TESIS – SM 142501

PENGEMBANGAN SISTEM ESTIMASI KECEPATAN PADA KENDARAAN BERGERAK BERBASIS PENGOLAHAN CITRA DIGITAL

Danang Wahyu Wicaksono
1215201008

Dosen Pembimbing:
Dr. Budi Setiyono, S.Si., M.T.

PROGRAM MAGISTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

THESIS – SM 142501

SYSTEM DEVELOPMENT FOR SPEED ESTIMATION ON MOVING VEHICLES BASED ON DIGITAL IMAGE PROCESSING

Danang Wahyu Wicaksono
1215201008

Supervisors:
Dr. Budi Setiyono, S.Si., M.T.

MASTER DEGREE
MATHEMATICS DEPARTMENT
FACULTY OF MATHEMATICS AND NATURAL SCIENCES
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2017

PENGEMBANGAN SISTEM ESTIMASI KECEPATAN PADA KENDARAAN BERGERAK BERBASIS PENGOLAHAN CITRA DIGITAL

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Sains (M.Si.)
di
Institut Teknologi Sepuluh Nopember

oleh:
DANANG WAHYU WICAKSONO
NRP. 1215 201 008

Tanggal Ujian : 9 Januari 2017
Periode Wisuda : Maret 2017

Disetujui oleh :

Dr. Budi Setiyono, S.Si., M.T. **(Pembimbing)**
NIP 19720207 199702 1 001

Prof. Dr. Mohammad Isa Irawan, M.T. **(Penguji)**
NIP 19631225 198903 1 001

Dr. Dwi Ratna Sulistyaningrum, S.Si., M.T. **(Penguji)**
NIP 19690405 199403 2 003

Dr. Hariyanto, M.Si. **(Penguji)**
NIP 19530414 198203 1 002

Direktur Program Pascasarjana,

an. Direktur Program Pascasarjana
Asisten Direktur

Prof. Dr. Ir. Tri Widjaja, M.Eng.
NIP. 19611021 198603 1 001

Prof. Ir. Djauhar Manfaat, M.Sc., Ph.D.
NIP. 19601202 198701 1 001

PENGEMBANGAN SISTEM ESTIMASI KECEPATAN PADA KENDARAAN BERGERAK BERBASIS PENGOLAHAN CITRA DIGITAL

Nama Mahasiswa : Danang Wahyu Wicaksono
NRP : 1215 201 008
Pembimbing : Dr. Budi Setiyono, S.Si., M.T.

ABSTRAK

Seiring dengan perkembangan teknologi informasi dan komunikasi, masyarakat urban dunia kini mengenal suatu istilah baru bernama *Smart City* yang artinya Kota Cerdas. Salah satu komponen *smart city* adalah *smart transportation* yang dikenal dengan *Intelligent Transportation System* (ITS) yang di dalamnya terdapat manajemen transportasi kendaraan di jalan raya. Pemasangan kamera CCTV (*Closed Circuit Television*) pada ruas-ruas jalan saat ini sudah banyak dilakukan. Hal ini dapat dimanfaatkan untuk memantau kondisi dan mendeteksi permasalahan lalu lintas seperti kemacetan dan pelanggaran batas kecepatan kendaraan. Pada penelitian ini dibahas mengenai estimasi kecepatan kendaraan bergerak berbasis pengolahan citra melalui data video dengan menggunakan metode *Euclidean Distance* secara *multi* objek dengan beberapa jenis sudut akuisisi video. Tahap pertama, video diekstrak menjadi kumpulan *frame* kemudian dilakukan *preprocessing* untuk minimalisir efek bayangan. Selanjutnya *frame* dicari *foreground*-nya menggunakan metode Gaussian Mixture Model. Kemudian hasil *foreground* melalui proses *filter*, *shadow removal*, dan operasi morfologi. Objek kendaraan yang terdeteksi kemudian dilakukan *tracking* untuk mendapatkan informasi lokasi objek di setiap *frame* untuk menghitung kecepatannya. Dari hasil uji coba, sistem yang dibangun mampu mengestimasi kecepatan kendaraan dengan tingkat akurasi terendah adalah 87,01% dan tertinggi adalah 99,38% dengan faktor pemilihan daerah ROI, deteksi objek, dan akurasi *input* parameter geometris yang baik.

Kata Kunci : *Estimasi Kecepatan Kendaraan, Sistem Cerdas Pemantauan Lalu Lintas, Smart City*

SYSTEM DEVELOPMENT FOR SPEED ESTIMATION ON MOVING VEHICLES BASED ON DIGITAL IMAGE PROCESSING

Name : Danang Wahyu Wicaksono
NRP : 1215 201 008
Supervisor : Dr. Budi Setiyono, S.Si., M.T.

ABSTRACT

Along with the development of information and communication technology, the world urban communities now recognize a new term called Smart City. One of Smart City components is smart transportation, known as Intelligent Transportation System (ITS) in which there is a vehicles transportation management on the highway. Installation of CCTV (Closed Circuit Television) on the streets are now widely performed. It can be used to monitor conditions and detect problems such as traffic jam and vehicle speed limit violation. This research discussed about vehicles speed estimation using image processing from video data with *Euclidean Distance* method for multiple objects with many different types of camera angle. The first step, video data is extracted into frames and applied preprocessing to extracted frames to minimize shadow effect. Then, using GMM method to extract foreground image. In the next step, the obtained foreground is filtered using median filter, shadow removing, and morphology operation. The detected vehicle objects will be tracked to determine the location of those objects in each pixel to estimate the speed. From the obtained results, this system is capable on estimating the speed of moving vehicles with degree of accuracy with the lowest is 87,01% and the highest is 99.38% with good support from ROI area selection, object detection, and geometric parameter input.

Keywords : *Vehicle Speed Estimation, Intelligent Traffic Monitoring System, Smart City*

KATA PENGANTAR

Dengan rahmat Allah SWT, syukur Alhamdulillah penulis dapat menyelesaikan tesis yang berjudul:

Pengembangan Sistem Estimasi Kecepatan pada Kendaraan Bergerak Berbasis Pengolahan Citra Digital

Tesis ini disusun sebagai salah satu syarat kelulusan Program Studi Strata 2 (S-2) Program Magister Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam (FMIPA) Institut Teknologi Sepuluh Nopember Surabaya.

Dalam penulisan tesis ini, tidak akan terselesaikan dengan baik tanpa adanya bantuan dan bimbingan dari berbagai pihak. Oleh karena itu, penulis mengucapkan banyak terima kasih kepada:

1. Kedua orang tua dengan segala kasih sayangnya selalu memberikan motivasi, saran, dukungan, dan doa kepada penulis sehingga dapat menyelesaikan tesis ini.
2. Bapak Prof. Ir. Joni Hermana, M.Sc.ES., Ph.D., selaku Rektor Institut Teknologi Sepuluh Nopember yang telah memberikan kesempatan dan fasilitas kepada penulis untuk menyelesaikan tesis ini.
3. Bapak Prof. Ir. Djauhar Manfaat, M.Sc., Ph.D., selaku Direktur Program Pascasarjana Institut Teknologi Sepuluh Nopember.
4. Bapak Prof. Dr. Basuki Widodo, M.Sc., selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam (FMIPA).
5. Bapak Dr. Imam Mukhlash, S.Si., M.T., selaku Ketua Jurusan Matematika Institut Teknologi Sepuluh Nopember.
6. Bapak Dr. Mahmud Yunus, M.Si., selaku koordinator Program Studi Pascasarjana Matematika ITS.
7. Bapak Dr. Budi Setiyono, S.Si, M.T. selaku dosen wali sekaligus dosen pembimbing tesis yang telah banyak mengarahkan, membimbing, dan memberikan motivasi sehingga tesis ini dapat terselesaikan.

8. Ibu Dr. Dwi Ratna Sulistyaningrum, S.Si, M.T., Bapak Prof. Dr. Mohammad Isa Irawan, M.T., dan Bapak Dr. Hariyanto, M.Si., selaku dosen penguji tesis yang telah banyak memberikan saran dan masukan sehingga tesis ini dapat diselesaikan dengan baik.
9. Seluruh dosen, staf, dan karyawan Jurusan Matematika ITS yang telah memberikan bantuan selama penulis menempuh perkuliahan ini.
10. Saudaraku Vian dan Fahmi yang telah meluangkan waktu dan bersedia membantu penulis dalam mengambil data untuk keperluan tesis ini.
11. Teman-teman S2 Matematika ITS angkatan 2015.
12. Seluruh pihak yang telah memberikan saran, dukungan, dan motivasi kepada penulis dalam penyelesaian tesis ini.

Penulis menyadari bahwa dalam penulisan ini masih banyak kekurangan, kesalahan, dan masih jauh dari kata sempurna sehingga segala kritik dan saran yang sifatnya membangun sangat diperlukan. Penulis berharap tesisi ini dapat memberikan manfaat bagi penulis pada khususnya dan pembaca pada umumnya.

Surabaya, Januari 2017

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan perkembangan teknologi informasi dan komunikasi, masyarakat urban dunia kini mengenal suatu istilah baru bernama *Smart City* yang artinya Kota Cerdas. Kota cerdas didefinisikan sebagai sebuah konsep pengembangan dan pengelolaan kota dengan pemanfaatan Teknologi Informasi dan Komunikasi (TIK) untuk menghubungkan, memonitor, dan mengendalikan berbagai sumber daya yang ada di dalam kota dengan lebih efektif dan efisien untuk memaksimalkan pelayanan kepada warganya serta mendukung pembangunan yang berkelanjutan [5].

Salah satu komponen *smart city* adalah *smart transportation* yang dikenal dengan *Intelligent Transportation System* yang di dalamnya terdapat manajemen transportasi kendaraan di jalan raya. Sistem pemantauan cerdas pada lalu lintas di jalan raya masih belum sempurna tanpa adanya sistem yang mampu mendeteksi permasalahan lalu lintas seperti pelanggaran-pelanggaran aturan lalu lintas dan kemacetan secara otomatis. Pemasangan kamera CCTV (*Closed Circuit Television*) pada ruas-ruas jalan saat ini telah banyak dilakukan. Hal ini dapat dimanfaatkan untuk memantau kondisi dan mendeteksi permasalahan lalu lintas yang sering terjadi di kota-kota besar seperti kemacetan dan pelanggaran batas kecepatan kendaraan, dimana kemacetan lalu lintas dapat dideteksi oleh kecepatan kendaraan yang rendah dan diikuti dengan volume kendaraan yang tinggi. Sehingga hasil dari penelitian ini diharapkan dapat dijadikan sebagai bahan penelitian lebih lanjut untuk otomasi sistem pemantuan cerdas pada lalu lintas dan dapat dijadikan acuan untuk melakukan rekayasa lalu lintas oleh pihak terkait untuk menanggulangi permasalahan-permasalahan tersebut berdasarkan profil kecepatan yang terjadi di jalan raya.

Pada penelitian-penelitian sebelumnya tentang deteksi dan estimasi kecepatan kendaraan berbasis pengolahan citra digital dilakukan dengan berbagai macam metode, antara lain metode *Gray Constraint Optical Flow* [7], *Euclidean Distance* [4][6], *Diagonal Hexadecimal Pattern* (DHP) [2], *Mean Filter* (CVS) [9], dan

Motion Vector Technique [8]. Dari penelitian-penelitian tersebut, pengaruh sudut kemiringan kamera dalam akuisisi video dan pemilihan daerah ROI (*Region of Interest*) terhadap hasil estimasi kecepatan serta kemampuan sistem dalam mendeteksi kecepatan kendaraan secara *multi* objek belum diteliti lebih lanjut. Berdasarkan hal tersebut, penulis mengusulkan ide untuk mengembangkan sistem estimasi kecepatan kendaraan secara *multi* objek menggunakan *euclidean distance* serta modifikasi skema uji coba. Modifikasi yang akan dilakukan adalah mengubah sudut kemiringan kamera dan pemilihan daerah ROI sedemikian hingga diharapkan dapat meningkatkan keakuratan deteksi kendaraan dan estimasi kecepatannya. Hasil dari penelitian ini adalah suatu sistem yang mampu menunjukkan estimasi kecepatan setiap kendaraan yang terdeteksi secara *multi* objek.

1.2 Rumusan Masalah

Dari latar belakang yang telah dipaparkan, masalah yang akan dibahas pada penelitian ini adalah:

1. Bagaimana melakukan kajian dan analisis algoritma untuk deteksi kecepatan kendaraan yang telah dilakukan pada penelitian sebelumnya?
2. Bagaimana analisis estimasi kecepatan kendaraan bergerak menggunakan metode *euclidean distance* yang akan digunakan untuk estimasi kecepatan kendaraan secara *multi* objek?
3. Bagaimana pengaruh kemiringan kamera dan pemilihan daerah ROI terhadap hasil dari estimasi kecepatan kendaraan yang didapatkan?

1.3 Batasan Masalah

Permasalahan yang akan dibahas dalam penelitian ini dibatasi sebagai berikut:

1. Data yang digunakan untuk uji coba adalah rekaman video bertipe *.avi menggunakan kamera biasa non *infrared* yang menampilkan arus lalu lintas kendaraan di jalan raya satu arah pada saat pagi dan siang hari dengan kondisi cuaca cerah.
2. Validasi hasil estimasi kecepatan dilakukan dengan uji empiris yaitu dengan menggunakan rekaman video kendaraan yang telah diatur kecepatannya.

3. Sudut pandang yang diambil ketika merekam adalah dari atas dengan jangkauan pandangan seluruh jalan dan posisi kamera tidak bergerak.
4. Pada eksperimen, informasi digital yang diproses dilakukan secara *offline*.
5. Deteksi pergerakan kendaraan dan estimasi kecepatan yang diteliti adalah objek-objek yang terekam pada daerah ROI (*Region of Interest*).

1.4 Tujuan

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

1. Melakukan kajian dan analisis algoritma untuk deteksi kendaraan yang telah dilakukan pada penelitian sebelumnya.
2. Merancang sistem yang dapat mengolah informasi dari video digital untuk mendeteksi pergerakan kendaraan dan mengestimasi kecepatannya dengan menggunakan metode *euclidean distance* secara *multi* objek.
3. Melakukan analisis pengaruh modifikasi posisi kamera dan pemilihan daerah ROI terhadap hasil deteksi kendaraan dan kecepatannya.

1.5 Manfaat

Manfaat yang diharapkan pada penelitian ini adalah memberikan informasi tentang kecepatan kendaraan yang melaju di jalan sehingga dapat dijadikan bahan penelitian lebih lanjut seperti integrasi ke sistem klasifikasi jenis kendaraan dan sistem penghitung jumlah kendaraan untuk otomasi sistem pemantauan cerdas pada lalu lintas.

BAB 2
KAJIAN PUSTAKA DAN DASAR TEORI

2.1 Penelitian Sebelumnya

Terdapat beberapa penelitian sebelumnya pada estimasi kecepatan kendaraan dengan berbagai macam metode. Arash Gholami R., dkk dalam jurnalnya yang berjudul "*Vehicle Speed Detection in Video Image Sequences using CVS method*" melakukan deteksi kecepatan dengan menggunakan metode CVS sebagai model *background* dan *euclidean distance* sebagai metode penghitungan kecepatannya, dalam penelitian tersebut juga dibandingkan hasil yang diperoleh dengan menggunakan model-model *background* yang berbeda dan berhasil melakukan estimasi kecepatan kendaraan dengan akurasi tertinggi adalah 90% [9].

Jinhui Lan, dkk dalam jurnalnya yang berjudul "*Vehicle Speed Measurement based on Gray Constraint Optical Flow*" menyatakan bahwa dasar metode VSM (*Vehicle Speed Measurement*) untuk menghitung estimasi kecepatan kendaraan adalah perbedaan tiga *frame*. Kemudian akurasi estimasi kecepatan ditingkatkan dengan menggunakan nilai *optical flow* dan berhasil dengan akurasi tertinggi mencapai 99% [7], namun metode *optical flow* yang digunakan untuk deteksi kecepatan sangat sensitif terhadap *noise* dan membutuhkan waktu komputasi yang lama.

Kemudian Hardy Santosa S dan Agus Harjoko dalam jurnalnya yang berjudul "*Vehicle Counting and Vehicle Speed Measurement Based On Video Processing*" menggunakan metode *euclidean distance* untuk estimasi kecepatan kendaraan dan menyatakan bahwa hasil estimasi kecepatan dari pengolahan data video masih lebih baik daripada menggunakan *speed gun* dengan tingkat akurasi tertinggi adalah 90,5% [3]. Kemudian Asif Khan, dkk melakukan penelitian untuk estimasi kecepatan kendaraan. Dalam penelitiannya yang berjudul *"Speed Estimation of Vehicle in Intelligent Traffic Surveillance System Using Video Image Processing"*, perangkat lunak yang dibangun mampu mendapatkan estimasi kecepatan kendaraan menggunakan metode *euclidean distance* dengan tingkat keberhasilan maksimal 98,12% [6]. Masih terkait deengan metode *euclidean distance*, sistem estimasi

kecepatan oleh Reena Gokule dan Amit Kulkarni pada penelitiannya yang berjudul "*Video Based Vehicle Speed Estimation and Stationary Vehicle Detection*" mampu menghasilkan akurasi maksimal mencapai 99% [4]. Namun pada penelitian-penelitian tersebut belum menjelaskan lebih lanjut tentang pengaruh kemiringan sudut kamera dalam akuisisi video dan pemilihan daerah ROI serta kemampuan sistem dalam mendeteksi kecepatan kendaraan secara *multi* objek.

Pada penelitian ini, digunakan metode *euclidean distance* untuk estimasi kecepatan kendaraan secara *multi* objek karena akurasi hasil yang baik yang didapat dari penelitian sebelumnya. Pada sub bab selanjutnya akan disajikan beberapa teori dasar dan teori pendukung dalam estimasi kecepatan kendaraan.

2.2 Citra Digital

Citra adalah suatu representasi (gambaran), kemiripan, atau imitasi dari suatu objek. Citra sebagai keluaran suatu sistem perekaman data dapat bersifat optik berupa foto, bersifat analog berupa sinyal-sinyal video seperti gambar pada monitor televisi, atau bersifat digital yang dapat langsung disimpan pada suatu media penyimpanan.

Citra terbagi menjadi dua yaitu citra diam (*still image*) dan citra bergerak (*moving image*). Citra diam adalah citra tunggal yang tidak bergerak. Sedangkan, citra bergerak adalah rangkaian citra diam yang ditampilkan secara beruntun (*sequential*) sehingga memberi kesan pada mata 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 kontinu tidak dapat direpresentasikan secara langsung oleh komputer. Oleh karena itu dilakukan sebuah proses untuk merubah nilai-nilai yang ada pada citra kontinu agar komputer dapat membaca dan menerjemahkan informasi yang terdapat pada citra kontinu. Hasil dari pemrosesan tersebut dinamakan sebagai citra digital.

Dalam buku *Digital Image Processing* yang ditulis oleh Rafael C. Gonzalez dan Richard E. Woods, menjelaskan bahwa citra digital merupakan fungsi dua

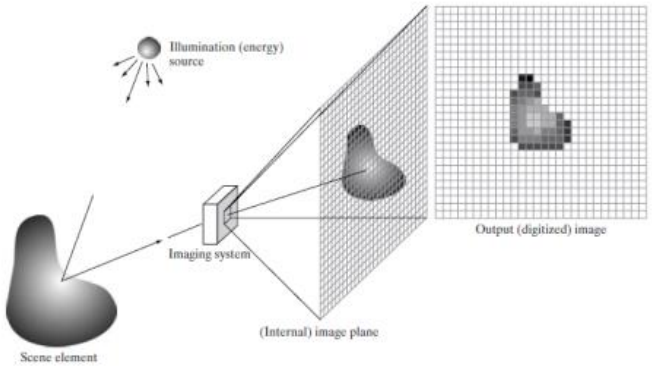
dimensi yang dapat dinyatakan dengan fungsi *f(x,y)*, dimana *x* dan *y* merupakan titik koordinat spasial. Dan nilai dari fungsi *f* pada sembarang koordinat (*x*,*y*) merupakan nilai intensitas cahaya, yang merupakan representasi dari warna cahaya yang ada pada citra kontinu. Citra digital adalah suatu citra dimana (*x*,*y*) dan nilai intensitas dari *f* terbatas (*discrete quantities*), dan telah dilakukan proses digitalisasi spasial dan digitalisisi kuantitas [12].

2.2.1 Digitalisasi Spasial (*Sampling*)

Sampling merupakan proses pengambilan informasi dari citra kontinu yang memiliki panjang dan lebar tertentu untuk membaginya ke beberapa blok kecil. Blok-blok tersebut disebut sebagai piksel. Sehingga citra digital yang lazim dinyatakan dalam bentuk matriks memiliki ukuran *M* x *N* dengan *M* sebagai baris dan *N* kolom. Bisa juga disebut sebagai citra digital yang memiliki *M* x *N* buah piksel. Notasi matriks citra digital dapat dinyatakan sebagai berikut:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,M-1) \\ f(1,0) & \cdots & \cdots & f(1,M-1) \\ \cdots & & & \cdots \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,M-1) \end{bmatrix} \quad (2.1)$$

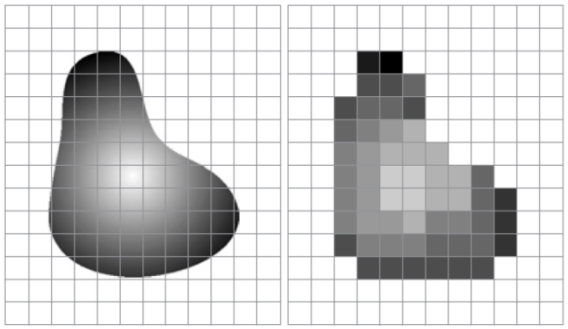
Proses *sampling* citra kontinu ke citra digital ditampilkan pada Gambar 2.1 di bawah ini:



Gambar 2.1: Proses sampling citra dari citra kontinu ke citra digital [12].

2.2.2 Digitalisasi Intensitas (Kuantisasi)

Kuantisasi adalah proses pemberian nilai derajat keabuan di setiap titik piksel yang merupakan representasi dari warna asli dari citra kontinu. Rentang nilai keabuan adalah 0 – 255.



Gambar 2.2: Proses sampling dan kuantisasi. (kiri) Gambar citra kontinu yang ditempatkan pada sensor array. (kanan) Gambar citra setelah proses sampling dan kuantisasi, tiap piksel pada citra tersebut memiliki derajat keabuan masing-masing [12].

2.3 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. 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 kontinu dan kecepatan untuk menampilkan citra-citra yang ada disebut sebagai *frame rate* dengan satuan *fps* (*frame per second*). Jika *frame rate* cukup tinggi, maka video akan telihat semakin 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 kontinu $I(x,y,t)$ dengan jumlah citra (x,y) tertentu dan waktu (t) tertentu.

2.4 *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 *background* model). Kemudian nilai citra hasil pengurangan tersebut

dibandingkan dengan nilai ambang batas (*threshold*) sesuai dengan metode yang dipakai untuk membangun *foreground image*.

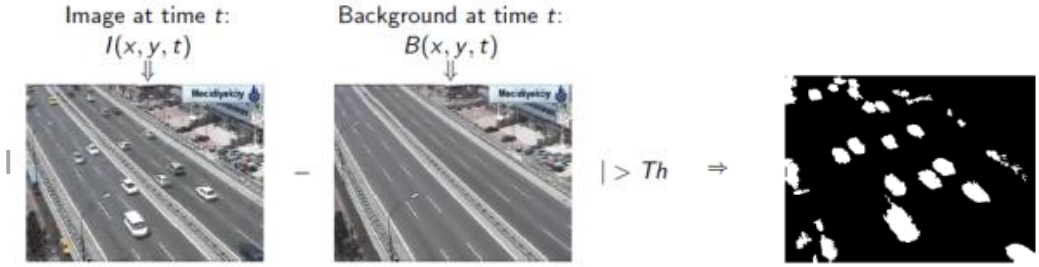
Secara matematis proses tersebut dapat ditulis sebagai berikut

$$|I(x,y,t) - B(x,y,t)| = d(x,y,t) \tag{2.2}$$

$$f(x,y,t)\begin{cases}1 & jika\ d(x,y,t) \geq \tau \\ 0 & jika\ d(x,y,t) < \tau\end{cases} \tag{2.3}$$

Dimana $I(x,y,t)$ adalah citra pada waktu *t*, $B(x,y,t)$ adalah *background model* pada waktu *t*, $d(x,y,t)$ adalah citra hasil pengurangan *I dan B* pada waktu *t*, $f(x,y,t)$adalah *foreground image* (gerak yang dideteksi) dan τ adalah *threshold*.

Ada berbagai macam metode untuk menginisialisasi background model. Salah satunya adalah metode *Gaussian Mixture Model* atau GMM.



Gambar 2.3: Proses *background subtraction* dengan metode *frame differencing* untuk mendapatkan *foreground image* [13].

2.5 Metode Gaussian Mixture Model (GMM)

Gaussian Mixture Model (GMM) adalah salah satu metode dari *Background Subtraction*. 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, yaitu 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 berupa skalar (citra *grayscale*) maupun vektor (citra berwarna). Prosedur dari metode *Gaussian Mixture Model* disajikan pada Gambar 2.4.

Terdapat beberapa tahaan dalam pemilihan distribusi yang mencerminkan *background* [13]. Tahapan-tahapan tersebut adalah:

1. Pencocokan *input* terhadap distribusi

   Pada tahap ini *input* dicocokkan dengan semua distribusi sampai ditemukan distribusi yang paling cocok. Suatu piksel dikatakan masuk dalam suatu distribusi jika nilai piksel tersebut masuk dalam jarak 2.5 standar deviasi dari sebuah distribusi. Untuk pencocokan *input* digunakan pertidaksamaan 2.4.

   $$\mu_{i,j,k} - 2.5 * \sigma_{i,j,k} < X_{i,j,t} < \mu_{i,j,k} + 2.5 * \sigma_{i,j,k} \quad (2.4)$$

   Dimana X_t adalah nilai intensitas dari suatu piksel (i, j) pada *frame* ke-t, μ_k adalah nilai *mean* pada piksel (i, j) dari gaussian ke-k, dan σ_k sebagai standar deviasi pada piksel (i, j) dari gaussian ke-k.

2. *Update* parameter

   Pada tahap ini dilakukan *update* terhadap nilai dari parameter-parameter GMM yang nantinya digunakan untuk mengolah *input* selanjutnya. Nilai yang di-*update* terdiri dari *weight, means,* dan *standard deviation*. Nilai *weight* di-*update* menggunakan Persamaan 2.5. Nilai *means* di-*update* menggunakan Persamaan 2.7 dengan ρ diberikan pada 2.6. Nilai *standard deviation* di-*update* menggunakan Persamaan 2.8. Setiap persamaan tersebut berlaku pada setiap piksel (i, j).

   $$\omega_{k,t} = (1 - \alpha)\omega_{k,t-1} + \alpha(M_{k,t}) \quad (2.5)$$

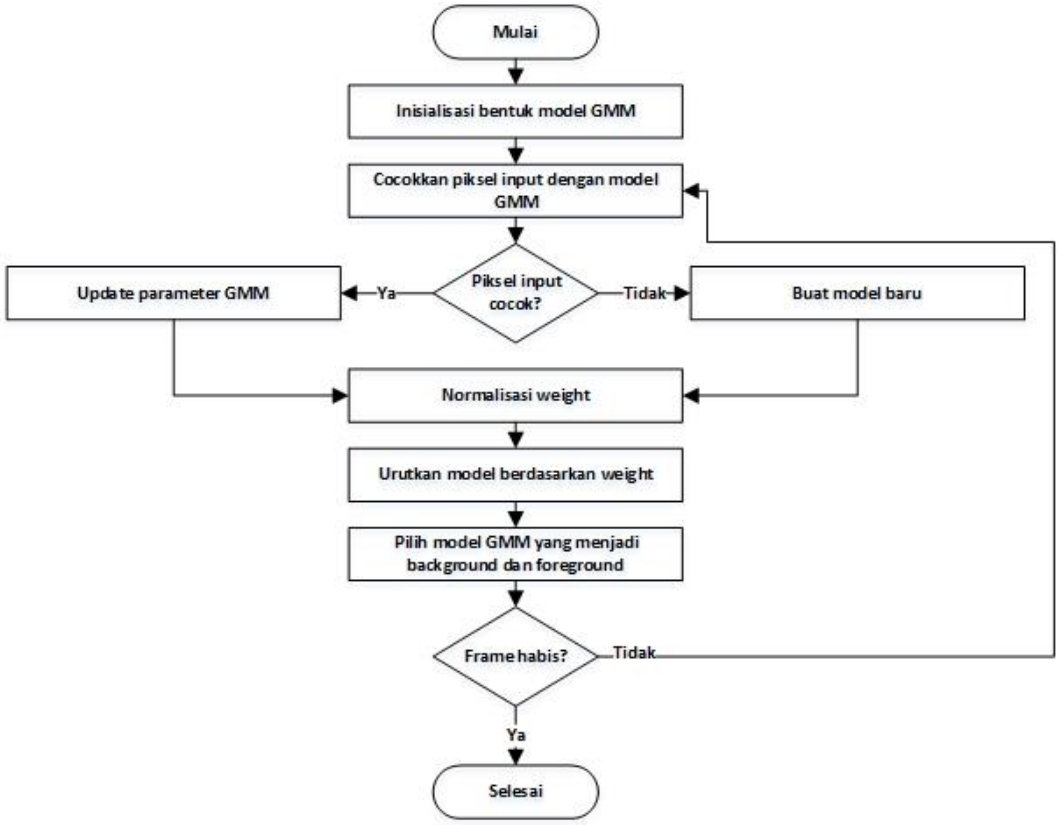
   $$\rho = \frac{\alpha}{\omega_{k,t}} \quad (2.6)$$

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

   $$\sigma_{k,t}^2 = (1 - \rho)\sigma_{k,t-1}^2 + \rho(X_t - \mu_{k,t})(X_t - \mu_{k,t}) \quad (2.8)$$

   Dimana $\omega_{k,t}$ adalah bobot dari gaussian ke-k pada *frame* t, $\mu_{k,t}$ adalah *mean* dari gaussian ke-k pada *frame* t, $\sigma_{k,t}$ adalah standar deviasi dari gaussian ke-k pada *frame* ke-t, α adalah *learning rate*, dan nilai $M_{k,t}$ adalah 1 untuk model yang cocok dan 0 untuk model yang tidak cocok. Setelah nilai *weight* di-*update* dilakukan normalisasi sehingga total bobot

dari semua distribusi tepat 1. Sementara *means* dan standar deviasi di-*update* hanya jika ada nilai piksel yang cocok dengan distribusi tersebut.



Gambar 2.4: Diagram Alir Metode GMM

3. Pemilihan Distribusi *Background*

   Pada tahap ini dipilih model-model yang mencerminkan *background*. Pertama model-model diurutkan berdasarkan $\frac{\omega}{\sigma^2}$ sehingga distribusi yang paling mencerminkan *background* tetap di atas dan yang tidak mencerminkan *background* ada di bawah yang nantinya digantikan oleh distribusi yang lain. Untuk memilih B distribusi pertama yang dijadikan distribusi *background* digunakan Persamaan 2.9, dengan T adalah nilai ambang batas yang telah ditentukan sebelumnya.

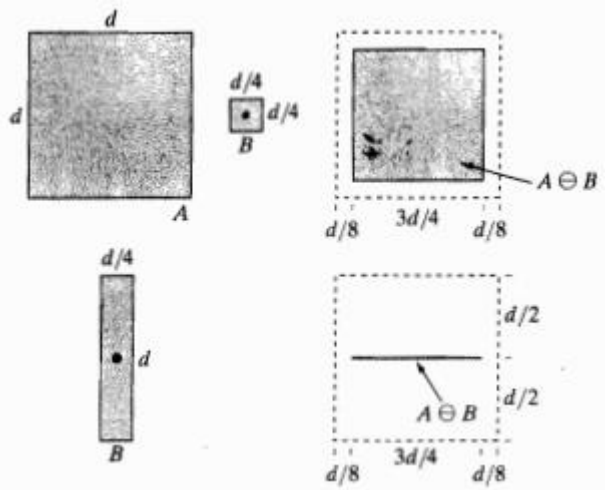
   $$B = argmin_b(\sum_{k=1}^{b} w_k > T) \quad (2.9)$$

2.6 Morfologi Citra

Morfologi dalam citra digital adalah suatu *tool* untuk ekstraksi komponen *image* yang berguna dalam representasi dan deskripsi dari bentuk daerah (*region shape*) dengan *structuring element* (SE) untuk menentukan *properties of interest* dari *image*. Di dalam morfologi, terdapat dua operasi fundamental yaitu *erosion* dan *dilation*. Secara matematis, dengan A dan Badalah himpunan pada Z^2, *erosion* dari A oleh B, dinotasikan dengan $A \ominus B$ didefinisikan oleh Persamaan 2.10.

$$A \ominus B = \{z|(B)_z \subseteq A\} \tag{2.10}$$

Dengan kata lain, *erosion* dari A oleh B adalah himpunan semua titik z sedemikian hingga semua titik z pada B termuat didalam A. Dalam hal ini, B diasumsikan sebagai *structuring element*. *Erosion* berguna untuk menyempitkan atau menipiskan objek pada *image* biner. Contoh *erosion* dapat dilihat pada Gambar 2.5.

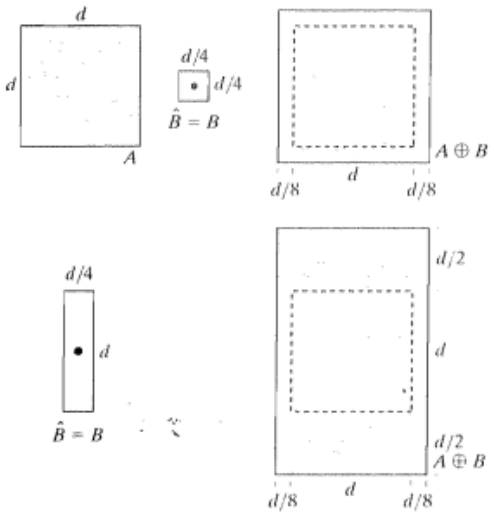


Gambar 2.5: Contoh *Erosion* pada Morfologi Citra [12].

Dengan asumsi yang sama dengan yang digunakan oleh Persamaan 2.10. *Dilation* dari A oleh B, dinotasikan dengan $A \oplus B$, didefinisikan oleh Persamaan 2.11.

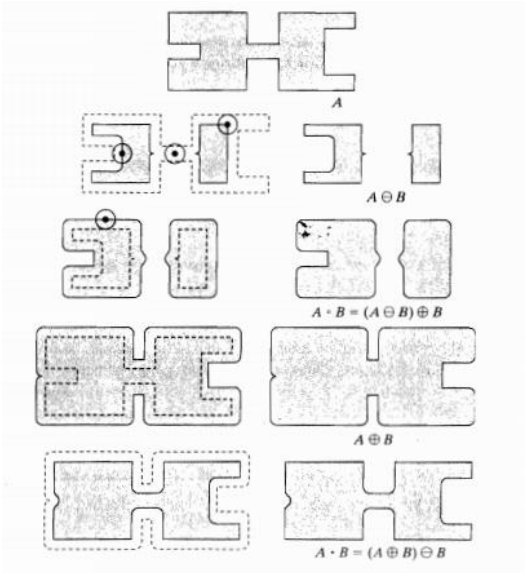
$$A \oplus B = \{z|(B)_z \cap A \neq \emptyset\} \tag{2.11}$$

Dilation berguna untuk memperluas atau menebalkan objek pada *image* biner. Contoh *dilation* dapat dilihat pada Gambar 2.6.



Gambar 2.6: Contoh *Dilation* pada Morfologi Citra [12].

Selain dua operasi fundamental *erosion* dan *dilation*, terdapat dua operasi lain yang merupakan kombinasi dari dua operasi fundamental tersebut, yaitu *opening* dan *closing*. Operasi *opening* adalah operasi yang melakukan proses *erosion* terlebih dahulu kemudian hasilnya akan dilakukan *dilation*. Sedangkan operasi *closing* adalah operasi yang melakukan proses *dilation* terlebih dahulu kemudian hasilnya akan dilakukan *erosion*. Operasi *opening* secara umum digunakan untuk memperhalus kontur objek, menghilangkan *gap* tipis dan tonjolan pada objek. Sedangkan operasi *closing* digunakan untuk menghubungkan *gap* tipis, menghilangkan lubang kecil (*small holes*), dan mengisi *gap* pada kontur objek. Contoh operasi *opening* dan *closing* dapat dilihat pada Gambar 2.7.



Gambar 2.7: Contoh Operasi *Opening* dan *Closing* [12].

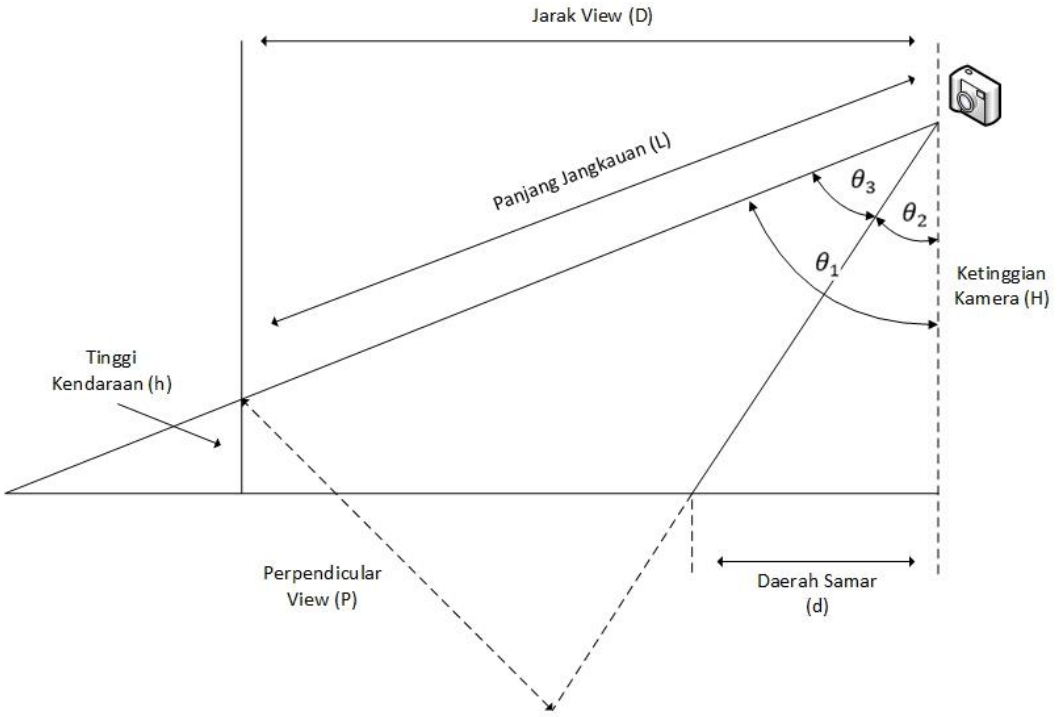
2.7 *Tracking*

Proses mencari objek bergerak dala urutan *frame* yang dikenal sebagai pelacakan (*tracking*). Pelacakan ini dapat dilakukan dengan menggunakan ekstraksi ciri benda dan mendeteksi objek/benda bergerak di urutan *frame*. Dengan menggunakan nilai posisi objek di setiap *frame*, bisa digunakan untuk menghitung posisi dan kecepatan objek bergerak [11]. Tujuan dari proses *tracking* adalah untuk mengetahui objek yang sama pada urutan *frame*.

2.8 *Speed Estimation* dengan Posisi Kamera di Atas

Kecepatan kendaraan diperoleh dari *frame* hasil deteksi *foreground*, yaitu dengan menentukan posisi kendaraan pada setiap *frame*. Jadi perlu ditentukan *boundingbox* dan *centroid* dari hasil *foreground* yang telah dilakukan sebelumnya. *Centroid* disini sangat penting untuk mengetahui jarak kendaraan yang bergerak dalam *frame* yang berurutan. Pada *frame* berurutan, setelah dideteksi objek kendaraan bergerak dan memberikan *boundingbox* maka harus ditentukan posisi objek pada *frame* ke t dan $t + 1$ dengan proses *tracking*.

Pada penelitian ini, digunakan posisi kamera di atas yang dapat direpresentasikan pada Gambar 2.8.



Gambar 2.8: Representasi Posisi Kamera di Atas.

Seperti yang ditunjukkan pada Gambar 2.8, kamera diatur pada ketinggan H di atas permukaan jalan dengan sumbu kemiringan optiknya yaitu θ_1 dari jalan. Hubungan antara sudut lensa kamera dan domain dari kamera dapat ditentukan dengan menggunakan persamaan geometris. Dari Gambar 2.8, persamaan yang sesuai dengan bidang tegak lurus bidang dapat ditulis sebagai berikut:

$$P = 2L \tan\left(\frac{\theta_3}{2}\right) \tag{2.12}$$

$$L = \sqrt{(H-h)^2 + D^2} \tag{2.13}$$

$$P = 2 \tan\left(\frac{\theta_3}{2}\right) \sqrt{(H-h)^2 + D^2} \tag{2.14}$$

dimana P adalah bidang tegak lurus pandang di layar 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,

h adalah tinggi dari kendaraan,

L adalah jarak nyata antara kamera dan kendaraan.

Jika $\theta_1 \rightarrow \angle 90°$ kemudian $L \rightarrow D$, maka persamaan diatas dapat disederhanakan menjadi:

$$P = 2D \tan\left(\frac{\theta_3}{2}\right) \tag{2.15}$$

Selain itu, θ_1 bisa didapatkan jika diasumsikan tidak ada kendaraan yang lewat, dengan kata lain $h \rightarrow 0$, maka dapat ditulis :

$$\theta_1 = \arctan\left(\frac{D}{H}\right) \tag{2.16}$$

dan sudut untuk daerah yang samar adalah :

$$\theta_2 = \arctan\left(\frac{d}{H}\right) \tag{2.17}$$

,d adalah daerah yang samar seperti yang ditunjukkan pada Gambar 2.8.

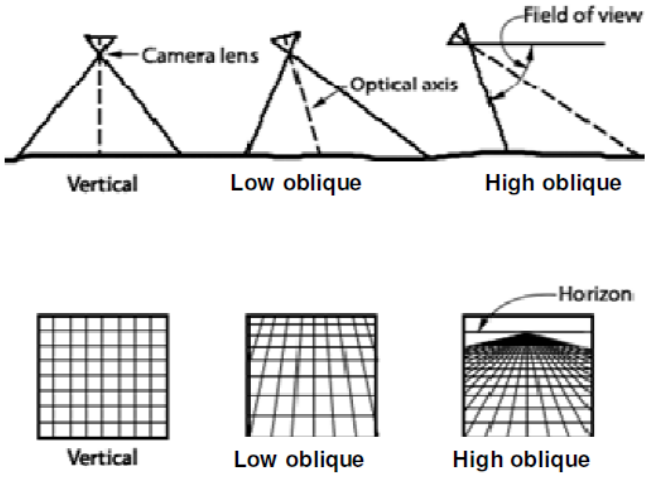
Selain itu juga, diketahui bahwa :

$$\theta_3 = \theta_1 - \theta_2 \text{ sehingga } \theta_2 = \theta_1 - \theta_3. \tag{2.18}$$

Daerah samar d bisa diperoleh dengan :

$$d = H \tan(\theta_1 - \theta_3) = H \tan \theta_2 \quad (2.19)$$

Kecenderungan dari kamera dengan posisi seperti Gambar 2.8 akan menghasilkan gambar dari tiga jenis area seperti yang ditunjukkan pada Gambar 2.9 yang bagian atas. Sedangkan Gambar 2.9 yang bagian bawah merupakan *grid* bagian garis yang terlihat pada berbagai jenis sudut kamera.



Gambar 2.9: Tendensi Kamera dan Tampilan *Grid* Hasil Rekaman dari Atas [9].

Untuk mengetahui jarak yang ditempuh dalam piksel, dapat dimisalkan koordinat suatu objek seperti berikut:

$$c_t(a, b) \text{ dan } c_{t+1}(c, d)$$

Dimana c_t dan c_{t+1} adalah posisi titik *centroid* pada *frame* t dan $t + 1$ untuk satu objek yaitu dengan koordinat (a, b) dan koordinat (c, d).

Perbedaan jarak jauh untuk kendaraan dengan *Euclidean Distance* [4] yaitu:

$$d = \sqrt{(a - c)^2 + (b - d)^2} \quad (2.20)$$

Sedangkan, untuk kecepatan (*speed*) dapat dihitung dengan:

$$V = k\frac{d}{t} \quad (2.21)$$

$$k = \frac{actual\_height}{image\_height} \quad (2.22)$$

$$t = \frac{1}{fps} \quad (2.23)$$

Dengan k adalah koefisien kalibrasi, t adalah waktu antar 2 *frame* yang berurutan.

BAB 3
METODE PENELITIAN

Pada bagian ini diuraikan beberapa metode penelitian yang akan digunakan untuk mencapai tujuan penelitian.

3.1 Objek Penelitian

Pada penelitian ini objek penelitian yang akan digunakan adalah kendaraan bergerak dalam data berupa video rekaman. Untuk kebutuhan uji coba akan digunakan video rekaman kendaraan bergerak. Sedangkan untuk validasi hasil estimasi kecepatan dan perhitungan akurasi digunakan video yang diketahui kecepatan kendaraan sesungguhnya sebagai pembanding hasil estimasi yang didapatkan dan kecepatan kendaraan sesungguhnya.

3.2 Peralatan

Peralatan yang akan digunakan adalah kamera digital dan *tripod* untuk proses merekam video lalu lintas dan *software* dengan bahasa JAVA sebagai alat bantu untuk melakukan uji coba sistem. Selain itu diperlukan juga OpenCV sebagai *library* pada JAVA untuk membantu proses pengolahan citra. *Software Tool* yang akan digunakan untuk implementasi bahasa JAVA dan OpenCV adalah Netbeans 8.0.2.

3.3 Tahap Penelitian

1. Studi Literatur

   Pada tahap ini, akan dilakukan pendalaman kajian dan analisis pada penelitian-penelitian sebelumnya dan pengumpulan berbagai informasi tentang pengolahan video digital, deteksi objek, dan hal lain yang berkaitan dengan penelitian ini. Berbagai informasi tersebut didapatkan dari berbagai sumber pustaka yaitu buku, jurnal, dan internet.
2. Akuisisi Video

   Pada tahap ini, akan dilakukan pengambilan data berupa rekaman video kendaraan yang bergerak di jalan raya pada saat pagi atau siang hari dengan cuaca cerah. Pada penelitian ini akan digunakan posisi kamera di atas sebagai media dalam mendapatkan rekaman video.

3. Analisis dan Perancangan Sistem

Pada tahap ini, akan dilakukan analisis data berupa rekaman video, analisis parameter-parameter yang dibutuhkan untuk mendapatkan estimasi kecepatan dengan satuan standar km/jam, penentuan parameter-parameter yang dibutuhkan, serta perancangan proses-proses yang dijalankan pada sistem. Berikut adalah prosedur dalam melakukan estimasi kecepatan kendaraan :

(a) *Input* parameter-parameter yang dibutuhkan untuk estimasi kecepatan dan konversi satuan kecepatan standar, seperti jumlah *frame* yang berkaitan dengan durasi dan kecepatan video dalam bentuk *fps* (*frame per second*), tingga posisi kamera, sudut pengambilan video, dan parameter lain yang dibutuhkan.

(b) *Input* video untuk proses ekstraksi menjadi *frame-frame* yang kemudian akan dilakukan *preprocessing* yaitu *contrast and brightness adjustment* sebelum *frame* masuk ke proses *background substraction* menggunakan GMM.

(c) GMM akan menghasilkan *background* dan *non-background* yang kemudian didapat citra *foreground* sebagai representasi objek bergerak.

(d) Metode *connected component* digunakan untuk deteksi objek dan menghitung *centroid* dari objek tersebut.

(e) Objek kemudian diberikan label dan dilakukan *tracking* berdasarkan titik *centroid*.

(f) Menghitung estimasi kecepatan kendaraan dalam satuan standar km/jam dari jarak *centroid* dan waktu antar *frame*.

(g) Kecepatan dihitung selama kendaraan berada di daerah ROI, kemudian ketika kendaraan keluar dari daerah ROI, akan dihitung kecepatan rata-rata dari kendaraan tersebut.

(h) Hasil estimasi kecepatan rata-rata kendaraan jam kemudian dibandingkan dengan kecepatan kendaraan sesungguhnya untuk mengetahui kesalahan relatif (*relative error*) dari hasil estimasi yang

didapat. Prosedur yang dipaparkan sebelumnya digambarkan pada Gambar 3.1.

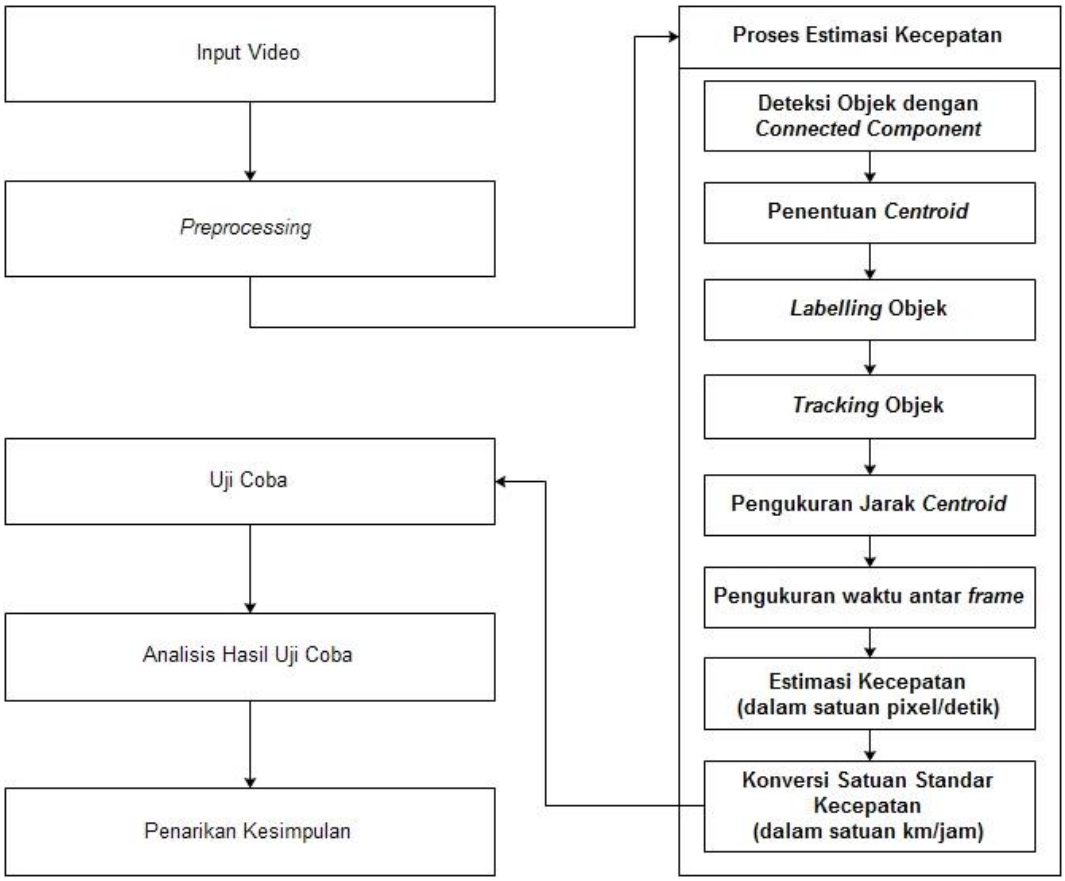
4. Implementasi Sistem

   Sistem yang telah dirancang pada langkah 3 kemudian diimplementasikan dalam bentuk aplikasi atau perangkat lunak dengan menggunakan peralatan yang telah dijelaskan pada subbab 3.2.

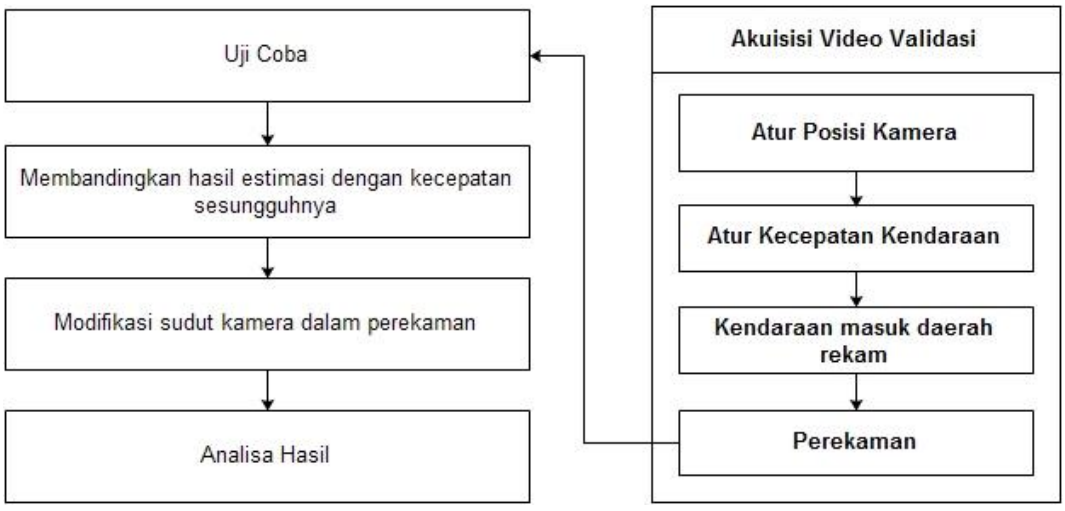
5. Uji Coba dan Analisis Hasil

   Sistem yang telah diimplementasikan kemudian dilakukan *testing* untuk estimasi kecepatan kendaraan pada video rekaman dari hasil akusisi video. Kemudian hasil estimasi kecepatan yang dihasilkan divalidasi dengan rekaman video kendaraan bergerak yang telah diatur kecepatannya, video kendaraan bergerak ini dibuat dengan parameter-parameter yang sama dengan video untuk uji coba atau *testing*, parameter yang dimaksud meliputi jenis kamera yang digunakan, *frame rate*, sudut kamera, dan ketinggian kamera, dan parameter lainnya yang dibutuhkan. Sedangkan kecepatan kendaraan ketika masuk dalam daerah rekam kamera adalah relatif konstan sehingga dapat dijadikan sebagai acuan untuk memvalidasi hasil estimasi. Hasil validasi sistem kemudian dianalisis keakuratannya. Adapun keakuratan tersebut dihitung dengan besar kesalahan relatif (*relative error*) nilai hasil estimasi kecepatan dibandingkan dengan nilai kecepatan sesungguhnya. Selanjutnya akan dilakukan modifikasi skema uji coba yaitu dengan mengubah sudut kamera dalam merekam video kemudia menganalisis hasil validasi dan *testing* pada video rekaman. Prosedur uji coba dan validasi dapat dilihat pada Gambar 3.2.

6. Penyusunan Tesis



Gambar 3.1: Blok Diagram Estimasi Kecepatan Kendaraan



Gambar 3.2: Blok Diagram Uji Coba dan Validasi

BAB 4
ESTIMASI KECEPATAN MENGGUNAKAN EUCLIDEAN DISTANCE

Pada bab ini dijelaskan mengenai analisis dan perancangan sistem pada penelitian ini, diantaranya adalah proses akuisisi video, proses *preprocessing* pada video input, proses *background substraction* menggunakan metode *Gaussian Mixture Model*, proses *smoothing* yaitu penghalusan citra yang dihasilkan oleh proses GMM dengan menggunakan metode *median filter*, *object detection enhancement* menggunakan teknik morfologi citra, proses *shadow removal* atau menghilangkan efek bayangan, proses pendeteksian objek, proses *tracking*, dan proses estimasi kecepatan. Selain proses-proses tersebut, perancangan sistem juga meliputi perancangan antarmuka (*interface*) untuk memudahkan dalam melakukan uji coba dan analisis. Pada bab ini juga akan dilakukan implementasi terhadap perancangan yang telah dibuat.

4.1 Analisis Sistem

Sesuai dengan tujuan dari penelitian ini, sistem yang dirancang harus mampu mengestimasi kecepatan kendaraan yang bergerak secara *multi* objek dari data masukan berupa video digital *offline*.

4.1.1 Deskripsi Sistem

Sebelum melakukan serangkaian proses yang sudah ditentukan sebelumnya, diperlukan data masukan dan beberapa parameter masukan. Data masukan yang diperlukan berupa rekaman video digital *offline* kendaraan bergerak yang diambil dari jembatan penyeberangan orang (JPO) dengan posisi kamera tidak bergerak atau statis. Parameter masukan adalah parameter-parameter yang diperlukan oleh metode-metode dari serangkaian proses yang digunakan oleh sistem dan dimasukkan oleh *user*. Parameter yang dibutuhkan pada sistem ini adalah parameter *Region of Interest* (ROI), parameter *preprocessing*, parameter *median filter*, parameter ukuran *kernel* morfologi, parameter pendeteksian objek, dan parameter-parameter geometris yang digunakan untuk kalibrasi kecepatan kendaraan yaitu mengkonversi dari satuan *image* piksel/s menjadi satuan standar yaitu km/jam.

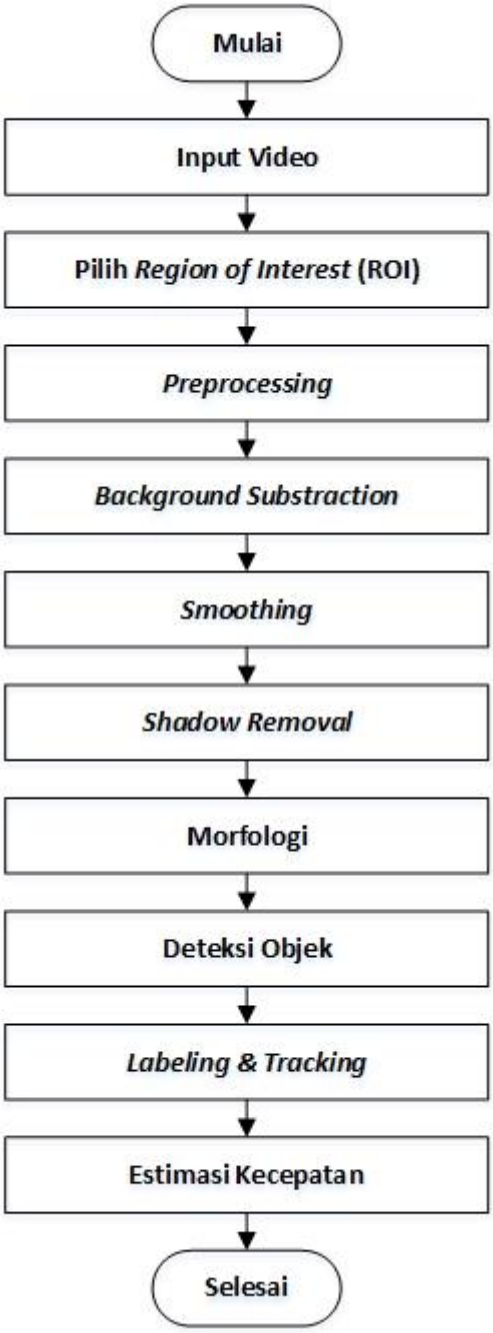
Setelah data masukan dan semua parameter diperoleh, sistem akan melakukan proses-proses selanjutnya hanya pada daerah ROI, hal ini dilakukan untuk mengurangi waktu komputasi dan mengurangi gangguan-gangguan yang tidak diinginkan dalam proses deteksi objek dan estimasi kecepatan kendaraan, seperti objek orang berjalan di trotoar, pohon bergerak di pinggir jalan, dan sebagainya. Proses pertama kali yang dilakukan adalah ekstraksi video menjadi rangkaian *frame*. Setiap *frame* yang berhasil diekstrak akan dilakukan *preprocessing* terlebih dahulu dengan tujuan untuk meminimalisir kemunculan bayangan objek yang solid sehingga dapat berpengaruh pada bentuk objek saat proses deteksi objek yang juga akan berpengaruh pada proses estimasi kecepatannya. *Preprocessing* dalam penelitian ini adalah berupa *contrast and brightness enhancement*. Setelah *frame* melalui proses *preprocessing* kemudian dicari model *background*-nya dengan menggunakan metode GMM. *Background* adalah *image* dari objek yang tidak bergerak, seperti jalan raya, trotoar, gedung, dan rumah. *Background* yang didapatkan akan dibandingkan dengan *frame* tersebut sehingga akan menghasilkan *image* baru yaitu *foreground*. *Foreground* merepresentasikan *image* dari objek yang bergerak. *Foreground* berbentuk *image greyscale* dengan 3 tingkat keabuan, yang masing-masing tingkat keabuan menyatakan objek tidak bergerak, objek bergerak, dan bayangan dari objek bergerak.

Hasil dari *image foreground* biasanya terdapat *noise*. *Noise* yang muncul biasanya akibat dari pantulan cahaya, daun yang jatuh, bayangan objek bergerak, atau gangguan-gangguan lain yang tidak diinginkan yang terdapat pada *image*. Untuk mengurangi *noise* diperlukan proses *smoothing*, yaitu proses untuk memperhalus *image*. Metode yang digunakan pada penelitian ini adalah *median filter*. Cara kerja *median filter* adalah dengan menggantikan nilai tiap piksel dengan nilai median dari piksel-piksel tetangganya. Setelah *image* menjadi lebih halus karena *noise* berkurang, diperlukan proses *shadow removal* untuk menghilangkan bayangan, yaitu dengan menyatakan bayangan objek bergerak sebagai objek yang tidak bergerak sehingga hasilnya adalah *image* biner dengan terdapat dua nilai untuk menyatakan objek bergerak dan objek tidak bergerak.

Setelah mendapatkan *image* biner, terkadang objek yang terdeteksi mempunyai lubang (*pixel holes*) akibat proses *shadow removal*, maka dilakukan proses *filling*

holes dengan memanfaatkan operasi morfologi untuk meningkatkan akurasi dan deteksi objek yang stabil. Proses selanjutnya adalah pendeteksian objek. Pendeteksian objek menggunakan konsep kontur (*contour*) yaitu *border following* dengan metode *connected component*, konsep kontur ini biasa digunakan untuk *shape analysis* dan *object detection and recognition* dalam pengolahan citra [14]. Dimana kontur yang terdeteksi pada *image* biner diidentifikasikan sebagai objek yang terdeteksi. Kontur kendaraan yang terdeteksi, kemudian dilakukan labelisasi dengan menggunakan *bounding box* sebagai representasi objek kendaraan yang terdeteksi. Setiap objek yang terdeteksi mempunyai luas area yang masing-masing merepresentasikan ukuran bentuk objek yang terdeteksi. Luas area ini digunakan sebagai suatu parameter *threshold* untuk menentukan bahwa objek bergerak yang diolah hanya kendaraan, dimana luas area kendaraan dapat bervariasi tergantung dari sudut kamera dan citra video yang dihasilkan dari proses akuisisi video. Dengan *bounding box* ini, dilakukan pelabelan lebih lanjut dengan memberikan ID objek dan posisi titik *centroid* untuk proses *tracking*.

Proses *tracking* bertujuan untuk mengetahui apakah kendaraan tersebut ada pada *frame* sebelumnya. Jika kendaraan tersebut ada pada *frame* sebelumnya, maka kendaraan tersebut menggantikan kendaraan yang sama pada *frame* sebelumnya, jika tidak maka kendaraan tersebut akan ditandai sebagai kendaraan yang baru terdeteksi. Pada proses *tracking* ini, dilakukan penghitungan jarak titik *centroid* antar *frame* kemudian dilakukan estimasi kecepatan kendaraan tersebut dan hal ini dilakukan selama kendaraan dalam daerah ROI sehingga yang terjadi adalah kecepatan kendaraan antar *frame* selalu dicatat. Ketika kendaraan meninggalkan ROI, kecepatan-kecepatan yang dicatat pada kendaraan tersebut kemudian dihitung rata-ratanya dan ditampilkan sebagai kecepatan rata-rata kendaraan tersebut. Kecepatan rata-rata pada proses ini masih dalam satuan standar yaitu km/jam. Proses konversi kecepatan dari satuan piksel/detik menjadi km/jam dilakukan dengan kalibrasi kalkulasi geometris berdasarkan parameter-parameter yang telah dimasukkan. Secara garis besar tahapan pada sistem dijelaskan pada Gambar 4.1.



Gambar 4.1: Tahapan Estimasi Kecepatan

4.1.2 Analisis Kebutuhan Sistem

Program akan dibuat dalam bentuk *desktop* dengan menggunakan bahasa pemrograman JAVA dengan bantuan IDE Netbeans 8.0.3. *Library* utama yang dibutuhkan untuk membangun sistem adalah OpenCV. OpenCV adalah sebuah *library* untuk pengolahan citra dan bersifat *open source*. Dikarenakan sistem membutuhkan data masukan berupa video digital maka juga dibutuhkan kamera digital untuk perangkat yang digunakan untuk merekam kendaraan.

4.2 Perancangan Sistem

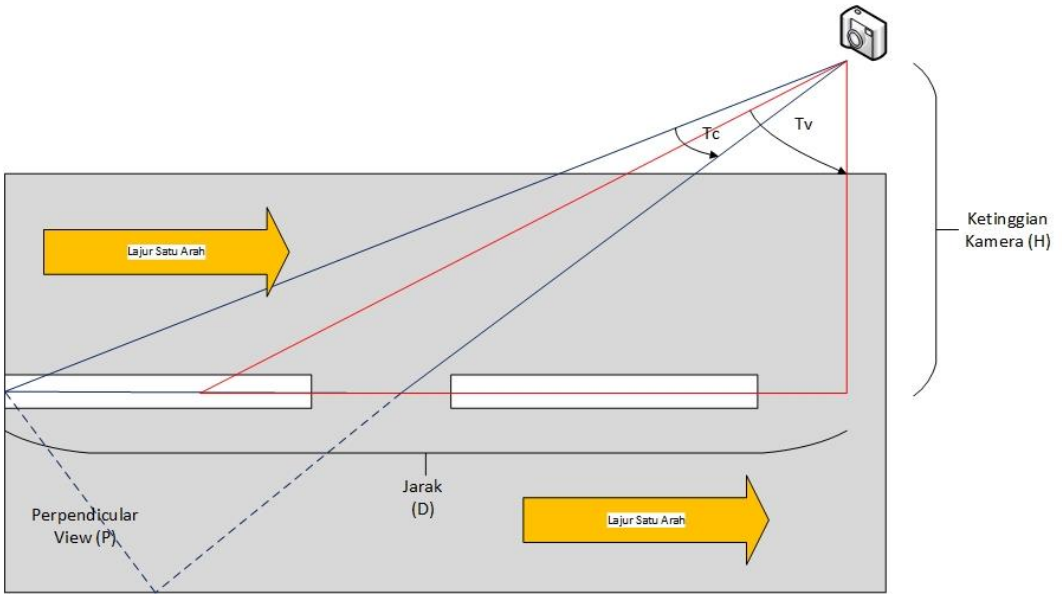
Pada subbab ini dijelaskan mengenai perancangan pada sistem yang telah dianalisis sebelumnya. Diawali dengan perancangan data yang meliputi data masukan, data proses, dan data keluaran. Kemudian penjelasan yang lebih detil mengenai tahapan proses estimasi kecepatan yang telah ditunjukkan pada Gambar 4.1. Dan terakhir adalah desain antarmuka sistem yang dibangun sebagai sarana untuk memproses data masukan dan parameter masukan yang ada dan melihat hasil yang diperoleh.

4.2.1 Perancangan Data

Data yang digunakan untuk estimasi kecepatan kendaraan bergerak terdiri dari data masukan, data proses, dan data keluaran. Data masukan merupakan video digital rekaman kendaraan bergerak di jalan raya yang dilakukan secara mandiri oleh penulis. Data proses adalah data masukan yang dibutuhkan oleh metode-metode dari serangkaian proses yang telah disebutkan sebelumnya. Sedangkan data keluaran adalah informasi mengenai hasil estimasi kecepatan kendaraan-kendaraan yang ada dalam video.

4.2.1.1 Akuisisi Data Video

Untuk memperoleh data masukan berupa video *offline*, dilakukan akusisi video terlebih dahulu. Adapun pengambilan video dilakukan di jalan satu arah di Surabaya dan diambil dari atas jembatan penyeberangan orang (JPO). Video harus mencakup semua bagian ruas jalan, sehingga sistem mampu memproses seluruh objek yang terdeteksi pada ruas jalan tersebut. Pada proses pengambilan video ini, parameter geometris untuk kalibrasi kecepatan harus dipenuhi, *layout* pengambilan video dapat dilihat pada Gambar 4.2.



Gambar 4.2: *Layout* Pengambilan Video

Parameter-parameter geometris yang terdapat pada Gambar 4.2 adalah sebagai berikut:

1. T_v adalah sudut kemiringan kamera terhadap garis tegak lurus dari permukaan jalan, digambarkan oleh garis berwarna merah.
2. T_c adalah sudut jangkauan FOV (*Field Of View*) dari kamera, digambarkan oleh garis berwarna biru.
3. H adalah tinggi kamera dari permukaan jalan.

Tiga parameter di atas digunakan untuk menghitung *perpendicular view* dan jarak nyata antara objek dengan kamera yang nantinya akan digunakan untuk kalibrasi kecepatan. Hasil rekaman video tersebut akan dijadikan data masukan ke sistem. Namun perlu dipertimbangkan resolusi piksel dari video, semakin kecil resolusi piksel semakin menghemat waktu komputasi dan semakin besar resolusi piksel semakin jelas objek yang terlihat. FPS (*frame per second*) dari kamera juga perlu diperhatikan karena semakin kecil FPS semakin menghemat waktu komputasi dan semakin besar FPS maka semakin baik model *background* yang didapatkan karena perubahan intensitas pencahayaan antar *frame* tidak terlalu signifikan. Kamera digital yang digunakan pada proses akuisisi video memiliki $\pm$30 FPS.

4.2.1.2 Data Input

Data proses adalah data yang digunakan dalam proses pengolahan data masukan. Setiap data masukan yang ada akan menghasilkan data proses sesuai dengan tahapan proses yang telah ditunjukkan pada Gambar 4.1. Selanjutnya, data proses tersebut digunakan sebagai data masukan untuk memproses tahapan selanjutnya. Tabel 4.1 menjelaskan data proses dalam sistem dan parameter-parameter setiap metode dijelaskan pada Tabel 4.2, 4.3, 4.4, 4.5, 4.6, dan 4.7.

Tabel 4.1: Tabel Data Input

No.	**Data *Input***	**Proses**	**Data *Output***
1	File video	Ekstraksi Video	*Image* RGB
2	*Image* RGB	*Preprocessing*	*Preprocessed image* RGB
3	*Preprocessed image* RGB	*Background Substraction*	*Image greyscale foreground*
4	*Image greyscale foreground*	*Smoothing*	*Image greyscale smoothing*
5	*Image greyscale smoothing*	*Shadow Removal*	*Image* biner
6	*Image* biner	Operasi Morfologi	*Image* biner morfologi
7	*Image* biner morfologi	Deteksi Objek	Kumpulan Objek
8	Kumpulan Objek	*Labeling* dan *Tracking*	Kumpulan Objek dengan ID dan *track record*
9	Objek dengan ID dan *track record*	Estimasi Kecepatan	Kecepatan rata-rata kendaraan

Tabel 4.2: Tabel Parameter *Region of Interest* (ROI)

No.	Variabel	Tipe Data	Keterangan
1	X1	Int	Koordinat x pada titik 1
2	Y1	Int	Koordinat y pada titik 1
3	X2	Int	Koordinat x pada titik 2
4	Y2	Int	Koordinat y pada titik 2

Tabel 4.3: Tabel Parameter *Preprocessing*

No.	Variabel	Tipe Data	Keterangan
1	Alpha	Double	Nilai pengaturan kontras *image* (*contrast adjustment*)
2	Beta	Double	Nilai pengaturan kecerahan *image* (*brightness adjustment*)

Tabel 4.4: Tabel Parameter *Median Filter*

No.	Variabel	Tipe Data	Keterangan
1	*Size*	Int	Jangkauan ketetanggaan piksel yang akan dicari nilai mediannya dan bernilai ganjil

Tabel 4.5: Tabel Parameter Morfologi

No.	Variabel	Tipe Data	Keterangan
1	*Size*	Int	Ukuran *kernel* yang akan digunakan untuk proses konvolusi morfologi tipe *close* dan bernilai ganjil

Tabel 4.6: Tabel Parameter Deteksi Objek

No.	Variabel	Tipe Data	Keterangan
1	Area	Double	Luas minimum area kontur objek yang akan dideteksi

Tabel 4.7: Tabel Parameter Estimasi dan Kalibrasi Kecepatan

No.	Variabel	Tipe Data	Keterangan
1	T_v	Double	Sudut kemiringan kamera terhadap garis tegak lurus dari permukaan jalan dalam satuan derajat
2	T_c	Double	Sudut jangkauan FOV (*Field of View*) kamera dalam satuan derajat
3	H	Double	Tinggi kamera dari permukaan jalan dalam satuan meter

4.2.1.3 Data Keluaran

Data keluaran dari sistem yang dikembangkan berupa tabel hasil estimasi kecepatan sesuai dengan ID yang sudah diberikan pada kendaraan. Selain itu sistem juga menghasilkan keluaran berupa *image* dari masing-masing proses, yaitu proses *preprocessing*, GMM, *smoothing*, *shadow removal*, operasi morfologi, dan deteksi objek.

4.2.2 Perancangan Algoritma Sistem

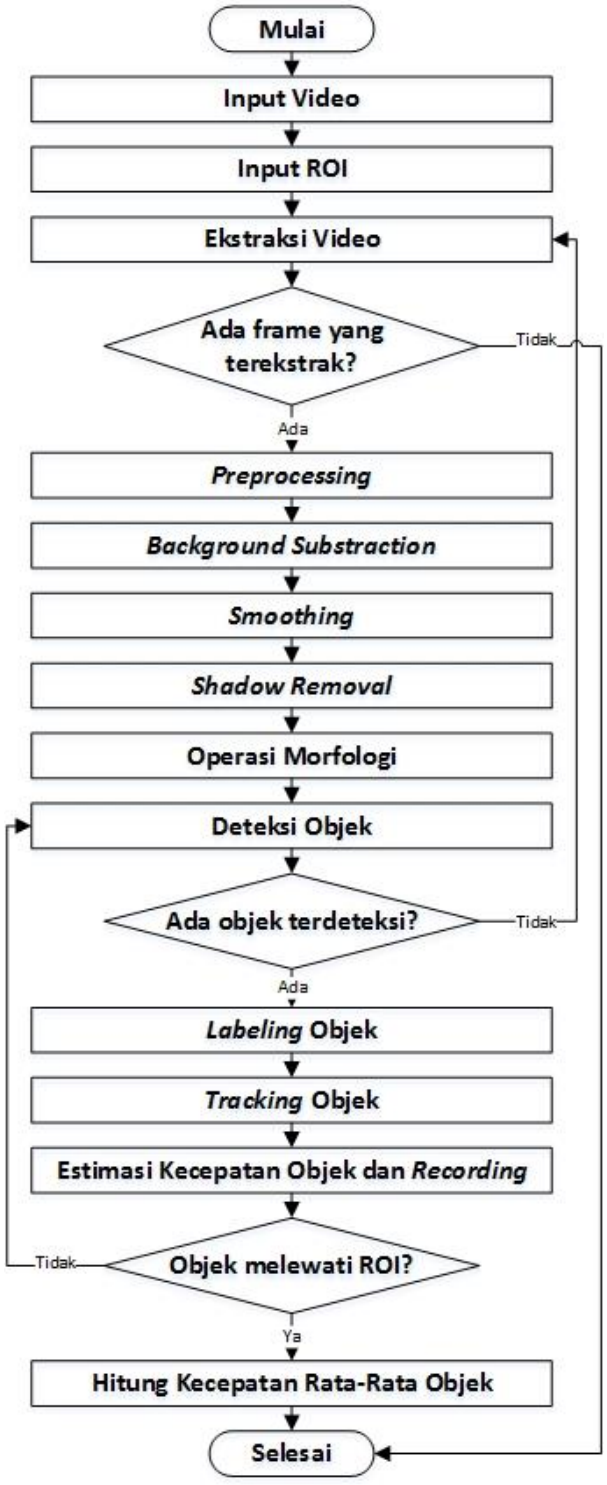
Diagram alir sistem yang dibuat dapat dilihat pada Gambar 4.3. Proses *background substraction*, *smoothing*, dan *shadow removal* pada penelitian ini direferensi dari penelitian yang telah dilakukan sebelumnya oleh Bayu Charisma (2016) [15].

4.2.2.1 Perancangan Daerah *Region of Interest* (ROI)

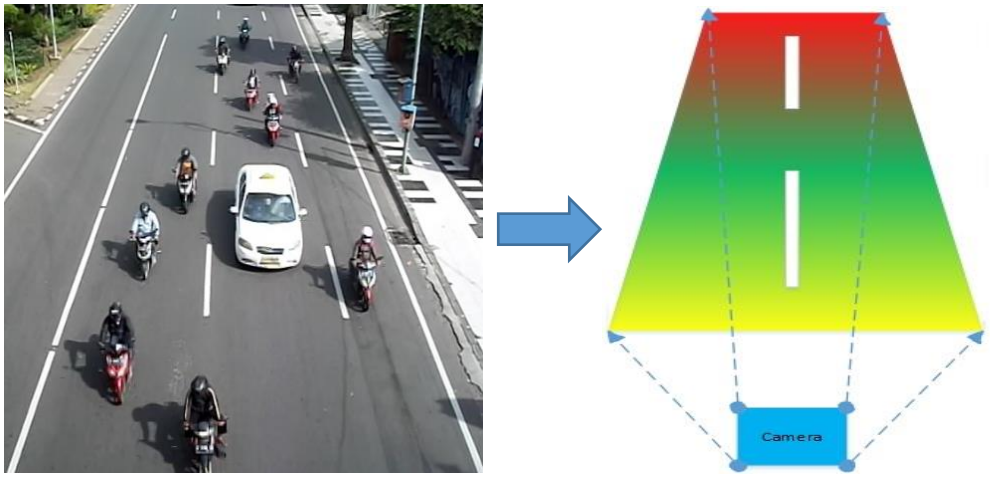
Permasalahan mengenai pemilihan ROI untuk daerah pencatatan kecepatan dikarenakan perspektif kamera yang dipakai pada penelitian ini adalah dari atas dengan posisi kamera statis, sehingga secara visual, *image* yang dihasilkan akan memiliki perspektif kecepatan seperti pada Gambar 4.4 yang mana hal ini juga berlaku ketika dilakukan estimasi kecepatan berdasarkan titik *centroid* antar *frame*.

Perspektif visual kamera menunjukkan bahwa saat objek bergerak mendekati kamera, maka akan terlihat berjalan lambat ketika objek tersebut masih jauh dari kamera, dan akan terlihat semakin cepat saat objek semakin dekat dengan kamera namun pada kenyataannya kecepatan kendaraan itu relatif konstan, hal ini juga

sama dengan apa yang dapat dilihat oleh mata manusia. Tentu hal ini akan sangat berpengaruh pada hasil estimasi kecepatan kendaraan yang melalui ROI sehingga perlu ditentukan ROI yang ideal untuk menentukan kecepatan rata-rata yang lebih tepat.



Gambar 4.3: Diagram Alir Proses pada Sistem



Gambar 4.4: Perspektif Visual Kamera terhadap Kecepatan Kendaraan. Kiri : Video Kendaraan Bergerak. Kanan : Perspektif Kecepatan Kendaraan

Pada Gambar 4.4, perspektif kecepatan kendaraan dibagi menjadi tiga bagian yaitu warna merah yang menandakan daerah profil kecepatan rendah, warna hijau menandakan daerah profil kecepatan sedang, dan warna kuning yang menandakan daerah profil kecepatan tinggi. Dengan daerah 3 profil kecepatan yang dihasilkan oleh perspektif visual kamera, terdapat 6 kemungkinan pemilihan daerah ROI yaitu:

1. Daerah ROI terletak hanya pada daerah merah, yaitu profil kecepatan rendah.
2. Daerah ROI terletak hanya pada daerah hijau yaitu profil kecepatan sedang.
3. Daerah ROI terletak hanya pada daerah kuning yaitu profil kecepatan tinggi.
4. Daerah ROI terletak antara daerah merah dan hijau, yaitu profil kecepatan rendah dan sedang.
5. Daerah ROI terletak antara daerah hijau dan kuning, profil kecepatan sedang dan tinggi.
6. Daerah ROI terletak antara daerah merah, hijau, dan kuning, yaitu profil kecepatan rendah, sedang, dan tinggi.

Pemilihan daerah ROI yang digunakan pada penelitian ini akan dijelaskan lebih detil pada tahap implementasi dan uji coba.

4.2.2.2 Perancangan Proses *Preprocessing*

Data masukan berupa video diekstrak menjadi *frame*. Setiap *frame* yang berhasil diekstrak akan dilakukan *preprocessing* terlebih dahulu dengan tujuan untuk meminimalkan kemunculan bayangan objek yang solid yang dapat berpengaruh pada bentuk objek sehingga mengganggu proses deteksi objek yang juga akan berpengaruh pada proses estimasi kecepatannya. *Preprocessing* dalam penelitian ini adalah berupa *contrast and brightness enhancement*. Misalkan piksel $f(i,j)$ adalah intensitas piksel asal pada koordinat (i,j) dan $g(i,j)$ adalah intensitas piksel hasil dengan $\alpha > 0$ adalah parameter *gain* (*contrast*) dan β adalah parameter *bias (brightness)* maka proses *contrast and brightness enhancement* didefinisikan oleh Persamaan 4.1.

$$g(i,j) = \alpha . f(i,j) + \beta \tag{4.1}$$

4.2.2.3 Perancangan Proses *Background Substraction*

Setiap *frame* yang terekstrak akan dicari model *background*-nya yang nantinya akan digunakan untuk men-*generate image foreground* berdasarkan model *background* tersebut. Proses pencarian model *background* dan *generate foreground* tersebut dilakukan dengan GMM (*Gaussian Mixture Model*). GMM juga tahan terhadap perubahan atau mampu beradaptasi karena setiap piksel pada setiap perubahan *frame* akan dievaluasi melalui proses *update* parameter *weight*, *standard deviation*, dan *mean*. Setiap piksel akan dikelompokkan berdasarkan distribusi yang dianggap paling efektif sebagai model *background*. Semakin besar nilai standar deviasi, maka semakin kuat penghalusan yang terjadi pada *image*. Setiap piksel memiliki model sendiri. Data yang diolah adalah intensitas pada piksel yang didapat dari *frame input*. Setiap ada *frame* yang terekstrak maka model pada setiap piksel akan di-*update*.

Pada sistem terdapat beberapa parameter yang telah didefinisikan sebelumnya yang dibutuhkan untuk proses *background subtraction* dengan GMM, diantara lain α (*learning rate*) dengan nilai 0.01, jumlah komponen gaussian yaitu 3, *T* (*threshold*) dengan nilai 0.4. Selain itu terdapat juga inisialisasi awal dari beberapa parameter GMM antara lain : ω_k yaitu bobot dari setiap piksel pada gaussian ke-*k* dengan nilai $\frac{1}{3}$ dimana 3 merupakan jumlah dari distribusi gaussian; μ_k yaitu *mean*

dari setiap piksel pada gaussian ke-*k* dimana masing-masing piksel dari setiap Gaussian mempunyai nilai random antara 0 sampai 255; σ_k yaitu standar deviasi dari setiap piksel pada gaussian ke-*k* dengan nilai 6. Contoh inisialisasi awal parameter GMM diberikan pada Gambar 4.5.

Berikutnya akan diberikan contoh bagaimana proses GMM bekerja. Misalkan *frame* yang terekstrak adalah seperti yang disajikan pada Gambar 4.6. Lalu dengan menggunakan pertidaksamaan 2.4 diperoleh pertidaksamaan 4.2

$$|X_{i,j,t} - \mu_{i,j,k}| < 2.5 * \sigma_{i,j,k} \tag{4.2}$$

Inisialisasi Awal Bobot (ω)

Gaussian 1			Gaussian 2			Gaussian 3		
1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3
1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3
1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3	1/3

Inisialisasi Awal Mean (μ)

Gaussian 1			Gaussian 2			Gaussian 3		
156	2	23	99	12	12	146	23	33
198	15	41	32	200	44	212	152	142
202	211	112	142	42	222	51	75	163

Inisialisasi Awal Standar Deviasi (σ)

Gaussian 1			Gaussian 2			Gaussian 3		
6	6	6	6	6	6	6	6	6
6	6	6	6	6	6	6	6	6
6	6	6	6	6	6	6	6	6

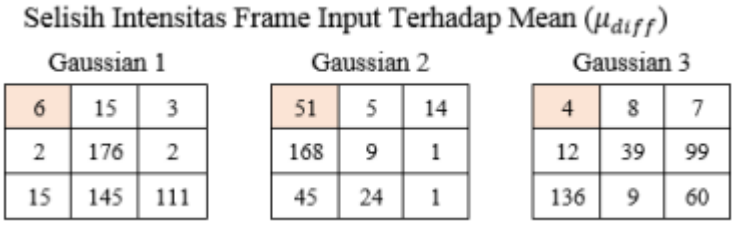
Gambar 4.5: Contoh Inisialisasi Parameter GMM [15].

dimana nilai intensitas *X* pada piksel (*i*, *j*) dari *frame input* yang memenuhi pertidaksamaan tersebut berarti masuk dalam distribusi. Dari inisialisasi awal σ sebesar 6 maka diperoleh nilai dari ruas kanan pada pertidaksamaan 4.2 sebesar 15. Sedangkan ruas kiri merupakan selisih antara intensitas *frame input* dan intensitas mean pada *frame* sebelumnya yang masing-masing diberikan pada Gambar 4.6 dan Gambar 4.5 dimana hasil dari ruas kiri disajikan pada Gambar 4.7.

Frame Input

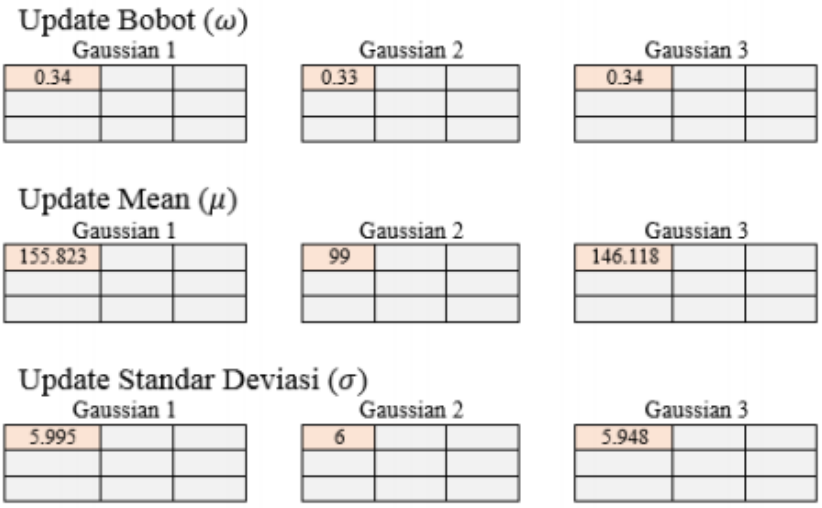
150	17	26
200	191	43
187	66	223

Gambar 4.6: Contoh Matriks Intensitas *Frame* yang Terekstrak [15].

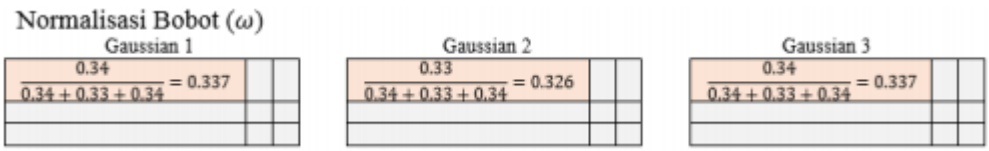


Gambar 4.7: Matriks Selisih Intensitas *Frame Input* dan *Mean* [15].

Untuk proses selanjutnya akan diberikan contoh hanya pada piksel (1,1). Langkah berikutnya adalah *update* bobot dari masing-masing gaussian dengan menggunakan Persamaan 2.5. Dari Gambar 4.7 dapat dilihat nilai intensitas $X_{1,1}$ yang cocok terhadap distribusinya adalah yang kurang dari 15 yaitu gaussian 1 dan gaussian 3, sehingga $\mu_{1,1,k}$ dan $\sigma_{1,1,k}$ dengan $k = 1$ dan $k = 3$ akan di-*update* dengan menggunakan Persamaan 2.7 dan Persamaan 2.8. Hasil *update* parameter GMM tersebut disajikan pada Gambar 4.8. Bobot yang di-*update* kemudian akan dinormalisasi agar jumlah dari bobot tiap komponen gaussian tepat nilainya 1. Hasil normalisasi bobot disajikan pada Gambar 4.9.



Gambar 4.8: Hasil *Update* Parameter GMM [15].



Gambar 4.9: Hasil Normalisasi Bobot [15].

Untuk memilih model mana yang mencerminkan *background*, terlebih dahulu bobot akan diurutkan berdasarkan $\frac{\omega}{\sigma^2}$. Menurut Gambar 4.10 urutan bobot dari besar ke kecil secara berturut-turut adalah gaussian 3, gaussian 1, gaussian 2. Selanjutnya akan dicari beberapa distribusi pertama yang sudah terurut menggunakan Persamaan 2.9 dimana *T* (*threshold*) adalah 0.4, sehingga pada contoh distribusi yang dipilih adalah gaussian 3 dan gaussian 1. Kemudian semua gaussian yang terpilih dicocokkan dengan intensitas dari *frame input* dengan menggunakan pertidaksamaan 4.1, jika ada distribusi gaussian yang cocok maka intensitas merupakan *background*, dan jika tidak maka intensitas merupakan *foreground*.

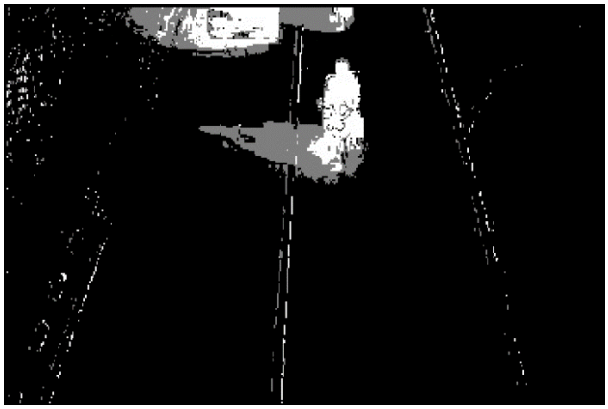
Perhitungan ω/σ^2

Gaussian 1	Gaussian 2	Gaussian 3
$\frac{0.337}{5.995^2} = 0.00938$	$\frac{0.326}{6^2} = 0.00906$	$\frac{0.337}{5.948^2} = 0.00953$

Gambar 4.10: Hasil Perhitungan $\frac{\omega}{\sigma^2}$ [15].

4.2.2.4 Perancangan Proses *Smoothing*

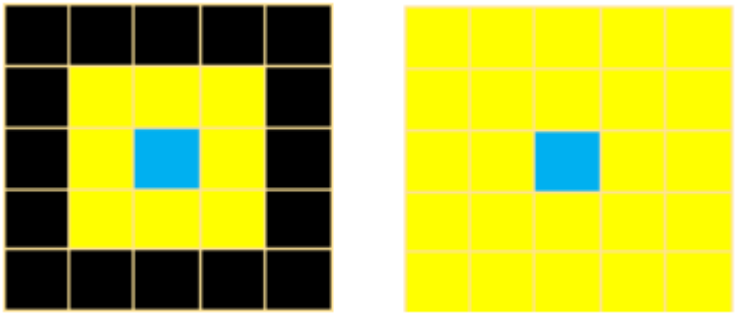
Smoothing adalah proses untuk memperhalus *image* yang dilakukan agar *noise* pada *image foreground* hasil dari proses *background subtraction* berkurang. Gambar 4.11 menyajikan *image output* dari proses *background subtraction* dimana pada *image* tersebut terdapat banyak *noise*.



Gambar 4.11: Contoh Hasil Proses dari *Background Substraction*

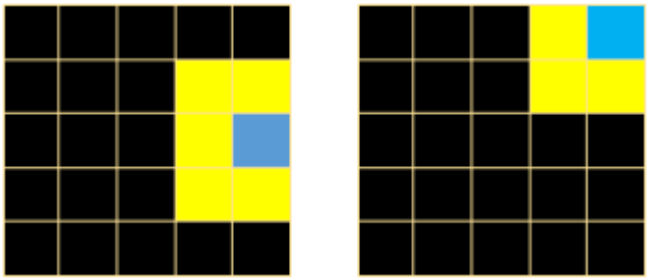
Dalam penelitian ini metode *smoothing* yang digunakan adalah median *filter*. Median *filter* membutuhkan parameter masukan berupa *size*, yaitu parameter untuk menentukan seberapa jauh jangkauan tetangga yang akan diproses untuk menentukan nilai mediannya. Gambar 4.12 menjelaskan bagaimana *size* pada

proses *smoothing*. Warna biru menjelaskan piksel yang sedang diproses, sementara warna kuning menjelaskan area dari *size*, dimana *size* selalu bernilai ganjil.



Gambar 4.12: Visualisasi Median Filter dengan Ketetanggaan. Kiri : *size* = 3. Kanan : *size* = 5.

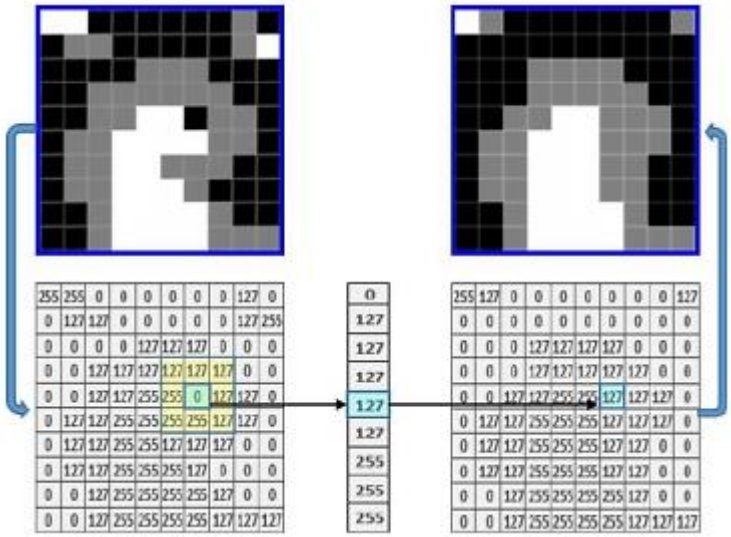
Proses *smoothing* berlaku pada semua piksel dari *image*. Tidak terkecuali piksel pada *border* atau tepi dari *image*. Namun *scale* atau jangkauan pada piksel *border* berbeda dengan *scale* pada piksel *non-border*. Jangkauan untuk piksel border disajikan pada Gambar 4.13.



Gambar 4.13: Visualisasi Median Filter dengan Ketetanggaan pada Piksel Border. Kiri : piksel border kanan akan mencari nilai median dari 6 piksel tetangga dengan *size* = 3. Kanan : piksel border kanan akan mencari nilai median dari 4 piksel tetangga dengan *size* = 3.

Semua piksel pada*image* akan dicari nilai median dari piksel tersebut dan semua piksel tetanggannya dimana piksel tetangga diperoleh berdasarkan parameter *scale*. Agar tidak mempengaruhi *smoothing* pada piksel berikutnya, maka nilai median dari hasil *smoothing* tidak menggantikan nilai lama, melainkan akan diletakkan pada *image* yang baru. *Output* dari proses ini berupa *image grayscale* dengan 3 tingkat keabuan yang berisi nilai median dari setiap piksel. Hitam bernilai 0, abuabu bernilai 127, dan putih bernilai 255. Masing-masing warna

atau nilai tersebut secara berturut-turut merepresentasikan *background*, Bayangan dari *foreground*, dan *foreground*. Contoh dari proses *smoothing* disajikan pada Gambar 4.14.



Gambar 4.14: Proses *Smoothing* dengan Median Filter

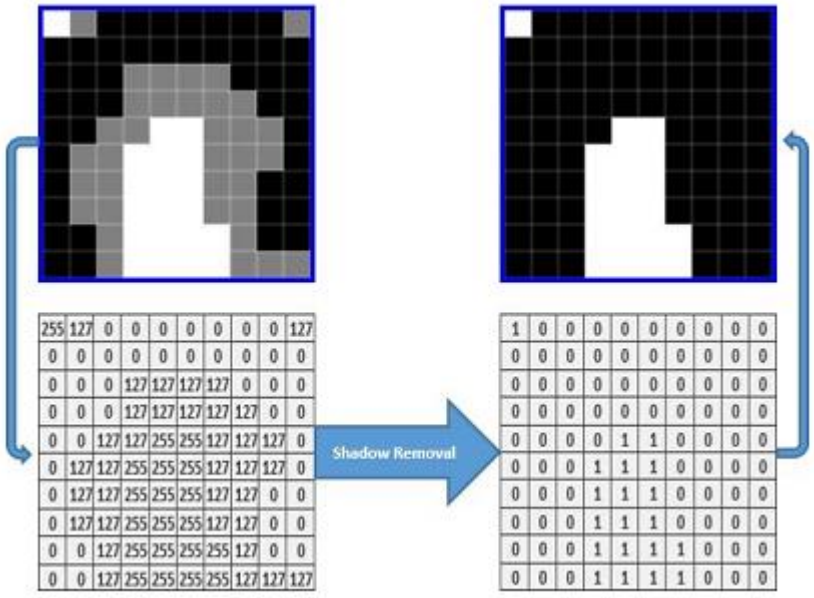
4.2.2.5 Perancangan Proses *Shadow Removal*

Proses *Background Subtraction* yang digunakan dalam sistem ini juga mendeteksi adanya bayangan. Namun untuk memaksimalkan pendeteksian bayangan, bayangan yang terdeteksi tidak langsung dihapus ketika terdeteksi, melainkan perlu proses *smoothing* agar bayangan yang terdeteksi lebih akurat. Maka dari itu proses *shadow removal* dalam sistem ini dilakukan setelah proses *smoothing*. Data masukan pada proses ini berupa *image grayscale* hasil dari proses *smoothing*. Seperti yang dijelaskan sebelumnya *image* hasil dari proses *smoothing* mempunyai 3 tingkat keabuan yaitu 0, 127, dan 255. *Shadow removal* adalah proses dimana nilai-nilai tersebut dipetakan ke angka biner yaitu 0 dan 1 seperti yang dijelaskan pada Persamaan 4.3.

$$I_{biner}(x,y) = \begin{cases} 0, & I_{gray}(x,y)=0 \; atau \; I_{gray}(x,y)=127 \\ 1, & I_{gray}(x,y)=255 \end{cases} \tag{4.3}$$

Proses *shadow removal* berlaku pada seluruh piksel dari *image*. Proses *shadow removal* akan mempermudah proses dari deteksi obyek bergerak karena hasil dari proses ini adalah *image* biner yang hanya mempunyai dua nilai, yaitu 0 atau hitam yang merepresentasikan *background* dan 1 atau putih yang merepresentasikan

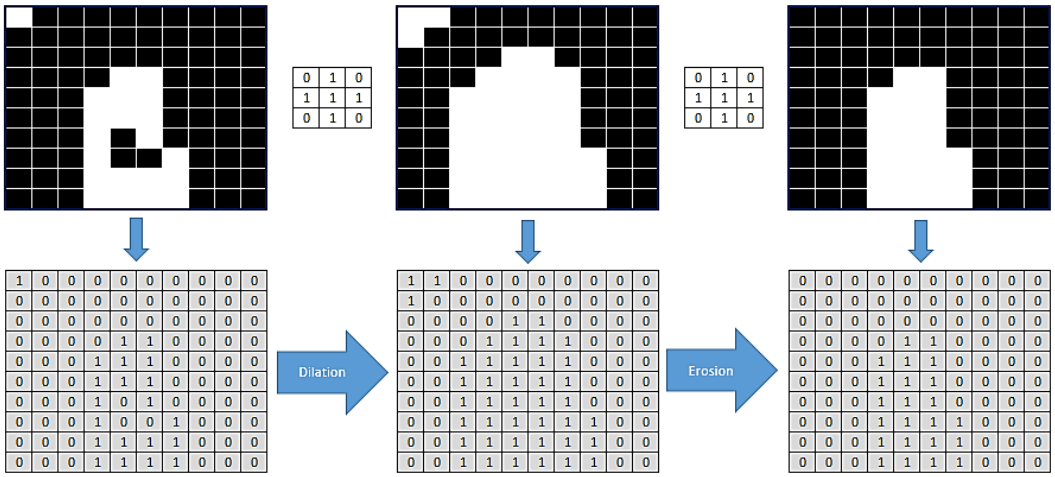
foreground atau obyek bergerak. contoh proses dari *shadow removal* dijelaskan pada Gambar 4.15.



Gambar 4.15: Proses *Shadow Removal*

4.2.2.6 Perancangan Proses Morfologi Citra

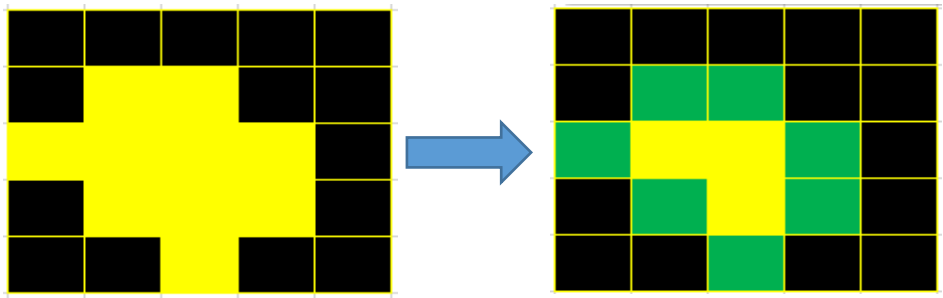
Image biner hasil *shadow removal* terkadang menghasilkan objek yang memiliki lubang (*holes*) di tengah-tengah objek. Sehingga perlu dilakukan proses *filling holes* dengan menggunakan operasi *closing* dengan tujuan untuk hasil deteksi objek yang lebih baik. Dengan melakukan operasi *closing* pada *image* biner hasil *shadow removal*, maka lubang-lubang kecil pada objek akan tertutup. Hasil proses *closing* pada piksel dapat dilihat pada Gambar 4.16.



Gambar 4.16: Proses Morfologi *Closing.*

4.2.2.7 Perancangan Proses Deteksi Objek

Deteksi objek dilakukan setelah *image* mengalami proses morfologi. Proses deteksi objek pada penelitian ini menggunakan konsep *connected component* untuk mendeteksi kontur pada *image biner*. Kontur yang dideteksi pada objek adalah berdasarkan pada *outer border* atau batas paling luar dengan metode *border following* [14]. Metode *border following* diilustrasikan pada Gambar 4.17 dan algoritma *border following* diberikan pada Gambar 4.18. Contoh *border following* pada *image* biner dapat dilihat pada Gambar 4.19.



Gambar 4.17: Ilustrasi *Border Following* dalam Deteksi Kontur Objek

1. Proses *scanning* berjalan dengan *raster scan*, yaitu dari atas ke bawah (*row*), kiri ke kanan (*column*).
2. Cari piksel $f(i,j)$ sebagai kandidat piksel *outer border* yang memenuhi kondisi berikut:

	j-1	j
i	0	1

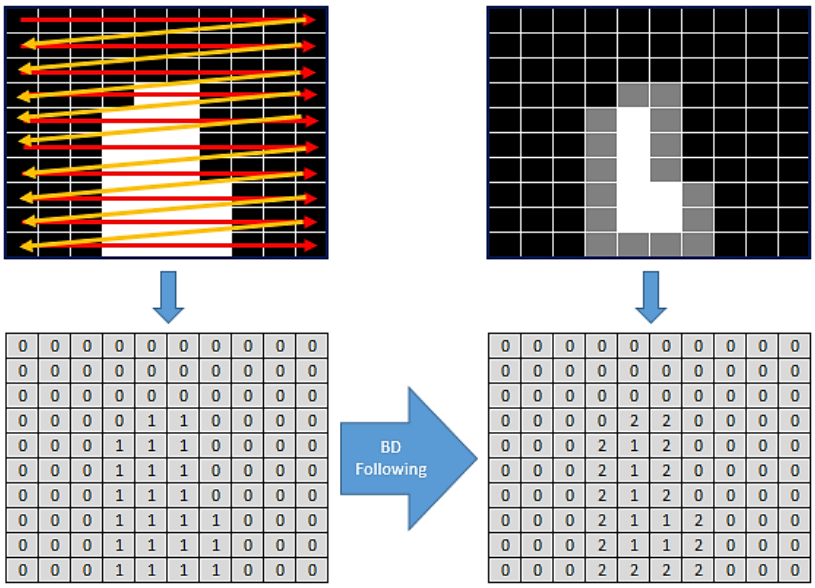
   dan jika piksel merupakan *outer most border*, yaitu memenuhi kondisi berikut:
   a. Piksel $f(i,1), f(i,2), \dots, f(i,j-1)$ adalah piksel 0, atau
   b. piksel $f(i,h)$ sejajar dengan titik *border*, dan piksel $f(i,h+1)$ adalah *background*.

   Lakukan *labelling* sebagai tanda *border*.
4. Jika batas akhir *border* telah ditemukan yaitu:

	j-1	j
i	1	0

   lakukan *labelling* sebagai tanda *border*.
5. Jalankan *raster scanning* hingga seluruh piksel terlewati.

Gambar 4.18: Algoritma *Border Following* dalam Deteksi Kontur Objek.



Gambar 4.19: Proses *Border Following*. Tanda *border* diberikan Label 2.

Setelah kontur objek terdeteksi, langkah selanjutnya adalah membentuk *bounding box* untuk analisis *blob* objek dengan koordinat (x, y) sebagai $B(x, y)$ dengan lebar (*width)* dan tinggi (*height*), kemudian dari *bounding box* ini ditentukan koordinat titik *centroid* sebagai $C(x, y)$ dengan Persamaan 4.4. Representasi *bounding box* dapat dilihat pada Gambar 4.20.

$$C(x, y) = (C_x, C_y), \tag{4.4}$$

dengan

$$C_x = B_x + \frac{B_{width}}{2},$$

$$C_y = B_y + \frac{B_{height}}{2}$$

,dimana:

C_x adalah koordinat x pada *centroid*,

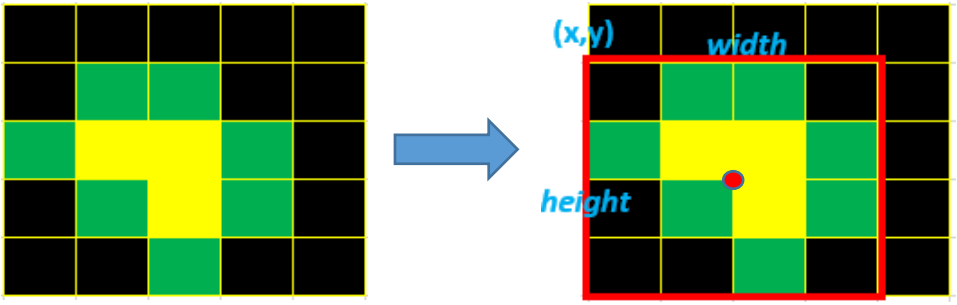
C_y adalah koordinat y pada *centroid*,

B_x adalah koordinat x pada *bounding box*,

B_y adalah koordinat y pada *bounding box*,

B_{width} adalah lebar *bounding box*, dan

B_{height} adalah tinggi *bounding box*.



Gambar 4.20: Ilustrasi *Bounding Box* dan *Centroid* pada Objek.

4.2.2.8 Perancangan Proses *Labelling* dan *Tracking* Objek

Kendaraan akan diberi label berupa ID berdasarkan urutan kemunculan kendaraan tersebut pada ROI. Ketika objek berada dalam ROI, kecepatan kendaraan antar *frame* akan selalu dicatat dan ketika objek meninggalkan ROI akan dihitung kecepatan rata-rata lalu disimpan ke dalam memori CPU. Data yang tersimpan tersebut ditampilkan dalam data keluaran berupa tabel hasil dari rata-rata estimasi kecepatan tiap kendaraan. Untuk memastikan objek tidak terhitung dan tersimpan lebih dari satu kali, maka proses *tracking object* antar *frame* perlu dilakukan. *Tracking* adalah suatu proses yang bertujuan untuk mencari objek yang sama pada *frame* sebelumnya. Dua objek dikatakan sama jika memenuhi pertidaksamaan 4.5 dan pertidaksamaan 4.6.

$$L_{objek1} \leq \frac{L_{objek2}+R_{objek2}}{2} \leq R_{objek1} \tag{4.5}$$

$$T_{objek2} \leq T_{objek1} \leq B_{objek2} \tag{4.6}$$

dimana $objek1$ adalah objek yang sedang diproses, $objek2$ adalah sebuah objek pada *frame* sebelumnya, *L* adalah batas kiri objek, *R* adalah batas kanan objek, *T* adalah batas atas objek, dan *B* adalah batas bawah objek. Jika kedua objek teridentifikasi sama maka ID pada objek sebelumnya akan berpindah ke objek yang sedang diproses dan melakukan penghitungan kecepatan berdasarkan jarak titik *centroid* objek antar 2 frame tersebut yang akan dijelaskan lebih lanjut pada sub bab selanjutnya. Proses tersebut akan terus berlaku hingga objek meninggalkan ROI.

4.2.2.9 Perancangan Proses Estimasi Kecepatan

Sebelum proses estimasi kecepatan dilakukan pada saat objek kendaraan bergerak, terlebih dahulu dibutuhkan parameter geometris untuk kalibrasi kecepatan kendaraan dari satuan piksel/detik menjadi km/jam. Representasi parameter geometris dapat dilihat pada Gambar 2.8 dan penulis merancang skema parameter geometris seperti pada Gambar 4.2 sebagai bentuk sederhananya. Dari Gambar 2.5 dan Gambar 4.2, diketahui bahwa $\theta_3 = T_c$ yaitu sudut jangkauan FOV (*Field Of View*) dari kamera yang didefinisikan pada Persamaan 4.7 dan θ_1 adalah sudut kemiringan kamera dan termasuk sudut jangkauan FOV, sehingga berdasarkan Gambar 4.2, θ_1 diperoleh dengan Persamaan 4.8.

$$\theta_3 = T_c = 2 \arctan\left(\frac{v}{2f}\right) \tag{4.7}$$

$$\theta_1 = T_v + \frac{T_c}{2} \tag{4.8}$$

Dengan f adalah *focal length* kamera, v adalah *vertical dimension of 35mm image format*, dan T_v adalah sudut kemiringan kamera saat akuisisi video. Sedangkan untuk jarak D dari Persamaan 2.16 didapat

$$D = H\ tan(\theta_1) \tag{4.9}$$

Dengan H adalah tinggi kamera dari permukaan jalan. Sehingga, dari Persamaan 4.7, 4.8, dan 4.9, dapat menghitung *perpendicular view P* dengan Persamaan 4.10.

$$P = 2 \tan\left(\frac{\theta_3}{2}\right) \sqrt{H^2 + D^2} \tag{4.10}$$

Sehingga parameter geometris yang harus dimasukkan oleh *user* adalah T_c, T_v, dan H dan sistem akan menghitung nilai D dan P secara otomatis. Kemudian kecepatan kendaraan dihitung dengan menggunakan *euclidean distance* yang sudah dijelaskan pada subbab 2.8 dimana kecepatan akan dihitung melalui jarak antar titik *centroid* kemudian dibagi dengan waktu antar 2 *frame* dan hasilnya dikalikan dengan faktor kalibrasi. Untuk mengetahui jarak yang ditempuh dalam piksel, dapat dimisalkan koordinat suatu objek seperti berikut:

$$c_t(a, b) \text{ dan } c_{t+1}(c, d)$$

dimana c_t dan c_{t+1} adalah posisi titik *centroid* pada *frame* t dan $t+1$ untuk satu objek yaitu dengan koordinat (a, b) dan koordinat (c, d).

Perbedaan jarak jauh untuk kendaraan dengan *Euclidean Distance* [4] yaitu:

$$d = \sqrt{(a-c)^2 + (b-d)^2} \tag{4.11}$$

Sedangkan, untuk kecepatan (*speed*) didapat

$$V = k\frac{d}{t} \tag{4.12}$$

Dengan k adalah koefisien kalibrasi dan t adalah waktu antar 2 *frame* yang berurutan, dimana

$$k = \frac{perpendicular\_view}{image\_height} \tag{4.13}$$

$$t = \frac{1}{fps} \tag{4.14}$$

Kecepatan V yang didapat dari Persamaan 4.12 masih dalam satuan meter per detik atau m/s. Untuk konversi dari satuan meter per detik menjadi kilometer per jam atau km/h maka hasil V dikalikan 3,6 yang didapat dari Persamaan 4.17.

$$1\ m = \frac{1}{1000}km \tag{4.15}$$

$$1\ detik = \frac{1}{3600}jam \tag{4.16}$$

Dari Persamaan 4.17 dan 4.18, maka didapat

$$1\ \frac{m}{s} = \frac{\left(\frac{1}{1000}\right)}{\left(\frac{1}{3600}\right)}\frac{km}{jam} = 3{,}6\,\frac{km}{jam} \tag{4.17}$$

Sehingga rumus kecepatan V dalam satuan kilometer per jam atau km/h dari Persamaan 4.12 dan 4.17 adalah

$$V = 3{,}6\ k\frac{d}{t} \tag{4.18}$$

Persamaan 4.18 adalah rumus kecepatan yang digunakan pada sistem ini sehingga kecepatan yang ditampilkan sudah dalam satuan standar yaitu kilometer per jam.

Karena sistem yang dibangun harus mampu mendeteksi kecepatan kendaraan *multiobject* dan selalu mencatat kecepatan kendaraan pada saat di dalam ROI, maka dibutuhkan suatu *list* untuk meyimpan objek-objek kendaraan yang terdeteksi dan

list untuk menyimpan kecepatan-kecepatan pada saat kendaraan berada di ROI, yaitu *listObj* dan *listSpeed*.

Pada saat pertama kali *listObj* kosong, maka kendaraan yang dideteksi merupakan objek baru yang langsung diberikan label ID, dimasukkan ke dalam *listObj* namun belum dilakukan proses estimasi kecepatannya. Pada *frame* selanjutnya, jika kendaraan yang terdeteksi sama dengan kendaraan pada *frame* sebelumnya yang ada pada *listObj* maka akan dilakukan proses *updating* objek, yaitu *update* ID pada objek kendaraan yang baru, *update* lokasi *bounding box*, menghitung jarak antar titik *centroid* pada *frame* sekarang dan *frame* sebelumnya, dan estimasi kecepatannya yang kemudian disimpan di suatu *speed list* pada objek kendaraan tersebut. Jika kendaraan tersebut tidak sama atau tidak cocok dengan objek-objek kendaraan yang ada pada *listObj*, dapat dikatakan bahwa kendaraan yang terdeteksi adalah kendaraan baru, diberikan ID dan dimasukkan *listObj*. Proses ini dilakukan sebanyak objek yang terdeteksi di dalam ROI. Ketika kendaraan keluar dari ROI, yang dilakukan adalah menghitung rata-rata kecepatan yang telah diestimasi dan disimpan dalam *speed list* dan hasilnya ditampilkan pada tabel hasil estimasi kecepatan sesuai dengan ID kendaraan tersebut. Untuk menghitung kecepatan rata-rata kendaraan yang telah keluar dari ROI, digunakan Persamaan 4.19. Diagram alir untuk proses estimasi kecepatan kendaraan *multi* objek dapat dilihat pada Gambar 4.21, sedangkan bentuk *pseudo code* dapat dilihat pada Gambar 4.22.

$$Va_{ID} = \frac{\sum_{i=0}^{n} listSpeed_{ID}(i)}{n}, \text{ dengan } ID = 0,1,2,...,m \tag{4.19}$$

dimana:

Va_{ID} = Kecepatan rata-rata kendaraan dengan *id*-nya adalah ID

$listSpeed_{ID}$ = *List* kecepatan yang disimpan saat objek berada di ROI

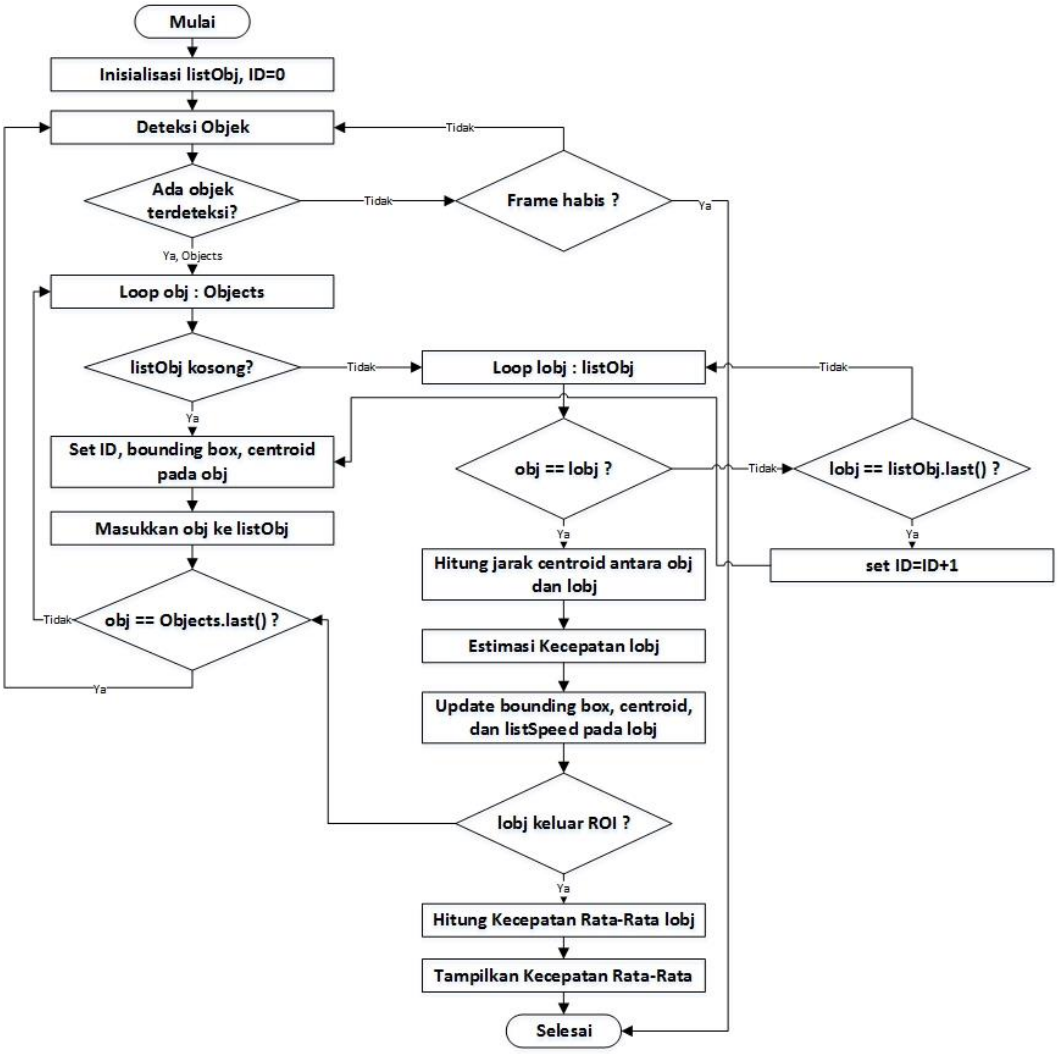
m = Jumlah kendaraan yang terdeteksi

n = Jumlah data kecepatan objek yang disimpan selama di ROI

Hasil dari estimasi kecepatan yang diperoleh dari sistem akan divalidasi untuk mendapatkan tingkat akurasi. Validasi dilakukan dengan membandingkan hasil estimasi kecepatan yang dihasilkan oleh sistem dengan kecepatan kendaraan sesungguhnya. Misal kecepatan kendaraan sesungguhnya adalah V_r dan kecepatan hasil estimasi sistem adalah V_e, dengan menggunakan konsep kesalahan relatif (*relative error*) maka tingkat akurasi didefinisikan oleh Persamaan 4.21.

$$Relative\ Error\ (\%) = \left(\frac{|V_e - V_r|}{V_r}\right) x\ 100 \tag{4.20}$$

$$Tingkat\ Akurasi\ (\%) = 100 - Relative\ Error \tag{4.21}$$



Gambar 4.21: Diagram Alir Estimasi Kecepatan Kendaraan secara *Multi* Objek

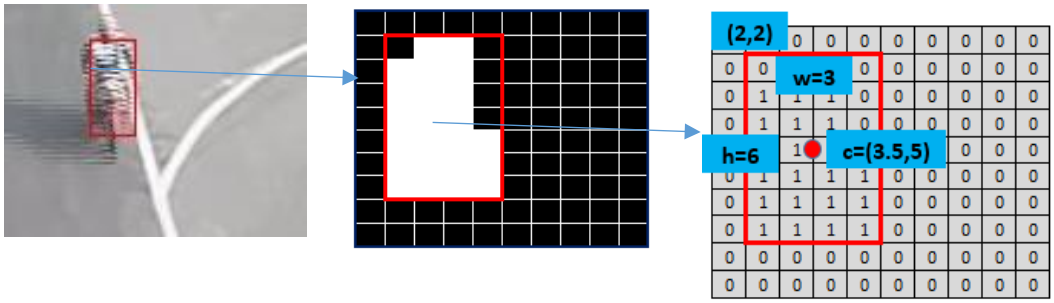
Algoritma Estimasi Kecepatan Kendaraan *Multi* Objek

Input : *Objects*
Output : *Average speeds* in *result repository* for each *object* in *Objects*
Process :
initialize *listObj* and *ID*
listObj ← {}
ID ← 0
for each *obj* **in** *Objects* **do**
  if *listObj* is empty, **then**
    let *boundingBox* is rectangle coordinate of object and *centroid* calculated from it.
    listSpeed ← {}
    obj ← *ID*
    obj ← *boundingBox*
    obj ← *centroid*
    obj ← *listSpeed*
    listObj ← *obj*
  if *listObj* is not empty, **then**
    found ← false
    for each *lobj* **in** *listObj* **do**
      if *obj* equals *lobj*, **then**
        let c_{obj}, c_{lobj} is centroid of *obj* and *lobj* respectively, v is velocity, d is distance, k is calibration factor, and t is time between 2 frames.
        d ← euclidean_distance(c_{obj}, c_{lobj})
        $v \leftarrow 3.6\, k \frac{d}{t}$
        listSpeed ← v
        lobj ← *boundingBox*
        lobj ← *centroid*
        lobj ← *listSpeed*
        found ← true
        if *lobj* is out of ROI, **then**
          let va is average velocity and res is result repository
          $va \leftarrow \frac{\sum_{v \in listSpeed} v}{size(listSpeed)}$
          $res \leftarrow va$
        end if
        break loop
      end if
    end for
    if *found* is false, **then**
      ID ← *ID* + 1
      listSpeed ← {}
      obj ← *ID*
      obj ← *boundingBox*
      obj ← *centroid*
      obj ← *listSpeed*
      listObj ← *obj*

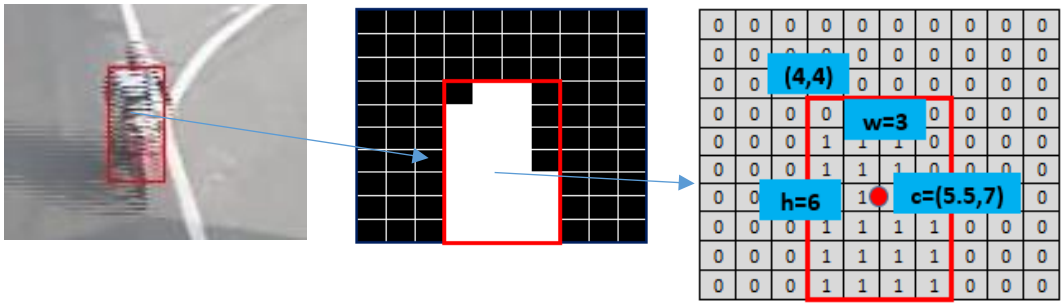
```
    end if
  end if
end for
```

Gambar 4.22: *Pseudo Code* Estimasi Kecepatan Kendaraan secara *Multi* Objek

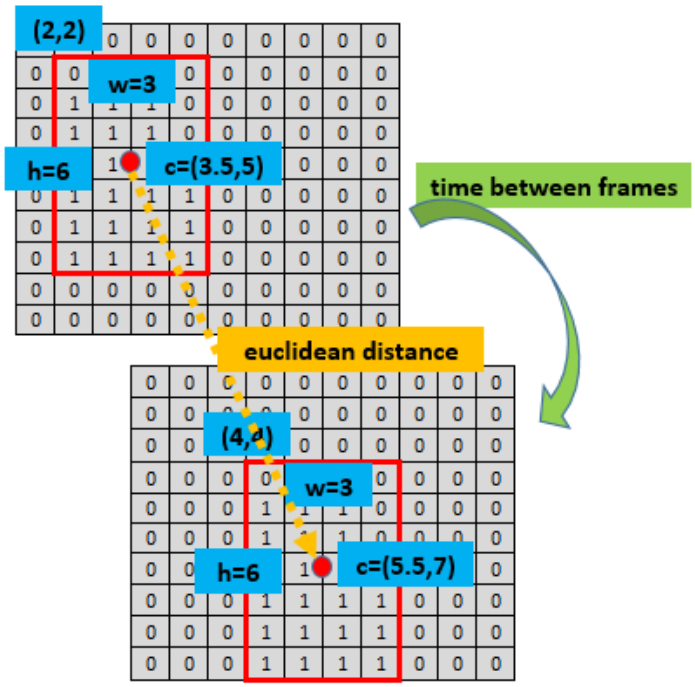
Ilustrasi estimasi kecepatan dengan menggunakan *euclidean distance* diberikan pada penjelasan berikut. Misalkan terdapat citra hasil proses deteksi objek pada *frame* t dan $t + 1$ diberikan pada Gambar 4.23 dan Gambar 4.24.



Gambar 4.23: Ilustrasi Estimasi Kecepatan : *Image* pada *frame* t dengan titik *centroid*.



Gambar 4.24: Ilustrasi Estimasi Kecepatan : *Image* pada *frame* $t + 1$ dengan titik *centroid*.



Gambar 4.25: Ilustrasi Estimasi Kecepatan : Kecepatan objek dari *frame* t dan *frame* $t+1$ dari jarak *Euclidean* dan waktu antar *frame*.

Gambar 4.25 merupakan ilustrasi penghitungan kecepatan objek dari jarak dan waktu antar *frame* oleh Gambar 4.23 dan 4.24. Pada Gambar 4.25 didapatkan dua titik *centroid* pada satu objek yang akan diestimasi kecepatannya, yaitu $c_t = (3.5,5)$ dan $c_{t+1} = (5.5,7)$. Kemudian akan dihitung jaraknya dengan *euclidean distance* menggunakan Persamaan 4.11 sehingga didapatkan $d = \sqrt{(5.5-3.5)^2 + (7-5)^2} \approx 2{,}83$ piksel. Dengan diketahui ukuran *image* adalah $320x240$, $fps = 30$, $T_v = 60$, $T_c = 41{,}10$, dan $H = 7{,}6$, maka didapatkan

$$t = \frac{1}{fps} = \frac{1}{30} = 0{,}033\ detik,$$

$$D = H \tan\left(T_v + \frac{T_c}{2}\right) = 7{,}6 \tan(80{,}55) = 45{,}661\ meter,$$

$$P = 2 \times \sqrt{H^2 + D^2} \times \tan\left(\frac{T_c}{2}\right) = 2 \times \sqrt{7{,}6^2 + 45{,}661^2} \times \tan\left(\frac{41{,}10}{2}\right) = 34{,}705\ meter,$$ dan

$$k = \frac{P}{image\_height} = \frac{34{,}705}{240} = 0{,}145$$

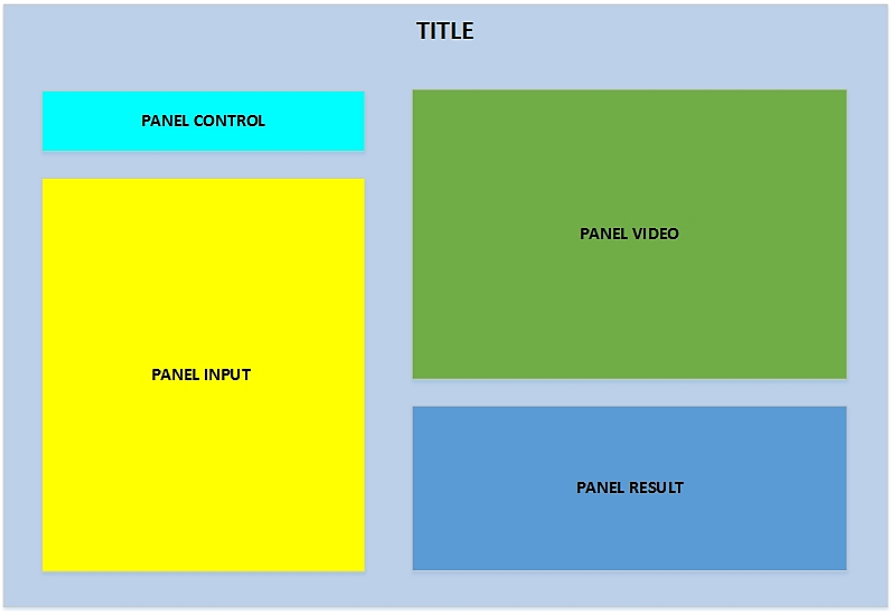
sehingga untuk mendapatkan kecepatan objek tersebut digunakan Persamaan 4.18, yaitu

$$V = 3{,}6\, k \frac{d}{t} = (3{,}6)(0{,}145)\left(\frac{2{,}83}{0{,}033}\right) \approx 44{,}765\ km/jam.$$

Hal ini juga berlaku untuk objek-objek yang lain yang terdeteksi di dalam ROI.

4.2.3 Perancangan Antar Muka Sistem

Untuk mempermudah eksperimen, dibutuhkan antar muka dari sistem yang telah dirancang. Desain antar muka dapat dilihat pada Gambar 4.26.



Gambar 4.26: Desain Antar Muka Sistem

4.3 Implementasi Sistem

4.3.1 Implementasi Akuisisi Video

Akuisisi video dilakukan di atas jembatan penyeberangan orang (JPO), dengan posisi kamera statis. Untuk mempermudah dalam menghitung sudut kemiringan kamera, penulis memanfaatkan busur derajat yang dipasang pada kamera, garis angka 90 pada busur derajat diletakkan sejajar dengan titik tengah dari optik kamera dan menggunakan tali serta beban kecil sebagai alat bantu untuk penunjuk besarnya sudut kemiringan kamera. Gambar 4.27 menjelaskan bagaimana posisi kamera saat akuisisi video.

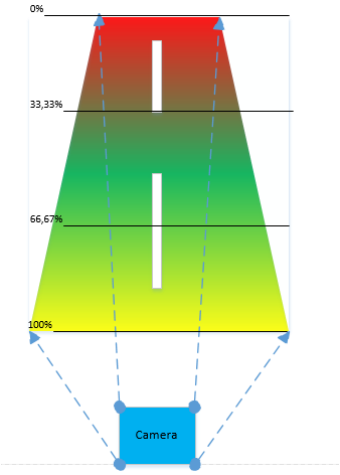
Gambar 4.27: Posisi Kamera saat Akuisisi Video. Kiri: Tampak Belakang. Kanan: Tampak Samping

4.3.2 Implementasi *Input* Video

Pada *input* video, *user* diharuskan memasukkan atau memilih berkas video yang akan diproses melalui panel *control* yang ada pada antar muka sistem. Setelah *user* memasukkan video, sistem akan menampilkan *image* awal dari video tersebut yang nantinya akan digunakan untuk kebutuhan pemilihan daerah ROI.

4.3.3 Implementasi Pemilihan Daerah ROI

ROI (*Region of Interest*) dalam sistem berfungsi untuk mempersempit ruang proses pengolahan pada video sehingga data yang diolah lebih fokus, mempercepat waktu komputasi, dan menghindari objek yang tidak diinginkan seperti orang yang berjalan atau pohon yang bergerak. Pada Gambar 4.28 ditunjukkan daerah jangkauan kamera dibagi 3 bagian yang ditandai dengan persentase. Untuk memilih daerah ROI yang tepat, perlu dilakukan analisis pemilihan daerah ROI untuk seluruh kemungkinan dari 3 daerah profil kecepatan, yaitu dengan melihat hasil estimasi kecepatan rata-rata yang didapatkan dan dibandingkan dengan kecepatan sesungguhnya, kemudian akan diambil daerah ROI yang menghasilkan tingkat akurasi tertinggi, untuk kebutuhan analisis hasil estimasi, akan digunakan salah satu data video yang telah diketahui kecepatan sesungguhnya. Analisis pemilihan daerah ROI dibahas pada tahap uji coba.

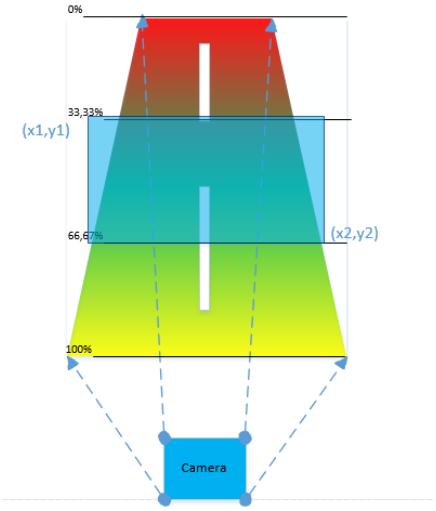


Gambar 4.28: Daerah Jangkauan Kamera Terbagi Menjadi Tiga Bagian

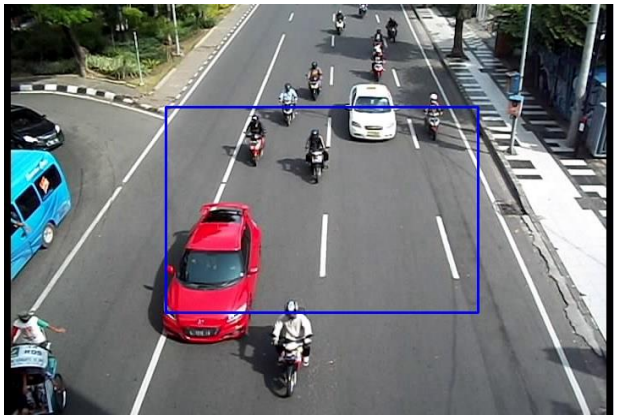
Dalam pemilihan daerah ROI, sistem akan menampilkan *image* awal saat *user* memasukkan video, kemudian *user* melakukan *click* dan *drag* pada panel video yang secara langsung akan diikuti oleh kotak biru sebagai tanda daerah ROI, terdapat dua jenis pemilihan daerah ROI:

1. *Default* yaitu *user* hanya menentukan koordinat x_1 dan x_2 atau lebar dari kotak biru yang menandakan daerah ROI sedangkan koordinat y_1 dan y_2 ditentukan oleh sistem sesuai satu pilihan dari 6 kemungkinan daerah ROI yang telah dijelaskan sebelumnya.
2. *Custom* yaitu *user* menentukan koordinat x_1, y_1, x_2, dan y_2 secara manual.

Gambar 4.29 menjelaskan bagaimana konsep daerah ROI dan Gambar 4.30 menjelaskan contoh pemilihan daerah ROI pada sistem.



Gambar 4.29: Konsep Pemilihan Daerah ROI



Gambar 4.30: Contoh Pemilihan Daerah ROI pada Sistem

4.3.4 Implementasi Proses *Preprocessing*

Sebelum video masuk proses *background substraction*, dilakukan *preprocessing* berupa *contrast* and *brightness adjustment* yang bertujuan untuk mengurangi efek bayangan kendaraan yang terlalu tebal akibat cahaya matahari yang terlalu terang yang mana akan menjadikan bayangan kendaraan terdeteksi menjadi suatu objek sehingga dapat mengurangi akurasi deteksi objek. Contoh bayangan yang terlalu tebal dapat dilihat pada Gambar 4.31 dan hasil *preprocessing* dengan parameter *alpha* dan *beta* untuk *contrast* dan *brightness adjustment* dapat dilihat pada Gambar 4.32. Analisis *preprocessing* untuk pengaruh bayangan dibahas lebih detil pada tahap uji coba.

Gambar 4.31: Contoh Bayangan Kendaraan yang Tebal

Gambar 4.32: Contoh *Preprocessing* dengan Nilai *alpha*=0.6 dan *beta*=40

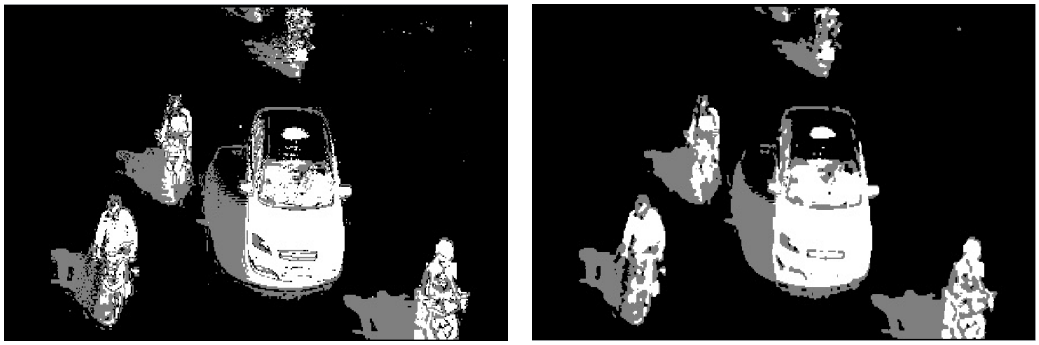
4.3.5 Implementasi Proses *Background Substraction*

Pada penelitian ini metode yang digunakan untuk *background subtraction* adalah GMM (*Gaussian Mixture Model*). Pada sistem, proses *background subtraction* dibantu dengan *class* BackgroundSubtractorMOG2 dari *library* OpenCV. OpenCV dapat mengidentifikasi bayangan dari sebuah obyek bergerak. Hasil keluaran dari class tersebut adalah *image grayscale* dengan 3 tingkat keabuan yaitu 0, 127, dan 255 yang menandakan *background*, *shadow*, dan *foreground* secara berurutan. Contoh hasil dari proses *background subtraction* pada program diberikan pada Gambar 4.33.

Gambar 4.33: Hasil *Background Substraction*

4.3.6 Implementasi Proses *Smoothing*

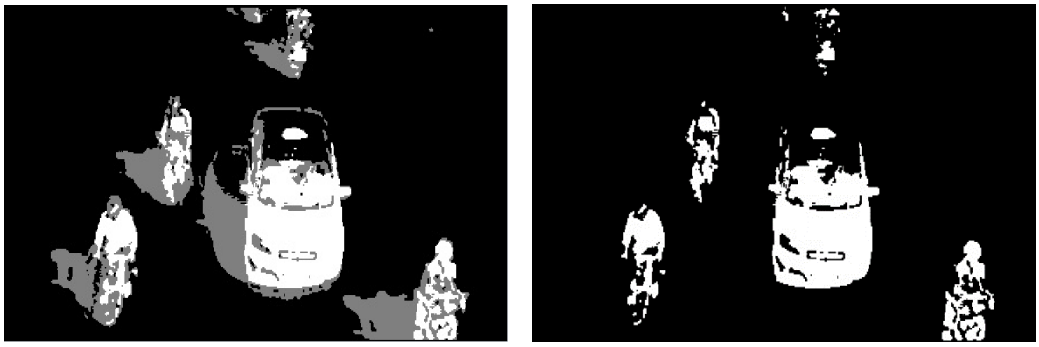
Metode *smoothing* yang digunakan adalah median *filter*. Data masukan pada proses *smoothing* adalah *image grayscale* hasil dari proses *background subtraction*. Tipe data dari hasil proses *smoothing* juga merupakan *image grayscale*. Contoh hasil dari proses *smoothing* diberikan pada Gambar 4.34.



Gambar 4.34: Hasil *Smoothing*. Kiri : sebelum. Kanan : sesudah

4.3.7 Implementasi Proses *Shadow Removal*

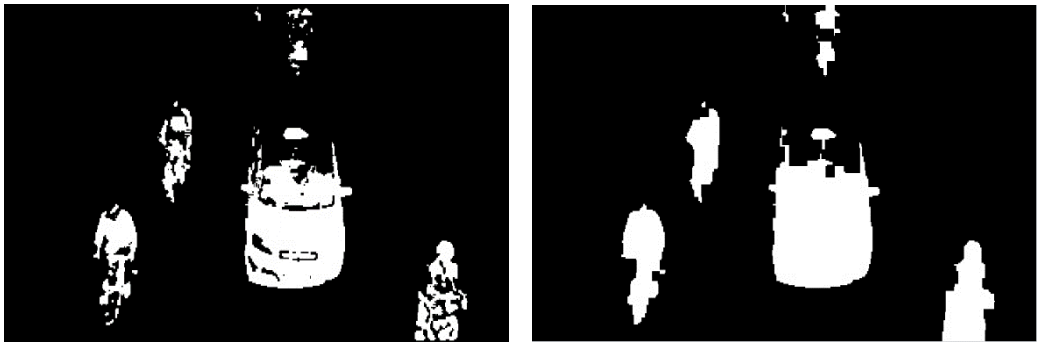
Proses *shadow removal* membutuhkan data masukan berupa *image grayscale* hasil dari *smoothing*. Pada sistem, semua piksel yang mempunyai nilai bayangan pada *image grayscale* hasil dari *smoothing* akan dijadikan nilai *background*. Sehingga *output* dari *shadow removal* merupakan *image* biner atau mempunyai dua nilai yaitu *background* dan *foreground*. Contoh hasil dari proses *shadow removal* diberikan pada Gambar 4.35.



Gambar 4.35: Hasil *Shadow Removal*. Kiri : sebelum. Kanan : sesudah

4.3.8 Implementasi Proses Morfologi

Proses morfologi membutuhkan data masukan berupa *image greyscale* hasil dari proses *shadow removal*. Pada Gambar 4.35 dapat dilihat bahwa objek pada *image* hasil proses *shadow removal* memiliki lubang (*holes*) pada objek yang terdeteksi yang akan mengganggu proses deteksi objek, sehingga perlu dilakukan proses *filling holes* dengan menggunakan operasi morfologi *closing* dengan tujuan untuk meningkatkan akurasi deteksi objek. Contoh hasil dari proses morfologi dapat dilihat pada Gambar 4.36.

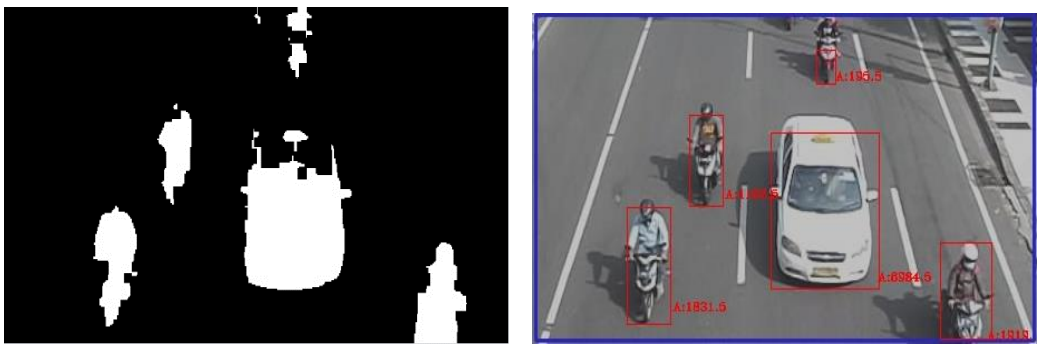


Gambar 4.36: Contoh Hasil Morfologi. Kiri : sebelum. Kanan : sesudah

4.3.9 Implementasi Proses Deteksi Objek

Data masukan pada proses deteksi obyek pada sistem adalah *image* biner hasil dari proses morfologi. Parameter masukan pada proses ini adalah *minimum area* atau luas area minimal dari objek yang akan dideteksi, hal ini bertujuan untuk mengurangi resiko terdeteksinya objek yang tidak dikehendaki seperti daun yang jatuh karena berukuran kecil, *noise* yang belum hilang setelah proses *smoothing*, dan benda-benda bergerak lain yang berukuran kecil. Penggunaan parameter *minimum area* pada deteksi objek harus sesuai pada kondisi *image*.

Setiap piksel objek yang diproses, sistem akan melakukan *border following* dengan menggunakan konsep *connected component* untuk mencari batas luar atau *outer border* yang digunakan sebagai penanda objek yang terdeteksi. Jika objek telah selesai diproses, sistem akan mempunyai data *border* dari objek tersebut. Kemudian dari batas piksel dari obyek tersebut akan diberikan *bounding box* dan dihitung luas area dari objek. Keluaran dari proses deteksi objek adalah objek-objek yang bergerak yang masing-masing objek memiliki *property* berupa luas area objek. Contoh hasil dari proses deteksi objek diberikan pada Gambar 4.37.



Gambar 4.37: Contoh Hasil Deteksi Objek. Kiri : sebelum. Kanan : sesudah

4.3.10 Implementasi Proses Estimasi Kecepatan

Proses estimasik kecepatan adalah proses yang merupakan tujuan utama dari sistem yang dibuat, yaitu menghitung kecepatan kendaraan bergerak. Data masukan pada proses ini adalah objek-objek bergerak yang terdeteksi pada sebuah *image*. Setiap objek yang terdeteksi memiliki *bounding box* dan titik *centroid*. Sistem akan memberikan label berupa ID sebagai identitas objek yang akan dicatat kecepatannya selama berada di dalam daerah ROI dalam proses *tracking* yang telah dijelaskan sebelumnya pada subbab 4.2.2.7 dan 4.2.2.8. Sehingga pada proses estimasi kecepatan, data keluaran pada proses ini berupa ID dan kecepatan kendaraan selama dalam daerah ROI yang ditampilkan pada *bounding box* objek sesuai dengan ID-nya. Ketika objek keluar dari daerah ROI maka kecepatan-kecepatan yang sudah diestimasi dan dicatat dihitung rata-ratanya kemudian hasilnya ditampilkan pada panel *result* berupa tabel dengan dua data yaitu ID kendaraan dan kecepatan rata-ratanya. Contoh hasil proses estimasi kecepatan dapat dilihat pada Gambar 4.38.

Gambar 4.38: Hasil Proses Estimasi Kecepatan Kendaraan dalam ROI

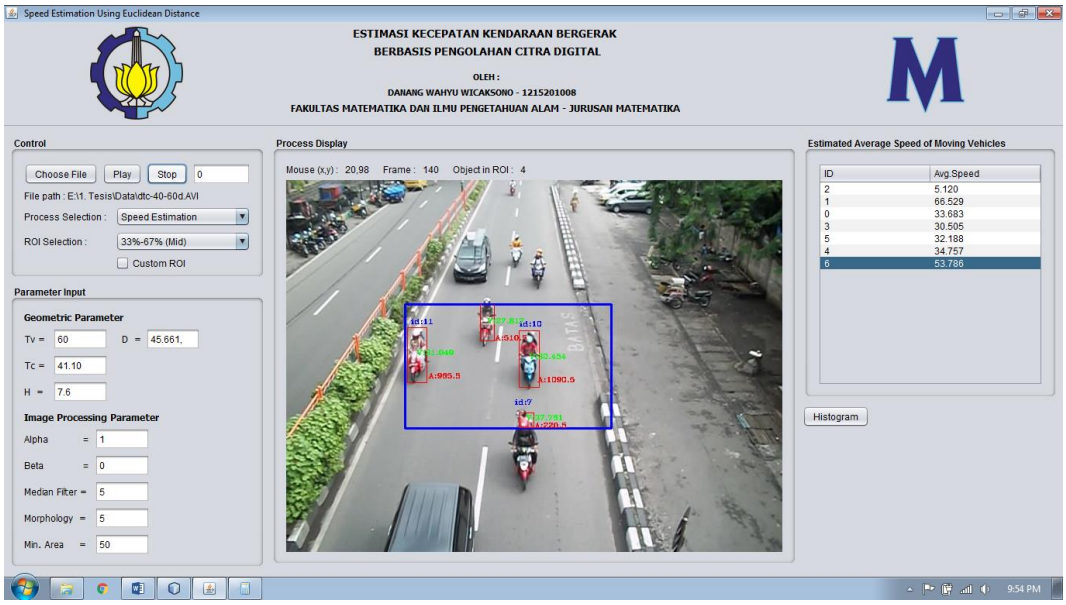
4.3.11 Implementasi Antar Muka Sistem

Antar muka sistem yang telah dirancang akan dibuat berdasarkan *layout* pada perancangan antar muka sistem pada subbab 4.2.3. Antar muka dibagi menjadi 4 bagian, yaitu panel *control*, panel *input*, panel video, dan panel *result*. Berikut adalah paparan dari tampilan utama yang terdeiri dari keempat bagian tersebut.

1. Implementasi Tampilan Utama

   Tampilan utama adalah halaman yang memuat 4 panel yaitu panel *control*¸ panel *input*, panel video, dan panel *result* dan menampilkannya dalam satu

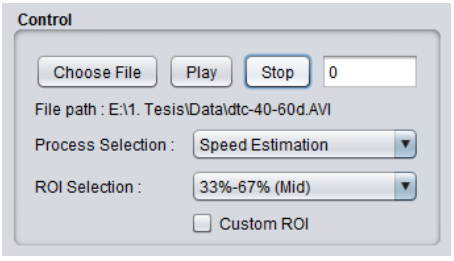
tampilan, sehingga memudahkan *user* dalam navigasi menu-menu proses dalam sistem. Desain tampilan utama diberikan pada Gambar 4.39.



Gambar 4.39: Tampilan Utama Sistem

2. Implementasi Panel *Control*

   Panel *control* adalah bagian halaman yang digunakan untuk memasukkan video, memutar video, dan menghentikan pemutaran video. Pada panel ini terdapat informasi lokasi berkas video yang telah dipilih dan pilihan proses untuk dapat memilih proses tertentu yang ingin ditampilkan hasilnya untuk analisis hasil pengolahan citra yang didapat dari parameter-parameter yang dimasukkan. Pilihan daerah ROI secara *default* atau *custom* terdapat pada panel ini. Panel *control* disajikan pada Gambar 4.40.

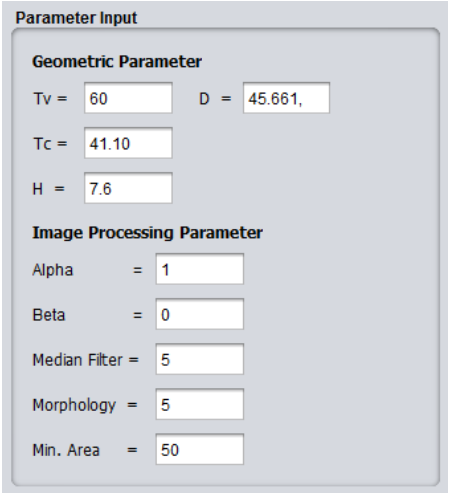


Gambar 4.40: Panel *Control*

3. Implementasi Panel *Input*

   Panel *input* adalah bagian halaman yang digunakan untuk memasukkan nilai parameter yang digunakan dalam proses estimasi kecepatan dimana terdapat

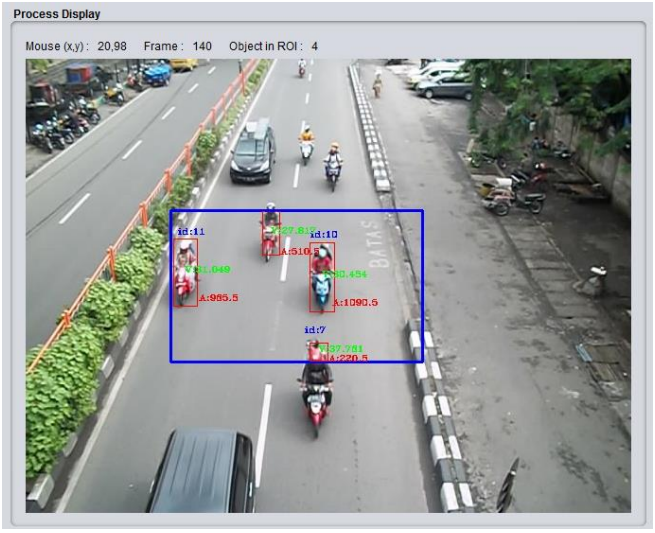
dua jenis parameter yaitu parameter geometris yang digunakan untuk kalibrasi kecepatan, contohnya sudut kemiringan kamera dan tinggi kamera. Selanjutnya adalah parameter pengolahan citra yang digunakan oleh proses-proses pengolahan citra contohnya *alpha* dan *beta* untuk *preprocessing*, ukuran median filter, ukuran kernel morfologi, dan luas area minimal untuk deteksi objek. Panel *input* disajikan pada Gambar 4.41.



Gambar 4.41: Panel *Input*

4. Implementasi Panel Video

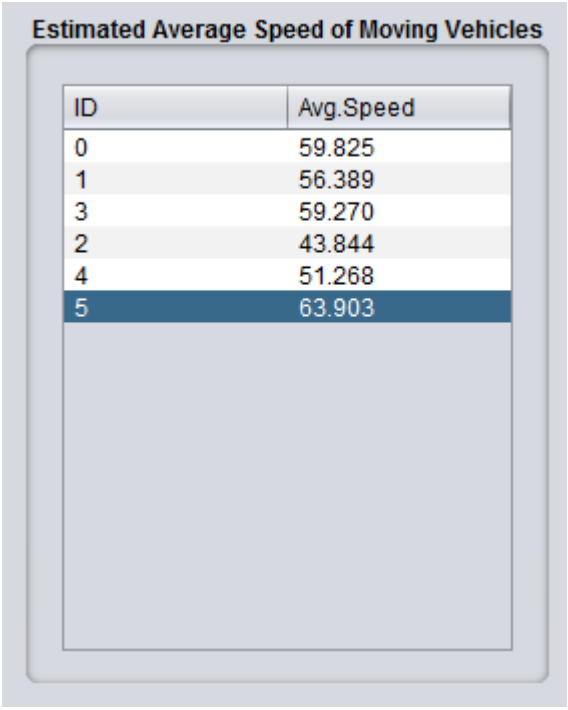
   Panel Video adalah bagian halaman yang berguna untuk menampilkan hasil dari proses-proses yang berjalan dalam sistem yang pada tahap implementasi bernama panel *Process Display*. Panel video ini juga sebagai tempat *user* memasukkan daerah ROI dengan melakukan *drag and drop* pada lokasi yang diinginkan. Pada panel ini terdapat informasi mengenai urutan *frame* yang sedang berjalan dan jumlah objek yang terdeteksi di dalam ROI. Panel video disajikan pada Gambar 4.42.



Gambar 4.42: Panel Video

5. Implementasi Panel *Result*

   Panel *Result* adalah bagian halaman yang berguna untuk menampilkan hasil estimasi kecepatan yang telah melalui proses sesuai rancangan algoritma pada subbab 4.2.2.9. Hasil estimasi kecepatan ditampilkan dalam bentuk tabel dengan dua informasi yaitu ID objek dan kecepatan rata-rata dalam satuan standar kilometer per jam atau km/jam. Panel *result* disajikan pada Gambar 4.43.



Gambar 4.43: Panel *Result*

BAB 5
UJI COBA DAN ANALISIS

Pada bab ini dijelaskan mengenai uji coba program yang telah dibuat dan kemudian hasil dari setiap uji coba akan dibahas. Uji coba yang pertama adalah pengujian deteksi objek menggunakan parameter *preprocessing* dan parameter pengolahan citra yang terbaik. Tujuan dari uji coba ini adalah untuk melihat tingkat keakurasian deteksi objek setelah proses *background substraction*. Tahap berikutnya analisis pemilihan daerah ROI yang tepat dan uji coba estimasi kecepatan kendaraan bergerak berdasarkan jarak titik *centroid* dan kalibrasi. Uji coba ini bertujuan untuk melihat dan menganalisis keakuratan hasil estimasi kecepatan kendaraan.

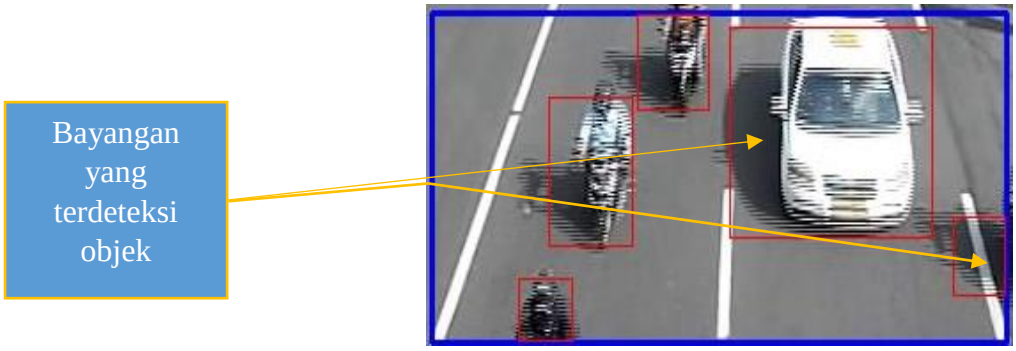
Uji coba yang dilakukan pada penelitian ini berorientasi pada keakuratan dan validasi hasil yang didapatkan, yaitu membandingkan hasil estimasi kecepatan yang didapatkan dari sistem dengan kecepatan kendaraan yang sesungguhnya. Hal ini diperoleh dengan melakukan akuisisi video kecepatan dari atas jembatan penyeberangan orang (JPO) kemudian mengatur daerah ROI pada sistem agar fokus pada satu objek kendaraan yang telah diketahui kecepatan sesungguhnya.

5.1 Uji Coba Deteksi Objek

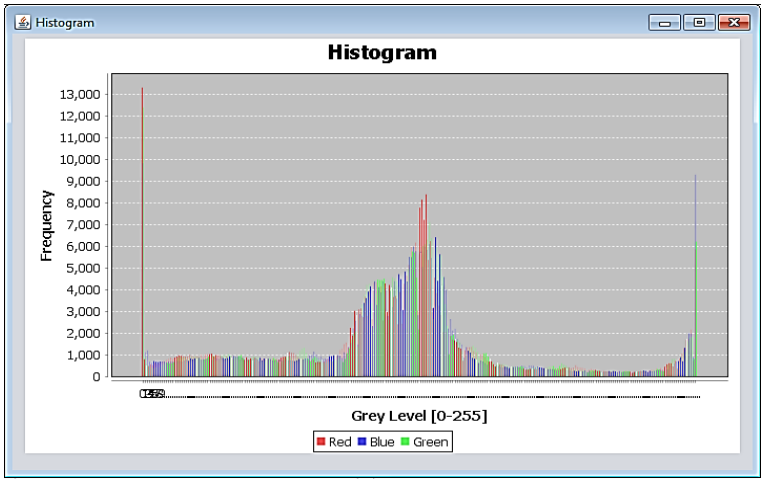
Sebelum data video masuk ke dalam proses *background substraction*, perlu dilakukan *preprocessing* terlebih dahulu, hal ini bertujuan untuk meningkatkan akurasi deteksi objek sehingga dapat meningkatkan akurasi estimasi kecepatannya. Pada penelitian ini, *preprocessing* yang dilakukan adalah berupa *contrast and brightness adjustment* dengan tujuan untuk mengurangi resiko munculnya bayangan objek yang terlalu tebal akibat cahaya matahari yang terlalu terang. Bayangan objek yang terlalu tebal akan mengganggu proses deteksi objek karena dapat menjadi suatu penghubung dengan objek lain sehingga ketika dijalankan proses *connected component*, dua objek tersebut akan terdeteksi sebagai satu objek karena piksel bayangan yang terdeteksi sebagai objek menyatukan kedua objek yang berbeda. Gambar 5.1 menunjukkan data video yang masuk ke dalam proses *background substraction* tanpa melalui *preprocessing* sehingga ketika terjadi

bayangan yang terlalu tebal maka akan mempengaruhi proses deteksi objek sehingga *bounding box* yang tidak optimal yang juga akan mempengaruhi titik *centroid*. Hal ini dapat dilihat pada Gambar 5.2 dengan histogram pada Gambar 5.3. Bayangan yang tebal pada Gambar 5.2 dapat ditunjukkan dengan bentuk histogram dimana piksel intensitas rendah mempunyai frekuensi yang cukup tinggi.

Gambar 5.1: Hasil *Background Substraction* dengan Bayangan Objek yang Tebal



Gambar 5.2: Hasil Deteksi Objek pada Objek dengan Bayangan Tebal

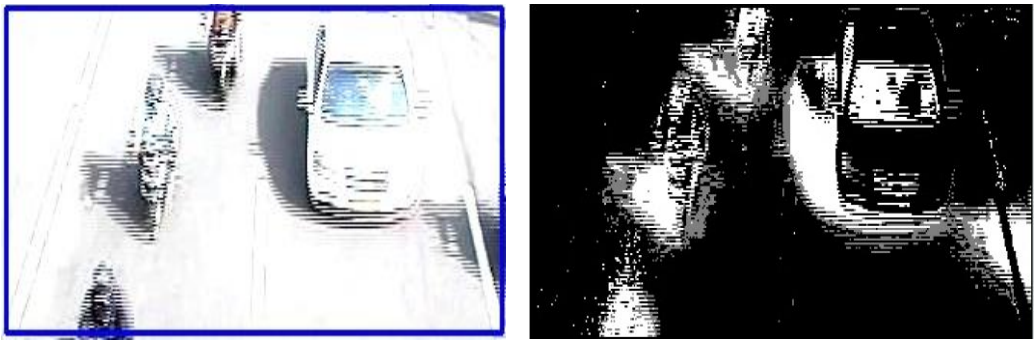


Gambar 5.3: Histogram *Image* pada Gambar 5.2

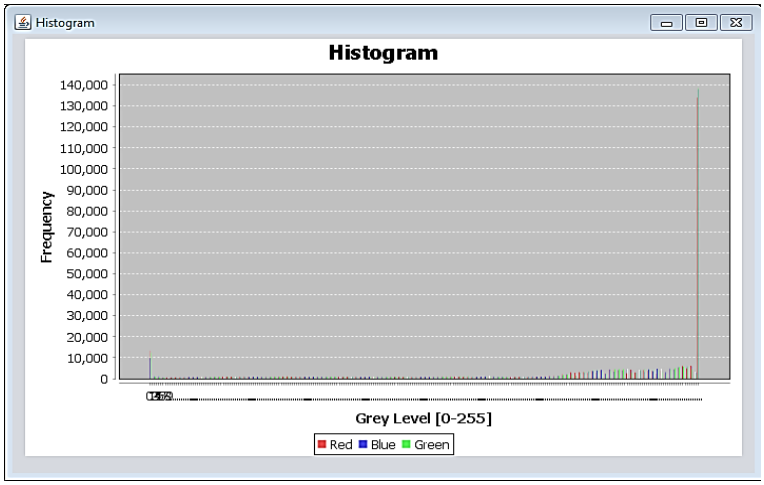
Dengan melakukan pengaturan *contrast* dan *brightness* melalui parameter *alpha* dan *beta* dimana nilai *alpha* mempunyai nilai *default* 1 dan *beta* mempunyai nilai *default* 0 (tidak ada perubahan nilai piksel) dan nilainya dapat naik atau turun, sehingga terdapat 8 kemungkinan kondisi nilai *alpha* dan *beta*, yaitu:

1. Nilai *Alpha* Naik dan *Beta* Tetap *Default*

   Ketika nilai *alpha* naik maka akan meningkatkan *contrast* atau ketajaman warna, semakin tinggi intensitas piksel, maka akan semakin menuju ke warna putih yang dapat ditunjukkan melalui bentuk histogram yaitu piksel dengan intensitas tinggi mempunyai frekuensi yang tinggi atau histogram cenderung condong ke kanan. Hal ini berakibat objek yang terdeteksi mendekati warna putih begitu juga dengan warna *background* yaitu jalan sehingga bayangan akan semakin terlihat jelas dan objek tidak terdeksi sempurna. Contoh hasil untuk kondisi ini disajikan pada Gambar 5.4 dengan histogram pada Gambar 5.5.



Gambar 5.4: Hasil *Image* dan *Background Substraction* dengan Nilai *Alpha* = 2 dan *Beta* = 0.

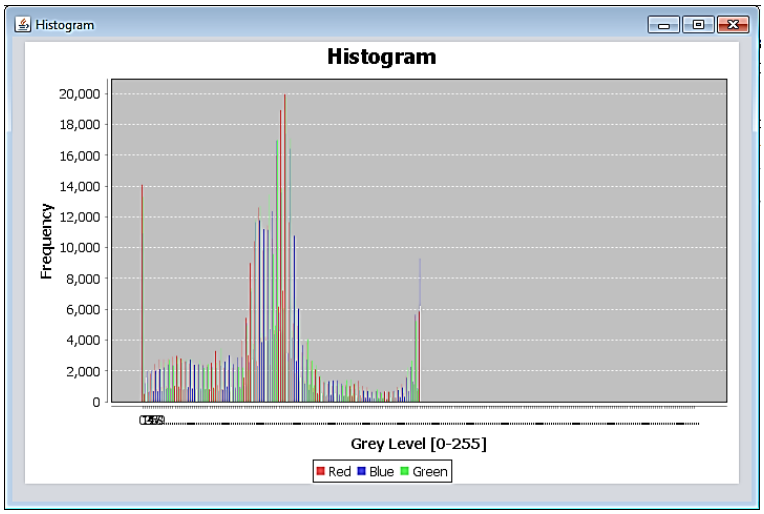


Gambar 5.5: Histogram *Image* dengan Nilai *Alpha* = 2 dan *Beta* = 0.

2. Nilai *Alpha* Turun dan *Beta* Tetap *Default*

   Ketika nilai *alpha* turun maka akan menurunkan *contrast* atau ketajaman warna, semakin rendah intensitas piksel, maka akan semakin menuju ke warna hitam yang dapat ditunjukkan melalui bentuk histogram yaitu piksel dengan intensitas rendah mempunyai frekuensi yang tinggi atau histogram cenderung condong ke kiri. Hal ini berakibat objek terdeteksi namun warna bayangan yang awalnya dominan ke hitam akan semakin hitam dan terlihat jelas. Contoh hasil untuk kondisi ini disajikan pada Gambar 5.6 dengan histogram pada Gambar 5.7.

Gambar 5.6: Hasil *Image* dan *Background Substraction* dengan Nilai *Alpha* = 0,5 dan *Beta* = 0.



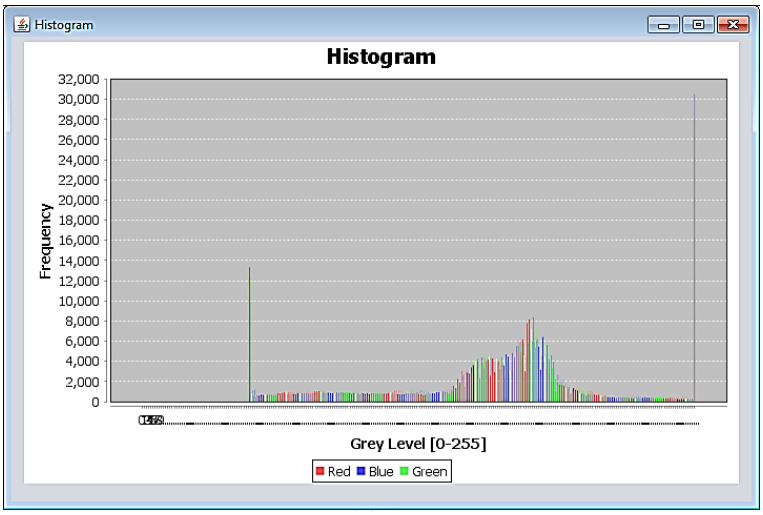
Gambar 5.7: Histogram *Image* dengan Nilai *Alpha* = 0,5 dan *Beta* = 0.

3. Nilai *Alpha* Tetap *Default* dan *Beta* Naik

   Ketika nilai *beta* naik maka akan meningkatkan *brightness* atau kecerahan warna. Hal ini berakibat objek dapat terdeteksi dan warna bayangan yang awalnya dominan ke hitam akan berkurang dan terlihat memudar yang dapat

ditunjukkan melalui bentuk histogram dimana piksel dengan intensitas rendah frekuensinya menurun. Contoh hasil untuk kondisi ini disajikan pada Gambar 5.8 dengan histogram pada Gambar 5.9.

Gambar 5.8: Hasil *Image* dan *Background Substraction* dengan Nilai *Alpha* = 1 dan *Beta* = 50.

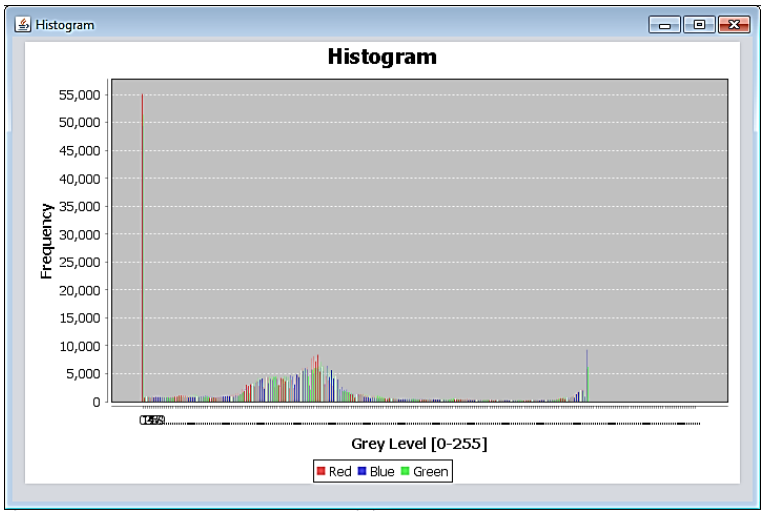


Gambar 5.9: Histogram *Image* dengan Nilai *Alpha* = 1 dan *Beta* = 50.

4. Nilai *Alpha* Tetap *Default* dan *Beta* Turun

   Ketika nilai *beta* turun maka akan menurunkan *brightness* atau kecerahan warna. Hal ini berakibat objek terdeteksi namun warna bayangan yang awalnya dominan ke hitam akan semakin hitam dan terlihat jelas yang dapat ditunjukkan melalui bentuk histogram dimana piksel dengan intensitas rendah frekuensinya meningkat. Contoh hasil untuk kondisi ini disajikan pada Gambar 5.10 dengan histogram pada Gambar 5.11.

Gambar 5.10: Hasil *Image* dan *Background Substraction* dengan Nilai *Alpha* = 1 dan *Beta* = -50.

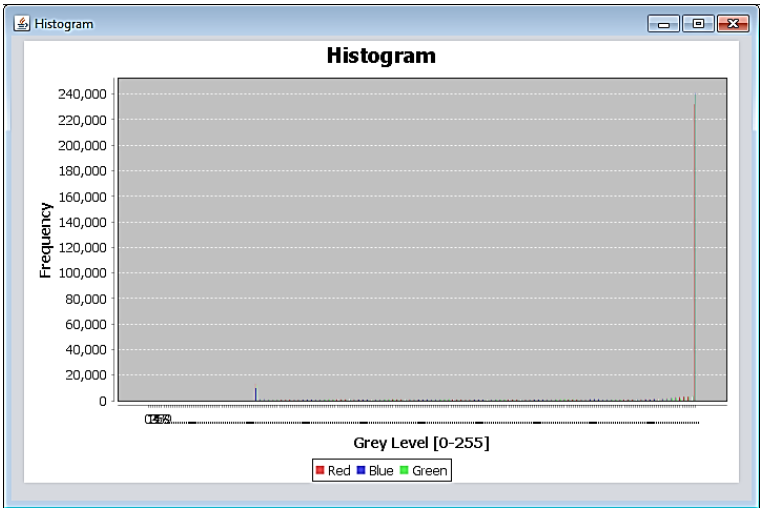


Gambar 5.11: Histogram *Image* dengan Nilai *Alpha* = 1 dan *Beta* = -50.

5. Nilai *Alpha* Naik dan *Beta* Naik

   Ketika nilai *alpha* dan *beta* naik maka akan meningkatkan *contrast* atau ketajaman warna sekaligus juga meningkatkan *brightness* atau kecerahan warna. Hal ini berakibat warna objek akan mendekati intensitas tertinggi yaitu warna putih dan bayangan memudar namun warna objek akan sama dengan *background* yang juga memutih sehingga bayangan terlihat jelas. Hal ini dapat ditunjukkan dengan bentuk histogram dimana piksel akan dominan pada intensitas tinggi dan mempunyai frekuensi tinggi. Contoh hasil untuk kondisi ini disajikan pada Gambar 5.12 dengan histogram pada Gambar 5.13.

Gambar 5.12: Hasil *Image* dan *Background Substraction* dengan Nilai *Alpha* = 2 dan *Beta* = 50.

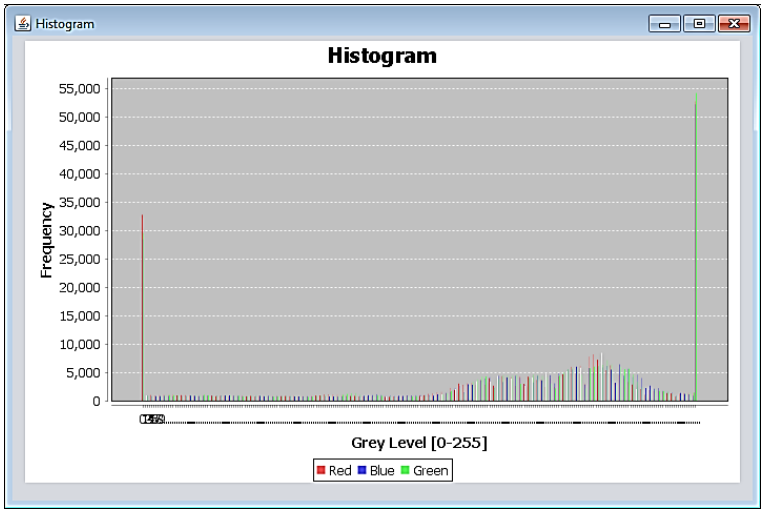


Gambar 5.13: Histogram *Image* dengan Nilai *Alpha* = 2 dan *Beta* = 50.

6. Nilai *Alpha* Naik dan *Beta* Turun

   Ketika nilai *alpha* naik maka akan meningkatkan *contrast* atau ketajaman warna dan ketika nilai *beta* turun maka akan menurunkan *brightness* atau kecerahan warna. Hal ini berakibat warna bayangan semakin tajam dan terlihat jelas yang dapat ditunjukkan dengan bentuk histogram dimana piksel intensitas tinggi frekuensinya meningkat dan piksel intensitas rendah juga meningkat. Contoh hasil untuk kondisi ini disajikan pada Gambar 5.14 dengan histogram pada Gambar 5.15.

Gambar 5.14: Hasil *Image* dan *Background Substraction* dengan Nilai *Alpha* = 2 dan *Beta* = -50.

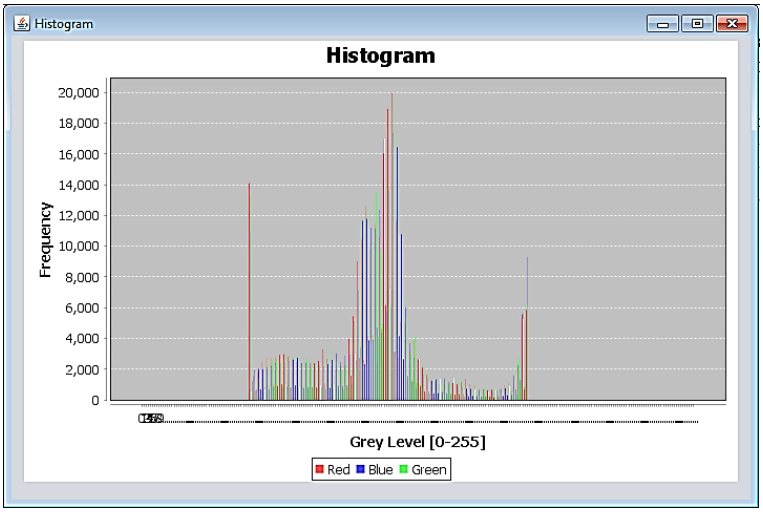


Gambar 5.15: Histogram *Image* dengan Nilai *Alpha* = 2 dan *Beta* = -50.

7. Nilai *Alpha* Turun dan *Beta* Naik

   Ketika nilai *alpha* turun maka akan menurunkan *contrast* atau ketajaman warna dan ketika nilai *beta* naik maka akan meningkatkan *brightness* atau kecerahan warna. Hal ini berakibat warna objek memudar dan bayangan juga memudar yang dapat ditunjukkan dengan bentuk histogram dimana piksel dengan intensitas tinggi frekuensinya menurun dan piksel dengan intensitas rendah frekuensinya juga menurun. Contoh hasil untuk kondisi ini disajikan pada Gambar 5.16 dengan histogram pada Gambar 5.17.

Gambar 5.16: Hasil *Image* dan *Background Substraction* dengan Nilai *Alpha* = 0,5 dan *Beta* = 50.

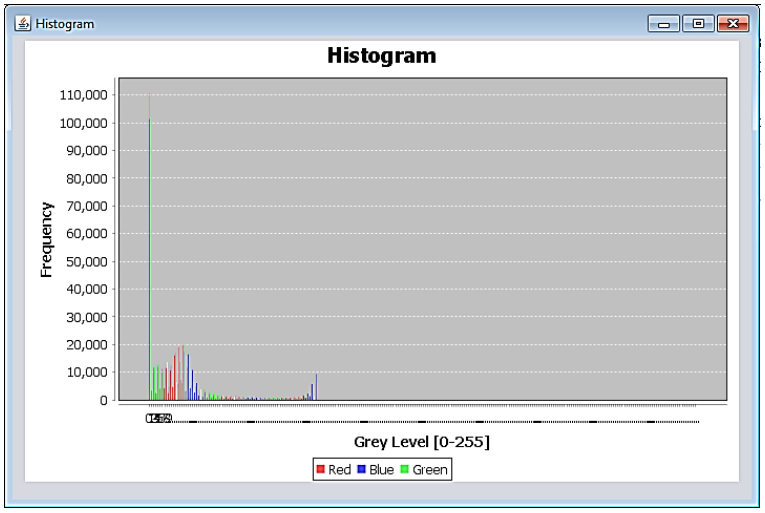


Gambar 5.17: Histogram *Image* dengan Nilai *Alpha* = 0.5 dan *Beta* = 50.

8. Nilai *Alpha* Turun dan *Beta* Turun

   Ketika nilai *alpha* turun maka akan menurunkan *contrast* atau ketajaman warna dan ketika nilai *beta* turun maka akan menurunkan *brightness* atau kecerahan warna. Hal ini berakibat warna objek dan bayangan akan mendekati intensitas terendah yaitu hitam sehingga bayangan akan terlihat jelas dan objek yang berwarna gelap tidak terdeteksi. Hal ini dapat ditunjukkan dengan bentuk histogram dimana piksel dengan intensitas rendah frekuensinya meningkat dan frekuensi piksel intensitas tinggi menurun. Contoh hasil untuk kondisi ini disajikan pada Gambar 5.18 dengan histogram pada Gambar 5.19.

Gambar 5.18: Hasil *Image* dan *Background Substraction* dengan Nilai *Alpha* = 0,5 dan *Beta* = -50.



Gambar 5.19: Histogram *Image* dengan Nilai *Alpha* = 0.5 dan *Beta* = -50.

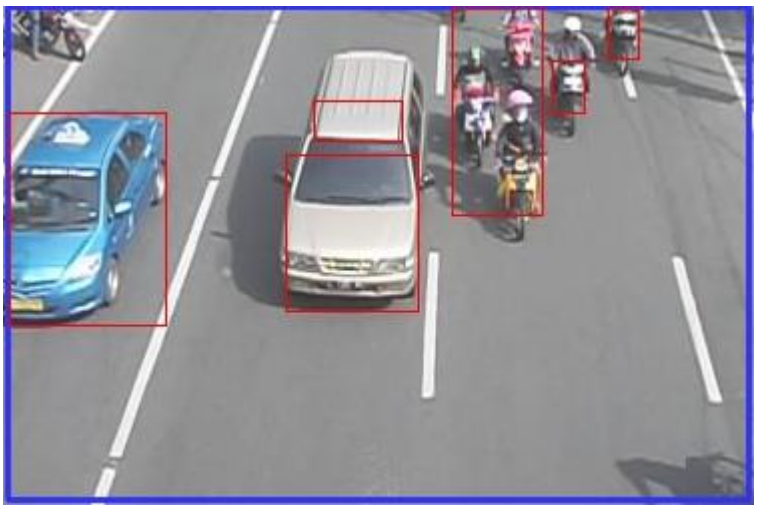
Dari hasil uji coba dan analisis parameter *alpha* dan *beta*, untuk dapat mengurangi atau bahkan menghilangkan efek bayangan pada proses *background substraction* maka perlu dilakukan kombinasi antara penurunan ketajaman warna atau *contrast* dan peningkatan kecerahan warna atau *brightness* yaitu menurunkan nilai *alpha* dan meningkatkan nilai *beta*. Gambar 5.20 menunjukkan hasil deteksi objek dengan penurunan *contrast* dan peningkatan *brightness* yang menghasilkan deteksi objek yang lebih baik dan Gambar 5.2 karena berkurangnya efek bayangan objek.

Gambar 5.20: Hasil *Background Substraction* dan Deteksi Objek dengan Nilai *Alpha* = 0,8 dan *Beta* = 50.

Selain efek bayangan yang terlalu tebal, ada beberapa faktor yang dapat mempengaruhi akurasi deteksi objek di antaranya adalah sebagai berikut:

1. Volume Lalu Lintas

   Jumlah kendaraan yang padat seringkali membuat kendaraan saling berdekatan dan berhimpitan yang berakibat beberapa objek kendaraan terdeteksi sebagai satu objek karena piksel-piksel yang terdeteksi saling berhubungan. Hasil deteksi objek dengan faktor jumlah kendaraan yang tinggi disajikan pada Gambar 5.21.



Gambar 5.21: Pengaruh Kepadatan Lalu Lintas pada Deteksi Objek.

2. Perubahan Intensitas Cahaya ke Kamera

   Perubahan intensitas cahaya yang masuk ke kamera yang diakibatkan oleh pantulan cahaya oleh kaca kendaraan atau perubahan intensitas cahaya matahari juga dapat mempengaruhi deteksi objek karena perubahan

intensitas citra yang tinggi secara tiba-tiba. Hasil deteksi objek dengan faktor pantulan cahaya ke kamera disajika pada Gambar 5.22.

Gambar 5.22: Pengaruh Perubahan Intensitas Cahaya pada Deteksi Objek.

3. Warna Kendaraan

   Warna kendaraan maupun bagian kendaraan yang hampir sama dengan warna *background*, dalam hal ini adalah jalan raya yang berwarna abu-abu akan terdeteksi sebagai background sehingga objek tidak terdeteksi sempurna, terlebih akan terdeteksi sebagai beberapa objek yang terpisah. Hasil deteksi objek dengan faktor warna kendaraan disajikan pada Gambar 5.23.

Gambar 5.23: Pengaruh Warna Kendaraan pada Deteksi Objek.

5.2 Pemilihan Nilai Parameter

Nilai parameter yang digunakan pada penelitian ini berorientasi pada hasil deteksi objek yang didapatkan dengan tujuan untuk mendapatkan hasil deteksi objek yang baik. Terdapat dua jenis parameter yaitu parameter geometris dan

parameter pengolahan citra. Parameter geometris didapat dari data lapangan yaitu berupa ketinggian kamera, sudut kemiringan kamera, dan sudut FOV kamera yang digunakan untuk akuisisi video. Sedangkan parameter pengolahan citra didapat dari uji coba deteksi objek terhadap data video sedemikian hingga didapat hasil deteksi objek yang optimal. Deteksi objek yang optimal dalam penelitian ini adalah bentuk dan jumlah objek yang terdeteksi oleh sistem mendekati atau sama dengan bentuk dan jumlah objek sebenarnya. Deteksi objek merupakan serangkaian proses yang berurutan sehingga proses sebelumnya akan menjadi data untuk proses selanjutnya. Rangkaian proses dalam deteksi objek disajikan pada Tabel 5.1.

Tabel 5.1: Detail Rangkaian Proses Deteksi Objek

Urutan	**Nama Proses**	**Parameter yang Digunakan**	**Tujuan**	***Output* yang diharapkan**
1	*Preprocessing*	*Alpha* dan *Beta*	Mengurangi efek bayangan pada objek	Objek tanpa bayangan yang terlalu tebal
2	*Smoothing*	*Size* Median Filter	Mengurangi *noise*	Objek dan *background* tanpa *noise*.
3	Morfologi	*Size Kernel* Morfologi	Rekonstruksi objek setelah *smoothing*	Bentuk objek lebih utuh, lubang dalam objek berkurang.
4	Filter Area Objek	*Minimum Area*	Mengurangi resiko deteksi benda-benda kecil	Objek dengan ukuran lebih dari sama dengan *minimum area*

Data yang akan digunakan pada uji coba deteksi objek adalah 3 data video yang didapat dari 3 sudut kemiringan yang berbeda.

5.2.1 Parameter *Alpha* dan *Beta*

Parameter *alpha* dan *beta* digunakan untuk tahap *preprocessing* yang bertujuan untuk mengurangi efek bayangan yang terlalu tebal yang dapat mengganggu proses deteksi objek. Penentuan nilai *alpha* dan *beta* mengacu pada bentuk objek yang terdeteksi sama dengan bentuk objek sebenarnya yaitu tidak mendeteksi bayangan

sebagai objek. Hasil analisis penentuan nilai *alpha* dan *beta* terhadap hasil deteksi objek yang didapatkan disajikan oleh Tabel 5.2, Tabel 5.3, dan Tabel 5.4.

Tabel 5.2: Nilai *Alpha* = 1 dan *Beta* = 0 (Nilai *Default*)

No.	**Nama Video**	**Durasi (detik)**	**Jumlah Objek Sebenarnya**	**Bentuk Deteksi Objek (T=Sesuai, F=Tidak Sesuai)**
1.	dtc-40-60d.avi	8	14	T
2.	dtc-40-50d.avi	3	4	T
3.	dtc-40-45d.avi	3	7	T
4.	waru-40-60d.avi	4	5	F

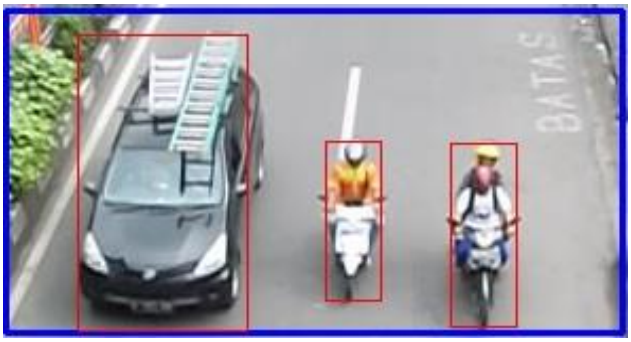
Tabel 5.3: Nilai *Alpha* = 0.9 dan *Beta* = 10

No.	**Nama Video**	**Durasi (detik)**	**Jumlah Objek Sebenarnya**	**Bentuk Deteksi Objek (T=Sesuai, F=Tidak Sesuai)**
1.	dtc-40-60d.avi	8	14	T
2.	dtc-40-50d.avi	3	4	T
3.	dtc-40-45d.avi	3	7	T
4.	waru-40-60d.avi	4	5	F

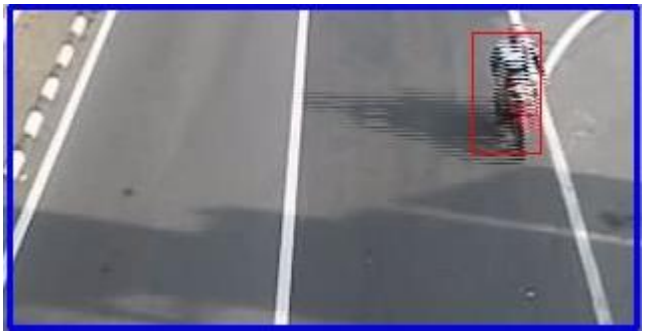
Tabel 5.4: Nilai *Alpha* = 0.8 dan *Beta* = 20

No.	**Nama Video**	**Durasi (detik)**	**Jumlah Objek Sebenarnya**	**Bentuk Deteksi Objek (T=Sesuai, F=Tidak Sesuai)**
1.	dtc-40-60d.avi	8	14	T
2.	dtc-40-50d.avi	3	4	T
3.	dtc-40-45d.avi	3	7	T
4.	waru-40-60d.avi	4	5	T

Contoh bentuk deteksi objek yang sesuai dengan bentuk objek sebenarnya disajikan pada Gambar 5.24 dan contoh bentuk deteksi objek yang tidak sesuai dengan bentuk objek sebenarnya disajikan pada Gambar 5.25, kemudian Gambar 5.26 menunjukkan bahwa dengan nilai *alpha* dan *beta* tertentu hasilnya akan sesuai dengan bentuk sebenarnya.



Gambar 5.24: Deteksi Objek pada Data dtc-40-60d



Gambar 5.25: Deteksi Objek pada Data waru-40-60d

Gambar 5.26: Deteksi Objek pada Data waru-40-60d dengan *alpha*=0.8 dan *beta*=20

5.2.2 Parameter *Size Median Filter*

Median filter digunakan untuk proses *smoothing* yaitu memperhalus *image* sehingga *noise* berkurang, contoh *noise* pada *image* dapat dilihat pada Gambar 5.27. Contoh *image* hasil *smoothing* dapat dilihat pada Gambar 5.28.

Gambar 5.27: *Noise* yang terdeteksi pada data dtc-40-60d tanpa proses *smoothing*.

Hasil analisis penentuan nilai *size* terhadap hasil deteksi objek yang didapatkan disajikan oleh Tabel 5.5 dan Tabel 5.6.

Tabel 5.5: Hasil Median Filter dengan *Size* = 3

No.	**Nama Video**	**Durasi (detik)**	**Jumlah Objek Sebenarnya**	***Noise* Berkurang (T=Ya, F=Tidak)**
1.	dtc-40-60d.avi	8	14	T
2.	dtc-40-50d.avi	3	4	T
3.	dtc-40-45d.avi	3	7	T
4.	waru-40-60d.avi	4	5	F

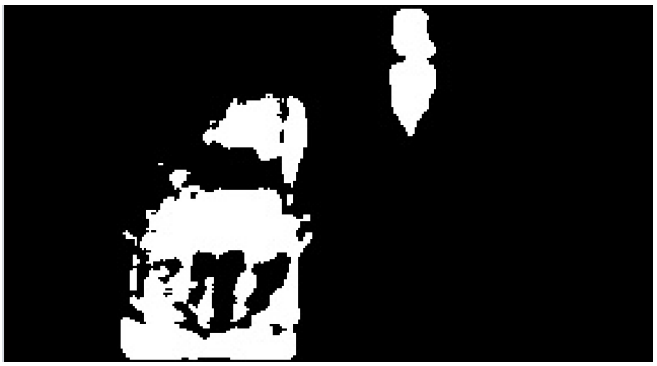
Tabel 5.6: Hasil Median Filter dengan *Size* = 5

No.	**Nama Video**	**Durasi (detik)**	**Jumlah Objek Sebenarnya**	***Noise* Berkurang (T=Ya, F=Tidak)**
1.	dtc-40-60d.avi	8	14	T
2.	dtc-40-50d.avi	3	4	T
3.	dtc-40-45d.avi	3	7	T
4.	waru-40-60d.avi	4	5	T

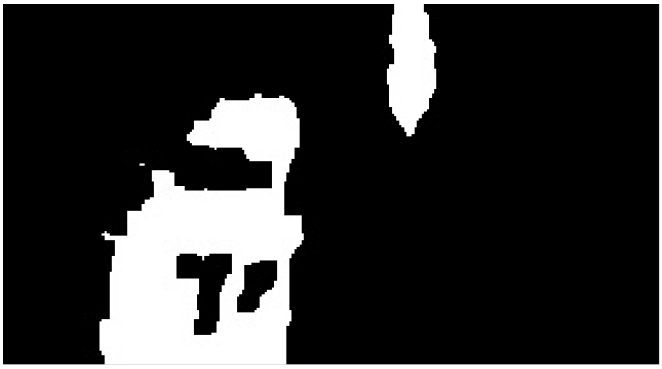
Gambar 5.28: *Noise* yang terdeteksi pada data dtc-40-60d setelah proses *smoothing* dengan *size*=3.

5.2.3 Parameter *Size Kernel* Morfologi

Proses morfologi digunakan untuk proses rekonstruksi *image*. Terkadang *image* hasil proses *smoothing* akan menghasilkan objek dengan lubang atau bentuk objek yang tidak sempurna sehingga dibutuhkan proses rekonstruksi *image*, salah satunya dengan menggunakan morfologi citra. Contoh *image* dengan bentuk tidak sempurna akibat *smoothing* dapat dilihat pada Gambar 5.29 dan *image* hasil morfologi dapat dilihat pada Gambar 5.30.



Gambar 5.29: Contoh bentuk objek yang tidak sempurna setelah proses *smoothing*



Gambar 5.30: Contoh bentuk objek setelah proses morfologi

Hasil analisis penentuan nilai *size kernel* morfologi terhadap hasil deteksi objek yang didapatkan disajikan oleh Tabel 5.7, Tabel 5.8, dan Tabel 5.9.

Tabel 5.7: Hasil Deteksi Objek tanpa Morfologi

No.	Nama Video	Durasi (detik)	Jumlah Objek Sebenarnya	Jumlah Objek yang terdeteksi
1.	dtc-40-60d.avi	8	14	249
2.	dtc-40-50d.avi	3	4	139
3.	dtc-40-45d.avi	3	7	35

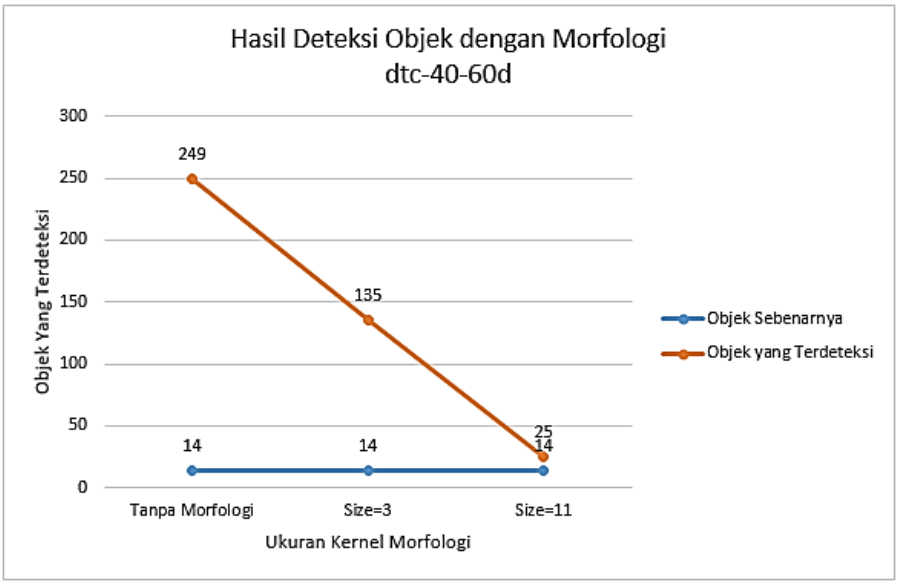
Tabel 5.8: Hasil Deteksi Objek dengan Morfologi *Size* = 3

No.	Nama Video	Durasi (detik)	Jumlah Objek Sebenarnya	Jumlah Objek yang terdeteksi
1.	dtc-40-60d.avi	8	14	135
2.	dtc-40-50d.avi	3	4	87
3.	dtc-40-45d.avi	3	7	26

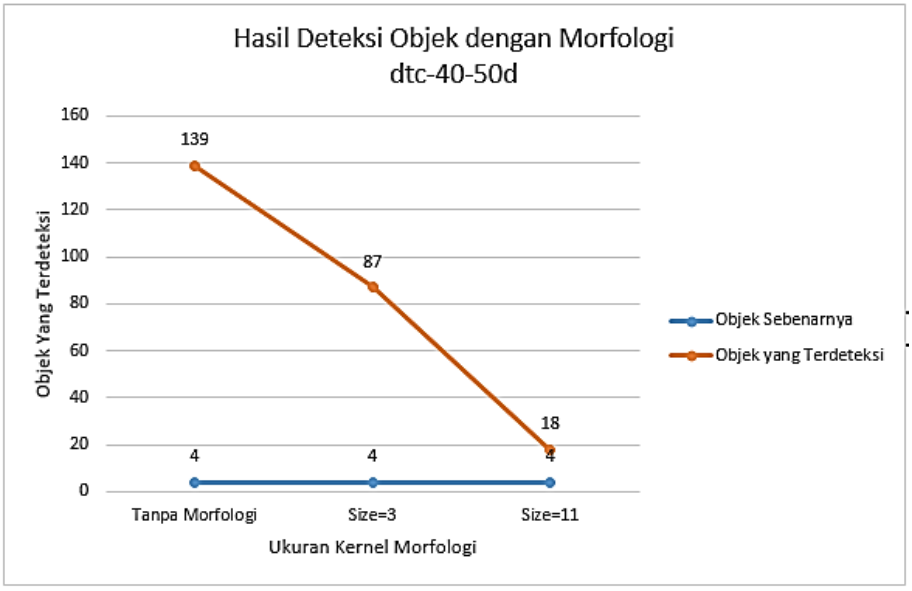
Tabel 5.9: Hasil Deteksi Objek dengan Morfologi *Size* = 11

No.	Nama Video	Durasi (detik)	Jumlah Objek Sebenarnya	Jumlah Objek yang terdeteksi
1.	dtc-40-60d.avi	8	14	25
2.	dtc-40-50d.avi	3	4	18
3.	dtc-40-45d.avi	3	7	12

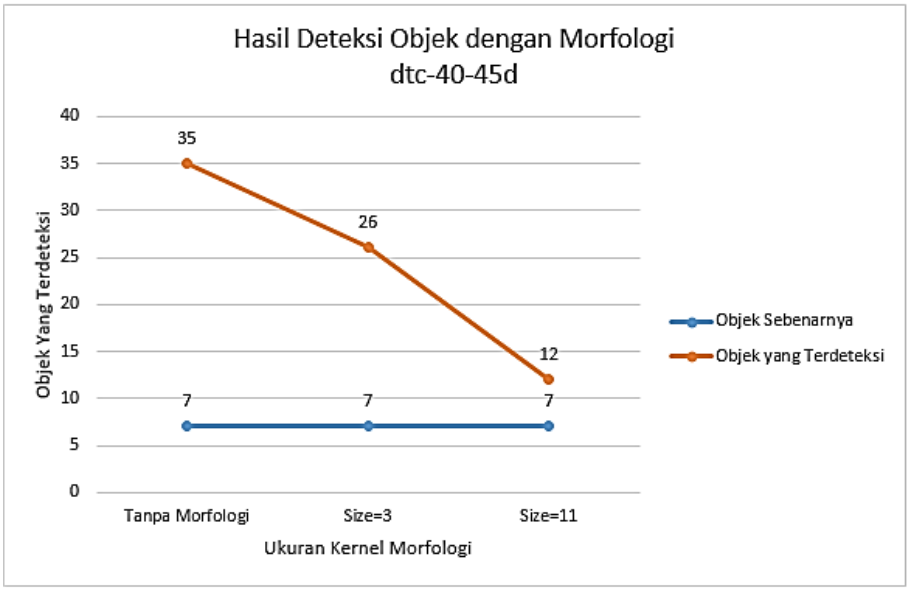
Dari ketiga tabel di atas dapat dilihat bahwa tanpa proses morfologi, objek yang terdeteksi jumlahnya jauh dari jumlah objek sebenarnya, hal ini dikarenakan objek yang tidak sempurna seperti pada Gambar 5.29 dihitung sebagai dua objek yang padahal dua objek tersebut adalah satu kesatuan objek yang tidak sempurna akibat proses *smoothing*. Dengan memberikan grafik pada Gambar 5.31, 5.32, dan 3.33 sebagai representasi hasil dari ketiga tabel di atas dapat disimpulkan bahwa rekonstruksi *image* menghasilkan deteksi objek yang lebih baik.



Gambar 5.31: Grafik Deteksi Objek dengan Morfologi pada Data dtc-40-60d



Gambar 5.32: Grafik Deteksi Objek dengan Morfologi pada Data dtc-40-50d



Gambar 5.33: Grafik Deteksi Objek dengan Morfologi pada Data dtc-40-45d

5.2.4 Parameter *Minimum Area*

Minimum area adalah parameter yang digunakan untuk proses filter area objek yang merupakan proses terakhir dalam deteksi objek setelah proses morfologi. Proses filter area ini bertujuan untuk lebih memfokuskan pada objek kendaraan berdasarkan ukurannya, menghindari objek-objek kecil yang bergerak, seperti daun-daun tanaman di tengah jalan maupun *noise* yang masih lolos dari proses *smoothing*. Ukuran *minimum area* objek berbeda-beda tergantung dari sudut pengambilan video atau kemiringan kamera. Semakin besar sudut kamera maka objek yang terdeteksi ukurannya lebih kecil dan semakin kecil sudut kamera maka objek yang terdeteksi ukurannya akan lebih besar dari ketinggian kamera yang sama. Contoh hasil objek yang terdeteksi dari sudut kemiringan kamera yang berbeda dapat dilihat pada Gambar 5.34.

Gambar 5.34: Contoh Objek yang Terdeteksi dari Sudut Berbeda. Kiri : 60 derajat. Kanan : 45 derajat

Area pada deteksi objek dihitung dari ukuran *bounding box* pada objek, dimana *bounding box* memiliki lebar dan tinggi seperti pada ilustrasi Gambar 4.23 sehingga area *bounding box* sama dengan luas *bounding box* itu sendiri. Hasil analisis penentuan nilai *minimum area* terhadap hasil deteksi objek yang didapatkan disajikan oleh Tabel 5.10, Tabel 5.11, dan Tabel 5.12.

Tabel 5.10: Hasil Deteksi Objek dengan *Minimum Area* = 50

No.	**Nama Video**	**Durasi (detik)**	**Jumlah Objek Sebenarnya**	**Jumlah Objek yang terdeteksi**
1.	dtc-40-60d.avi	8	14	15
2.	dtc-40-50d.avi	3	4	6
3.	dtc-40-45d.avi	3	7	8

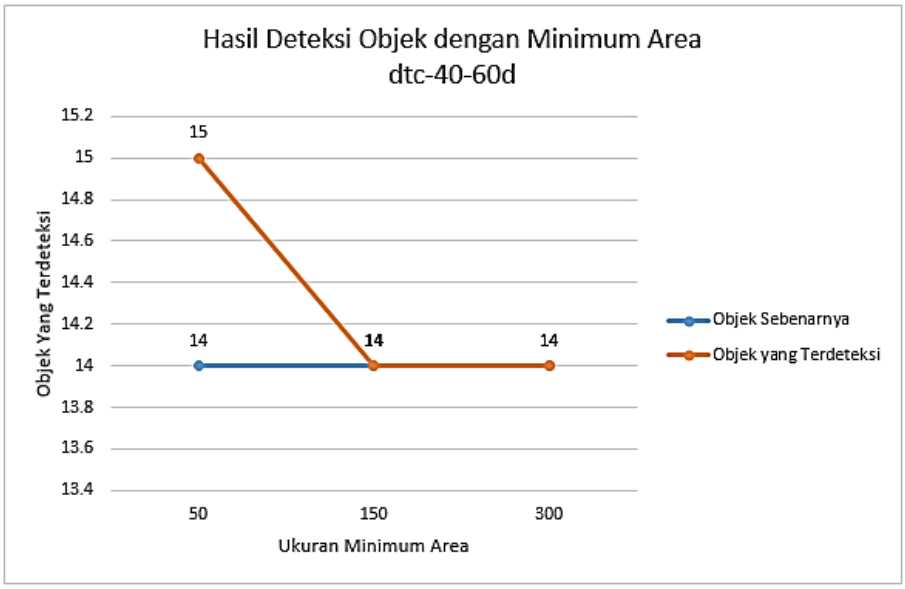
Tabel 5.11: Hasil Deteksi Objek dengan *Minimum Area* = 150

No.	**Nama Video**	**Durasi (detik)**	**Jumlah Objek Sebenarnya**	**Jumlah Objek yang terdeteksi**
1.	dtc-40-60d.avi	8	14	14
2.	dtc-40-50d.avi	3	4	6
3.	dtc-40-45d.avi	3	7	7

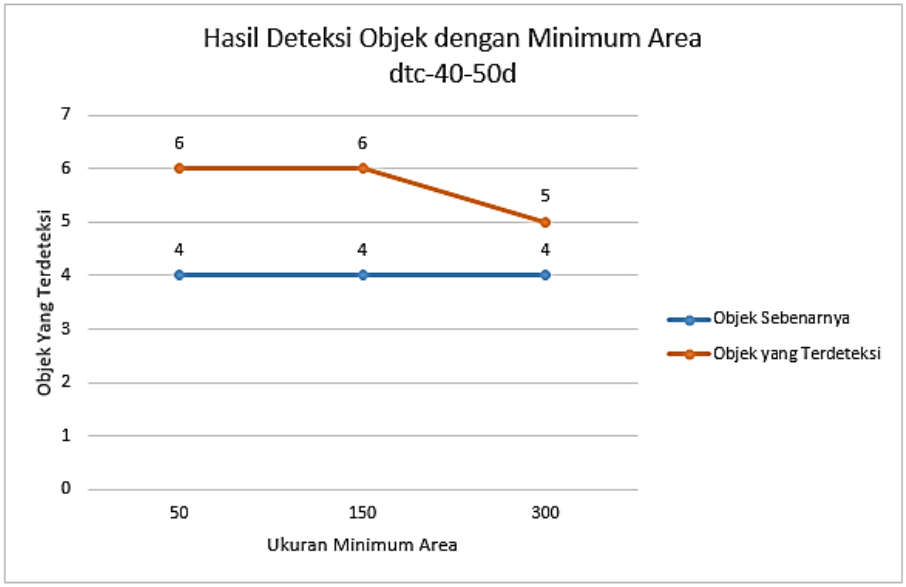
Tabel 5.12: Hasil Deteksi Objek dengan *Minimum Area* = 300

No.	**Nama Video**	**Durasi (detik)**	**Jumlah Objek Sebenarnya**	**Jumlah Objek yang terdeteksi**
1.	dtc-40-60d.avi	8	14	14
2.	dtc-40-50d.avi	3	4	5
3.	dtc-40-45d.avi	3	7	7

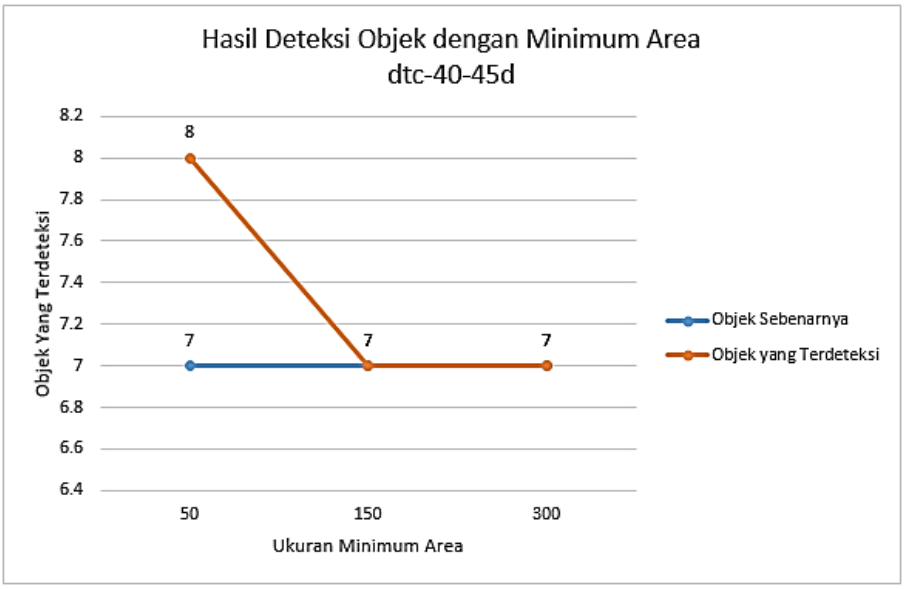
Dengan memberikan grafik pada Gambar 5.35, 5.36, dan 3.37 sebagai representasi hasil dari ketiga tabel di atas dapat disimpulkan bahwa filter area objek pada *image* menghasilkan deteksi objek yang lebih baik karena mengurangi dampak objek-objek kecil yang terdeteksi seperti daun tanaman di tengah jalan dan *noise* yang masih lolos dari proses *smoothing*.



Gambar 5.35: Grafik Deteksi Objek dengan *Minimum Area* pada Data dtc-40-60d



Gambar 5.36: Grafik Deteksi Objek dengan *Minimum Area* pada Data dtc-40-50d



Gambar 5.37: Grafik Deteksi Objek dengan *Minimum Area* pada Data dtc-40-45d

5.3 Pemilihan Daerah ROI

Daerah ROI dalam penelitian ini menjadi hal yang sangat penting karena perspektif visual kamera dengan sudut kemiringan akan memberikan dampak dalam estimasi kecepatan sehingga diperlukan analisis daerah ROI untuk menentukan daerah ROI yang tepat untuk mendapatkan hasil estimasi kecepatan yang akurat. Hasil analisis pemilihan daerah ROI terhadap hasil estimasi kecepatan yang didapatkan:

1. Daerah ROI terletak hanya pada daerah merah, yaitu profil kecepatan rendah (persentase 0% – 33,33%) didapat hasil sebagai berikut:

No.	Nama Video	Kecepatan sesungguhnya	T_v	Hasil Estimasi	*Relative Error* (%)	Tingkat Akurasi (%)
1.	dtc-40-60d.avi	40 km/jam	60	22,688 km/jam	43,28	56,72
2.	dtc-40-50d.avi	40 km/jam	50	13,931 km/jam	65,1725	34,8275
3.	dtc-40-45d.avi	40 km/jam	45	14,802 km/jam	62,995	37,005

2. Daerah ROI terletak hanya pada daerah hijau yaitu profil kecepatan sedang (persentase 33,33% – 66,67%) didapat hasil sebagai berikut:

No.	Nama Video	Kecepatan sesungguhnya	T_v	Hasil Estimasi	*Relative Error* (%)	Tingkat Akurasi (%)
1.	dtc-40-60d.avi	40 km/jam	60	34,344 km/jam	14,14	85,86
2.	dtc-40-50d.avi	40 km/jam	50	25,235 km/jam	36,9125	63,0875
3.	dtc-40-45d.avi	40 km/jam	45	23,978 km/jam	40,055	59,945

3. Daerah ROI terletak hanya pada daerah kuning yaitu profil kecepatan tinggi (persentase 66,67% – 100%) didapat hasil sebagai berikut:

No.	Nama Video	Kecepatan sesungguhnya	T_v	Hasil Estimasi	*Relative Error* (%)	Tingkat Akurasi (%)
1.	dtc-40-60d.avi	40 km/jam	60	57,694 km/jam	44,235	55,765
2.	dtc-40-50d.avi	40 km/jam	50	36,209 km/jam	9,4775	90,5225
3.	dtc-40-45d.avi	40 km/jam	45	33,554 km/jam	16,115	83,885

4. Daerah ROI terletak antara daerah merah dan hijau, yaitu profil kecepatan rendah dan sedang (persentase 0% – 66,67%) didapat hasil sebagai berikut:

No.	Nama Video	Kecepatan sesungguhnya	T_v	Hasil Estimasi	*Relative Error* (%)	Tingkat Akurasi (%)
1.	dtc-40-60d.avi	40 km/jam	60	30,098 km/jam	24,755	75,245
2.	dtc-40-50d.avi	40 km/jam	50	21,422 km/jam	46,445	53,555
3.	dtc-40-45d.avi	40 km/jam	45	22,669 km/jam	43,3275	56,6725

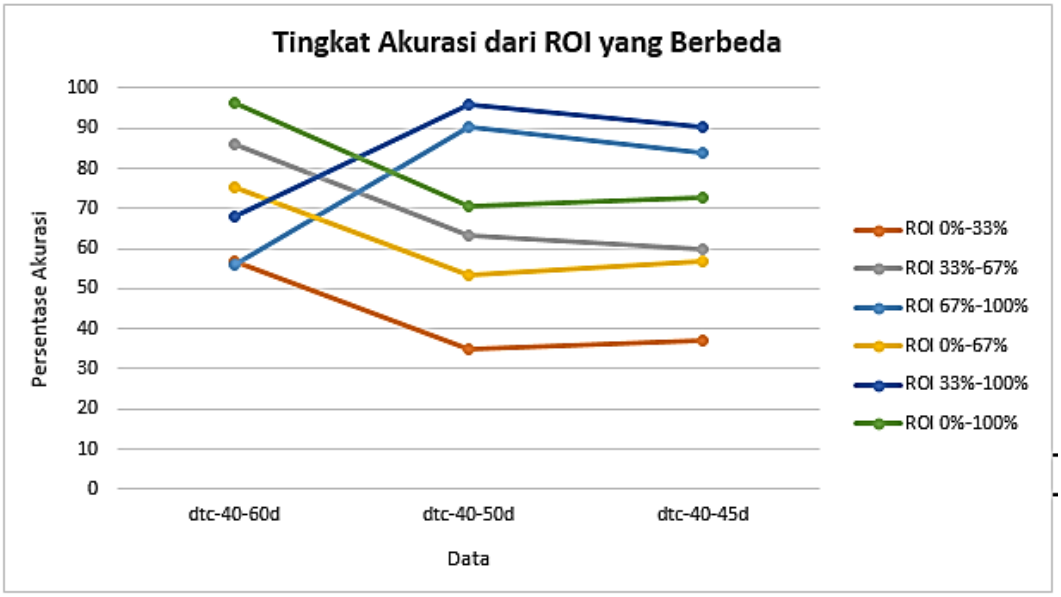
5. Daerah ROI terletak antara daerah hijau dan kuning, profil kecepatan sedang dan tinggi (persentase 33,33% – 100%) didapat hasil sebagai berikut:

No.	**Nama Video**	**Kecepatan sesungguhnya**	T_v	**Hasil Estimasi**	***Relative Error* (%)**	**Tingkat Akurasi (%)**
1.	dtc-40-60d.avi	40 km/jam	60	52,806 km/jam	32,015	67,985
2.	dtc-40-50d.avi	40 km/jam	50	38,348 km/jam	4.13	95,87
3.	dtc-40-45d.avi	40 km/jam	45	36,186 km/jam	9,535	90,465

6. Daerah ROI terletak antara daerah merah, hijau, dan kuning, yaitu profil kecepatan rendah, sedang, dan tinggi (persentase 0% – 100%) didapat hasil sebagai berikut:

No.	**Nama Video**	**Kecepatan sesungguhnya**	T_v	**Hasil Estimasi**	***Relative Error* (%)**	**Tingkat Akurasi (%)**
1.	dtc-40-60d.avi	40 km/jam	60	38,520 km/jam	3,7	96,3
2.	dtc-40-50d.avi	40 km/jam	50	28,151 km/jam	29,6225	70,3775
3.	dtc-40-45d.avi	40 km/jam	45	29,084 km/jam	27,29	72,71

Dari hasil yang didapat tersebut, dapat direpresentasikan dalam bentuk grafik pada Gambar 5.38.



Gambar 5.38: Grafik Tingkat Akurasi berdasarkan Daerah ROI.

Dari hasil tingkat akurasi yang dihasilkan untuk pemilihan daerah ROI yang berbeda, didapat hasil berikut:

1. Tingkat akurasi tertinggi pada data video dengan kemiringan kamera 60 derajat adalah daerah ROI dengan proporsi 0%-100% atau seluruh layar.
2. Tingkat akurasi tertinggi pada data video dengan kemiringan kamera 50 derajat adalah daerah ROI dengan proporsi 33%-100% atau dua sepertiga layar.
3. Tingkat akurasi tertinggi pada data video dengan kemiringan kamera 45 derajat adalah daerah ROI dengan proporsi 33%-100% atau dua sepertiga layar.

Sehingga dalam uji coba nantinya akan digunakan daerah ROI yang berbeda-beda sesuai dengan sudut kemiringan kamera pada video uji coba dengan tujuan untuk mendapatkan hasil estimasi yang terbaik.

5.4 Uji Coba Estimasi Kecepatan Kendaraan Bergerak

Dalam penelitian ini, setiap kendaraan yang melalui daerah ROI akan diestimasi kecepatan rata-ratanya dengan menggunakan *euclidean distance*. Data video yang akan diproses harus diketahui nilai parameter geometrisnya, yaitu meliputi sudut kemiringan kamera, sudut jangkauan FOV dari kamera, dan tinggi kamera dari permukaan jalan saat proses akuisisi video.

Uji coba yang dilakukan pada penelitian ini berorientasi pada akurasi dan validitas hasil yaitu dengan menjalankan sistem yang telah dibuat untuk mengestimasi kecepatan kendaraan yang bergerak kemudian membandingkan hasilnya dengan kecepatan sesungguhnya. Proses akuisisi video dengan mengetahui kecepatan kendaraan sesungguhnya dilakukan secara mandiri oleh penulis dengan meminta bantuan teman untuk menjalankan kendaraan pada kecepatan-kecepatan yang telah diatur yaitu 15, 20, 30, 40, 50, dan 60 km/jam dengan relatif konstan mulai dari kendaraan masuk daerah rekam hingga kendaraan keluar dari daerah rekam. Setiap kecepatan yang diatur tersebut juga diberlakukan untuk sudut kemiringan kamera yang berbeda yaitu 60, 50, dan 45 derajat.

Dalam uji coba estimasi kecepatan, hasil yang dilihat dan divalidasi adalah hasil kecepatan rata-rata kendaraan saat keluar dari ROI. Hasil estimasi akan divalidasi dengan menghitung kesalahan relatif dan tingkat akurasi berdasarkan Persamaan 4.22 dan 4.23.

5.4.1 Data Uji Coba

Data yang digunakan dalam uji coba estimasi kecepatan adalah video kendaraan di jalan raya yang satu dari kendaraan tersebut diketahui kecepatannya dengan parameter geometris yang berbeda-beda. Kamera yang digunakan adalah Kodak Easyshare C1530 yang diketahui *focal length* 32 mm dan *vertical dimension* 24 mm sehingga berdasarkan Persamaan 4.9, didapat $T_c = 41{,}10$ derajat. Detil data uji coba disajikan pada Tabel 5.13.

Tabel 5.13: Data Video yang Digunakan Uji Coba

No.	Nama Video	*Frame rate*	T_v	T_C	H	Kecepatan sesungguhnya
1.	dtc-15-60d.avi	30	60	41,10	7.6	15 km/jam
2.	dtc-20-60d.avi	30	60	41,10	7.6	20 km/jam
3.	dtc-30-60d.avi	30	60	41,10	7.6	30 km/jam
4.	dtc-40-60d.avi	30	60	41,10	7.6	40 km/jam
5.	dtc-50-60d.avi	30	60	41,10	7.6	50 km/jam
6.	dtc-60-60d.avi	30	60	41,10	7.6	60 km/jam
7.	dtc-15-50d.avi	30	50	41,10	7.6	15 km/jam
8.	dtc-20-50d.avi	30	50	41,10	7.6	20 km/jam
9.	dtc-30-50d.avi	30	50	41,10	7.6	30 km/jam

10.	dtc-40-50d.avi	30	50	41,10	7.6	40 km/jam
11.	dtc-50-50d.avi	30	50	41,10	7.6	50 km/jam
12.	dtc-60-50d.avi	30	50	41,10	7.6	60 km/jam
13.	dtc-15-45d.avi	30	45	41,10	7.6	15 km/jam
14.	dtc-20-45d.avi	30	45	41,10	7.6	20 km/jam
15.	dtc-30-45d.avi	30	45	41,10	7.6	30 km/jam
16.	dtc-40-45d.avi	30	45	41,10	7.6	40 km/jam
17.	dtc-50-45d.avi	30	45	41,10	7.6	50 km/jam
18.	dtc-60-45d.avi	30	45	41,10	7.6	60 km/jam

5.4.2 Hasil Uji Coba

Dengan data uji coba yang telah dipaparkan pada Tabel 5.13, hasil estimasi kecepatan oleh sistem disertai dengan tingkat akurasi dalam satuan persen (%) disajikan oleh Tabel 5.14.

Tabel 5.14: Hasil Uji Coba

No.	**Nama**	**Kecepatan sesungguhnya**	**Hasil Estimasi**	***Relative Error* (%)**	**Tingkat Akurasi (%)**
1.	dtc-15-60d.avi	15 km/jam	16,335 km/jam	8,9	91,1
2.	dtc-20-60d.avi	20 km/jam	21,424 km/jam	7,12	92,88
3.	dtc-30-60d.avi	30 km/jam	31,884 km/jam	6,28	93,72
4.	dtc-40-60d.avi	40 km/jam	38,520 km/jam	3,7	96,3
5.	dtc-50-60d.avi	50 km/jam	48,632 km/jam	2,736	97,264
6.	dtc-60-60d.avi	60 km/jam	53,487 km/jam	10,855	89,145
7.	dtc-15-50d.avi	15 km/jam	15,874 km/jam	5,83	94,17
8.	dtc-20-50d.avi	20 km/jam	21,154 km/jam	5,77	94,23
9.	dtc-30-50d.avi	30 km/jam	31,201 km/jam	4	96
10.	dtc-40-50d.avi	40 km/jam	38,348 km/jam	4,13	95,87

11.	dtc-50-50d.avi	50 km/jam	48,446 km/jam	3,108	96,892
12.	dtc-60-50d.avi	60 km/jam	53,807 km/jam	10,32	89,68
13.	dtc-15-45d.avi	15 km/jam	15,346 km/jam	2,30	97,7
14.	dtc-20-45d.avi	20 km/jam	20,124 km/jam	0,62	99,38
15.	dtc-30-45d.avi	30 km/jam	27,934 km/jam	6,88	93,12
16.	dtc-40-45d.avi	40 km/jam	36,922 km/jam	7,695	92,305
17.	dtc-50-45d.avi	50 km/jam	46,192 km/jam	7,616	92,384
18.	dtc-60-45d.avi	60 km/jam	52,201 km/jam	12,99	87,01

Selanjutnya adalah uji coba untuk kondisi khusus yaitu dengan video yang terdapat bayangan objek. Hasil estimasi untuk kondisi khusus disajikan oleh Tabel 5.15.

Tabel 5.15: Hasil Uji Coba pada Kendaraan Berbayangan

No.	**Nama Video**	**Kecepatan sesungguhnya**	**Hasil Estimasi**	***Relative Error* (%)**	**Tingkat Akurasi (%)**
1.	waru-40-60d.avi	40 km/jam	44,779 km/jam	11,95	88,05

Kemudian data yang digunakan pada Tabel 5.15 dilakukan *preprocessing* untuk menghilangkan efek bayangan pada deteksi objek sehingga didapat hasil estimasi kecepatan yang lebih baik. Hasil estimasi kecepatan setelah dilakukan *preprocessing* dapat dilihat pada Tabel 5.16.

Tabel 5.16: Hasil Uji Coba pada Kendaraan Berbayangan setelah *Preprocessing*

No.	**Nama Video**	**Kecepatan sesungguhnya**	**Hasil Estimasi**	***Relative Error* (%)**	**Tingkat Akurasi (%)**
1.	waru-40-60d.avi	40 km/jam	39,612 km/jam	0,97	99,03

Untuk kebutuhan validasi estimasi kecepatan kendaraan secara *multi* objek, digunakan data video yang disajikan pada Tabel 5.17, 5.18, dan 5.19 yaitu terdapat dua kendaraan yang telah diatur kecepatannya kemudian bergerak dengan tiga keadaan sebagai berikut:

1. Kendaraan pertama bergerak sejajar dengan kendaraan kedua dengan kecepatan yang sama.
2. Kendaraan pertama bergerak lebih cepat dari kendaraan kedua.
3. Kendaraan pertama dan kendaraan kedua bergerak dengan kecepatan yang sama namun dengan posisi yang tidak sejajar.

Tabel 5.17: Uji Coba pada Dua Kendaraan dengan Posisi Sejajar

No.	**Nama Video**				
1.	dtc-multi-20-50d.avi	**Kecepatan sesungguhnya kendaraan 1**	**Hasil Estimasi kendaraan 1**	***Relative Error* (%)**	**Tingkat Akurasi (%)**
		20 km/jam	19,345 km/jam	3,275	96,275
		Kecepatan sesungguhnya kendaraan 2	**Hasil Estimasi kendaraan 2**	***Relative Error* (%)**	**Tingkat Akurasi (%)**
		20 km/jam	20,848 km/jam	4,24	95,76

Tabel 5.18: Uji Coba pada Dua Kendaraan dengan Kecepatan Berbeda

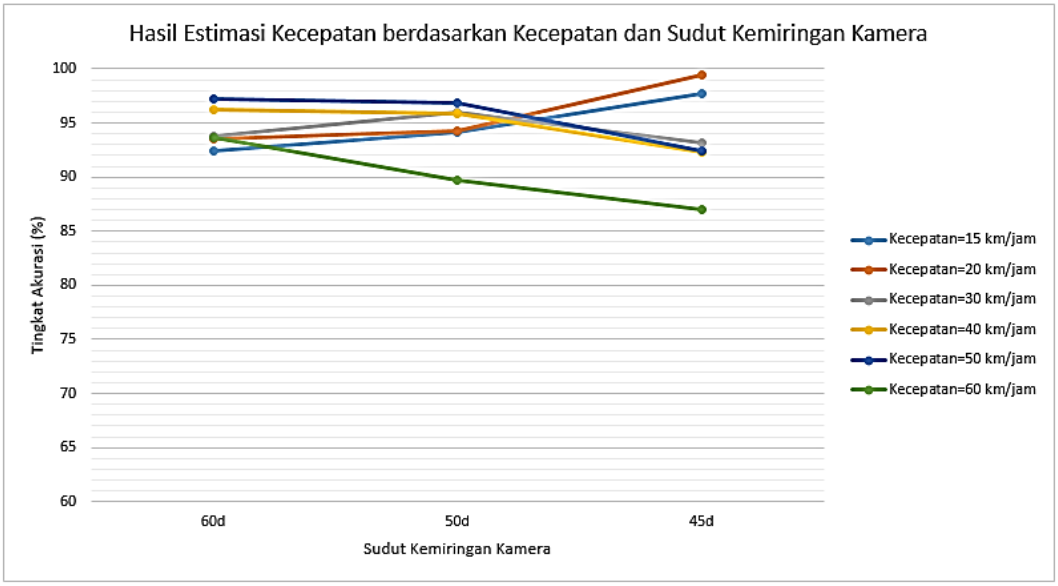
No.	**Nama Video**				
1.	dtc-multi-25-30-50d.avi	**Kecepatan sesungguhnya kendaraan 1**	**Hasil Estimasi kendaraan 1**	***Relative Error* (%)**	**Tingkat Akurasi (%)**
		25 km/jam	25,198 km/jam	0,792	99,208
		Kecepatan sesungguhnya kendaraan 2	**Hasil Estimasi kendaraan 2**	***Relative Error* (%)**	**Tingkat Akurasi (%)**
		30 km/jam	31,221 km/jam	4,07	95,93

Tabel 5.19: Uji Coba pada Dua Kendaraan dengan Kecepatan Sama namun Posisinya Tidak Sejajar

No.	**Nama Video**				
1.	dtc-multi-20-20-50d.avi	**Kecepatan sesungguhnya kendaraan 1**	**Hasil Estimasi kendaraan 1**	***Relative Error* (%)**	**Tingkat Akurasi (%)**
		20 km/jam	20,926 km/jam	4,63	95,37
		Kecepatan sesungguhnya kendaraan 2	**Hasil Estimasi kendaraan 2**	***Relative Error* (%)**	**Tingkat Akurasi (%)**
		20 km/jam	20,580 km/jam	2,9	97,1

5.4.3 Analisis Hasil Uji Coba

Dari hasil uji coba beberapa data, didapat hasil yang cukup memuaskan yaitu dengan tingkat akurasi terendah adalah 87,01 % dan tertinggi adalah 99,38 %. Selanjutnya adalah validasi untuk estimasi kecepatan kendaraan secara *multi* objek dengan tiga jenis keadaan didapat hasil yang konsisten dengan hasil yang didapat pada *single* objek. Faktor pengaruh terbesar pada akurasi hasil estimasi kecepatan adalah proses deteksi objek, akurasi parameter geometris, dan pemilihan daerah ROI. Telah dipaparkan pada penjelasan sebelumnya bahwa terdapat permasalahan tertentu yang masih menjadi kekurangan dalam proses deteksi objek. Hasil uji coba direpresentasikan dalam grafik pada Gambar 5.39.



Gambar 5.39: Grafik Tingkat Akurasi berdasarkan Kecepatan dan Sudut Kemiringan Kamera.

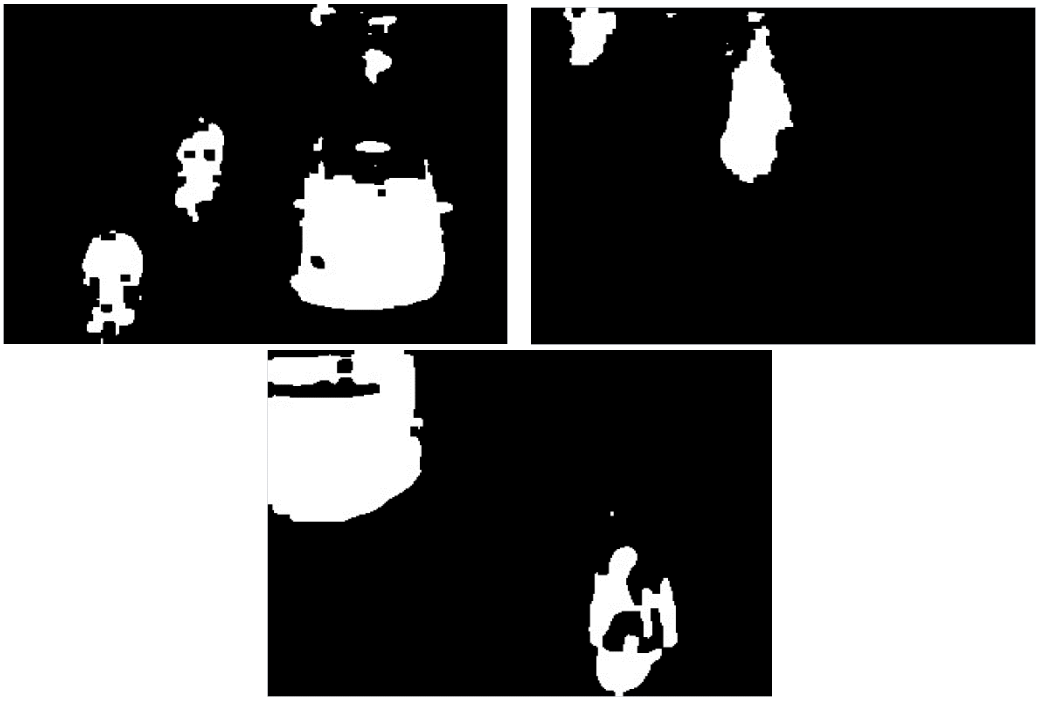
Dari grafik pada Gambar 5.39, dan dengan mengelompokkan kecepatan menjadi 3 jenis berdasarkan standar kecepatan dalam kota yaitu:

- Kecepatan rendah : 15 km/jam dan 20 km/jam
- Kecepatan sedang : 30 km/jam dan 40 km/jam
- Kecepatan tinggi : 50 km/jam dan 60 km/jam

Didapat hasil bahwa untuk kecepatan rendah, tingkat akurasi tertinggi adalah pada sudut kemiringan kamera 45 derajat, untuk kecepatan sedang, tingkat akurasi tertinggi adalah pada sudut kemiringan kamera 50 derajat, dan untuk kecepatan tinggi, tingkat akurasi tertinggi adalah pada sudut kamera 60 derajat. Sedangkan *error* atau kesalahan terbesar adalah estimasi pada kecepatan tinggi.

Sudut kemiringan kamera menghasilkan representasi objek bergerak yang berbeda, semakin besar sudut kemiringan kamera maka jangkauan kamera akan semakin luas sehingga objek kendaraan yang terdeteksi semakin kecil, dan semakin kecil sudut kemiringan kamera maka jangkauan kamera akan semakin sempit sehingga objek kendaraan yang terdeteksi semakin besar. Contoh objek yang diambil dengan kemiringan sudut berbeda disajikan oleh Gambar 5.40.

Kelebihan dari sudut kemiringan yang kecil adalah objek yang dideteksi berukuran lebih besar sehingga jika objek yang dideteksi cukup jelas, dengan proses morfologi akan menjadil sangat jelas, namun kekurangannya jika terjadi *noise* semisal akibat perubahan intensitas cahaya, maka *noise* akan tersebar dengan jumlah yang lebih besar daripada jika terjadi *noise* di *image* dengan pengambilan sudut kemiringan kamera lebih besar. Sehingga ketika terjadi *noise* pada *image* yang diambil dengan sudut lebih kecil, maka parameter-parameter median filter dan morfologi akan lebih besar. Begitu juga jika objek yang terdeteksi tidak sempurna yang dikarenakan warna kendaraan yang hampir sama dengan *background* maka akan sulit untuk mendeteksi keseluruhan objek karena terdapat lubang atau *holes* yang cukup besar.



Gambar 5.40: Hasil Deteksi Objek dengan Sudut Kemiringan Kamera yang Berbeda. Kiri : 60 derajat, Tengah : 50 derajat, Kanan : 20 derajat

BAB 6
KESIMPULAN DAN SARAN

Bab ini berisi tentang beberapa kesimpulan yang dapat diambil dari penelitian yang telah dilaksanakan. Di samping itu, pada bab ini juga dimasukkan beberapa saran yang dapat digunakan jika penelitian ini ingin dikembangkan.

6.1 Kesimpulan

Berdasarkan analisis terhadap hasil pengujian yang telah dilakukan, beberapa kesimpulan yang dapat diambil dari penelitian ini adalah sebagai berikut:

1. Efek bayangan yang terlalu tebal akibat intensitas cahaya matahari yang terlalu tinggi dapat dikurangi atau bahkan dihilangkan dengan melakukan pengaturan *contrast* dan *brightness* yaitu dengan kombinasi penurunan *contrast* dan peningkatan *brightness*.
2. Penelitian ini berhasil melakukan estimasi kecepatan kendaraan pada sudut kemiringan kamera yang berbeda-beda dengan hasil akurasi terendah adalah 87,01 % dan tertinggi adalah 99,38 % dan menghasilkan tingkat akurasi untuk estimasi kecepatan kendaraan *multi* objek yang konsisten.
3. Faktor-faktor yang mempengaruhi akurasi hasil estimasi kecepatan adalah proses deteksi objek, akurasi parameter geometris yang dimasukkan, dan pemilihan daerah ROI, dimana proses deteksi objek masih mempunyai beberapa kekurangan dalam kondisi tertentu.
4. Dari hasil yang didapat, dengan mengelompokkan data uji menjadi tiga jenis yaitu kecepatan rendah, sedang, dan tinggi bahwa sudut kamera dapat mempengaruhi akurasi hasil estimasi kecepatan. Sudut kemiringan kamera 45 derajat memberikan akurasi tertinggi untuk kecepatan rendah, sudut 50 derajat memberikan akurasi tertinggi untuk kecepatan sedang, dan sudut 50 derajat memberikan akurasi tertinggi untuk kecepatan tinggi sehingga hal ini dapat dijadikan bahan pertimbangan oleh pihak terkait.

6.2 Saran

Dengan melihat hasil yang dicapai pada penelitian ini, ada beberapa saran yang penulis sampaikan untuk pengembangan selanjutnya, yaitu:

1. Sistem deteksi objek masih memiliki kekurangan pada kondisi-kondisi tertentu sehingga diperlukan algoritma deteksi objek yang lebih handal sehingga estimasi kecepatan yang dihasilkan akan lebih akurat.
2. Diperlukan suatu pendekatan metode lain dan algoritma yang bersifat *robust* untuk dapat menghilangkan bayangan dan menentukan parameter untuk filter dan morfologi sesuai dengan kondisi *image*.
3. Data yang diproses dan hasil masih disimpan sementara pada memori CPU, sehingga diperlukan suatu *database* untuk dapat menyimpan data dalam jumlah besar dan dapat diolah untuk penelitian lebih lanjut.
4. Video masukan pada penelitian ini masih bersifat *offline* sehingga perlu dikembangkan untuk dapat memproses video *online* yaitu terhubung langsung dengan kamera CCTV untuk estimasi kecepatan kendaraan secara *realtime*.

DAFTAR PUSTAKA

[1] Kumar, T. & Dharmender Singh Kushwaha. (2016). "An Efficient Approach for Detection and Speed Estimation of Moving Vehicles". *Procedia Computer Science*. India, hal. 726-731.

[2] Jeyabharathi, D. & Dr. D Dejey. (2016). "Vehicle Tracking and Speed Measurement System (VTSM) Based On Novel Feature Descriptor: Diagonal Hexadecimal Pattern (DHP)". Journal of Vis. Commun. Image R Vol. 40, Hal. 816-830.

[3] Santosa S., Hardy & Agus Harjoko. (2016). "Vehicle Counting and Vehicle Speed Measurement Based On Video Processing". Journal of Theoretical and Applied Information Technology, Vol. 84, No. 2.

[4] Gokule, Reena & Amit Kulkarni. (2014). "Video Based Vehicle Speed Estimation and Stationary Vehicle Detection". International Journal of Advance Foundation and Research in Computer (IJAFRC), Vol. 1, Issue 11, ISSN 2348-4853.

[5] Supangkat, Suhono H, dkk. (2015). "Pengenalan dan Pengembangan Smart City". Bandung: Ell.

[6] Khan, A., dkk. (2014). "Speed Estimation of Vehicle in Intelligent Traffic Surveillance System Using Video Image Processing". *International Journal of Scienctific & Engineering Research,* Vol. 5, Issue 12.

[7] Lan, Jinhui, dkk. (2013). "Vehicle Speed Measurement Based On Gray Constraint Optical Flow". Journal of Optic Vol. 125, Hal. 289-295.

[8] Ranjit, S. S. S., dkk. (2012). "Real-Time Vehicle Speed Detection Algorithm using Motion Vector Technique". Proc. Of International Conference on Advances in Electrical & Electronics.

[9] Rad, A. G, Deghani. A, Karim. M. R.. (2010). "Vehicle Speed Detection in Video Image Squence using CVS Method". International Journal of the Physical Sciences Vol. 5(17), Hal. 2555-2563.

[10] 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.

[11] Alper Yilmaz, Omar Javed and Mubarak Shah. (2006). "Object tracking: A survey". ACM Comput. Suv., 38(4):13.

[12] Rafael C. Gonzalez, Richard E. Woods. (2002). "Digital Image Processing". Prentice Hall, New Jersey.

[13] Amaluddin, F., dkk. (2015). "Klasifikasi Kendaraan Menggunakan Gaussian Mixture Model (GMM) dan Fuzzy Cluster Mean (FCM)". Jurnal EECCIS Vol. 9, No. 1.

[14] Satoshi Suzuki and others. (1985). "Topological structural analysis of digitized binary images by border following". *Computer Vision, Graphics, and Image Processing*, 30(1):32–46.

[15] Putra, Bayu Charisma. (2016). "Klasifikasi Kendaraan Bergerak dengan Logika Fuzzy berbasis Pengolahan Citra". Tesis Pascasarjana pada Jurusan Matematika ITS.

LAMPIRAN

- Source Code Parameter Geometris

```
double teth1,teth3,H,D,tv;
teth3 = Double.valueOf(txtTetha3.getText()) * Math.PI / 180;
tv=Double.valueOf(txtTetha1.getText())*Math.PI/180;
teth1=tv+(teth3/2);
H=Double.valueOf(txtH.getText());

D=H*Math.tan(teth1);
txtD.setText(""+String.format("%.3f,", D));
scale=2*Math.sqrt(Math.pow(H,2)+Math.pow(D,2))*Math.tan(teth3/2;
```

- Source Code Kelas Obj untuk Objek Kendaraan

```
    class Obj{
        private Rect rect;
        private int id;
        private double estimated_speed;
        private boolean isOut;
        private ArrayList<Double> speeds;
        public Obj(Rect rect,int id){
            this.rect=rect;
            this.id=id;
            this.speeds=new ArrayList();
        }
        public Obj(Rect rect,int id,ArrayList speeds){
            this.rect=rect;
            this.id=id;
            this.speeds=speeds;
            this.isOut=false;
        }
        public Obj(Rect rect,int id,ArrayList speeds,boolean
isOut){
            this.rect=rect;
            this.id=id;
            this.speeds=speeds;
            this.isOut=isOut;
        }
        public void setRect(Rect rect){
            this.rect=rect;
        }
        public Rect getRect(){
            return rect;
        }
        public void setId(int id){
            this.id=id;
        }
        public int getId(){
            return id;
```

```
        }
        public void setIsOut(boolean isOut){
            this.isOut=isOut;
        }
        public boolean getIsOut(){
            return isOut;
        }
        public ArrayList<Double> getListSpeeds(){
            return speeds;
        }
        public double getEstimatedSpeed() {
            double total_speed=0;
            for(int i=0;i<speeds.size();i++){
                total_speed+=speeds.get(i);
            }
            estimated_speed=total_speed/speeds.size();

            return estimated_speed;
        }
    }
```

- Source Code Estimasi Kecepatan

```
        public void run() {
            synchronized (this) {
                while (runnable) {
                    if (videoSource.read(frame)) {
                        //untuk ROI
                        Imgproc.rectangle(frame, point1, point2,
new Scalar(255, 0, 0), 2);
                        frameROI=frame.submat(rect);
                        contours = new ArrayList();
                        System.out.println("Frame ke-
"+frame_count);

                        //get current process selection
                        processID=cbProcess.getSelectedIndex();
                        setProcessState(processID);

                        //contrast and brightness
                        if(statusPreprocessing){
                            alpha =
Double.valueOf(txtAlpha.getText());
                            beta =
Double.valueOf(txtBeta.getText());
                            frame.convertTo(frame, -1, alpha,
beta);
                        }

                        try {
                            //BS dengan GMM
                            if(statusBS){
```

```
                                mBGSub.apply(frameROI, mFGMask,
learning_rate);
                                //smoothing using median
                                if (statusMedianFilter) {
                                    int ksize =
Integer.parseInt(txtKSize.getText());
                                    Imgproc.medianBlur(mFGMask,
mFGMask, ksize);
                                }
                                //remove shadow
                                if(statusRemoveShadow){
                                    Imgproc.threshold(mFGMask,
mFGMask, 128, 255, Imgproc.THRESH_BINARY);
                                }
                                if (statusMorph) {
                                    int ksize =
Integer.parseInt(txtMSize.getText());
                                    Mat kernel =
Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new
Size(ksize, ksize));

Imgproc.morphologyEx(mFGMask, mFGMask, Imgproc.MORPH_CLOSE,
kernel);
                                }
                                if (statusDetectObject) {
                                    detect_obj = 1;

Imgproc.findContours(mFGMask, contours, hierarchy,
Imgproc.RETR_EXTERNAL, Imgproc.CHAIN_APPROX_SIMPLE);
                                    listObjIn.clear();//untuk
list id obj yang ada di roi
                                    for (MatOfPoint contour :
contours) { //looping objects
                                        double
area=Imgproc.contourArea(contour);
                                        if (area >=
Double.valueOf(txtArea.getText())) { //memenuhi area minimal
                                            Rect boundingBox =
Imgproc.boundingRect(contour);
                                            Point center = new
Point(boundingBox.x + (boundingBox.width / 2), boundingBox.y +
(boundingBox.height / 2));


if(statusEstimateSpeed){
                                                detect_speed=1;
                                                if
(listObj.isEmpty()) { //objek awal

System.out.println("Ada object baru "+idObj);

listObj.add(new FrameTest.Obj(boundingBox, idObj));

listObjIn.add(idObj);
```

```
                                                    } else {
                                                        boolean
check = false;
                                                        for
(FrameTest.Obj obj : listObj) {
                                                            Rect
objRect = obj.getRect();
                                                            if
(center.inside(objRect) && !obj.isOut) {
check = true; //ketemu objek yang sama

System.out.println("Sama dengan object " + obj.getId());

Point center1 = new Point(objRect.x + (objRect.width / 2),
objRect.y + (objRect.height / 2));

double distance = Math.sqrt(Math.pow(center.x - center1.x, 2) +
Math.pow(center.y - center1.y, 2));

double time = 1 / fps;

double calibration = scale / maxHeight;

double estimated_speed = (distance / time) * calibration * 3.6;
//km/h

Imgproc.rectangle(frameROI, boundingBox.tl(), boundingBox.br(),
new Scalar(0, 0, 255));

Imgproc.putText(frameROI, "V:"+String.format("%.3f",
obj.getEstimatedSpeed()), center,
Core.FONT_HERSHEY_COMPLEX_SMALL, 0.5, new Scalar(0, 255, 0));

Imgproc.putText(frameROI, "id:"+obj.getId(), obj.getRect().tl(),
Core.FONT_HERSHEY_COMPLEX_SMALL, 0.5, new Scalar(255, 0, 0));

Imgproc.putText(frameROI, "A:"+area, obj.getRect().br(),
Core.FONT_HERSHEY_COMPLEX_SMALL, 0.5, new Scalar(0, 0, 255));


listSpeeds=obj.getListSpeeds();

listSpeeds.add(estimated_speed);

listObj.set(obj.getId(), new FrameTest.Obj(boundingBox,
obj.getId(), listSpeeds)); //updating location of object

listObjIn.add(obj.getId());

break;
                                                            }
                                                        }
                                                        if (!check)
{
```

```
                                                    idObj++;

listObj.add(new FrameTest.Obj(boundingBox, idObj));

listObjIn.add(idObj);

System.out.println("Ada object baru "+idObj);
                                                        }
                                                    }
                                                }else{
                                                    detect_speed=0;

dtm.setRowCount(0);
                                                    listObj.clear();

listObjIn.clear();
                                                    idObj=0;

Imgproc.rectangle(frameROI, boundingBox.tl(), boundingBox.br(),
new Scalar(0, 0, 255));
                                                }
                                            }
                                        }
                                        //cek objek yang keluar roi
                                        if(detect_speed==1){
                                            for (FrameTest.Obj obj :
listObj) {

if(!listObjIn.contains(obj.getId()) && !obj.isOut){

System.out.println("Obj out of roi = " + obj.getId());

listSpeeds=obj.getListSpeeds();

listObj.set(obj.getId(), new FrameTest.Obj(obj.getRect(),
obj.getId(), listSpeeds, true)); //updating status of object
                                                    dtm.addRow(new
Object[]{obj.getId(),String.format("%.3f",
obj.getEstimatedSpeed())});

tableResults.changeSelection(tableResults.getRowCount() - 1, 0,
false, false);
                                                }
                                            }
                                        }
                                    } else {
                                        detect_obj = 0;
                                    }
                                    //untuk display saja
                                    if(detect_obj==0){
                                        imencode(".jpg", mFGMask,
mem);
                                    }else{
```

```
                                    imencode(".jpg", frame,
mem);
                                }
                            }else{
                                imencode(".jpg", frame, mem);
                            }
                            Image im = ImageIO.read(new
ByteArrayInputStream(mem.toArray()));

                            BufferedImage buff = (BufferedImage)
im;
                            Graphics g =
panelCamera.getGraphics();

                            if (g.drawImage(buff, 0, 0,
frame.width(), frame.height(), null)) {
                                if (runnable == false) {
                                    System.out.println("Going to
wait()");
                                    this.wait();
                                }
                            }
                            else{
                                System.out.println("Failed to
draw()");
                                this.wait();
                            }

                            Thread.sleep(delay);
                        } catch (Exception ex) {
                            System.out.println("Error :
"+ex.toString());
                            runnable=false;
                        }
                        frame_count++;
                    }
                    else{
                        System.out.println("Failed to play
video");
                        runnable=false;
                    }

                }
            }
        }
```

BIODATA PENULIS

Penulis memiliki nama lengkap Danang Wahyu Wicaksono yang biasa dipanggil Danang, dilahirkan di Blitar tanggal 31 Desember 1991 dan merupakan anak kedua dari lima bersaudara pasangan Bapak Gatut Hadi Prayogo dan Ibu Saropah. Pendidikan formal yang pernah ditempuh yaitu TK Pertiwi, SDN Bendogerit II Blitar, SMPN 1 Blitar, SMKN 1 Blitar jurusan Teknik Komputer dan Jaringan (TKJ). Setelah lulus SMK, penulis melanjutkan S1 Matematika di ITS yang masuk melalui jalur PMDK reguler tahun 2010 dan lulus pada tahun 2014. Setelah lulus S1, penulis bekerja sebagai *programmer* di PT. Dutacipta Konsultama (www.dckkonsulting.com) hingga akhirnya penulis melanjutkan studi kembali di Jurusan Matematika ITS pada tahun 2015, bidang minat penulis saat S1 dan S2 adalah Ilmu Komputer (ILKOM) karena penulis suka dengan dunia teknologi termasuk komputer yang dipadu dengan kesukaan penulis kepada matematika. Judul skripsi penulis saat S1 adalah "Sistem Deteksi Kemiripan antar Dokumen Teks Menggunakan Model Bayesian pada *Term Latent Semantic Analysis* (LSA)". Keseharian penulis selain kuliah juga bekerja *freelance* sebagai *trainer* di Cartenz HRD (www.cartenzhrd.com), *freelance* sebagai *programmer*, dan pengajar di Pendidikan Ahli Pemrograman Sistem Informasi (PAPSI) ITS. Jika ingin memberikan saran, kritik dan pertanyaan mengenai tesis ini, dapat dikirimkan melalui *e-mail* danangww@yahoo.com.