TESIS - SM 142501

PENGEMBANGAN ALGORITMA ROBUST UNTUK MENGHITUNG KENDARAAN BERGERAK BERBASIS PENGOLAHAN CITRA

REZA AUGUSTA JANNATUL FIRDAUS
NRP 1214 201 011

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

PROGRAM MAGISTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

THESIS - SM 142501

AN IMPROVED ROBUST ALGORITHMS FOR COUNTING MOVING VEHICLE BASED ON IMAGE PROCESSING

REZA AUGUSTA JANNATUL FIRDAUS
NRP 1214 201 011

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

MASTER'S DEGREE
MATHEMATICS DEPARTMENT
FACULTY OF MATHEMATICS AND NATURAL SCIENCES
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2016

PENGEMBANGAN ALGORITMA *ROBUST* UNTUK MENGHITUNG KENDARAAN BERGERAK BERBASIS PENGOLAHAN CITRA

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

oleh:
REZA AUGUSTA JANNATUL FIRDAUS
NRP. 1214 201 011

Tanggal Ujian : 21 Juli 2016
Periode Wisuda : September 2016

Disetujui oleh:

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

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

Dr. Darmaji, S.Si., MT.
NIP. 19691015 199412 1 001 (Penguji)

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

Dr. Dieky Adzkiya, S.Si., M.Si.
NIP. 19830517200812 1 003 (Penguji)

Direktur Program Pascasarjana,

Prof. Ir. DjauharManfaat, M.Sc., Ph.D
NIP. 19601202198701 1 001

PENGEMBANGAN ALGORITMA ROBUST UNTUK MENGHITUNG KENDARAAN BERGERAK BERBASIS PENGOLAHAN CITRA

Nama Mahasiswa	:	Reza Augusta Jannatul Firdaus
NRP	:	1214 201 011
Pembimbing	:	1. Dr. Budi Setiyono S.Si, M.T
		2. Dr. Dwi Ratna S.Si, M.T

ABSTRAK

Permasalahan lalu lintas merupakan salah satu permasalahan yang dihadapi di kota-kota besar pada umumnya. Hal ini dikarenakan meningkatnya jumlah volume kendaraan sehingga berpotensi menimbulkan kemacetan. Oleh karena itu diperlukan analisis kepadatan lalu lintas untuk mendapatkan informasi yang dibutuhkan seperti banyaknya kendaraan yang melintas, yang mana informasi tersebut nantinya akan dapat digunakan oleh pihak terkait sebagai pertimbangan pengaturan *traffic light*, pelebaran jalan, ataupun kebijakan-kebijakan lainnya. Salah satu metode yang dapat diterapkan adalah dengan cara melakukan *counting* berbasis pengolahan citra digital yang lebih efisien dari proses *counting* secara manual. Penelitian tentang *counting* berbasis pengolahan citra telah banyak dilakukan sebelumnya, namun beberapa terkendala waktu pengambilan video. Maka dari itu pada penelitian ini dibahas pengembangan algoritma *robust* terhadap video pagi, siang, dan sore untuk menghitung kendaraan pada video lalu lintas. Proses seperti *background subtraction*, *noise removal*, *object detection* dan *counting* akan dibahas didalamnya. Pada proses *background subtraction* digunakan metode *GMM* (*Gaussian Mixture Model*). Proses *robust* yang dilakukan adalah dengan cara melakukan proses *updating foreground* dari hasil proses *GMM* yang diperoleh, hal ini dilakukan untuk mendeteksi bayangan yang bergerak menyerupai objeknya. Proses tersebut dilakukan dengan cara melakukan pengecekan terhadap piksel pada suatu koordinat citra foreground yang bernilai 1 (terdeteksi sebagai foreground) dan dilakukan proses seleksi berdasarkan nilai mean dan modusnya pada intensitas frame saat itu. Pada *object detection* akan diperiksa tetangga pada tiap-tiap piksel *image*. Hasil yang diperoleh menunjukkan bahwa proses *updating foreground* memperoleh hasil yang lebih akurat dari proses *GMM*.

Kata kunci: *counting*, deteksi kendaraan bergerak, pengolahan citra digital, *Robust*, *Background Subtraction*

AN IMPROVED ROBUST ALGORITHMS FOR COUNTING MOVING VEHICLE BASED ON IMAGE PROCESSING

Name	:	Reza Augusta Jannatul Firdaus
NRP	:	1214 201 011
Supervisors	:	1. Dr. Budi Setiyono S.Si, M.T
		2. Dr. Dwi Ratna S.Si, M.T

ABSTRACT

The traffic problems are faced almost in every big city. This is due to the increase number of vehicles that potentially cause a traffic jam. Therefore, it is necessary to analyze the traffic density to obtain information, such as the number of passing vehicles, that information will become refference for taking some policies like traffic light time setting, road widening or the other policy. One of methods than can be implemented is by counting vehicle based on digital image processing, that more efficient than the manual counting. Research about image processing for counting has been done, but some of it depend on the video time taken. This research discussed about an improved robust algorithm for counting vehicle in a traffic video for morning, afternoon and evening video. Processes such as background subtraction, noise removal, object detection and counting will be discussed therein. In the background subtraction method is used *GMM* (Gaussian Mixture Model), then the obtained result from GMM method will be processed in order to remove shadow that move similarly like an vehicle based on its intensity with frame mean and mode. While in object detection will be checked neighbors on each pixel of the image. Robust process was carried out by analyzing shadow that detected in GMM process, then shadow removal will be executed in order to obtain detecting and counting results that was accurated, this process called updating foreground process. Updating foreground process will update the obtained result from GMM depend on its mean and mode on present frame. The obtained result show that updating foreground result is more accurate than *GMM*.

Keywords: counting, moving vehicle detection, image processing, robust, background subtraction

KATA PENGANTAR

Tidak henti-hentinya ucapan syukur Alhamdulillah atas nikmat, karunia, dan petunjuk Allah SWT sehingga penulis dapat menyelesaikan Thesis yang berjudul:

Pengembangan Algoritma Robust untuk Menghitung Kendaraan Bergerak Berbasis Pengolahan Citra

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, Ibu alm.Rina Herdiani, Ayah Muslim, dan Ibu Erna yang dengan segala kasih sayangnya telah memberikan motivasi, saran, dukungan dan doa kepada penulis sehingga dapat menyelesaikan Tesis ini.

2. Bapak Prof. Ir. Djauhar Manfaat, M.Sc., Ph.D, selaku Direktur Program Pascasarjana Institut Teknologi Sepuluh Nopember.

3. Bapak Dr. Imam Mukhlash, S.Si., MT., selaku Ketua Jurusan Matematika Institut Teknologi Sepuluh Nopember dan dosen penguji tesis yang telah banyak memberikan nasihat serta saran sehingga tesis ini dapat terselesaikan

4. Bapak Prof. DR. Basuki Widodo, M.Sc., selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam (FMIPA) sekaligus sebagai dosen wali yang telah banyak memberikan saran dan semangat kepada penulis.

5. Bapak Dr. Mahmud Yunus, M.Si., selaku Koordinator Program Studi Pascasarjana Matematika ITS.

6. Bapak Dr. Budi Setiyono S.Si, M.T, selaku pembimbing 1 tesis yang telah banyak mengarahkan, membimbing, dan memberi motivasi sehingga tesis ini dapat terselesaikan.

7. Bapak Dr. Dwi Ratna S.Si, M.T selaku pembimbing 2 tesis yang telah banyak mengarahkan, membimbing, dan memberi motivasi sehingga tesis ini dapat terselesaikan.

8. Bapak Dr. Darmaji, S.Si., MT, Prof., Bapak Dr. Mohammad Isa Irawan, M.T. dan Bapak Dr. Dieky Adzkiya, S.Si., M.Si., selaku penguji tesis yang telah banyak memberikan saran sehingga tesis ini dapat diselesaikan.

9. Seluruh Dosen, staf dan karyawan Jurusan Matematika ITS yang telah memberikan bantuan selama penulis menempuh perkuliahan ini.

10. Teman-teman S2 Matematika ITS angkatan 2014 khususnya teman-teman *'Ajah community'*, ada Bayu, Pandi, Kempa, Mba Nonop, Mba Dew, Farly, Indira, Ruzika dan Imam Fauzi. Tidak lupa juga penulis mengucapkan banyak terimakasih buat teman-teman yang lain, ada Lissa, Titak, Cynthia, Shinta, Aryani, Dya, Aminatu dan teman-teman yang lain yang tidak bisa diucapkan satu-persatu.

11. Teman-teman dekat dari penulis khususnya Bang Satria, Ipin, Ferry, Angga, Mimin dan Athok yang telah memberikan semangat dan hiburan selama penulis mengerjakan Tesis ini.

12. Seluruh pihak yang telah memberikan saran, dukungan dan motivasi dalam menyelesaikan tesis ini.

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

Surabaya, Juli 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Permasalahan lalu lintas merupakan salah satu permasalahan yang dihadapi di kota-kota besar pada umumnya. Hal ini dikarenakan meningkatnya jumlah volume kendaraan sehingga berpotensi menimbulkan kemacetan. Oleh karena itu diperlukan analisis kepadatan lalu lintas untuk mendapatkan informasi yang dibutuhkan seperti banyaknya kendaraan yang melintas, yang mana informasi tersebut nantinya akan dapat digunakan sebagai pertimbangan pengaturan *traffic light*, pelebaran jalan, ataupun kebijakan-kebijakan lainnya. Untuk mendapatkan informasi tersebut dibutuhkan rekayasa lalu lintas dengan cara melakukan *counting* kendaraan. Namun yang menjadi masalah di beberapa tempat pada umumnya, metode yang digunakan untuk mendapatkan data jumlah kendaraan yang melintas masih menggunakan metode manual yakni menggunakan alat hitung manual. Untuk itu dibutuhkan suatu metode alternatif yang lebih efisien dari metode perhitungan secara manual data. Salah satu metode yang dapat diterapkan adalah dengan cara melakukan *counting* berbasis pengolahan citra digital, yang mana *counting* berbasis pengolahan citra digital lebih efisien dan ekonomis dari *counting* secara manual. Metode ini diterapkan pada video lalu lintas yang ada dengan memperhatikan beberapa parameter seperti kecepatan kendaraan, pencahayaan, dan kepadatan lalu lintas.

Penelitian tentang deteksi dan perhitungan kendaraan berbasis citra telah banyak dilakukan sebelumnya, salah satunya oleh Pratistha Gupta dkk(2009). Dalam penelitiannya, metode yang digunakan untuk proses deteksi adalah *edge detection*. Kelemahan dari metode *edge detection* adalah ketika pada saat deteksi, masih dibutuhkan suatu metode koneksi agar obyek yang dihasilkan dapat dikenali polanya dengan jelas, serta penelitian ini hanya dilakukan pada satu kondisi waktu, dan apabila dilakukan pada kondisi yang lain maka akan diperlukan nilai parameter yang berbeda dengan keadaan lainnya. Kaushik Deb dan Ashraful Huq Suny(2014) melakukan penelitian tentang *shadow detection and removal* pada suatu citra dengan memperhatikan rata-rata intensitas citra tersebut. Metode ini digunakan untuk mendeteksi bayangan kemudian menghilangkan bayangan yang ada pada citra, pada algoritma ini hanya digunakan pada citra.

Proses *counting* dari penelitian sebelumnya tersebut bergantung pada waktu

pengambilan video. Oleh karena itu penulis pada penelitian ini melakukan *counting* kendaraan bergerak pada video pagi, siang, dan sore dengan menggunakan algoritma *Gaussian Mixture Model* (*GMM*) kemudian dari hasil GMM yang diperoleh akan dilakukan proses *updating foreground* untuk menganalisa bayangan yang terdeteksi sebagai objek pada proses GMM. Proses *robust* dilakukan dengan cara melakukan pengecekan terhadap nilai modus dan mean frame pada saat itu, kemudian nilai tersebut dijadikan nilai acuan untuk proses *updating foreground*. Proses tersebut dapat meminimalisir kesalahan perhitungan yang diakibatkan oleh beberapa kendaraan yang terkoneksi oleh bayangan yang menyebabkan objek hanya terdeteksi sebagai satu objek. Setelah bayangan dan *noise* telah diminimalisir, proses *tracking* dan *counting* dilakukan sehingga hasil *detecting* dan *counting* yang diperoleh akurat. Proses *updating foreground* ini belum dilakukan pada penelitian-penelitian sebelumnya, sehingga dari hasil algoritma yang digunakan mampu menyesuaikan pada video pagi, siang, dan sore.

1.2 Rumusan Masalah

Rumusan masalah dari penelitian ini adalah bagaimana mengembangkan algoritma *robust* untuk perhitungan pada rekaman video lalu lintas yang akurat dan adaptif terhadap waktu(pagi, siang, dan sore) pengambilan video.

1.3 Batasan Masalah

Permasalahan yang akan dibahas dalam penelitian ini dibatasi sebagai berikut:

1. Data yang digunakan berupa video yang diambil dari jembatan penyebrangan salah satu jalan di Surabaya pada saat pagi, siang, dan sore hari.

2. Kamera yang digunakan merupakan kamera standar dan dilakukan pada jalan satu arah.

3. Pada eksperimen, input informasi digital yang diproses dilakukan secara offline, serta tidak membedakan jenis kendaraan yang bergerak dan terbatas oleh kundisi cuaca seperti hujan.

4. Algoritma *robust* yang dimaksud disini adalah pengembangan dari algoritma GMM dengan cara melakukan pengecekan ulang terhadap piksel yang didefinisikan sebagai foreground, dan algoritma ini robust terhadap beberapa waktu pengambilan video(pagi, siang, dan sore).

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah memgembangkan algoritma *robust* untuk counting pada rekaman video lalu lintas yang akurat dan adaptif terhadap waktu(pagi, siang, dan sore) pengambilan video.

1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah algoritma yang telah diterapkan dapat memberikan informasi yang akurat tentang banyaknya jumlah kendaraan yang melintas sebagai bahan pertimbangan pengambilan keputusan dalam kebijakan lalu lintas pada video pagi, siang, dan sore hari, serta algoritma ini dapat diterapkan sebagai pelengkap algoritma lain.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

2.1 Penelitian Sebelumnya

Terdapat beberapa penelitian sebelumnya yang membahas masalah *counting* pada kendaraan bergerak. Pratistha Gupta dkk(2013) dalam penelitiannya yang berjudul *"Traffic Load Computation Using Corner Detection Technique in Matlab Simulink Model"*, melakukan counting pada kendaraan bergerak dengan menggunakan metode *edge detection*. Dalam penelitiannya diperoleh bahwa metode *edge detection* dapat mendeteksi tepi dari kendaraan dengan sangat sensitif, namun diperlukan metode deteksi lain untuk mengenali pola dari kendaraan yang telah dideteksi oleh metode *edge detection*. Penelitian selanjutnya berjudul *"Real Time Vehicle Detection and Counting Method for Unsupervised Traffic Video on Highways"* oleh Mrs. P. M. Daigavane dan Dr. P. R. Bajaj(2010). Penelitian ini membahas tentang *counting* menggunakan metode *morphological transformation* dan hasil yang diperoleh dari penelitian ini adalah tingkat keakurasian deteksi kendaraan bergerak sebesar 96%. Penelitian ini dilakukan pada video dengan intensitas yang serupa dan pengambilan video dilakukan pada waktu pagi hari.

Kemudian dalam penelitian Kaushik Deb dan Ashraful Huq Suny(2014) berjudul *"Shadow Detection and Removal Based on YCbCr Color Space"*, membahas tentang algoritma untuk mendeteksi kemudian menghilangkan bayangan pada image, bayangan dihilangkan dengan cara mengganti intensitas bayangan tersebut dengan intensitas objek maupun background. Pada penelitian ini, proses shadow removal hanya dilakukan pada citra dan pengambilan gambar dilakukan pada waktu tertentu. Kristian K.(2014) dkk dalam penelitiannya yang berjudul "Real Time Vehicle Detection and Tracking on Multiple Lanes" melakukan counting dengan menggunakan *foreground* dan *background image segmentation*. Dalam penelitian ini diperoleh tingkat keakurasian sebesar 95%. Sama seperti halnya penelitian lainya, video yang digunakan dalam penelitian ini, diambil pada suatu waktu tertentu.

2.2 Citra Digital

Citra menurut Kamus Besar Bahasa Indonesia (KBBI) memiliki makna rupa, gambar, atau gambaran. Sedangkan menurut kamus Webster citra adalah suatu

representasi, kemiripan, atau imitasi dari suatu objek atau benda. Citra terbagi menjadi dua yaitu citra diam dan citra bergerak. Citra diam adalah citra tunggal yang tidak bergerak. Sedangkan, citra bergerak adalah rangkaian citra diam yang ditampilkan secara beruntun sehingga memberi kesan pada mata kita sebagai gambar yang bergerak.

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

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

Secara matematis citra digital dapat dinyatakan sebagai fungsi dua variabel $f(x, y)$, dimana x dan y merupakan koordinat spasial horizontal dan vertikal. Fungsi f pada sembarang pasangan koordinat (x, y) merupakan nilai intensitas cahaya, representasi dari warna cahaya yang ada pada citra analog. Ketika (x, y) dan nilai intensitas dari f terbatas (*discrete quantities*), dan telah dilakukan proses digitalisasi spasial dan digitalisisai kuantitas maka citra itu disebut sebagai citra digital. Citra digital terdiri dari beberapa elemen terhingga yang disebut dengan piksel.

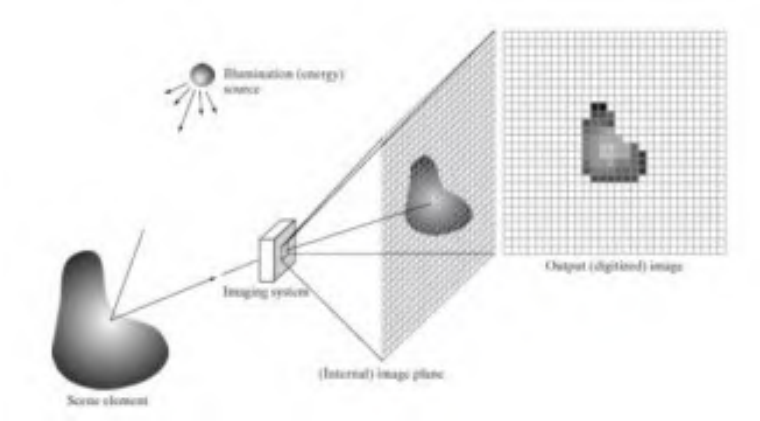
2.2.1 Digitalisasi Spasial(Sampling)

Sampling merupakan proses pengambilan informasi dari citra analog 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:

$$f = \begin{bmatrix} f(0,0) & \dots & f(0,N-1) \\ \vdots & \ddots & \vdots \\ f(M-1,0) & \dots & f(M-1,N-1) \end{bmatrix} \tag{2.1}$$

Sampling menyatakan besarnya blok-blok yang disusun dalam baris dan kolom. Dengan kata lain sampling pada citra menyatakan besar kecilnya ukuran piksel

pada citra. Untuk memudahkan implementasi, jumlah sampling umumnya diasumsikan sebagai perpangkatan dari dua, yaitu $N = 2^n$. Dengan n merupakan bilangan positif, dan N merupakan jumlah suatu baris/kolom. Partisi gambar menjadi ukuran tertentu menentukan resolusi spasial yang diperoleh. Semakin tinggi resolusi yang digunakan, semakin kecil ukuran piksel (atau semakin banyak jumlah pikselnya), sehingga semakin halus gambar yang diperoleh karena informasi yang hilang akibat pengelompokkan derajat keabuan pada proses sampling semakin kecil. Proses sampling citra analog ke citra digital ditampilkan pada Gambar 2.1 di bawah ini :



Gambar 2.1: Proses sampling citra dari citra analog ke citra digital(sumber: Gonzales dan Woods, 2002)

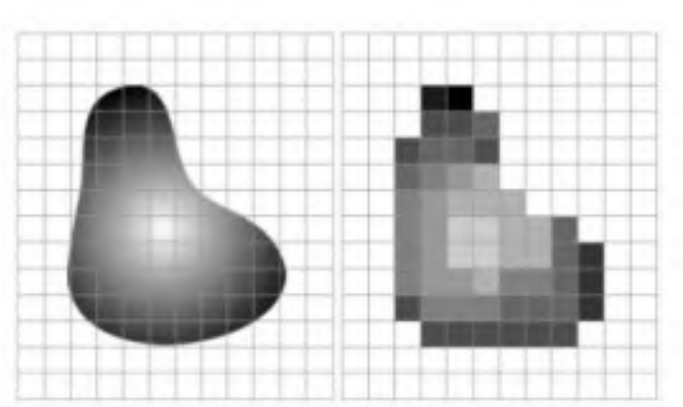
2.2.2 Digitalisasi Intensitas(Kuantisasi)

Setelah proses sampling pada citra maka proses selanjutnya adalah kuantisasi. Kuantisasi menyatakan besarnya nilai tingkat kecerahan yang dinyatakan dalam derajat keabuan(grayscale) sesuai dengan jumlah bit biner yang digunakan, dengan kata lain kuantisasi menyatakan jumlah warna yang ada pada citra. Proses kuantisasi membagi skala keabuan $(0, L)$ menjadi G buah level yang dinyatakan dengan dengan suatu nilai bilangan bulat(integer), umumnya G diambil perpangkatan dari dua.

Tabel 2.1: Kuantisasi Citra dengan Skala Keabuan yang Berbeda

Skala Keabuan	Rentang Nilai keabuan	Pixel Depth
2^1 (2 nilai)	0, 1	1 bit
2^2 (4 nilai)	0, ... , 3	2 bit
2^4 (16 nilai)	0, ... ,15	4 bit
2^8 (256 nilai)	0, ... , 255	8 bit

Pada tabel 2.1 ditunjukkan hasil kuantisasi citra dengan skala keabuan yang berbeda-beda. Jumlah bit yang dibutuhkan untuk merepresentasikan nilai keabuan suatu piksel disebut kedalaman piksel (*pixel depth*). Citra sering diasosiasikan dengan kedalaman pikselnya. Jadi, citra dengan kedalaman 8 bit disebut citra 8-bit(atau citra 256 warna, G=256=2^8). Semakin banyak jumlah derajat keabuan, maka semakin bagus gambar yang diperoleh.



Gambar 2.2: Proses sampling dan kuantisasi. (kiri) Gambar citra analog yang ditempatkan pada sensor array. (kanan) Gambar citra setelah proses sampling dan kuantisasi, tiap piksel pada citra memiliki derajat keabuan masing-masing.(sumber: Gonzales dan Woods, 2002)

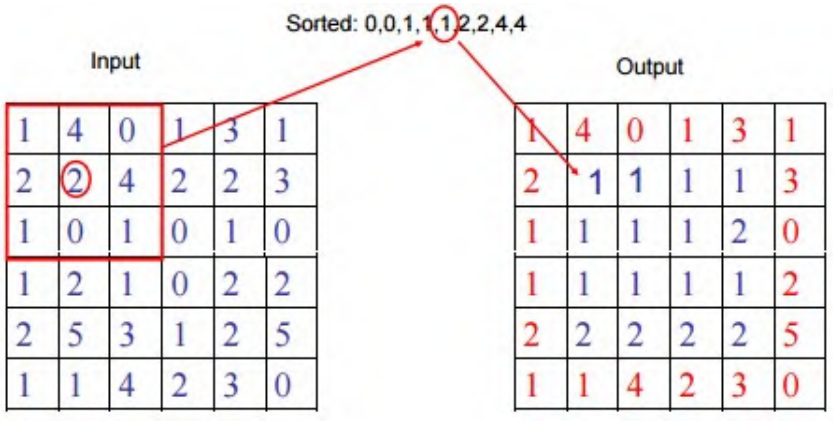
2.2.3 Video Digital

Video adalah teknologi untuk menangkap, merekam, memproses, menyimpan, dan merekonstruksi suatu urutan dari beberapa citra. Video digital merupakan jenis sistem video recording yang bekerja menggunakan sistem digital dibandingkan dengan analog dalam hal representasi video. Sedangkan menurut Alan C. Bovik, video digital merupakan hasil sampling dan kuantisasi dari video analog (Bovik, 2000). Secara mendasar, tidak ada perbedaan proses sampling dan kuantisasi antara citra digital dan video digital. Salah satu alat yang dapat digunakan untuk menghasilkan video digital adalah camrecorder, yang digunakan untuk merekam gambar-gambar video dan audio.

Sebenarnya video tidak memiliki sistem pengambilan gambar yang kontinu, hal ini dikarenakan video merupakan kumpulan beberapa frame yang ditangkap dalam satuan detik. Banyaknya frame yang tertangkap dalam satu detik umumnya dinyatakan dalam bentuk *frame per second* (*fps*), dimana tiap frame dari suatu video adalah sebuah image. Semakin besar fps, maka video akan terlihat semakin halus. (Solomon dan Breckon, 2013)

2.3 Smoothing

Smoothing atau *filter low pass* merupakan suatu metode yang bertujuan untuk mengurangi *noise*. Noise tersebut muncul sebagai akibat dari hasil pensamplingan yang tidak bagus. Piksel komponen yang mempunyai noise pada umumnya memiliki frekuensi yang tinggi. Komponen citra yang berfrekuensi rendah akan diloloskan dan komponen yang memiliki frekuensi tinggi akan dilakukan proses smoothing. Pada penelitian ini proses *smoothing* menggunakan median filter. Prinsip kerja dari median filter adalah dengan cara mengganti nilai intensitas dari tiap piksel dengan nilai median dari piksel tetangganya. Pola dari piksel tetangga telah ditentukan terlebih dahulu sebelumnya. Pola dari piksel tetangga disebut dengan *window*, yang bergeser tiap piksel sampai menjangkau image secara keseluruhan. Nilai median dihitung dengan melakukan *sorting* pada bagian *window* kemudian mengganti pusat dari *window* tersebut dengan nilai median yang diperoleh. Proses tersebut dapat digambarkan pada gambar di bawah ini.



Gambar 2.3: Proses Median Filter

2.4 Background Subtraction

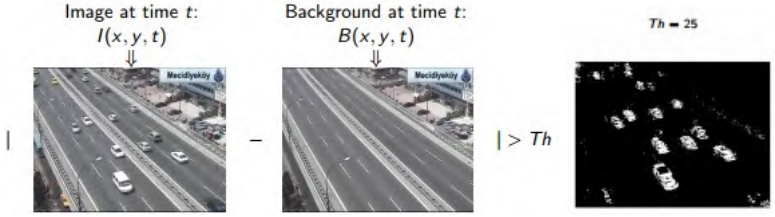
Background Subtraction (*BS*) merupakan suatu metode untuk mendapatkan *foregound object* (objek bergerak) dari suatu video. Dengan tujuan proses pendeteksian objek akan menjadi lebih sederhana, karena objek yang bergerak telah dipisahkan dari *background*. Konsep dasar dari *background subtraction* adalah dengan melakukan pengurangan frame pada waktu t dengan *background* dari video, kemudian hasil pengurangan tersebut dibandingkan suatu nilai ambang batas (*threshold* tertentu), proses ini disebut dengan proses *differencing*. Secara matematis proses tersebut dapat dinyatakan dalam bentuk

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

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

dengan $(I(x,y,t))$ merupakan frame pada waktu (t), $(B(x,y,t))$ merupakan *background* pada waktu (t), $(d(x,y,t))$ merupakan hasil pengurangan *background* terhadap *foreground* (τ) merupakan *threshold*, dan $(f(x,y,t))$ adalah *foreground image*. Contoh proses ini dapat dilihat pada Gambar 2.4.

Terdapat berbagai macam metode untuk menginisialisasi *background* model. Salah satunya adalah metode *Gaussian Mixture Model*.



Gambar 2.4: Proses *Background Subtraction* dengan metode frame *differencing* untuk mendapatkan citra *foreground*.

2.5 Metode Gaussian Mixture Model

Gaussian Mixture Model (*GMM*) merupakan metode yang tepat untuk berbagai kondisi yang terdapat pada citra, seperti background citra yang selalu statis, multimodal, maupun yang mengandung *noise* (gangguan atau objek yang tidak diinginkan terdapat pada citra). *GMM* merupakan tipe density model yang terdiri dari komponen fungsi-fungsi Gaussian. Komponen fungsi tersebut terdiri dari weight yang berbeda untuk menghasilkan multi model density. Model-model GMM terbentuk dari data warna piksel berdasarkan waktu. Hasil model

tersebut akan menjadi 2 bagian, model yang mencerminkan background dan model non-background. Jumlah model *GMM* yang digunakan mempengaruhi jumlah model background. Semakin besar jumlah model *GMM* yang dipakai semakin banyak model background yang dimiliki suatu piksel. GMM memproses tiap piksel pada citra, baik citra tersebut berupa skalar (citra grayscale) maupun vektor (citra berwarna/ RGB). Untuk sebarang waktu t, citra tersebut dimodelkan sebagai berikut(Staufer dan Grimson, 1999):

$$\{X_1, ..., X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\} \tag{2.4}$$

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

Untuk setiap piksel, dimodelkan dengan $\{X_1, ..., X_t\}$ mixture K distribusi Gaussian. Dengan K merupakan banyak model gaussian yang dibangun dan bernilai antara 3 hingga 5. Secara singkat, terdapat dua proses pada GMM; yaitu tahap pencocokan input terhadap distribusi dan tahap pemilihan distribusi yang mencerminkan background.

2.5.1 Tahap pencocokan input terhadap distribusi

Pada tahap ini akan dilakukan update parameter GMM yang akan digunakan untuk memproses input selanjutnya. Berikut model matematis GMM dari tahap pencocokan input hingga update parameter.

1. Tahap pencocokan input terhadap distribusi Suatu piksel masuk dalam distribusi jika nilai piksel masuk dalam jarak 2.5 dikalikan standar deviasi dari sebuah distribusi

   $$\mu_i - 2.5 * \sigma_i < X_t < \mu_i + 2.5 * \sigma_i \tag{2.5}$$

   μ_i adalah nilai mean citra RGB dari Gaussian ke-i, σ_i adalah standar deviasi dari Gaussian ke-i, dan X_t adalah vektor dari citra RGB. Dengan X_t merupakan vektor dari warna piksel RGB, sehingga jika channel yang digunakan hanya satu (Grayscale) maka X_t bernilai skalar.

2. Tahap *updating* komponen GMM

   Pada tahap ini terdapat beberapa parameter GMM yang akan di-update nilainya adalah $\omega_{i,t}$ (*weight*), $\mu_{i,t}$ (*mean*), $\sigma_{i,t}$ (*standar deviasi*). Update *weight* dilakukan setiap saat

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

α adalah *learning rate* dan $M_{i,t}$ bernilai 1 untuk model yang cocok dan 0 untuk lainnya. Update mean dilakukan jika dan hanya jika ada model yang cocok:

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

Update standar deviasi dilakukan jika dan hanya jika ada model yang cocok:

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

dengan,

$$\rho = \frac{\alpha}{\omega_{i,t}} \tag{2.9}$$

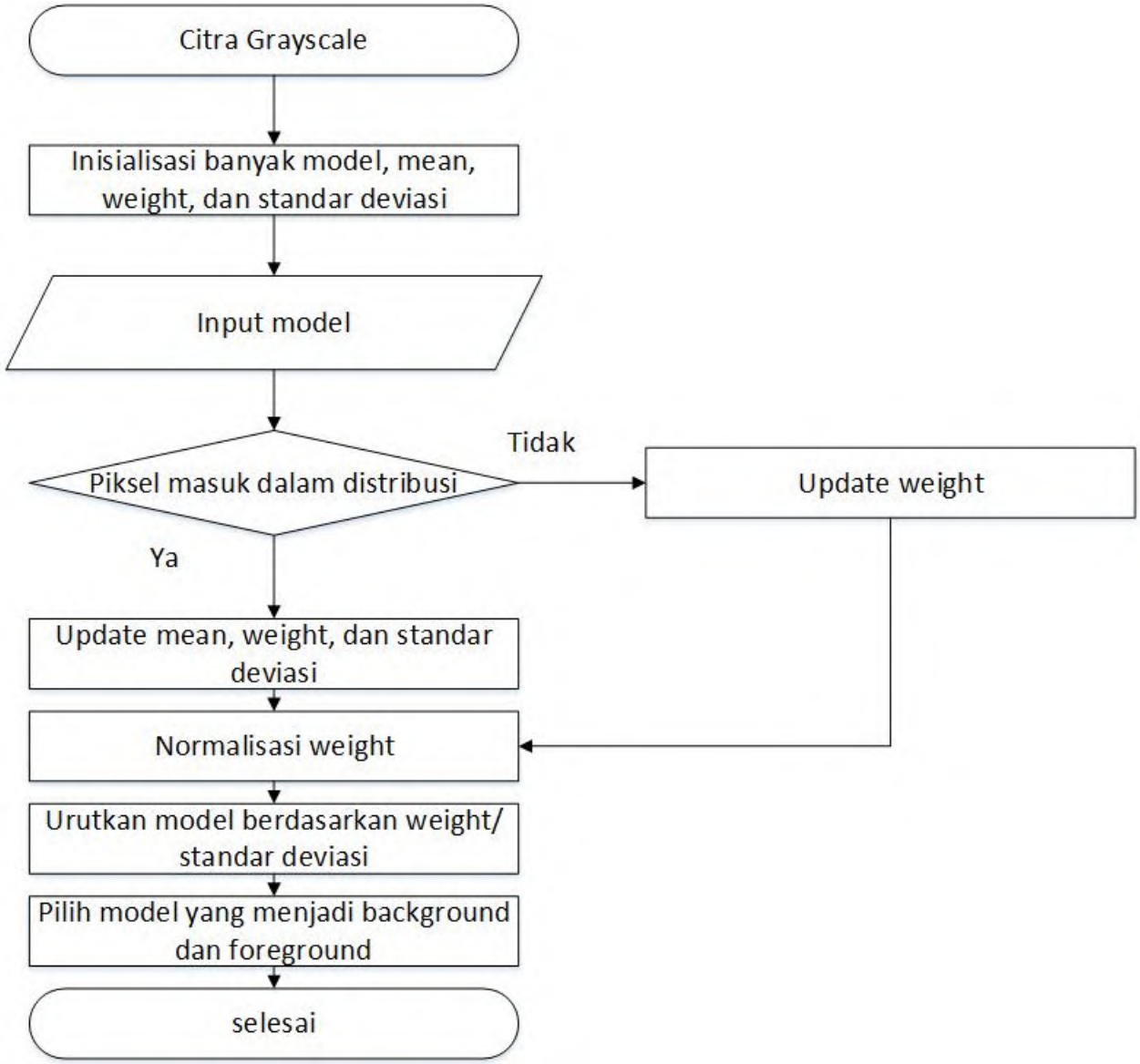
2.5.2 Tahap pemilihan distribusi yang mencerminkan background

Pada tahap ini akan dipilih model yang menyerupai background. Model akan diurutkan berdasarkan $\frac{\omega}{\sigma^2}$ sehingga distribusi yang menyerupai background akan berada di urutan atas sedangkan yang tidak menyerupai akan berada di urutan bawah yang nantinya digantikan oleh distribusi lain yang cocok. Model matematis untuk memilih B distribusi pertama yang dijadikan background adalah

$$B = argmin_b(\sum_{k=1}^{b} \omega_k > \tau) \tag{2.10}$$

dengan nilai *default* τ berkisar antara 0.25 sampai dengan 0.75.

Berikut ini disajikan diagram alir dari proses GMM



Gambar 2.5: Diagram alir proses GMM.

2.6 *Shadow Removal*

Bayangan (*shadow*) merupakan bidang yang terbentuk akibat hilangnya cahaya karena mengenai suatu objek yang tidak dapat ditembus oleh cahaya tersebut. Terdapat dua jenis bayangan yaitu *self-shadow* dan *cast-shadow*.

Bayangan berpengaruh dalam perhitungan dan pengenalan jenis kendaraan, oleh karena itu diperlukan proses untuk menghilangkan bayangan (*shadow removal*) sebelum dilakukan perhitungan.

Shadow removal adalah proses dimana bayangan yang terdeteksi pada foreground akan dihilangkan untuk memperjelas deteksi objek. Salah satu contoh proses shadow removal adalah dengan cara memproses lebih lanjut output dari *GMM* yang berupa citra biner dengan dua jenis intensitas yaitu 0 untuk objek diam dan 1 untuk objek bergerak. Pada awalnya diberikan nilai 1 pada bayangan hal ini dikarenakan bayangan juga bergerak menyerupai objeknya. Untuk itu ditentukan terlebih dahulu nilai mean dan modus dari intensitas image pada frame saat itu, kemudian dicek apakah intensitas nilai image biner hasil *GMM* adalah 1, jika iya maka piksel tersebut akan dilakukan proses seleksi. Seleksi yang berbeda akan dilakukan pada piksel foreground tersebut berdasarkan nilai modus dan mean dari frame pada saat itu. Untuk lebih jelasnya, proses ini akan dijelaskan pada bab 4.

2.7 Deteksi Objek Bergerak

Pada penelitian ini, deteksi objek bergerak yang digunakan adalah dengan memeriksa semua piksel tetangga dengan radius tertentu di dalam *ROI* (*Region of Interest*) yang sebelumnya telah ditentukan. Berikut ini dijelaskan prosedur dari algoritma deteksi objek bergerak.

1. Input citra hasil *updating foreground*
   Diinputkan citra biner hasil proses *updating foreground*. Pada proses ini, telah dilakukan proses smoothing pada input citra.

2. Inisialisasi matriks temp
   Diinisialisasikan matriks temp dengan semua elemen matriks adalah nol dan memiliki ukuran yang sama dengan input citra. Fungsi dari matriks temp ini adalah sebagai penanda apakah telah dilakukan pengecekan tetangga pada piksel tersebut, apabila telah dilakukan pengecekan maka nilai elemen matriks temp pada piksel tersebut diubah menjadi 1. Pengecekan hanya dilakukan pada piksel citra updating foreground yang bernilai 1.

3. Pencarian tetangga
   Proses ini dilakukan pada piksel dari citra updating foreground yang bernilai 1 dan nilai elemen dari matriks temp pada koordinat piksel tersebut adalah nol(belum dilakukan pengecekan). Apabila dua syarat tersebut terpenuhi maka dilakukan proses pencarian tetangga pada piksel tersebut, proses tersebut dilakukan pada 4 tetangganya. Apabila piksel tersebut memiliki minimal 1 tetangga maka piksel tersebut merupakan satu objek yang sama. Untuk lebih jelasnya proses ini akan dibahas lebih detail pada bab 4.

BAB III
METODE PENELITIAN

Bab ini menjelaskan tahapan penelitian yang dilakukan untuk menerapkan algoritma *robust* pada *counting* kendaraan bergerak. Adapun tempat, tahapan penelitian dan jadwal pelaksanaan yang digunakan adalah sebagai berikut.

3.1 Objek Penelitian

Pada penelitian ini objek penelitian yang digunakan adalah video kendaraan bergerak dalam suatu rekaman suatu lalu lintas yang diambil pada pagi, siang, dan sore hari dengan menggunakan kamera.

3.2 Peralatan

Peralatan yang digunakan adalah Matlab GUI sebagai software utama untuk melakukan counting pada kendaraan bergerak, dan sebuah kamera untuk proses merekam video lalu lintas.

3.3 Tahapan Penelitian

Tahapan penelitian yang dilakukan dalam penelitian ini adalah sebagai berikut.

1. Studi Literatur

   Mengkaji teori mengenai pengolahan citra, seperti *background subtraction*, *shadow removal*, *object detection*, yang akan diimplementasikan pada proses counting kendaraan bergerak. Kajian ini dilakukan dengan membaca jurnal yang berkaitan dengan topik yang diajukan.

2. Akuisisi Data

   Pada tahap ini akan dilakukan pengumpulan data video. Video yang diperoleh merupakan hasil rekaman langsung oleh peneliti dari beberapa jembatan di Surabaya pada saat pagi, siang, dan sore hari diantaranya jembatan penyebrangan Kedung Cowek, jalan Pemuda, dan jalan Diponegoro.

3. Analisa Algoritma Robust

   Pada tahap ini dilakukan analisa metode dari data berupa rekaman video. Prosedur algoritma robust untuk *counting* kendaraan bergerak berbasis pengolahan citra adalah sebagai berikut.

(a) Untuk eksperimen input data berupa rekaman video *offline*

(b) Setelah itu dari input video yang ada, diekstrak menjadi *frame* kemudian intensitas *frame* tersebut diubah menjadi *grayscale*.

(c) Penentuan area *ROI* (*Region of Interest*).

(d) *Smoothing* dilakukan pada tiap *frame* untuk memperhalus citra dengan tujuan untuk meminimalisir noise.

(e) Dilakukan proses *background subtraction* menggunakan metode *GMM* dimana output dari *GMM* merupakan citra dengan 2 nilai di tiap pikselnya, yaitu 0 untuk *background*, dan 1 untuk *foreground*. Pada proses ini bayangan masih memiliki intensitas bernilai 1.

(f) Proses *Shadow removal* yaitu tahap dimana dilakukan pengecekan pada setiap piksel yang mempunyai nilai 1 (objek dan *shadow*) dari citra output GMM. Proses tersebut dilakukan dengan membandingkan piksel foreground terhadap intensitas mean dan modus pada saat *frame* itu. Seleksi yang berbeda akan dilakukan pada piksel foreground tersebut berdasarkan nilai modus dan mean dari *frame* pada saat itu. Apabila intensitas grayscale pikssel tersebut masuk dalam suatu interval yang bergantung pada mean dan modus, maka piksel tersebut merupakan background.

(g) Proses *smoothing* untuk meminimalisir noise dari citra biner hasil *shadow removal*.

(h) Proses *tracking* dengan cara memeriksa apakah objek pada *frame* sebelumnya dan *frame* sekarang adalah sama atau tidak. Selain itu diberikan kondisi apabila banyaknya piksel dalam suatu objek telalu sedikit maka objek tersebut tidak dihiraukan, dan apabila ada objek di dalam objek lain maka objek tersebut juga tidak dihiraukan.

(i) Proses *counting* dilakukan pada objek yang telah terdeteksi pada ROI.

4. Uji Coba dan Pembahasan

   Pada tahap ini dilakukan percobaan dengan input video yang berbeda waktu pengambilannya. Dari uji coba tersebut akan dilihat tingkat kesalahan dari penghitungan jumlah kendaraan.

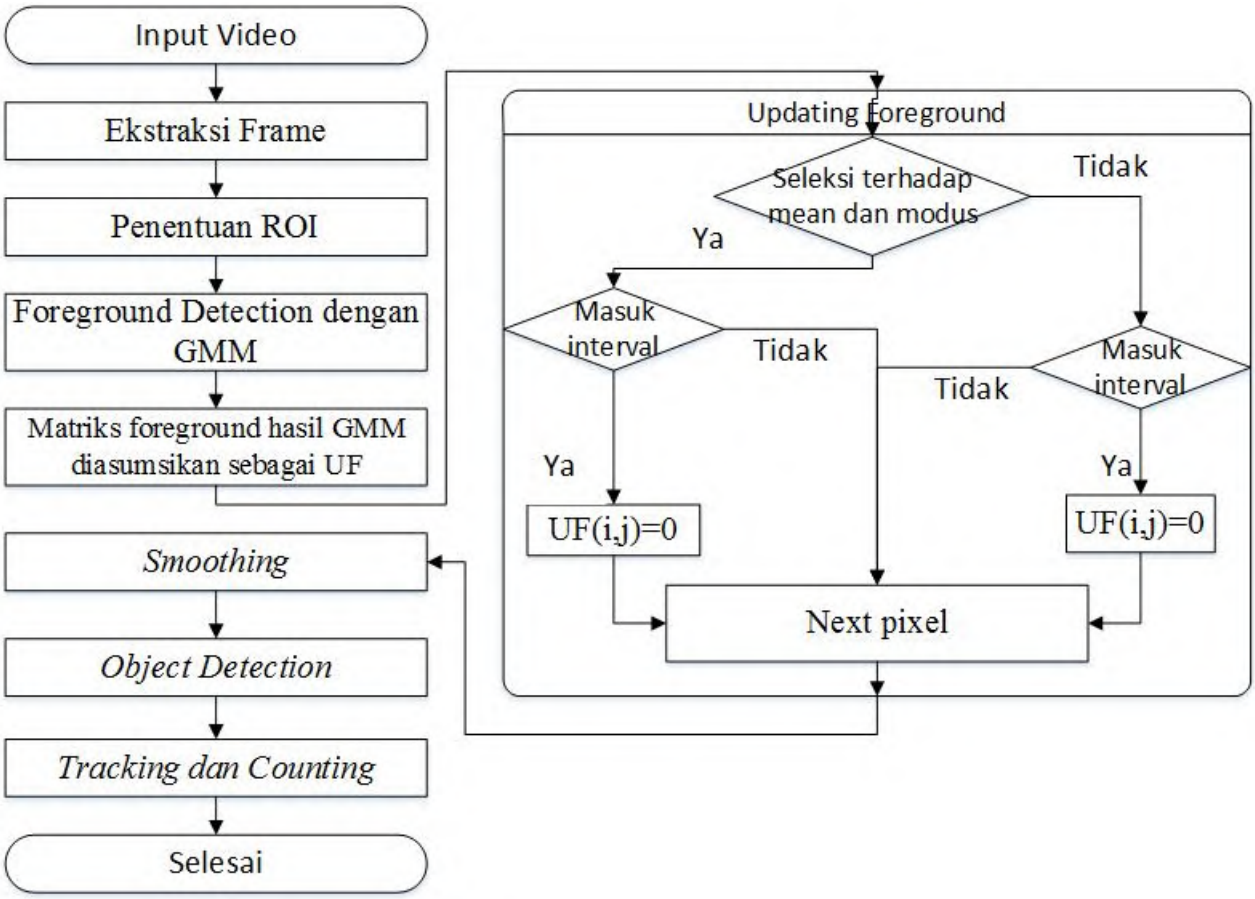
5. Publikasi

   Pada tahap ini, penelitian yang telah dilakukan dipublikasikan serta dipresentasikan pada suatu seminar internasional.

6. Penulisan Tesis

3.4 Diagram Penelitian

Diagram alir yang digunakan pada metode penelitian ini adalah:



Gambar 3.1: Diagram alir algoritma yang digunakan

BAB IV
PERANCANGAN DAN PENERAPAN ALGORITMA

Pada bab ini dijelaskan mengenai analisis perancangan algoritma yang terdiri dari penjelasan tentang proses akuisisi video, proses pengambilan data masukan, pengolahan data masukan dengan algoritma *Background Subtraction* dan metode *Gaussian Mixture Model*, serta penjelasan mengenai *shadow removal* dalam proses *updating foreground*.

4.1 Perancangan Algoritma

Untuk dapat menghitung kendaraan bergerak berbasis video digital, pada saat eksperimen data video yang digunakan berupa rekaman video digital offline kendaraan bergerak di jalan. Data masukan tersebut nantinya akan diolah menggunakan algoritma *Background Subtraction* dengan metode *Gaussian Mixture Model*. Kemudian citra biner hasil dari metode *Gaussian Mixture Model* akan diproses lebih lanjut untuk meminimalisir eror pada *counting* yang disebabkan oleh bayangan yang muncul disekitar objek.

Berikut ini adalah proses dari algoritma yang digunakan dalam penelitian ini

1. *Scanning*

   Scanning merupakan proses partisi video menjadi beberapa citra yang disebut dengan frame. Banyaknya frame yang terbentuk sesuai dengan kapasitas fps kamera yang digunakan untuk merekam video tersebut.

2. Pemilihan *ROI*

   Proses ini dilakukan setelah didapatkan frame dari proses *scanning*, ROI dipilih sebagai fokus area yang dilakukan perhitungan kendaraan yang bergerak.

3. *Proses Background Subtraction* menggunakan *GMM*

   Proses GMM dilakukan untuk memperoleh *background* dan *foreground*, dimana hasil dari proses ini adalah citra biner dengan piksel berintensitas 1 adalah *foreground* dan piksel yang memiliki intensitas 0 adalah *background*. Pada proses ini, bayangan dikenali objek bergerak sesuai dengan objek dari bayangan tersebut, hal ini dapat menimbulkan kesalahan perhitungan seperti 2 objek yang berdekatan dikenali sebagai 1 objek.

4. Proses *Updating Foreground*

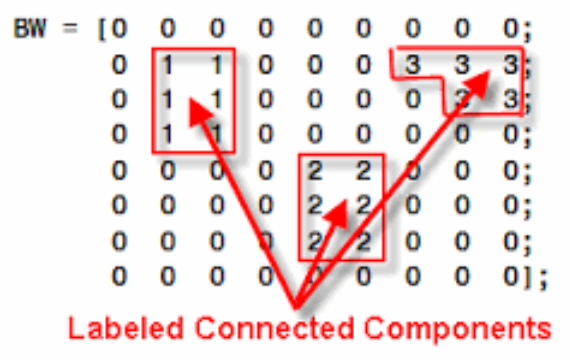
   Proses ini bertujuan untuk meminimalisir bayangan yang tedeteksi sebagai *foreground*. Awalnya dilakukan pengecekan pada citra *foreground*(piksel dengan intensitas 1) hasil dari proses GMM, pengecekan dilakukan dengan perilaku tertentu sesuai dengan nilai mean dan modus frame pada saat itu, apabila kondisi terpenuhi dan piksel itu masuk dalam interval itu maka piksel tersebut diidentifikasikan sebagai bayangan dan diberi intensitas piksel 0, apabila tidak maka piksel itu tetap berintensitas 1.

5. *Filtering*

   Proses ini bertujuan untuk mengurangi noise yang timbul akibat terdapat objek selain kendaraan yang bergerak, seperti daun yang tertiup angin atau gangguan-gangguan lainnya. Pada proses filtering ini metode yang digunakan adalah metode median filter. Cara kerja dari median filter adalah dengan menggantikan nilai tiap piksel dengan nilai median dari piksel-piksel tetangganya.

6. Proses Deteksi Objek

   Setelah didapat citra dengan *noise* yang terminimalisir, dilakukan proses deteksi objek dengan cara melakukan pengecekan terhadap masing-masing piksel pada citra biner yang telah diperoleh. Pengecekan dilakukan pada tetangga tiap-tiap piksel dengan 4 nilai ketetanggan. Piksel kemudian diberi label yang bernilai sama dengan tetangganya apabila memiliki tetangga paling tidak 1 serta diberi tanda persegi pada piksel terluar dari objek tersebut. Proses tersebut dapat ditampilkan pada Gambar 4.1 berikut.

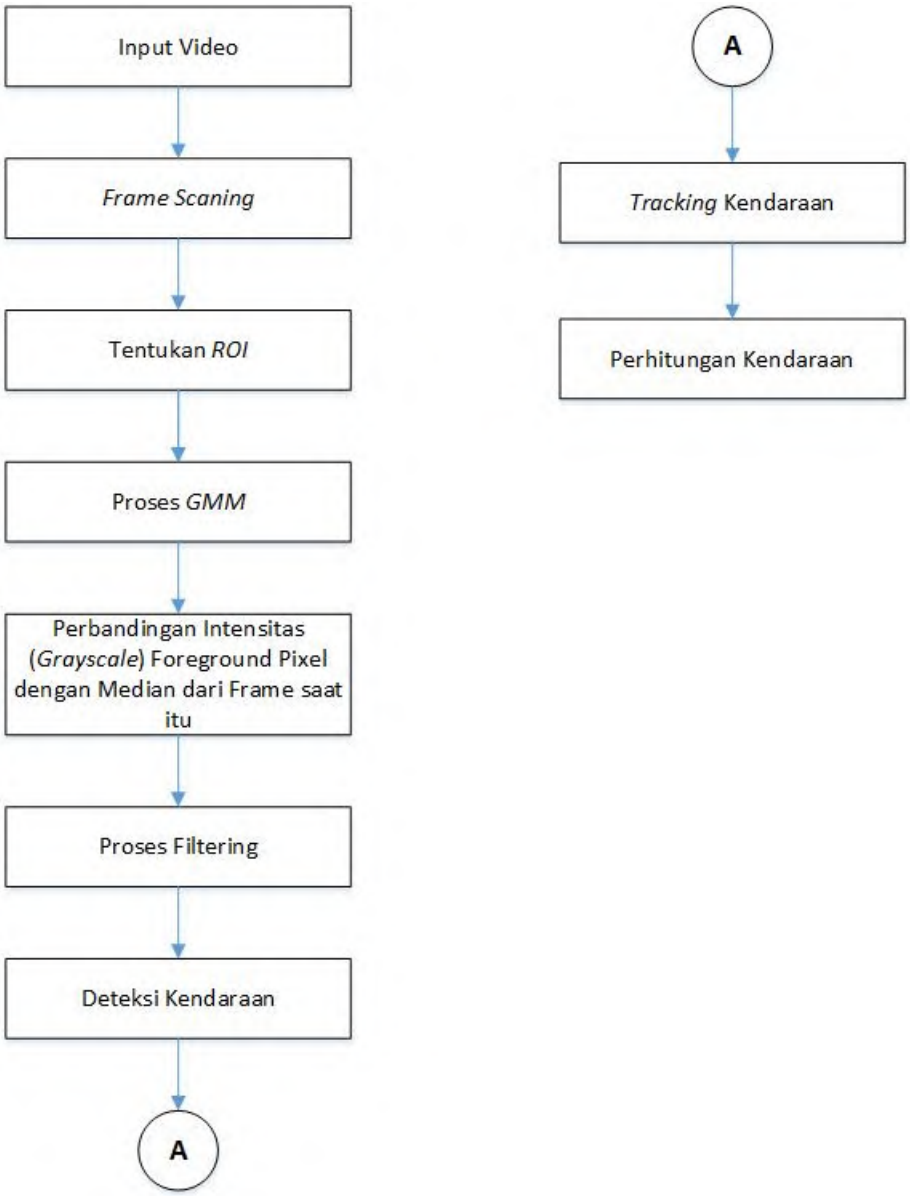


Gambar 4.1: Proses Deteksi Objek dengan Batas Ketetanggaan 1.

7. Proses *Tracking*

   Proses tracking bertujuan untuk mengetahui apakah objek yang terdeteksi

pada frame ke-$(t-1)$ dengan frame ke-(t) merupakan objek yang sama. Proses ini dilakukan dengan cara mengecek apakah centroid pada frame ke dari objek tersebut kurang dari batas batas dari frame sebelumnya, apabila kurang maka objek tersebut dianggap sama. Setelah itu dilakukan proses *counting* dimana proses *counting* ini dilakukan apabila objek telah melewati *ROI* serta objek tersebut bukan objek yang sama pada frame sebelumnya. Dari penjelasan tersebut, proses-proses tersebut dapat dinyatakan dalam bentuk diagram alir seperti pada Gambar 4.2 berikut.



Gambar 4.2: Diagram Alir Algoritma yang diterapkan.

4.2 Perancangan Data

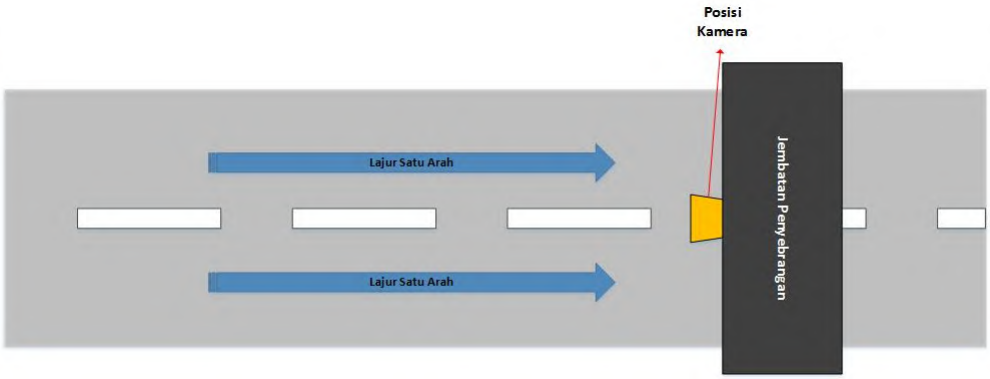
Data yang digunakan untuk menghitung jumlah kendaraan bergerak di suatu jalan yaitu data masukan, dan data keluaran. Data masukan merupakan video rekaman offline di beberapa ruas jalan yang dilakukan secara mandiri oleh peneliti dan data diambil pada waktu yang berbeda yakni pagi, siang, dan sore. Sedangkan data keluaran adalah informasi mengenai jumlah penghitungan kendaraan bergerak yang terdeteksi oleh sistem.

4.2.1 Akuisisi Video

Pada eksperimen, data video yang digunakan adalah hasil rekaman video yang diambil dari jembatan penyebrangan beberapa jembatan penyebrangan di Surabaya pada waktu pagi, siang, dan sore hari. Posisi kamera berada pada pegangan jembatan penyebrangan seperti gambar di bawah ini.

Gambar 4.3: Posisi Kamera saat Pengambilan Data.

Berikut adalah gambar layout lapangan untuk akuisisi video arus lalu lintas.



Gambar 4.4: Layout Lapangan untuk Akuisisi Video Arus Lalu Lintas.

4.2.2 Data Proses

Data proses adalah data yang digunakan di dalam proses pengolahan data masukan. Setiap data masukan yang ada membutuhkan data proses sesuai dengan tahapan algoritma dan metode yang telah disusun. Selanjutnya data proses tersebut digunakan sebagai parameter untuk memproses tahapan selanjutnya. Berikut tabel yang menjelaskan data proses tersebut beserta kegunaannya dalam program.

Tabel 4.1: Tabel data proses

No	Tahapan	Input	Output
1	Ekstraksi Video	Video	Frame video
2	Penentuan *ROI*	Frame video	Citra ROI
3	Proses *GMM*	Citra ROI	Citra *Foreground*
4	Update Citra *GMM*	Citra *Foreground*	*Updated Foreground*
5	Filtering	*Updated Foreground*	*Filtered Foreground*
6	Deteksi Objek	*Filtered Foreground*	Batas koordinat objek
7	Tracking dan Counting	Batas koordinat objek	Jumlah Kendaraan

4.2.3 Data Keluaran

Data keluaran dari algoritma di atas adalah jumlah kendaraan yang melintas. Selain itu dihasilkan pula data keluaran seperti *citra foreground*, citra *updated foreground*, citra *filtered foreground*, citra hasil deteksi objek.

4.3 Perancangan Sistem

Pada proses ini akan dijelaskan lebih detail lagi mengenai beberapa proses dari sistem yang akan dibangun seperti yang dapat dilihat pada Gambar 4.2. Proses yang akan dijelaskan adalah proses untuk memperoleh *background* dengan menggunakan *background subtraction (GMM)*, deteksi objek, serta proses *tracking*.

4.3.1 Perancangan Proses Background Subtraction Menggunakan GMM

Proses ini dilakukan untuk memperoleh background, dan dalam proses ini metode yang digunakan adalah GMM. Sebelumnya pada bab 2 (subbab 2.6) telah dijelaskan mengenai formula pada proses GMM. Pada subbab ini akan diberikan contoh bagaimana langkah-langkah dari algoritma GMM tersebut. Misal matriks dari citra grayscale berintensitas sebagai berikut.

$$A = \begin{bmatrix} 100 & 120 & 150 \\ 75 & 90 & 60 \\ 50 & 70 & 80 \end{bmatrix} \tag{4.1}$$

Setelah itu dilakukan inisialisasi dari nilai mean, bobot, dan standar deviasi. Inisialisasi itu dilakukan sebanyak K model Gaussian yang dibentuk, dengan memisalkan nilai awal seperti dibawah ini.

$$Mean = \begin{bmatrix} Mean(1,1,k) & \dots & Mean(1,N,k) \\ \vdots & \ddots & \vdots \\ Mean(M,1,k) & \dots & Mean(M,N,k) \end{bmatrix} \tag{4.2}$$

dengan $i = 1, ..., M$, $j = 1, ..., N$, dan $k = 1, ..., K$, dengan i dan j menyatakan koordinat piksel dan k menyatakan gausian ke-K dari banyak distribusi gaussian yang diinginkan. Dengan nilai awal mean bernilai random dengan interval $0 \leq Mean(1,1,k) \leq 255$. Misal pada contoh ini nilai dari masing-masing mean adalah sebagai berikut.

$Mean(i,j,1) =$

102	89	95
97	100	120
130	105	96

$Mean(i,j,2) =$

150	110	80
51	90	115
140	85	100

$Mean(i,j,3) =$

95	100	120
120	95	80
90	100	130

Gambar 4.5: Contoh inisialisasi mean.

Sedangkan untuk inisialisasi dari bobot diberikan dengan nilai *uniform* ($\frac{1}{K}$) atau dapat dinyatakan dalam bentuk sebagai berikut.

$$Bobot = \begin{bmatrix} \frac{1}{K} & \frac{1}{K} & \frac{1}{K} \\ \frac{1}{K} & \frac{1}{K} & \frac{1}{K} \\ \frac{1}{K} & \frac{1}{K} & \frac{1}{K} \end{bmatrix} \tag{4.3}$$

Dan untuk inisialisasi dari standar deviasi, juga diberikan nilai *uniform* (σ_0), dengan (σ_0) nilai awal standar deviasi (pada paper Stauffer dan Grimson(1999) nilai σ_0 yang digunakan adalah 6) atau dapat dinyatakan dalam bentuk di bawah ini.

$$Standardeviasi = \begin{bmatrix} \sigma_0 & \sigma_0 & \sigma_0 \\ \sigma_0 & \sigma_0 & \sigma_0 \\ \sigma_0 & \sigma_0 & \sigma_0 \end{bmatrix} \tag{4.4}$$

Proses GMM ini akan dilakukan pada masing-masing piksel dalam masing-masing distribusi. Sebelum dilakukan proses selanjutnya, dari persamaan (2.5)

diperoleh bentuk

$$|A(i,j) - Mean(i,j,k)| < 2.5 * Standardeviasi(i,j,k) \tag{4.5}$$

dengan $|A(i,j) - Mean(i,j,k)|$ merupakan selisih intensitas piksel citra grayscale terhadap mean, oleh karena itu akan ditentukan terlebih dahulu nilai tersebut. Nilai ruas kiri dari persamaan (4.5) diperoleh dari nilai absolut selisih dari matriks pada persamaan (4.1) terhadap matriks yang terdapat pada Gambar 4.5. Hasil dari proses tersebut dapat dilihat pada Gambar 4.6 di bawah ini.

$|A(i,j) - Mean(i,j,1)| =$

2	31	55
22	10	60
80	35	16

$|A(i,j) - Mean(i,j,2)| =$

50	10	70
24	0	55
90	15	20

$|A(i,j) - Mean(i,j,3)| =$

5	20	30
45	5	20
40	30	50

Gambar 4.6: Hasil Selisih mean terhadap piksel.

Sebelum dilakukan pengecekan seperti pada persamaan (4.5), diketahui nilai ruas kanan dari persamaan (4.5) adalah sebagai berikut.

$$2.5 * Standardeviasi = \begin{bmatrix} 15 & 15 & 15 \\ 15 & 15 & 15 \\ 15 & 15 & 15 \end{bmatrix} \tag{4.6}$$

Setelah itu, seperti yang dapat dilihat pada persamaan 4.5 dilakukan pengecekan yang dimulai dari koordinat (1,1) dari matriks yang terdapat pada Gambar 4.6 apakah piksel dari model background tersebut masuk ke dalam distribusi. Operasi tersebut dilakukan pada masing-masing piksel terhadap tiap-tiap model yang dibentuk (sebanyak K). Hasil dari operasi tersebut dapat dilihat pada Gambar 4.8 dibawah ini.

$|A(i,j) - Mean(i,j,1)| =$

2	31	55
22	10	60
80	35	16

Piksel masuk dalam distribusi

$|A(i,j) - Mean(i,j,2)| =$

50	10	70
24	0	55
90	15	20

Piksel tidak masuk dalam distribusi

$|A(i,j) - Mean(i,j,3)| =$

5	20	30
6	5	20
40	30	50

Piksel masuk dalam distribusi

Gambar 4.7: Proses pengecekan piksel.

Kemudian dilakukan proses *update* komponen, sesuai dengan hasil proses pengecekan yang dilakukan sebelumnya. Apabila piksel masuk dalam distribusi, maka update dilakukan pada bobot, mean dan dan standar deviasi. *Update* nilai komponen tersebut dilakukan sesuai dengan yang dijelaskan pada subbab 2.6.1. Hasil update tersebut dapat dilihat pada gambar dibawah ini.

$|A(i,j) - Mean(i,j,1)| =$

2	31	55
22	10	60
80	35	16

Piksel masuk dalam distribusi

Hasil proses update $Bobot(i,j,1) =$

0.3397	0.333	0.333
0.333	0.333	0.333
0.333	0.333	0.333

Hasil proses update $Mean(i,j,1) =$

101.9411	89	95
97	100	120
130	105	96

Hasil proses update $Standardeviasi(i,j,1) =$

5.9210	6	6
6	6	6
6	6	6

Gambar 4.8: Hasil proses update komponen GMM pada model pertama.

$|A(i,j) - Mean(i,j,2)| =$

50	10	70
24	0	55
90	15	20

Piksel tidak masuk dalam distribusi

Hasil proses update $Bobot(i,j,2) =$

0.3297	0.333	0.333
0.333	0.333	0.333
0.333	0.333	0.333

Gambar 4.9: Hasil proses update komponen GMM pada model kedua.

$|A(i,j) - Mean(i,j,3)| =$

5	20	30
6	5	20
40	30	50

Piksel masuk dalam distribusi

Hasil proses update $Bobot(i,j,3) =$

0.3397	0.333	0.333
0.333	0.333	0.333
0.333	0.333	0.333

Hasil proses update $Mean(i,j,3) =$

95.1472	100	120
120	95	80
90	100	130

Hasil proses update $Standardeviasi(i,j,3) =$

5.9730	6	6
6	6	6
6	6	6

Gambar 4.10: Hasil proses update komponen GMM pada model ketiga.

Pada penjelasan langkah ini terlihat pada Gambar 4.9, komponen yang diupdate hanya bobot, hal ini dikarenakan piksel dari model ke-2 ini tidak masuk dalam distribusi sedangkan sebaliknya untuk Gambar 4.8 dan Gambar4.10 dilakukan

update terhadap ketiga komponen. Apabila pada proses ini tidak terdapat piksel yang masuk ke dalam distribusi, maka dilakukan penggantian terhadap mean dari model yang memiliki bobot terkecil sesuai dengan intensitas terkecil atau dapat dituliskan dengan rumus.

$$Mean(i, j, min_w) = A(i, j) \tag{4.7}$$

$$Standardeviasi(i, j, min_w) = \sigma_0 \tag{4.8}$$

dengan min_w merupakan index minimum dari model yang memiliki bobot terkecil. Setelah itu dilakukan normalisasi terhadap nilai bobot dengan rumus

$$W(i, j, k) = \frac{Bobot(i, j, k)}{\sum_{k=1}^{K} Bobot(i, j, k)} \tag{4.9}$$

Sehingga dari proses ini diperoleh

$W(i,j,1) =$

0.3366	0.333	0.333
0.333	0.333	0.333
0.333	0.333	0.333

$W(i,j,2) =$

0.3267	0.333	0.333
0.333	0.333	0.333
0.333	0.333	0.333

$W(i,j,3) =$

0.3366	0.333	0.333
0.333	0.333	0.333
0.333	0.333	0.333

Gambar 4.11: Hasil Normalisasi Bobot.

Dengan W merupakan bobot yang telah dinormalisasi. Kemudian dihitung nilai $W/Standardeviasi^2$ pada masing-masing piksel tersebut sesuai dengan apa yang telah dijelaskan pada tahap pemilihan distribusi yang mencerminkan background. Hasil pada tahap ini adalah sebagai berikut:

$$W(1, 1, k)/Standardeviasi(1, 1, k)^2 = \begin{bmatrix} 0.0096 & 0.0091 & 0.0094 \end{bmatrix} \tag{4.10}$$

dengan indeks model yang awalnya

$$indeks = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \tag{4.11}$$

Nilai $W/Standardeviasi^2$ kemudian diurutkan secara *descending* sehingga diperoleh

$$W(1, 1, k)/Standardeviasi(1, 1, k)^2 = \begin{bmatrix} 0.0096 & 0.0094 & 0.0091 \end{bmatrix} \tag{4.12}$$

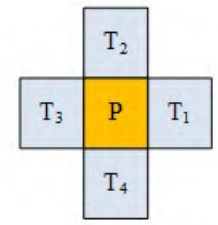
dengan k bernilai $1 \leq k \leq K$, dan indeks model sebagai berikut

$$indeks = \begin{bmatrix} 1 & 3 & 2 \end{bmatrix} \tag{4.13}$$

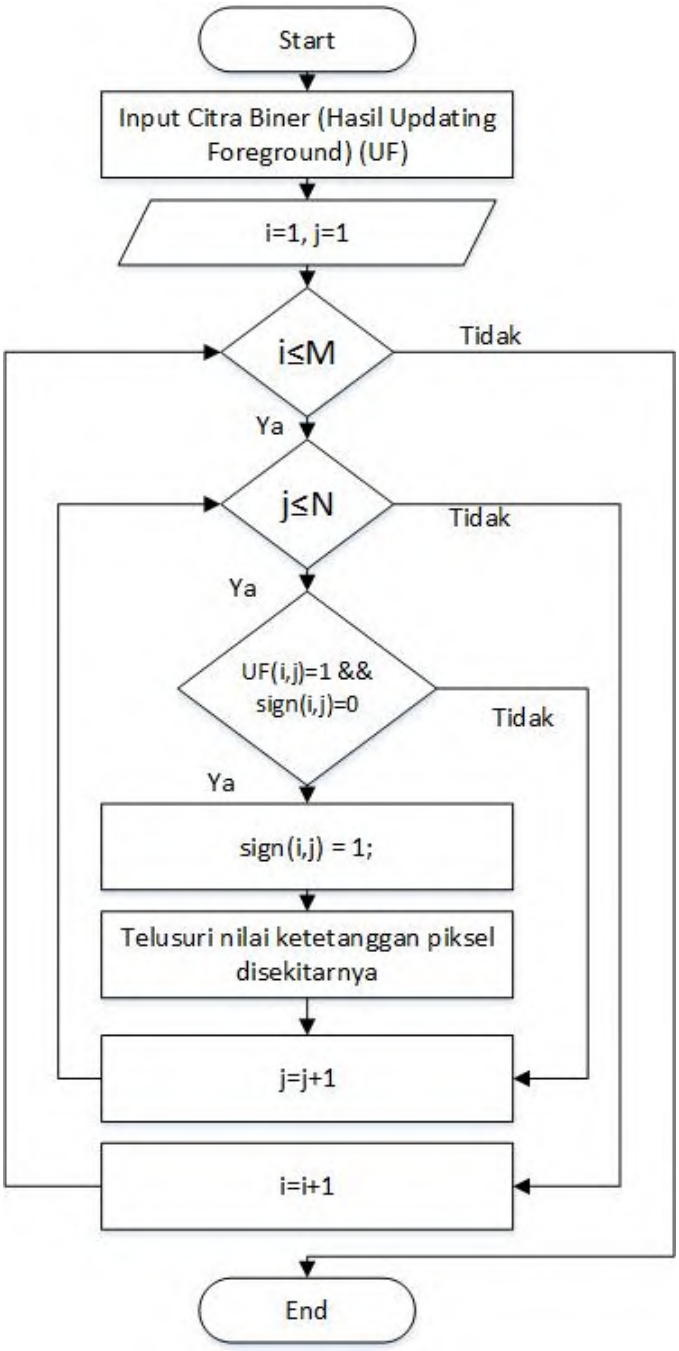
Setelah itu dipilih B distribusi pertama yang mencerminkan background berdasarkan nilai τ. Kemudian dilakukan pengecekan terhadap B distribusi tersebut apabila terdapat salah satu dari beberapa model tersebut yang berada diluar distribusi maka piksel tersebut merupakan *foreground*.

4.3.1.1 Perancangan Proses Deteksi Objek

Proses deteksi dilakukan dengan cara melakukan pengecekan terhadap masing-masing piksel pada citra *foreground*. Piksel tersebut dicek dengan tetangganya(4 tetangga) yang dapat dilihat pada Gambar 4.12. Apabila piksel tersebut memiliki minimal 1 tetangga yang memiliki intensitas 1, maka piksel tersebut masih dianggap 1 objek namun apabila piksel tersebut tidak memiliki tetangga maka dianggap objek yang berbeda. Sedangkan untuk diagram alir dari proses deteksi dapat dilihat pada gambar 4.13.



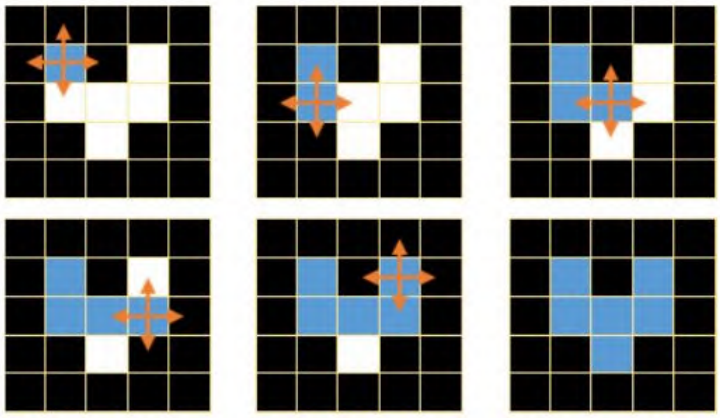
Gambar 4.12: Piksel dengan 4 ketentanggan.



Gambar 4.13: Diagram alir dari proses deteksi objek

Untuk lebih jelasnya proses pencarian tetangga dari deteksi objek dapat dilihat pada Gambar 4.14. Pada Gambar 4.14 $(M$ x $N)$ merupakan ukuran dari input matriks citra hasil proses updating foreground. Proses deteksi dimulai dari piksel yang terdapat pada sisi kiri atas dengan melakukan pengecekan terhadap 4 tetangga, piksel tersebut dijadikan pusat pengecekan. Kemudian piksel yang merupakan 1 objek akan diberi warna yang sama. Kemudian pusat piksel dipindahkan ke piksel

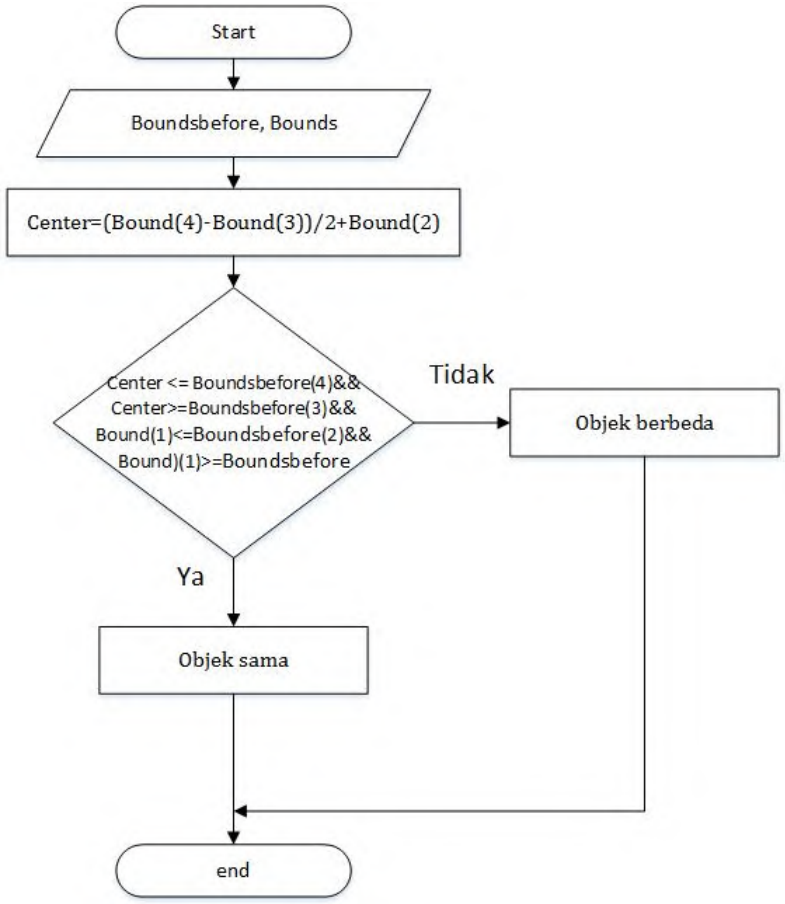
tetangga dari pusat sebelumnya selama piksel tersebut belum pernah dilakukan proses pengecekan. Proses tersebut dilakukan terhadap seluruh piksel, selama piksel tersebut memiliki minimal 1 tetangga.



Gambar 4.14: Proses pencarian tetangga pada proses deteksi objek

4.3.1.2 Perancangan Proses *Tracking* Objek

Proses ini dilakukan untuk mengetahui apakah objek pada saat frame sebelum (frame ke-(t-1)) sama dengan objek pada saat ini (frame ke-(t)), sehingga tidak terjadi kesalahan seperti dilakukannya perhitungan ulang terhadap objek yang sama pada frame yang berbeda (pada saat (t) dan (t-1)). Diagram alir proses dari *tracking* objek dapat dilihat pada gambar 4.15. Pada Gambar 4.15, *Boundsbefore* merupakan batas(batas kiri atas, kiri bawah, kanan atas, dan kanan bawah) dari objek pada frame sebelumnya yang merupakan array dengan ukuran (1x4), serta *Bounds* yang merupakan batas objek pada frame sekarang. Dengan cara menentukan titik pusat dari objek kemudian membandingkan dengan frame sebelumnya, apakah kondisi yang terdapat pada Gambar 4.15 itu terpenuhi, jika terpenuhi maka objek tersebut merupakan objek yang sama. Proses tersebut dilakukan pada semua frame, dan ketika objek tersebut melewati ROI maka objek tersebut akan dihitung.



Gambar 4.15: Diagram alir dari proses *tracking* objek

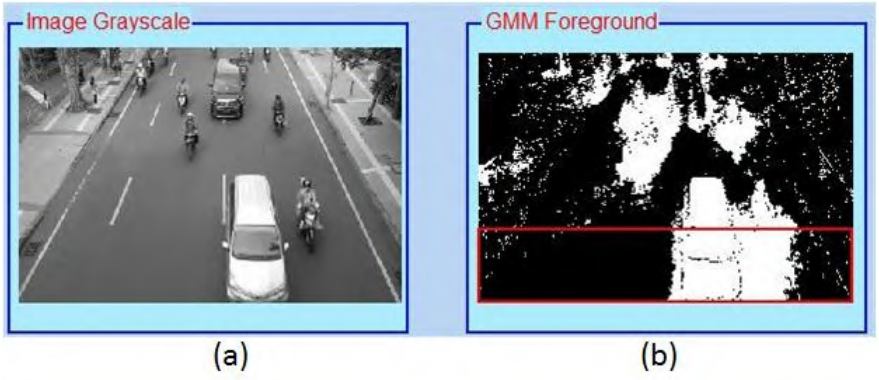
4.4 Perancangan Proses *Updating Foreground*

Pada sesi ini akan dilakukan proses *updating foreground* dari proses GMM yang telah dijelaskan sebelumnya. Proses ini bertujuan untuk mengurangi shadow yang terdeteksi sebagai *foreground*. Sebelum dilakukan proses *updating foreground* berikut diberikan contoh citra hasil dari proses GMM yang telah dijelaskan sebelumnya. Pada citra grayscale Gambar 4.16(a) meskipun bayangan tidak terlihat secara visual, namun pada citra yang dihasilkan dari proses *GMM* Gambar 4.16(b) menunjukkan terdapat objek selain *foreground* yang diidentifikasikan sebagai *foreground*. Meskipun pada frame ini tidak akan terjadi kesalahan perhitungan, hal ini dikarenakan tidak terdapat objek lain disekitar objek tersebut yang tercover oleh bayangan sehingga diidentifikasikan sebagai 1 objek. Namun beda halnya apabila terdapat kasus yang dicontohkan oleh Gambar 4.17 dan 4.18. Pada Gambar 4.17 sebagian background tertutupi oleh bayangan, selain itu terdapat objek yang berada didalam bayangan dari objek lain, hal ini menyebabkan terjadinya kesalahan perhitungan yaitu beberapa objek yang terdeteksi sebagai satu objek. Sama halnya

Gambar 4.16: Citra hasil dari proses GMM (1).(a)citra grayscale(b)hasil *GMM*

dengan Gambar 4.17, Gambar 4.18 juga memperlihatkan posisi objek yang berada di dalam objek lain. Sehingga dari kasus-kasus ini, pada penelitian ini akan dilakukan proses *updating foreground* dari hasil proses *GMM* untuk meminimalisir *shadow* dan *noise* yang timbul pada proses *GMM*.

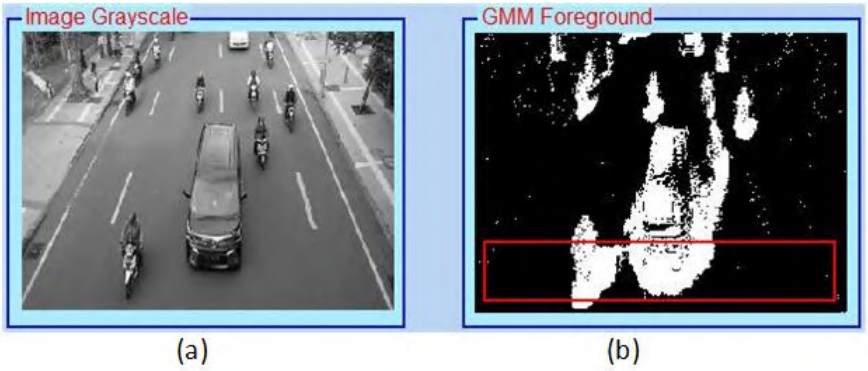
Gambar 4.17: Citra hasil dari proses GMM (2).(a)citra grayscale(b)hasil *GMM*



Gambar 4.18: Citra Hasil dari proses GMM (3).(a)citra grayscale(b)hasil *GMM*

Berikut ini diberikan kesalahan kesalahan yang lain yang terjadi dari hasil proses *GMM*.

Gambar 4.19: Citra Hasil dari proses GMM (4).(a)citra *grayscale*(b)hasil *GMM*



Gambar 4.20: Citra Hasil dari proses GMM (5).(a)citra *grayscale*(b)hasil *GMM*

Gambar 4.21: Citra Hasil dari proses GMM (6).(a)citra *grayscale*(b)hasil *GMM*

Pada penelitian ini proses *updating foreground* dilakukan pada citra hasil dari proses GMM yang menggunakan parameter-parameter *default* GMM, yang mana

pada penelitian sebelumnya input parameter GMM ini berubah-ubah sesuai dengan waktu pengambilan video. Oleh karena itu ditentukan 2 nilai yang difokuskan sebagai titik tumpu dari algoritma ini yaitu nilai mean, dan modus dari matriks intensitas citra pada saat itu, yang mana sesuai kondisi dan waktu kedua nilai ini akan berubah. Berikut ini adalah tahapan dari proses *updating foreground*

1. Menghitung nilai mean dan modus
   Hal yang pertama kali dilakukan adalah menghitung nilai mean dan modus pada frame saat itu. Nilai mean dan modus inilah yang akan menentukan berapa interval intensitas yang dikategorikan sebagai bayangan.

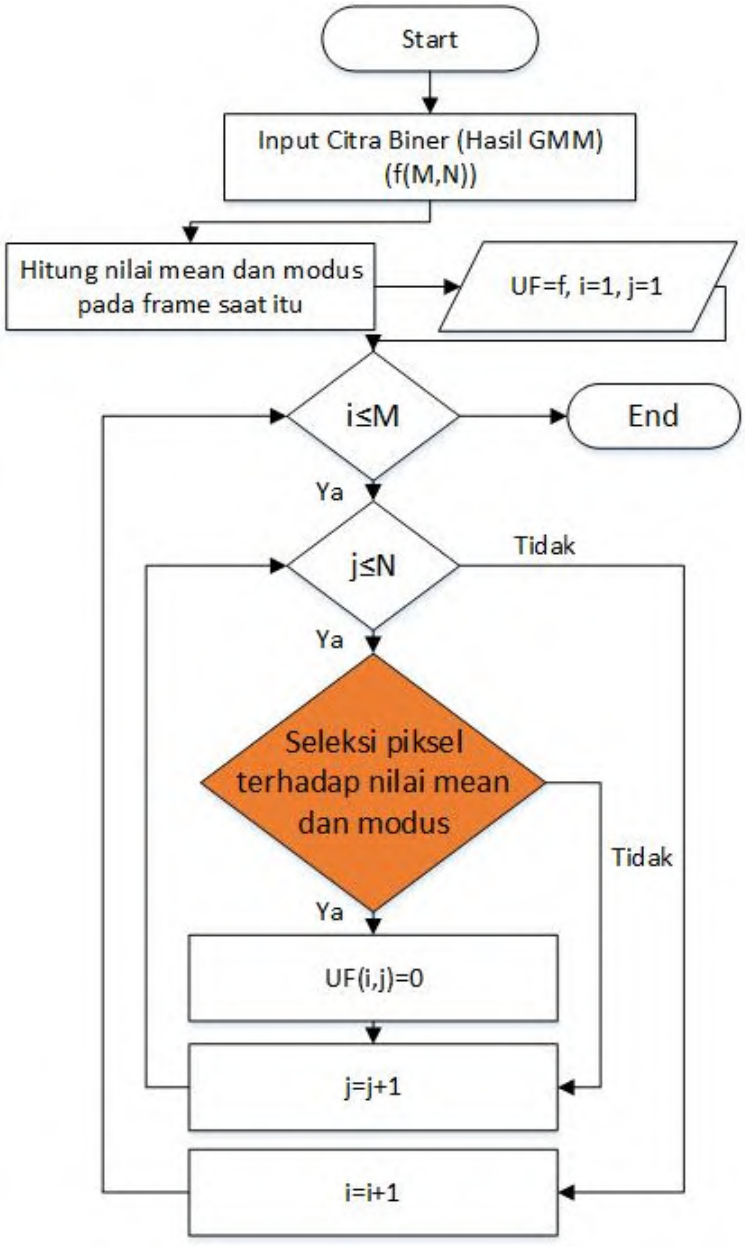
2. Proses seleksi piksel terhadap nilai mean dan modus
   Berdasarkan nilai mean dan modus yang telah ditentukan sebelumnya, dilakukan proses seleksi terhadap tiap piksel yang diasumsikan sebagai *foreground* hasil dari proses *GMM*. Hal ini dilakukan dengan cara melihat nilai intensitas grayscale pada koordinat piksel yang dianggap *foreground*, apabila piksel tersebut masuk dalam interval yang dapat dilihat pada Gambar 4.22, maka piksel tersebut merupakan bayangan yang kemudian diberikan nilai intensitas nol. Namun apabila piksel tersebut tidak masuk dalam interval tersebut, maka piksel tersebut adalah *foreground* dan diberikan nilai intensitas satu. Hal ini akan meminimalisir shadow atau noise yang terdeteksi sebagai *foreground* dari citra hasil *GMM*.

Kondisi	Interval yang dinolkan
$45 <$ modus-mean < 100	$60 < A(i,j) < 171,\ UF(i,j) = 0$
modus-mean ≥ 100	$77 < A(i,j) < 145,\ UF(i,j) = 0$
mean-modus ≥ 45	$50 < A(i,j) < 180,\ UF(i,j) = 0$
$20 <$ mean-modus < 30	$50 < A(i,j) < 100,\ UF(i,j) = 0$
$30 \leq$ mean-modus < 45	$70 < A(i,j) < 150,\ UF(i,j) = 0$
$0 \leq$ mean-modus ≤ 20	$70 < A(i,j) < 150,\ UF(i,j) = 0$
else	$60 < A(i,j) < 171,\ UF(i,j) = 0$

Gambar 4.22: Interval yang digunakan pada proses seleksi *updating foreground*.

Langkah-langkah yang telah dijelaskan sebelumnya, dapat dinyatakan dalam bentuk Diagram alir proses *updating foreground* pada Gambar 4.23. Box berwarna orange pada Gambar 4.20 merupakan proses seleksi yang disesuaikan oleh kondisi mean dan modus frame pada saat itu. Nilai yang digunakan pada box berwarna orange tersebut, diberikan pada Gambar 4.22. Beberapa kondisi tersebut diperoleh dari

hasil eksperimen dari beberapa jenis video (pagi, siang, dan sore). Sedangkan pada Gambar 4.24 diberikan contoh sederhana proses *updating foreground* pada suatu frame sesuai dengan langkah yang dijelaskan sebelumnya.



Gambar 4.23: Diagram alir proses *updating foreground*.

Hasil GMM

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

Updating Foreground

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

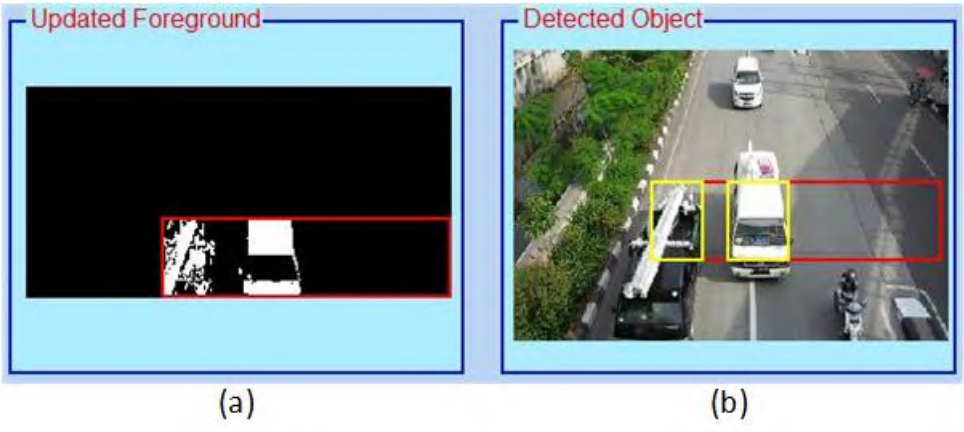
50	50	61	80	80	80	80	80	80
50	50	80	80	80	80	80	80	80
61	80	80	80	80	80	80	80	80
80	80	80	61	80	80	50	80	80
80	80	61	50	50	80	50	50	61
80	80	50	50	50	80	50	50	61
80	61	50	50	50	80	50	50	61
80	61	50	50	50	80	50	50	80

Mean = 67.5
Modus=80
$60 < A(i,j) < 171, \quad UF(i,j) = 0$
Dengan
A = matriks intensitas,
UF = matriks updating foreground hasil dari GMM

Gambar 4.24: Contoh proses *updating foreground*.

Hasil yang diperoleh proses *updating foreground* ini dapat dilihat pada Gambar 4.25, 4.26, dan 4.27.

Gambar 4.25: Hasil proses *updating foreground* (1). (a)citra hasil proses *updating foreground*(b)citra RGB



Gambar 4.26: Hasil proses *updating foreground* (2). (a)citra hasil proses *updating foreground*(b)citra RGB

Gambar 4.27: Hasil proses *updating foreground* (3). (a)citra hasil proses *updating foreground*(b)citra RGB

Pada gambar 4.25 terlihat bahwa noise yang sebelumnya terdapat pada sekeliling objek dapat diminimalisir. Sedangkan pada Gambar 4.26 terlihat bahwa bahwa objek yang termasuk dalam bayangan dari objek lain, mampu dipisahkan satu sama lain. Begitu pula pada Gambar 4.27, objek yang berdekatan mampu diidentifikasikan sebagai objek yang terpisah. Baik bayangan yang terdeteksi sebagai *foreground* maupun noise yang terjadi pada proses GMM, mampu diminimalisir dengan algoritma ini. Selain itu, berikut diberikan hasil proses *updating foreground* untuk kasus lainnya.

Gambar 4.28: Hasil proses *updating foreground* (4). (a)hasil *updating foreground*(b)citra RGB

Gambar 4.29: Hasil proses *updating foreground* (5). (a)hasil *updating foreground*(b)citra RGB

Gambar 4.30: Hasil proses *updating foreground* (6). (a)hasil *updating foreground*(b)citra RGB

Beberapa proses tersebut dapat dinyatakan dalam bentuk algoritma seperti di bawah ini

Judul → Pengembangan Algoritma Robust untuk Menghitung Kendaraan Bergerak Berbasis Pengolahan Citra

Input → Video

Output → Jumlah Kendaraan

Deskripsi → Algoritma ini digunakan untuk menghitung kendaraan dari suatu rekaman lalu lintas dan robust terhadap waktu pengambilan video(pagi, siang, dan sore)

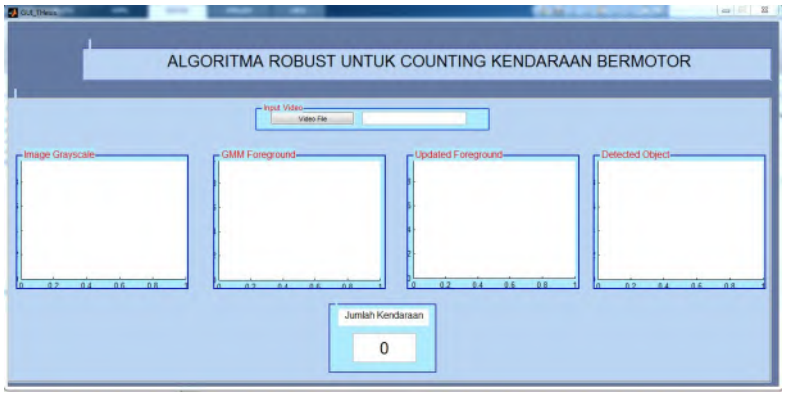
Algoritma

1. Input video
2. Ekstraksi frame
3. Konversi frame ke dalam *grayscale*
4. Proses *GMM* (Gambar 4.13)
5. Proses *Updating Foreground* (Gambar 4.23)
6. Proses Deteksi Objek (Gambar 4.13)
7. Proses *Tracking Object* (Gambar 4.15)
8. Proses *Counting*

Gambar 4.31: Hasil proses *updating foreground* (6). (a)hasil *updating foreground*(b)citra RGB

4.5 Perancangan *Interface* Program

Berikut ini adalah perancangan *interface* program.



Gambar 4.32: Interface Program

Gambar 4.32 merupakan gambar *interface* dari program, hasil yang akan ditampilkan pada program tersebut adalah, citra grayscale, citra *foreground* hasil *GMM*, citra hasil *updating foreground* pada area *ROI*, citra RGB dengan hasil proses *detecting* objek, jumlah kendaraan yang dihitung, dan nilai frame saat ini.

4.6 Implementasi Algoritma pada Program

Pada subab ini akan ditampilakn penggalan program dari beberapa proses dibawah ini.

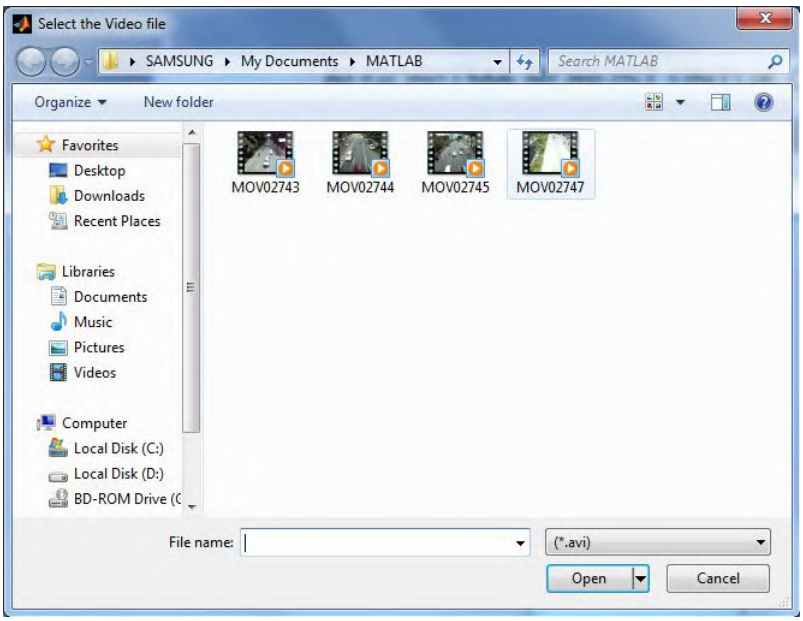
4.6.1 Implementasi Input Video

Program dapat menerima input video dengan berbagai tipe, *.avi, *.flv,*.mov, *.3gp dan lain sebagainya. Dianjurkan untuk menggunakan video dengan resolusi 240x320 piksel. Agar proses pengolahan informasi digital tidak berlangsung lama. Semakin tinggi resolusi video tersebut, maka durasi proses pun semakin lama. Setelah diinputkan, video tersebut kemudian diekstrak ke dalam bentuk frame-frame. Kode program untuk menerima input video adalah sebagai berikut.

```
1     [FileName,PathName] = uigetfile('~/matlab/*.avi', 'Select the Video file');
2     set(handles.edit1,'string',[PathName FileName]);
3     for i=1:400
4     frame = step(videoReader);
5     BBB(:,:,:,i)=im2uint8(frame);
6     end
```

Gambar 4.33: *listing* program untuk implementasi input video

Apabila proses tersebut dilakukan maka tampilan interface yang muncul adalah sebagai berikut



Gambar 4.34: Tampilan input video

4.6.2 *Implementasi Proses GMM dan Proses Updating Foreground*

Proses GMM digunakan untuk mencari citra *foreground*. Yaitu citra yang merepresentasikan keadaan objek yang bergerak. Parameter untuk menggunakan proses GMM ini adalah; a. Banyaknya komponen gaussian, b. Dan nilai learning rate dengan nilai kedua parameter tersebut tidak berubah-ubah meskipun input video berubah-ubah. Sedangkan proses *updating foreground* dilakukan untuk meminimalisir *noise* berdasarkan nilai mean dan modus frame pada saat itu. Code untuk proses GMM dan proses *updating foreground* dapat dilihat pada Gambar 4.35.

```
set(0, 'RecursionLimit', 10000)
[FileName,PathName] = uigetfile('~/matlab/*.avi', 'Select the Video file');
set(handles.edit1,'string',[PathName FileName]);
foregroundDetector = vision.ForegroundDetector('NumGaussians', 7, ...
    'NumTrainingFrames', 50);
videoReader = vision.VideoFileReader(FileName);

for i=1:400
frame = step(videoReader);
BBB(:,:,:,i)=im2uint8(frame);
foreground(:,:,i)=step(foregroundDetector, frame);
end

    roi = [10 180 300 50];
    boundBefore=[];

for i=1:400
    A(:,:)=rgb2gray(BBB(:,:,:,i));
    frameke=frameke+1;
    meanie=round(mean2(A(:,:)));
    BAA=(A(:,:));
    [M,F] = mode(BAA(:));
    for j=roi(2):roi(2)+roi(4)
        for k=roi(1):roi(3)+roi(1)
            approach(j,k)=foreground(j,k,i);

        if M-meanie>45 && M-meanie< 100
            if A(j,k)>60&&A(j,k)<171
            approach(j,k)=0;
            end
        elseif M-meanie> 100
             if A(j,k)<=145&&A(j,k)>=77
                approach(j,k)=0;
             end
        elseif meanie-M>45
            if A(j,k)>=50 && A(j,k) <= 180
            approach(j,k)=0;
            end
```

Gambar 4.35: *listing* program untuk *GMM* dan *updating foreground*

Pada Gambar 4.35 merupakan *code* untuk proses GMM dan *updating foreground*, dimana terdapat beberapa seleksi di dalam proses *updating foreground*.

4.6.3 Implementasi Proses Tracking dan Counting

Input yang digunakan pada saat ini merupakan citra biner hasil proses *updating foreground*. Proses *tracking* bertujuan untuk memeriksa apakah objek tersebut adalah objek yang sama sehingga tidak dihitung ulang, sedangkan proses *counting* dilakukan apabila terdapat objek yang melewati ROI (batas bawah). Berikut ini merupakan code untuk proses *tracking* dan *counting*. Pada Gambar 4.36 dilakukan pengecekan apakah piksel pada koordinat itu bernilai satu, apabila iya maka dilakukan deteksi tetangga dan kemudian piksel tersebut dikelompokan. Setelah itu dilihat apakah titik tengah objek tersebut berada dalam objek pada frame sebelumnya jika iya maka objek tersebut merupakan objek yang sama. Setelah objek itu melewati ROI maka akan dilakukan proses counting.

```
for i=roi(2):roi(2)+roi(4)
        for j=roi(1):roi(3)+roi(1)
            if (bin(i,j) == 1) && (sign(i,j) == 0)
                sign(i,j) = 1;
                Boundawal=[i i j j];
                [bin, sign, Bound] = find(i, j, bin, sign, Boundawal, roi);
                object = object + 1;
                Bounds(object,:)=Bound;
            end
        end
end
for i=1:size(boundBefore,1)
        for j=1 : size(Boundsnew,1)
            if track(boundBefore(i,:),Boundsnew(j,:))
                if (count (boundBefore(i,:),Boundsnew(j,:), roi(2)+roi(4)))
                    counter=counter+1;
                end
            end
        end
end
```

Gambar 4.36: *listing* program untuk *tracking* dan *counting*

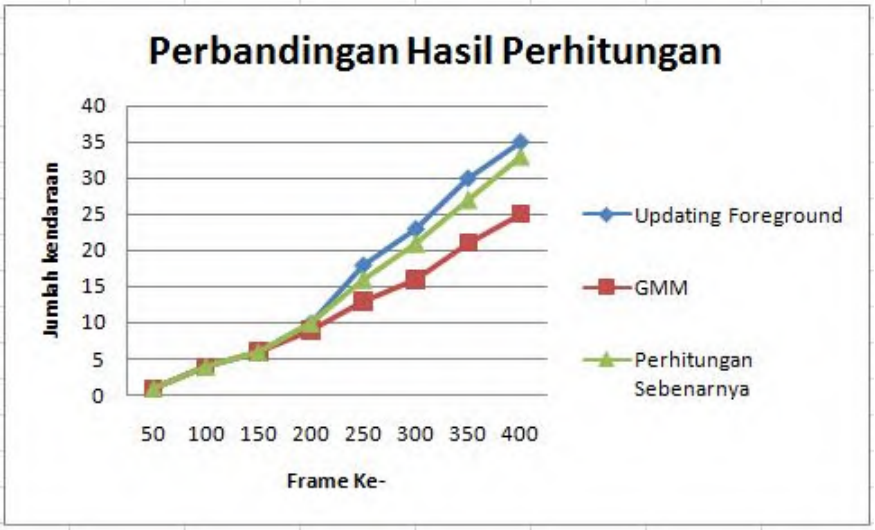
BAB V
UJI COBA DAN PEMBAHASAN

Pada bab ini akan dilakukan pengujian program serta pembahasan hasil uji coba terhadap beberapa jenis waktu pengambilan video yaitu pagi, siang, dan sore hari. Untuk uji coba pertama dilakukan pada video yang diambil pada waktu pagi hari dengan kondisi bayangan normal tidak terlalu tampak secara visual (*soft shadow*). Data video tersebut diambil dari jembatan penyebrangan di daerah jalan Diponegoro Surabaya, diambil dengan menggunakan kamera hp yang diikatkan pada pegangan jembatan dengan bantuan tripod *octopus*. Sedangkan data berikutnya diambil dari jembatan penyebrangan di daerah Suramadu, dengan kondisi posisi kamera yang serupa dengan kondisi sebelumnya. Data video yang diperoleh merupakan data video pada siang hari dengan adanya bayangan yang dapat terlihat secara visual. Sedangkan video ketiga diambil dari jembatan penyebrangan di sekitar jalan Pemuda dengan keadaan relatif ramai dan posisi bayangan yang menutupi background .Selain itu dilakukan antara hasil perbandingan perhitungan antara metode GMM dengan metode yang diusulkan pada penelitian ini. Data uji coba disajikan dalam bentuk tabel di bawah ini.

Tabel 5.1: Tabel Data uji coba video

Video	Waktu	Kondisi bayangan	keramaian
1	pagi	normal	relatif ramai
2	siang	tampak secara visual	lengang
3	sore	tampak secara visual	ramai

Ketiga data video tersebut masing-masing diambil dari jembatan penyebrangan di jalan Diponegoro (video pertama), daerah kedung cowek(video kedua), dan jalan Pemuda(video ketiga) dengan kondisi bayangan dan tingkat keramaian yang berbeda-beda. Untuk video pertama diambil pada waktu pagi hari, video kedua pada waktu siang hari dan video ketiga pada waktu sore hari. Berikut ini diberikan grafik hasil uji coba pada video ketiga (video dengan tingkat keramaian tinggi) untuk memperlihatkan hasil perbandingan perhitungan dengan metode *GMM*, *updating foreground* dan perhitungan sebenarnya.



Gambar 5.1: Grafik perbandingan hasil perhitungan.

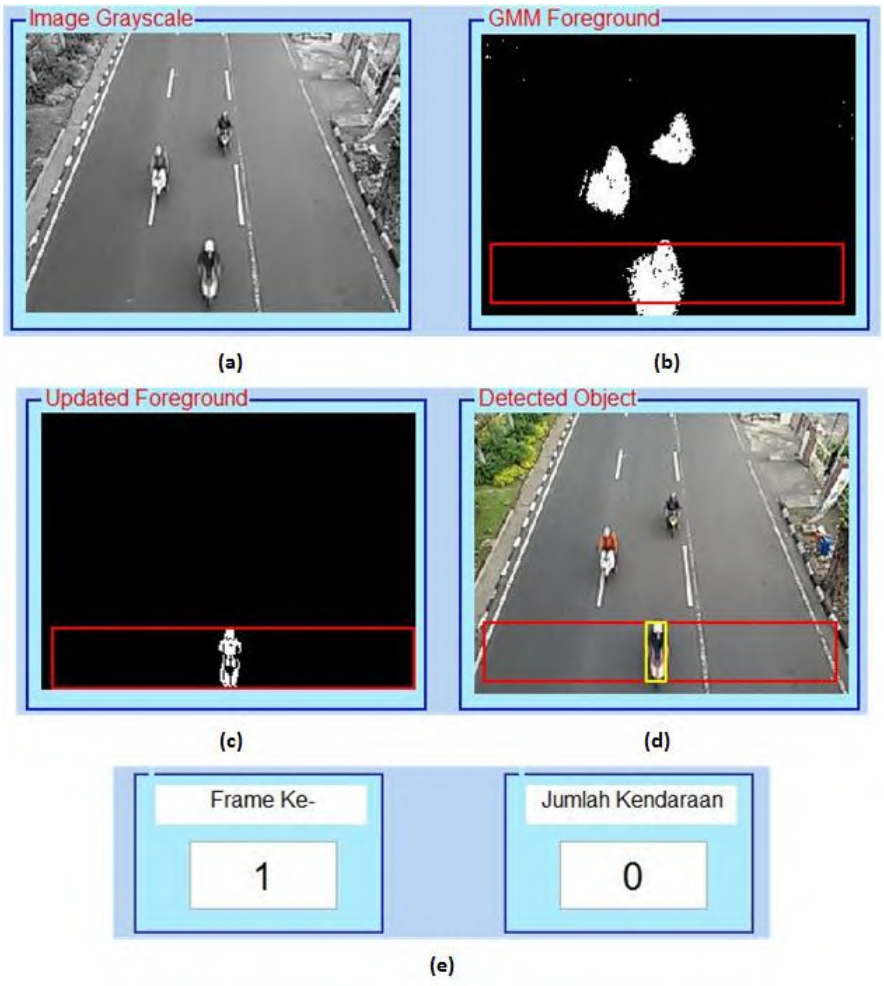
Gambar 5.1 menunjukkan hasil yang diperoleh pada proses running menggunakan software Matlab. Pada grafik yang diperlihatkan Gambar 5.1 menunjukkan sebanyak 35 kendaraan terdeteksi dengan nilai sebenarnya adalah 33 atau dengan kata lain didapatkan tingkat kesalahan metode *updating foreground* yang diterapkan pada video 3 adalah sebesar 6.06%. Sedangkan dengan metode *GMM* diperoleh 25 kendaraan terdeteksi dari 33 kendaraan atau dengan kata lain tingkat kesalahan sebesar 24.24%. Kesalahan perhitungan yang terjadi pada proses GMM disebabkan kondisi jalan yang ramai serta pengaruh bayangan yang mengakibatkan beberapa kendaraan hanya terdeteksi sebagai 1 kendaraan. Untuk lebih jelasnya berikut ini diberikan hasil uji coba pada masing-masing video.

5.1 Uji Coba Video Pertama

5.1.1 Uji Video Pertama

Pengujian pertama dilakukan pada video dari jembatan penyebrangan di sekitar daerah jalan Diponegoro Surabaya pada waktu pagi hari dengan bayangan normal dan tidak terlalu tampak secara visual(*soft shadow*). Gambar 5.1 adalah contoh hasil program ketika video pertama diinputkan ke dalam program dengan ROI yang telah ditentukan sebelumnya. Gambar 5.2(a) merupakan gambar citra ketika telah diubah ke dalam bentuk *grayscale* dimana terlihat bahwa bayangan pada objek tidak terlalu tampak secara visual (*soft shadow*). Sedangkan pada Gambar5.2(b) menunjukkan hasil deteksi foreground menggunakan metode *GMM* terlihat bahwa pada masing-masing objek-objek tersebut masih terdapat bayangan meskipun pada gambar 5.2(a) tidak nampak secara visual. Kemudian pada Gambar 5.2(c) merupakan hasil dari proses *updating foreground* dimana bayangan yang terdeteksi pada proses *GMM*

telah diminimalisir. Proses ini dilakukan dengan cara menentukan mean pada saat frame itu dan nilai modus, nilai modus menggambarkan bagaimana keadaan pada saat pengambilan video, cerah atau mendung. Setelah diketahui nilai mean dan modus pada frame saat itu, kemudian dilakukan proses seleksi berdasarkan nilai mean dan modus, nilai interval intensitas piksel foreground yang diasumsikan sebagai yang diasumsikan sebagai shadow bergantung pada proses seleksi tersebut. Pada Gambar 5.2(d) merupakan tampilan input dalam bentuk *RGB* dan proses *Tracking* dan *Counting*, pada proses ini objek telah dideteksi kemudian diberi tanda persegi disekitar objek tersebut, setelah itu di-*track* apakah objek pada frame(t) sama dengan frame (t-1) dan apabila objek tersebut telah melewati roi bagian bawah. Untuk selanjutnya proses akan dilakukan untuk 400 frame dan disajikan pada tabel 5.2 berikut.

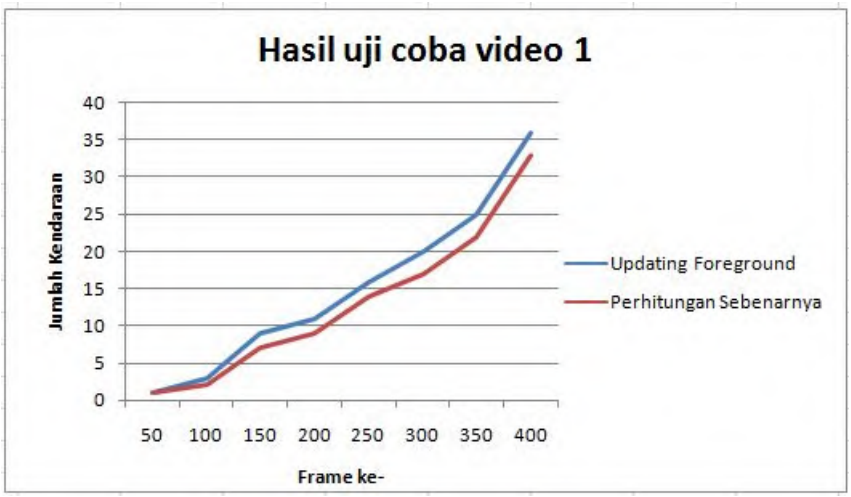


Gambar 5.2: Contoh proses *counting* pada input video pertama (a)citra *grayscale* (b)foreground hasil *GMM* (c)updated foreground (d)tracking dan *counting* (e)indeks frame dan hasil *counting*.

Tabel 5.2: Tabel Perbandingan hasil *counting updating foreground* dengan perhitungan sebenarnya pada input video pertama

Metode/Frame ke-	50	100	150	200	250	300	350	400
Updating Foreground	1	3	9	11	16	20	25	36
Sebenarnya	1	2	7	9	14	17	22	33

Pada tabel 5.2 dapat dilihat hasil perhitungan untuk 400 frame, hasil perhitungan disajikan dalam bentuk tiap 50 frame. Data yang terdapat pada tabel 5.2 tersebut dapat dinyatakan dalam bentuk grafik yang terdapat pada Gambar 5.3 berikut.



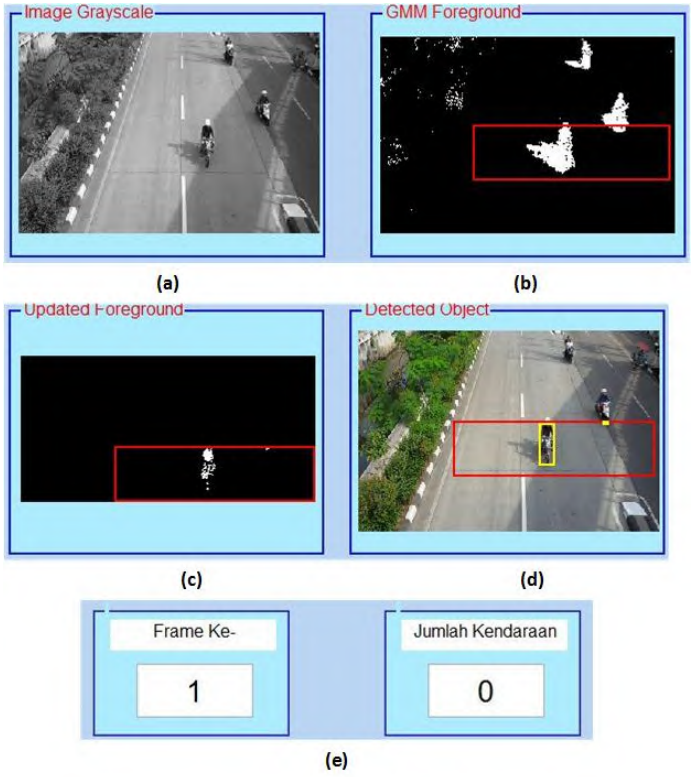
Gambar 5.3: Grafik Perbandingan hasil *counting updating foreground* dengan perhitungan sebenarnya pada input video pertama.

Dari hasil *running* program untuk 400 frame diperoleh hasil yang dapat dilihat pada grafik yang disajikan pada Gambar 5.3, yaitu sebanyak 36 kendaraan terdeteksi. Sedangkan hasil yang diperoleh dari perhitungan secara manual adalah 33 sehingga diketahui tingkat kesalahanya adalah sebesar 9.09%, kesalahan ini bisa saja terjadi dikarenakan antara lain perubahan cuaca yang terjadi secara tiba-tiba contohnya munculnya sinar matahari secara tiba-tiba setelah keadaan mendung, apabila hal ini terjadi maka proses *GMM* akan menjadi kurang sesuai dikarenakan *GMM* menggunakan history dari frame-frame sebelumnya. Dan karena proses *updating foreground* menggunakan hasil dari proses *GMM* maka akan terjadi kesalahan juga saat deteksi objek. Selain itu kesalahan juga mungkin diakibatkan oleh warna objek yang identik dengan *background* atau *shadow*.

5.2 Uji Coba Video Kedua

5.2.1 Uji Video Kedua

Pengujian kedua dilakukan pada video dari jembatan penyebrangan di sekitar daerah Suramadu Surabaya pada waktu siang hari dengan bayangan yang tampak secara visual. Uji coba ini dilakukan untuk mengetahui apakah bayangan yang secara visual tampak dapat tidak terdeteksi sebagai bagian dari objek. Proses-proses dari *running* program dengan input video kedua disajikan pada Gambar 5.4. Berbeda dengan data input untuk eksperimen pertama, pada input kedua bayangan terlihat jelas secara visual hal ini dapat dilihat pada Gambar 5.4(a), yang menyebabkan bayangan tersebut terdeteksi sebagai *foreground*. Hal yang membedakan video pertama dan kedua adalah nilai mean dan modus dari masing-masing video. Pada Gambar 5.4(b) terlihat bahwa bayangan juga terdeteksi sebagai *foreground*. Sedangkan pada Gambar 5.4(b), setelah melalui proses seleksi, bayangan telah diasumsikan sebagai *background*, namun karena objek tersebut belum melewati ROI maka belum dilakukan perhitungan terhadap objek tersebut. Untuk selanjutnya proses akan dilakukan untuk 400 frame dan disajikan pada Tabel 5.3 berikut.

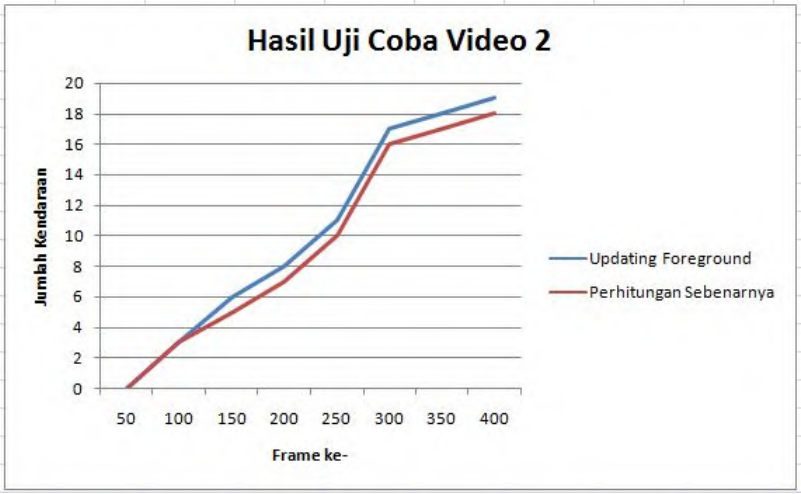


Gambar 5.4: Contoh proses *counting* pada input video kedua (a)citra *grayscale* (b)foreground hasil *GMM* (c)updated foreground (d)tracking dan *counting* (e)indeks frame dan hasil *counting*.

Tabel 5.3: Tabel Perbandingan hasil *counting updating foreground* dengan perhitungan sebenarnya pada input video kedua

Metode/Frame ke-	50	100	150	200	250	300	350	400
Updating Foreground	0	3	6	8	11	17	18	19
Sebenarnya	0	3	5	7	10	16	17	18

Data yang terdapat pada tabel 5.3 tersebut dapat dinyatakan dalam bentuk grafik yang terdapat pada Gambar 5.4 berikut.



Gambar 5.5: Perbandingan hasil *counting updating foreground* dengan perhitungan sebenarnya pada input video kedua.

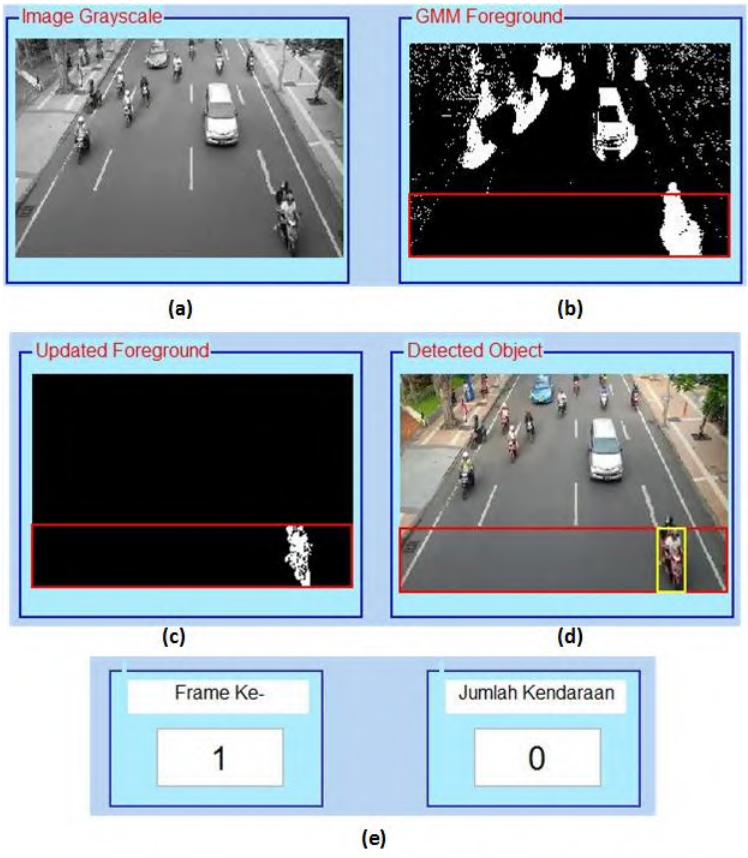
Pada grafik yang disajikan dalam Gambar 5.5 terdeteksi 19 kendaraan yang melintas, sedangkan dari hasil perhitungan secara manual didapatkan 18 kendaraan yang melintas. Dari hasil tersebut diperoleh eror sebesar 5.55%.

5.3 Uji Video Ketiga

5.3.1 Uji Coba Video Ketiga

Pengujian ketiga dilakukan pada video yang diambil dari jembatan penyebrangan di sekitar daerah jalan Pemuda Surabaya pada waktu sore hari dengan bayangan normal yang tidak tampak secara visual, namun terdapat bayangan pada background dan tingkat keramaian jalan cukup ramai. Uji coba ini dilakukan untuk mengetahui apakah algoritma mampu mendeteksi kendaraan dengan keadaan yang telah disebutkan. Proses-proses dari *running* program dengan input video ketiga dapat dilihat pada Gambar 5.6. Meskipun bayangan secara visual tidak terlalu terlihat pada gambar, namun hasil *GMM* yang tidak bagus dapat mempengaruhi proses deteksi, hal tersebut dapat dilihat pada Gambar 5.6(a) dan Gambar 5.6(b). Pada gambar 5.6(a) citra *grayscale* menunjukkan tidak terdapat bayangan pada

objek, namun hasil *GMM* pada Gambar 5.6(b) dapat menyebabkan kesalahan perhitungan pada proses deteksi objek jika terdapat objek lain disekitar objek tersebut. Setelah proses *updated foreground* tersebut dilakukan, bayangan dapat diminimalisir.



Gambar 5.6: Contoh proses *counting* pada input video ketiga (a)citra *grayscale* (b)foreground hasil *GMM* (c)updated foreground (d)tracking dan *counting* (e)indeks frame dan hasil *counting*.

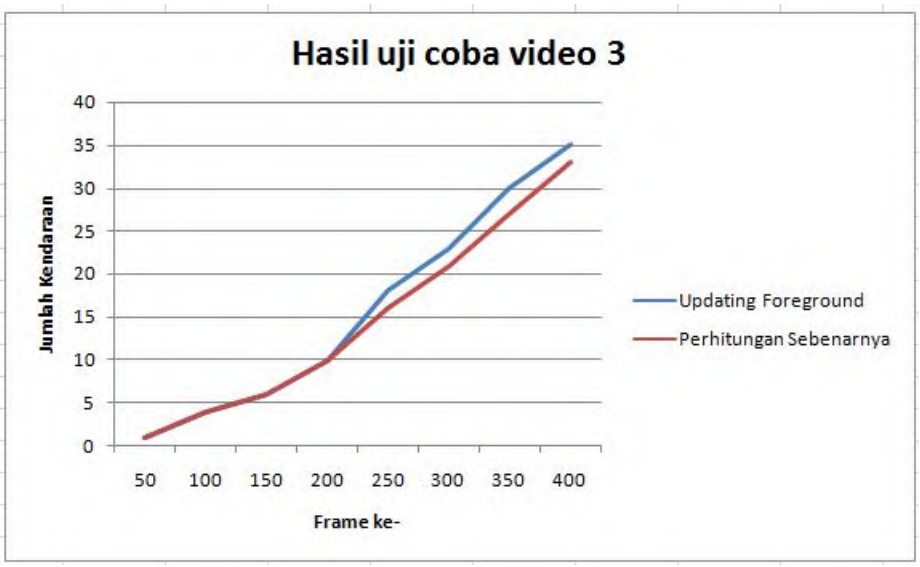
Sedangkan hasil perhitungan untuk 400 frame dapat dilihat pada grafik yang terdapat pada Tabel 5.4 dibawah ini.

Tabel 5.4: Tabel Perbandingan hasil *counting updating foreground* dengan perhitungan sebenarnya pada input video ketiga

Metode/Frame ke-	50	100	150	200	250	300	350	400
Updating Foreground	1	4	6	10	18	23	30	35
Sebenarnya	1	4	6	10	16	21	27	33

Pada tabel 5.4 dapat dilihat hasil perhitungan untuk 400 frame, hasil perhitungan disajikan dalam bentuk tiap 50 frame. Data yang terdapat pada tabel 5.4 tersebut

dapat dinyatakan dalam bentuk grafik yang terdapat pada Gambar 5.7 berikut.



Gambar 5.7: Perbandingan hasil *counting updating foreground* dengan perhitungan sebenarnya pada input video ketiga.

Dapat dilihat pada grafik yang disajikan pada Gambar 5.7, jumlah kendaraan yang terdeteksi oleh metode *updating foreground* sebanyak 35 dari hasil perhitungan sebenarnya adalah sebanyak 33, dengan kata lain tingkat kesalahan yang didapatkan adalah sebesar 6.06%.

Hasil pembahasan di atas dapat ditampilkan dalam bentuk tabel berikut.

Tabel 5.5: Tabel hasil uji video

Video	Hasil *updating foreground*	Hasil Sebenarnya	Eror
1	36 Kendaraan	33 Kendaraan	9.09%
2	19 Kendaraan	18 Kendaraan	5.55%
3	35 Kendaraan	33 Kendaraan	6.06%

Dari Tabel 5.5 di atas, pada video pertama diperoleh tingkat kesalahan hasil proses *updating foreground* sebesar 9.09%. Pada video kedua diperoleh tingkat kesalahan hasil proses *updating foreground* sebesar 5.55% . Pada video ketiga diperoleh tingkat kesalahan hasil proses *updating foreground* sebesar 6.06%. Selain itu dari hasil tersebut diperoleh bahwa pada video pertama rata-rata terdeteksi tiap 50 frame(2 detik) adalah 4.625 kendaraan, pada video 2 diperoleh rata-rata kendaraan terdeteksi tiap 50 frame masuk sebesar 2.25 kendaraan sedangkan pada video 3 diperoleh rata-rata kendaraan terdeteksi tiap 50 frame sebesar 4.5. Tingginya

tingkat kendaraan yang terdeteksi pada gambar video 1 dan video 3 berpotensi menimbulkan kemacetan, oleh karena itu pada video pertama dan kedua ini perlu dilakukan kebijakan-kebijakan seperti pelebaran jalan, pencarian jalur alternatif maupun pengaturan lampu lalu lintas baik pada posisi sebelum ataupun sesudah jalan pada video 1 dan 2 ini.

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini dibahas kesimpulan dari hasil uji coba dan pembahasan. Selain itu akan diberikan saran yang dapat digunakan untuk penelitian selanjutnya.

6.1 Kesimpulan

Dari hasil uji coba di atas dapat diambil kesimpulan sebagai berikut.

1. Algoritma yang diterapkan dapat menghitung kendaraan bergerak dari video rekaman arus lalu lintas jalan dengan melakukan *update* pada foreground yang telah terbentuk dari metode *GMM*. Tahapan dalam penghitungan ini adalah:
   (a) Penentuan area *ROI* (*Region of Interest*) sebagai area perhitungan.
   (b) Proses *Background Subtraction* menggunakan metode *GMM* untuk mendapatkan citra foreground
   (c) *Shadow removal* yaitu tahap dimana dilakukan proses *updating foreground* dari hasil *GMM* untuk meminimalisir *shadow* dan *noise*.
   (d) setelah itu dilakukan proses *tracking* dan *counting* untuk menghitung kendaraan yang melewati ROI
2. Algoritma mampu meminimalisir *noise* dan *shadow* hal ini dapat dilihat dari kasus khusus yang dibahas dalam pembahasan, sehingga objek yang awalnya terkoneksi oleh *shadow* pada proses *GMM* dapat dikenali sebagai objek yang terpisah. Hal ini terlihat dari hasil yang diperoleh pada bab 5 yang mengatakan bahwa semakin ramai volume kendaraaan pada video itu maka tingkat kesalahan dari metode GMM akan semakin besar dan selisih tingkat kesalahan dari proses *updating foreground* dan *GMM* juga akan semakin signifikan (dengan performa updating foreground yang lebih baik tentunya).
3. Rata-rata akurasi keberhasilan proses *updating foreground* dalam menghitung kendaraan bergerak dengan kondisi yang berbeda-beda adalah 93.1%.
4. Perubahan keadaan dan cahaya matahari secara tiba tiba, pergerakan kamera, serta warna kendaraan yang menyerupai background mempengaruhi performa program dalam menghitung. kendaraan.

6.2 Saran

Dengan melihat hasil yang dicapai pada penelitian ini, ada beberapa hal yang penulis sarankan untuk pengembangan selanjutnya yaitu untuk penelitian selanjutnya algoritma ini dapat diterapkan pada algoritma lain. Dengan cara kerja apabila ukuran dari kendaraan melebihi ukuran kendaraan secara umum, maka algoritma ini diterapkan untuk mengecek apakah objek tersebut terkoneksi oleh *shadow* atau tidak.

DAFTAR PUSTAKA

Al., Bovik. 2000. "*Handbook of Image and Video Processing*". Academic Press Publisher. San Diego.

C., Stauffer, W., E., L., Grimson. 1999. "*Adaptive Background Mixture Models for Real-time Tracking*". The Artificial Intelligence Laboratory. Cambridge.

Daigavane, P. M., Bajaj, P. R. 2010. "*Real Time Vehicle Detection and Counting Method for Unsupervised Traffic Video on Highways*". *International Journal of Computer Science and Network Security*.Volume 10 No.8.

Gupta, P., Purohit, G. N., Gupta, A. 2013. "Traffic Load Computation using Corner Detection Technique in Matlab Simulink Model".*International Journal of Computer Applications (0975 8887)*.Volume 72 No.16.

Kaushik, D., Suny, A., H. 2014. "Shadow Detection and Removal Based on YCbCr Color Space".*Smart Computing Review, vol. 4, no. 1*.

Kovacic, K., Ivanjko, E., H., Gold H. G. 2014. "Real Time Vehicle Detection and Tracking on Multiple Lanes".*Conference on Computer Graphics, Visualization and Computer Vision*.

Murat, Tekalp, A,. 2002."*Digital Video Processing*". Pearson Education, Inc. New Jersey.

Rafael, C., Gonzales, Richard, E., Woods. 2002."*Digital Image Processing*". Tom Robbins Publisher. United States of America.

Solomon, C., Breckon, T. 2013. "*Fundamental of Digital Image Processing*". John Wiley and Sons, Ltd. Chichester, UK.

BIODATA PENULIS

Penulis bernama lengkap Reza Augusta Jannatul Firdaus, dilahirkan di Kab. Jombang pada tanggal 20 Agustus 1991. Merupakan anak kedua dari tiga bersaudara dari pasangan Bapak Muslim dan ibu Rina Herdiani. Penulis telah menempuh pendidikan formal yaitu di SD Tugu Kepatihan 2 Jombang pada tahun 1997 dan lulus pada tahun 2003 selanjutnya SMP Negeri 2 Jombang pada tahun 2003-2006, SMAN 2 Jombang pada tahun 2006-2009. Setelah itu penulis melanjutkan pendidikan sarjana di Jurusan Matematika FMIPA, Universitas Brawijaya Malang pada tahun 2009 dan lulus pada tahun 2013. Pada tahun 2014 penulis melanjutkan kuliah di S2 Matematika ITS Surabaya. Di jurusan Matematika penulis mengambil bidang minat Ilmu Komputer, dengan judul penelitian ”Pengembangan Algoritma Robust untuk Menghitung Kendaraan Bergerak Berbasis Pengolahan Citra”. Informasi, kritik, dan saran yang berhubungan dengan Tesis ini dapat ditujukan ke alamat e-mail: `reza.8th@gmail.com`