TESIS - SM 142501

# KLASIFIKASI KENDARAAN BERGERAK DENGAN LOGIKA *FUZZY* BERBASIS PENGOLAHAN CITRA

BAYU CHARISMA PUTRA
NRP 1214 201 013

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

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

THESIS - SM 142501

# MOVING VEHICLE CLASSIFICATION WITH FUZZY LOGIC BASED ON IMAGE PROCESSING

BAYU CHARISMA PUTRA
NRP 1214 201 013

Supervisor:
Dr. Budi Setiyono S.Si, M.T
Dr. Imam Mukhlash S.Si, M.T

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

# KLASIFIKASI KENDARAAN BERGERAK DENGAN LOGIKA *FUZZY* BERBASIS PENGOLAHAN CITRA

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

**oleh:**
**BAYU CHARISMA PUTRA**
**NRP. 1214 201 013**

**Tanggal Ujian : 18 Juli 2016**
**Periode Wisuda : September 2016**

**Disetujui oleh:**

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

**Dr. Imam Mukhlash, S.Si., M.T.** **(Pembimbing II)**
**NIP. 19700631 199403 1 003**

**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. Dieky Adzkiya, S.Si., M.Si.** **(Penguji)**
**NIP. 19830517 200812 1 003**

**Direktur Program Pascasarjana,**

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

# KLASIFIKASI KENDARAAN BERGERAK DENGAN LOGIKA *FUZZY* BERBASIS PENGOLAHAN CITRA


| | | |
|---|---|---|
| Nama Mahasiswa | : | Bayu Charisma Putra |
| NRP | : | 1214 201 013 |
| Pembimbing | : | 1. Dr. Budi Setiyono S.Si, M.T |
| | | 2. Dr. Imam Mukhlash S.Si, M.T |


## ABSTRAK


Transportasi memiliki peranan penting bagi kehidupan baik dari sisi ekonomi, sosial, politik, dan banyak lainnya. Salah satu permasalahan transportasi yang sering muncul adalah kemacetan lalu lintas. Untuk mengurangi kemacetan dibutuhkan informasi-informasi yang terkait dengan lalu lintas, salah satunya jenis kendaraan yang melintas. Pada penelitian ini dibahas klasifikasi dan penghitungan jenis kendaraan menggunakan logika *fuzzy* berbasis pengolahan citra. Tahap pertama, *input* berupa rekaman video diekstrak menjadi *frame-frame*. *Frame* tersebut dicari *foreground*-nya menggunakan metode *Gaussian Mixture Model*. Kemudian *frame* di-*filter* untuk meminimalkan *noise*. Tahap selanjutnya adalah *shadow removal* yaitu menghilangkan bayangan yang terdeteksi di *foreground*, Setelah itu obyek akan dideteksi dengan memeriksa seluruh tetangga setiap piksel. Obyek yang terdeteksi diklasifikasi menggunakan logika *fuzzy* lalu diberi label sesuai jenis kendaraan. Tahap terakhir adalah *tracking object* dan menghitung kendaraan yang melewati ROI (*Region of Interest*) sesuai dengan jenis kendaraannya. Dari hasil uji coba pada penelitian ini algoritma yang dibangun mampu mengklasifikasikan jenis kendaraan dengan tingkat akurasi sebagai berikut; Jalan dengan kondisi lengang yaitu Jalan Kedung Cowek 100% dan Jalan Wonokromo 97.67%, Jalan dengan kondisi padat yaitu Jalan Raya Diponegoro 92.21%, dan Jalan Pemuda 84.78%.



**Kata kunci:** Kemacetan Lalu Lintas, *Gaussian Mixture Model*, *Shadow Removal*, Deteksi Obyek, Klasifikasi, Logika *Fuzzy*.

# MOVING VEHICLE CLASSIFICATION WITH FUZZY LOGIC BASED ON IMAGE PROCESSING


| | | |
|---|---|---|
| Name | : | Bayu Charisma Putra |
| NRP | : | 1214 201 013 |
| Supervisors | : | 1. Dr. Budi Setiyono S.Si, M.T |
| | | 2. Dr. Imam Mukhlash S.Si, M.T |


## ABSTRACT


Transportation has a vital part in terms of economy, social, politics, and many others. One of transportation problems that happens almost everyday is the traffic jam. In order to reduce that problem, one of informations that is needed is the kind of vehicle that passed the street. This research discussed counting and classification of vehicles using fuzzy logic-based on image processing. The first step, input in the form of recorded video is extracted into frames. Then, the foreground from the extracted frames is determined by GMM (Gaussian Mixture Model) methods. In the next step, the obtained foreground were filtered using median filter in order to reduce noises. The next one is the shadow removal process which removes the shadow that is detected in the foreground, then the object will be detected by examining the whole neighborhood of each pixel. Then the detected and classified object are labeled according to the type of vehicle. The last step is tracking object and counting vehicles that pass the ROI (Region of Interest) according the type of vehicle. From the obtained results this algorithm is capable on classifying type of vehicles with a high degree of accuracy as follows; The road to the deserted condition Kedung Cowek street with 100% and Wonokromo street with 97.67%, The road with high traffic density Raya Diponegoro street with 92.21% and Pemuda street with 84.78%.



**Keywords:** Traffic Jam, Gaussian Mixture Model, Shadow Removal, Object Detection, Classification, Fuzzy Logic.

# DAFTAR ISI


HALAMAN JUDUL i

LEMBAR PENGESAHAN v

ABSTRAK vii

ABSTRACT ix

KATA PENGANTAR xi

DAFTAR ISI xiii

DAFTAR GAMBAR xvii

DAFTAR TABEL xxi

BAB I PENDAHULUAN 1

1.1 Latar Belakang . . . . . 1

1.2 Rumusan Masalah . . . . . 2

1.3 Batasan Masalah . . . . . 2

1.4 Tujuan Penelitian . . . . . 3

1.5 Manfaat Penelitian . . . . . 3

BAB II KAJIAN PUSTAKA DAN DASAR TEORI 5

2.1 Penelitian Sebelumnya . . . . . 5

2.2 Citra . . . . . 6

2.3 Citra Digital . . . . . 6

2.3.1 Digitalisasi Spasial (*Sampling*) . . . . . 6

2.3.2 Digitalisasi Intensitas (Kuantisasi) . . . . . 7

2.4 Video Digital . . . . . 8

2.5 *Background Subtraction* . . . . . 8

2.6 Metode *Gaussian Mixture Model* . . . . . 9

2.7 *Smoothing* . . . . . 11

2.8 *Shadow Removal* . . . . . 12

2.9 Deteksi Obyek . . . . . 12

2.10 Logika *Fuzzy* . . . . . 12

2.11 Fungsi Keanggotaan *Fuzzy* . . . . . 13

2.12 Sistem Inferensi Mamdani . . . . . 15
BAB III METODE PENELITIAN 17
3.1 Obyek Penelitian . . . . . 17
3.2 Peralatan . . . . . 17
3.3 Tahapan Penelitian . . . . . 17
BAB IV PERANCANGAN DAN IMPLEMENTASI SISTEM 21
4.1 Analisis Sistem . . . . . 21
4.1.1 Deskripsi Sistem . . . . . 21
4.1.2 Analisis Kebutuhan Sistem . . . . . 23
4.2 Perancangan Sistem . . . . . 23
4.2.1 Perancangan Data Sistem . . . . . 24
4.2.2 Perancangan Algoritma Sistem . . . . . 26
4.2.3 Perancangan Antar Muka Sistem . . . . . 46
4.3 Implementasi Sistem . . . . . 46
4.3.1 Implementasi Akuisisi Video . . . . . 46
4.3.2 Implementasi *Input* Video . . . . . 47
4.3.3 Implementasi Pemilihan Area ROI . . . . . 47
4.3.4 Implementasi Proses *Background Subtraction* . . . . . 48
4.3.5 Implementasi Proses *Smoothing* . . . . . 48
4.3.6 Implementasi Proses *Shadow Removal* . . . . . 49
4.3.7 Implementasi Proses Deteksi Obyek Bergerak . . . . . 49
4.3.8 Implementasi Proses Klasifikasi Kendaraan Bergerak . . . . . 50
4.3.9 Implementasi Antar Muka Sistem . . . . . 51
BAB V UJI COBA DAN ANALISIS 53
5.1 Uji Coba Deteksi obyek . . . . . 53
5.2 Uji Coba Klasifikasi Kendaraan Bergerak . . . . . 56
5.3 Uji Coba Klasifikasi Berdurasi Panjang . . . . . 60
BAB VI KESIMPULAN DAN SARAN 63
6.1 Kesimpulan . . . . . 63
6.2 Saran . . . . . 64
DAFTAR PUSTAKA 65
LAMPIRAN 67
A Kode Program Deteksi Obyek 69
B Kode Program Klasifikasi Kendaraan Bergerak 71

BIOGRAFI 77

# DAFTAR TABEL


Tabel 4.1 Tabel kebutuhan sistem . . . . . 24
Tabel 4.2 Tabel data proses . . . . . 25
Tabel 4.3 Tabel Parameter *Region of Interest* . . . . . 26
Tabel 4.4 Tabel Parameter Median *Filter* . . . . . 26
Tabel 4.5 Tabel Parameter Algoritma Deteksi Obyek . . . . . 26
Tabel 4.6 Tabel Parameter Logika *Fuzzy* . . . . . 26
Tabel 4.7 Tabel jumlah deteksi obyek benar tiap jalan . . . . . 37
Tabel 4.8 Tabel rata-rata jumlah piksel dan luas area tiap jalan . . . . . 40

Tabel 5.1 Tabel jumlah deteksi obyek benar tiap jalan . . . . . 54
Tabel 5.2 Tabel total kendaraan sebenarnya . . . . . 58
Tabel 5.3 Tabel jumlah klasifikasi benar dari program . . . . . 59
Tabel 5.4 Tabel hasil klasifikasi durasi panjang. . . . . 60

# DAFTAR GAMBAR


Gambar 2.1 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. . . . . 7

Gambar 2.2 Proses *sampling* citra dari citra kontinu ke citra digital. . . . . 7

Gambar 2.3 Proses *Background Subtraction* dengan metode *frame differencing* untuk mendapatkan *foreground image*. . . . . 8

Gambar 2.4 Diagram Alir Metode GMM. . . . . 10

Gambar 2.5 Proses *smoothing* dengan median *filter*. . . . . 12

Gambar 2.6 Fungsi keanggotaan linear naik. . . . . 13

Gambar 2.7 Fungsi keanggotaan linear turun. . . . . 13

Gambar 2.8 Fungsi keanggotaan segitiga. . . . . 14

Gambar 2.9 Fungsi keanggotaan trapesium. . . . . 14

Gambar 3.1 Blok diagram klasifikasi kendaraan. . . . . 19

Gambar 4.1 Tahapan dalam klasifikasi. . . . . 23

Gambar 4.2 Perancangan akuisisi video. . . . . 24

Gambar 4.3 Diagram alir proses pada sistem. . . . . 27

Gambar 4.4 Contoh inisialisasi awal parameter GMM. . . . . 28

Gambar 4.5 Contoh matriks intensitas dari *frame* yang terkestrak. . . . . 29

Gambar 4.6 Matriks selisih intensitas *frame input* dan *mean*. . . . . 29

Gambar 4.7 Hasil *update* parameter GMM. . . . . 30

Gambar 4.8 Hasil normalisasi bobot. . . . . 30

Gambar 4.9 Hasil perhitungan $\frac{\omega}{\sigma^2}$. . . . . 30

Gambar 4.10 Contoh hasil dari proses background subtraction. . . . . 31

Gambar 4.11 *Scale* dengan 8 ketetanggaan pada proses *smoothing*. Kiri : *scale* = 1. Kanan : *scale* = 2. . . . . 31

Gambar 4.12 *Scale* dengan 8 ketetanggaan pada piksel *border*. Kiri : piksel *border* kanan akan mencari nilai median dari 6 piksel berdasarkan *scale*. Kanan : piksel *border* kanan atas akan mencari nilai median dari 4 piksel berdasarkan *scale*. . . . . 32
Gambar 4.13 Proses *smoothing* dengan median *filter*. . . . . 32
Gambar 4.14 Proses *shadow removal*. . . . . 33
Gambar 4.15 Tahapan pada deteksi obyek. Kiri : arah pencarian piksel suatu obyek. Kanan : memulai proses pencarian piksel lainnya pada obyek. . . . . 34
Gambar 4.16 *Scale* pada deteksi obyek. Kiri : *scale* deteksi obyek = 1. Kanan : *scale* deteksi obyek = 2. . . . . 34
Gambar 4.17 Proses pencarian tetangga pada deteksi obyek. . . . . 35
Gambar 4.18 Contoh *image* yang akan dideteksi obyek. . . . . 35
Gambar 4.19 Perbedaan hasil berdasarkan *scale*. Kiri : deteksi dengan *scale* = 1. Tengah : deteksi dengan *scale* = 3. Kanan : deteksi dengan *scale* = 7. . . . . 36
Gambar 4.20 Prosentase deteksi obyek pada *scale* 1 sampai 8. . . . . 37
Gambar 4.21 *Output* dari proses deteksi obyek. Kiri : jumlah piksel pada obyek adalah 6. Kanan : luas area pada obyek adalah 9. 38
Gambar 4.22 Variabel pada himpunan *fuzzy*. . . . . 38
Gambar 4.23 Inisialisasi himpunan *fuzzy input*. . . . . 40
Gambar 4.24 Inisialisasi himpunan *fuzzy* bobot ukuran. . . . . 40
Gambar 4.25 Aturan *fuzzy*. . . . . 41
Gambar 4.26 Contoh pencarian nilai keanggotaan dengan jumlah piksel = $x$. . . . . 41
Gambar 4.27 Contoh pencarian nilai keanggotaan dengan luas area = $y$. . . 41
Gambar 4.28 Contoh fungsi implikasi antara jumlah piksel sedikit dan luas area sedikit dengan aturan bobot ukuran sedikit. . . . . 42
Gambar 4.29 Contoh komposisi aturan variabel sedikit. . . . . 42
Gambar 4.30 Contoh Daerah hasil masing-masing variabel. . . . . 43
Gambar 4.31 Contoh komposisi aturan setiap variabel. . . . . 43
Gambar 4.32 Contoh klasifikasi : pembentukan himpunan *fuzzy*. Kiri : himpunan jumlah piksel. Kanan : himpunan luas area. . . . . 44
Gambar 4.33 Contoh klasifikasi : komposisi aturan setiap variabel. . . . . 44
Gambar 4.34 Contoh klasifikasi : pembagian daerah hasil. . . . . 45

Gambar 4.35 Masing-masing batas pada obyek yang berbeda. Kiri : obyek pada $frame_{t-1}$. Kanan : obyek pada $frame_t$. . . . . 45
Gambar 4.36 Desain antar muka sistem. . . . . 46
Gambar 4.37 Posisi kamera saat akuisisi video. . . . . 47
Gambar 4.38 Parameter pemilihan ROI. . . . . 47
Gambar 4.39 Hasil *background subtraction*. . . . . 48
Gambar 4.40 Hasil *smoothing*. . . . . 48
Gambar 4.41 Hasil *shadow removal*. . . . . 49
Gambar 4.42 Hasil deteksi obyek. . . . . 50
Gambar 4.43 Hasil klasifikasi. . . . . 50
Gambar 4.44 Hasil proses dari sistem. . . . . 51
Gambar 4.45 Antar muka halaman utama. . . . . 51
Gambar 4.46 Antar muka halaman *input*. . . . . 52
Gambar 4.47 Antar muka halaman hasil. . . . . 52

Gambar 5.1 Faktor volume lalu lintas. . . . . 54
Gambar 5.2 Faktor kondisi cuaca. . . . . 55
Gambar 5.3 Faktor pantulan cahaya. . . . . 55
Gambar 5.4 Faktor warna kendaraan. . . . . 56
Gambar 5.5 Faktor bayangan. . . . . 56
Gambar 5.6 Kendaraan dengan ukuran jenis sepeda motor. . . . . 57
Gambar 5.7 Kendaraan dengan ukuran jenis mobil. . . . . 57
Gambar 5.8 Kendaraan dengan ukuran jenis mini bis / mini truk. . . . . 57
Gambar 5.9 Kendaraan dengan ukuran jenis bis / truk . . . . . 57
Gambar 5.10 Perbedaan klasifikasi terhadap jarak. Kiri : mini truk dianggap motor. Tengah : mini truk dianggap mobil. Kanan : klasifikasi tepat. . . . . 58
Gambar 5.11 Tingkat keakuratan klasifikasi kendaraan pada program. . . . 59
Gambar 5.12 Tingkat Keakuratan klasifikasi program pada video berdurasi panjang. . . . . 60
Gambar 5.13 Faktor obyek selain kendaraan. . . . . 61
Gambar 5.14 Faktor ukuran yang tidak wajar. . . . . 62
Gambar 5.15 Faktor bayangan dalam klasifikasi. . . . . 62

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Kemacetan lalu lintas adalah masalah transportasi yang sangat krusial yang dihadapi hampir di semua kota besar. Untuk mengurangi kemacetan tersebut dibutuhkan informasi-informasi yang terkait dengan lalu lintas, salah satunya adalah jenis kendaraan yang melintas. Klasifikasi jenis kendaraan dapat berfungsi sebagai data dasar untuk membuat kebijakan rekayasa lalu lintas untuk pemeliharaan jalan dan membangun jalan yang baru. Dari sudut pandang manusia, pengenalan pola untuk klasifikasi terlihat mudah. Namun yang menjadi permasalahan adalah tingkat kesulitan yang dihadapi ketika dilakukan secara otomatis oleh mesin.

Pesatnya perkembangan teknologi di semua aspek seperti saat ini sangat mendukung penyelesaian masalah secara komputasi. Dengan menggunakan data *input* berupa video, klasifikasi kendaraan dengan menggunakan pengolahan citra dapat menjadi pilihan yang akan menghemat tenaga, waktu, dan biaya. Video akan dijadikan sebagai sumber untuk memperoleh obyek yang bergerak dengan algoritma *Background Subtraction*. Rosenberg C melakukan perbandingan dari 7 metode *Background Subtraction*; *Basic*, *One Gaussian* (1-G), *MinMax Inter-Frame Difference*, *Gaussian Mixture Model* (GMM), *Kernel Density Estimation* (KDE), *Codebook* (CBRGB), dan *Eigen Backgrounds* (Eigen). Dari penelitan tersebut diperoleh kesimpulan bahwa metode GMM memiliki tingkat akurasi yang tinggi dengan berbagai kondisi *input* video; *static*, *multimodal*, maupun *noise background* serta memiliki waktu komputasi dan kebutuhan memori yang lebih efisien dibandingkan metode lainnya (Benezith dkk, 2012).

Pada penelitian-penelitian sebelumnya, klasifikasi kendaraan bergerak dilakukan dengan berbagai macam metode, antara lain metode *Fuzzy Cluster Means*, *Probabilistic Neural Network*, *SVM Classifier*, *Geometric Invariant Moment*, dan *Fuzzy Logic*. Dalam penelitian yang dilakukan Jailson A. dan Luis Edmundo Prando dalam jurnalnya yang berjudul *"Automatic Vehicle Classification Using Learning-based Computer Vision and Fuzzy Logic"*, akuisisi video diambil dari sisi samping kendaraan, dan parameter *fuzzy* adalah jumlah roda dan jarak roda. Hal tersebut dapat menyebabkan kesalahan klasifikasi jika terdapat kendaraan lain yang melintas di samping kendaraan yang akan diklasifikasi. Dari paparan

di atas, penulis akan melakukan penelitian klasifikasi kendaraan bergerak dengan mempertimbangkan jumlah piksel dan luas *square* dari obyek dengan logika *fuzzy* dimana data video diambil dari atas. Klasifikasi dalam penelitian ini berbasis pada citra, sehingga kendaraan bergerak akan diklasifikasi berdasarkan ukuran pada citra.

Adapun obyek yang akan diklasifikasi diperoleh dari beberapa langkah yaitu *background subtraction*, *smoothing*, *shadow removal*, dan *object detection*. *Background subtraction* adalah cara untuk memperoleh *foreground* tiap *frame* dengan menggunakan metode *Gaussian Mixture Model*. *Smoothing* adalah tahap untuk mengurangi kontras citra dengan menggunakan metode median *filter*. *Shadow removal* adalah tahap menghilangkan bayangan yang terdeteksi pada *foreground*. *Object detection* adalah tahap pengambilan obyek kendaraan pada ROI (*Region of Interest*) yang akan diklasifikasi.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, pokok permasalahan yang dikaji dalam penelitian ini adalah sebagai berikut:

1. Bagaimana melakukan pendeteksian obyek dengan memanfaatkan *output* dari *Gaussian Mixture Model*.

2. Bagaimana mengklasifikasi kendaraan bergerak menggunakan logika *fuzzy* berbasis pengolahan citra.

## 1.3 Batasan Masalah

Penelitian ini difokuskan pada pembahasan dengan beberapa batasan masalah sebagai berikut:

1. Data yang digunakan pada eksperimen berupa rekaman video *offline* yang diambil di jalan protokol satu arah di Surabaya pada kondisi jalan lengang dan padat dengan kondisi cuaca cerah.

2. Kamera yang digunakan untuk merekam berada dalam posisi diam atau tidak bergerak.

3. Sudut pandang kamera pada saat merekam adalah dari atas dan menjangkau seluruh lebar ruas jalan.

4. Mengklasifikasikan jenis kendaraan dalam jenis berdasarkan ukuran, yaitu sepeda motor, mobil, mini bus/truk, bus/kontainer.

## 1.4 Tujuan Penelitian

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

1. Melakukan pendeteksian obyek dengan memanfaatkan *output* dari *Gaussian Mixture Model*.

2. Mengklasifikasi kendaraan bergerak menggunakan logika *fuzzy* berbasis pengolahan citra.

## 1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah memberikan informasi jumlah dan jenis kendaraan yang melintas pada jalan protokol satu arah sehingga dapat membantu instansi terkait dalam merumuskan kebijakan rekayasa lalu lintas. Selain itu penelitian ini juga diharapkan dapat memberikan kontribusi dalam pengembangan algoritma untuk klasifikasi jenis kendaraan.

# BAB II

# KAJIAN PUSTAKA DAN DASAR TEORI

## 2.1 Penelitian Sebelumnya

Terdapat beberapa penelitian sebelumnya pada klasifikasi kendaraan dengan berbagai macam metode. Paulraj P. A. dkk dalam jurnalnya yang berjudul *"Moving Vehicle Recognition and Classification Based on Time Domain Approach"* melakukan klasifikasi kendaraan bergerak dengan metode PNN (*Probabilistic Neural Network*) berbasis sinyal suara dan berhasil mengklasifikasikan kendaraan dengan tingkat akurasi tertinggi sebesar 92,8%. Kemudian penelitian tersebut dikembangkan oleh N. Abdul Rahim dkk dalam jurnalnya yang berjudul *"Adaptive Boosting with SVM Classifier for Moving Vehicle Classification"* yang melakukan klasifikasi kendaraan bergerak dengan metode (*Support Vector Machine Classifier*) berbasis sinyal suara.

Fitroh Amaluddin dalam jurnalnya yang berjudul *"Klasifikasi Kendaraan Menggunakan Gaussian Mixture Model (GMM) dan Fuzzy Cluster Means (FCM)"* berhasil mengklasifikasikan kendaraan dengan tingkat keakurasian tertinggi sebesar 91,3% yaitu pada sore hari.

Sung-Wook Kim dkk dalam jurnalnya yang berjudul *"Application of Fuzzy Logic to Vehicle Classification Algorithm in Loop/Piezo-Sensor Fusion System"* mengklasifikasi kendaraan menggunakan logika *fuzzy* dengan kecepatan dan berat sebagai *input* dari *fuzzy*. Dalam penelitian tersebut disimpulkan bahwa klasifikasi kendaraan menggunakan logika *fuzzy* dapat mengurangi *error* dari klasifikasi tanpa logika *fuzzy* sebesar 6,56%. Sedangkan Jailson A. dan Luis Edmundo Prando dalam jurnalnya yang berjudul *"Automatic Vehicle Classification Using Learning-based Computer Vision and Fuzzy Logic"* juga mengklasifikasikan kendaraan menggunakan logika *fuzzy*, namun yang menjadi *input* dari *fuzzy* adalah jumlah roda dan jarak roda.

Pada penelitian ini kendaraan bergerak diklasifikasikan dengan menggunakan logika *fuzzy* dengan jumlah piksel dan luas area sebagai *input* dari logika *fuzzy*. Pada subbab selanjutnya akan dijelaskan dasar teori yang digunakan dalam penelitian ini.

## 2.2 Citra

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 penyimpan (Sutoyo dkk, 2009).

Citra dapat dikelompokkan menjadi dua bagian 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.

Manusia pada awalnya mengenal citra dalam bentuk citra kontinu. Citra kontinu merupakan 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, seperti mata manusia dan kamera analog. Citra kontinu tidak dapat direpresentasikan secara langsung oleh komputer. Oleh sebab 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.

## 2.3 Citra Digital

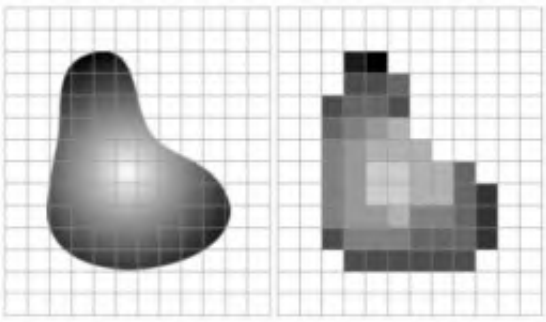
Citra Digital dapat diasumsikan sebagai fungsi dua variabel yang dapat dinyatakan dengan fungsi $f(x, y)$, dimana $x$ dan $y$ merupakan koordinat spasial. Fungsi $f$ pada sembarang pasangan koordinat $(x, y)$ merupakan nilai intensitas cahaya, representasi dari warna cahaya yang ada pada citra kontinu. Ketika $(x, y)$ dan nilai intensitas dari $f$ terbatas (*discrete quantities*), dan telah dilakukan proses digitalisasi spasial dan digitalisasi kuantitas maka citra itu disebut sebagai citra digital (Gonzales dan Woods, 2001). Proses digitalisasi spasial (*sampling*) dan digitalisasi kuantitas disajikan pada Gambar 2.1.

### *2.3.1 Digitalisasi Spasial (Sampling)*

*Sampling* merupakan proses pengambilan informasi dari citra kontinu yang memiliki panjang dan lebar tertentu untuk membaginya ke beberapa blok kecil yang disebut sebagai piksel, sehingga citra digital yang lazim dinyatakan dalam bentuk matriks memiliki ukuran $(M \times N)$ dengan $M$ sebagai baris dan $N$ sebagai kolom, atau bisa juga disebut sebagai citra digital yang memiliki $(M \times N)$ buah piksel.

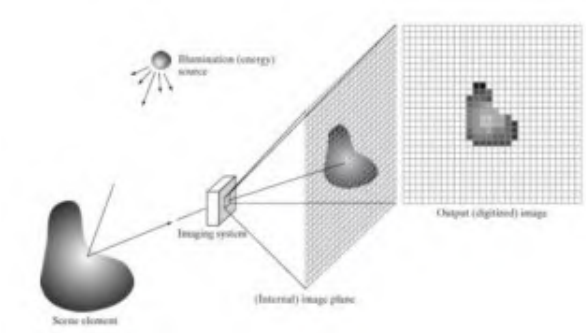
Notasi matriks citra digital dapat dinyatakan pada persamaan 2.1.

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



Gambar 2.1: 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. (Gonzales dan Woods, 2001).

Proses *sampling* citra kontinu ke citra digital ditampilkan pada Gambar 2.2.



Gambar 2.2: Proses *sampling* citra dari citra kontinu ke citra digital. (Gonzales dan Woods, 2001).

### *2.3.2 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.

## 2.4 Video Digital

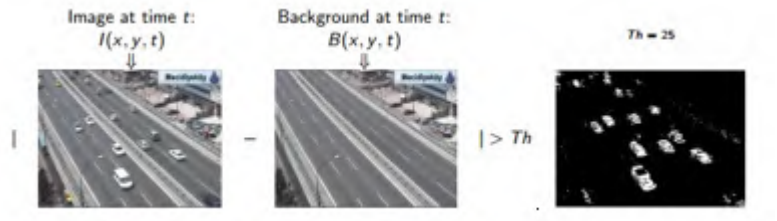
Video adalah teknologi untuk menangkap, merekam, memproses, menyimpan, dan merekonstruksi suatu urutan dari beberapa citra. Video digital merupakan hasil *sampling* dan kuantisasi dari video analog (Bovik, 2000). Proses *sampling* dan kuantisasi pada video digital sama dengan proses *sampling* dan kuantisasi pada citra digital.

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* (Solomon dan Breckon, 2013). Dari paparan tersebut menunjukan bahwa semakin besar fps maka video yang terlihat akan 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.

## 2.5 *Background Subtraction*

*Background Subtraction* adalah proses untuk mendapatkan *foreground object* (obyek yang bergerak) dari serangkaian citra. *Foreground* diperoleh dengan melakukan pengurangan *frame* pada waktu $t$ dengan *background* model dari video, kemudian hasil pengurangan tersebut dibandingkan dengan suatu nilai ambang batas (*threshold*) tertentu. Contoh proses *background subtraction* diberikan pada Gambar 2.3.




Gambar 2.3: Proses *Background Subtraction* dengan metode *frame differencing* untuk mendapatkan *foreground image*.

Secara matematis proses tersebut dapat dinyatakan dalam persamaan 2.2 dan persamaan 2.3.

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

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

dengan $I(x,y,t)$ merupakan intensitas dari citra $(x,y)$ pada *frame* $t$, $B(x,y,t)$ merupakan intensitas dari *background* citra $(x,y)$ pada *frame* $t$, $d(x,y,t)$ merupakan hasil pengurangan intensitas dari *background* citra terhadap intensitas dari citra pada *frame* $t$, $\tau$ merupakan ambang batas atau *threshold*, dan $f(x,y,t)$ adalah intensitas dari *foreground* citra.

## 2.6 Metode *Gaussian Mixture Model*

*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, 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). Prosedur dari metode *Gaussian Mixture Model* disajikan pada Gambar 2.4.

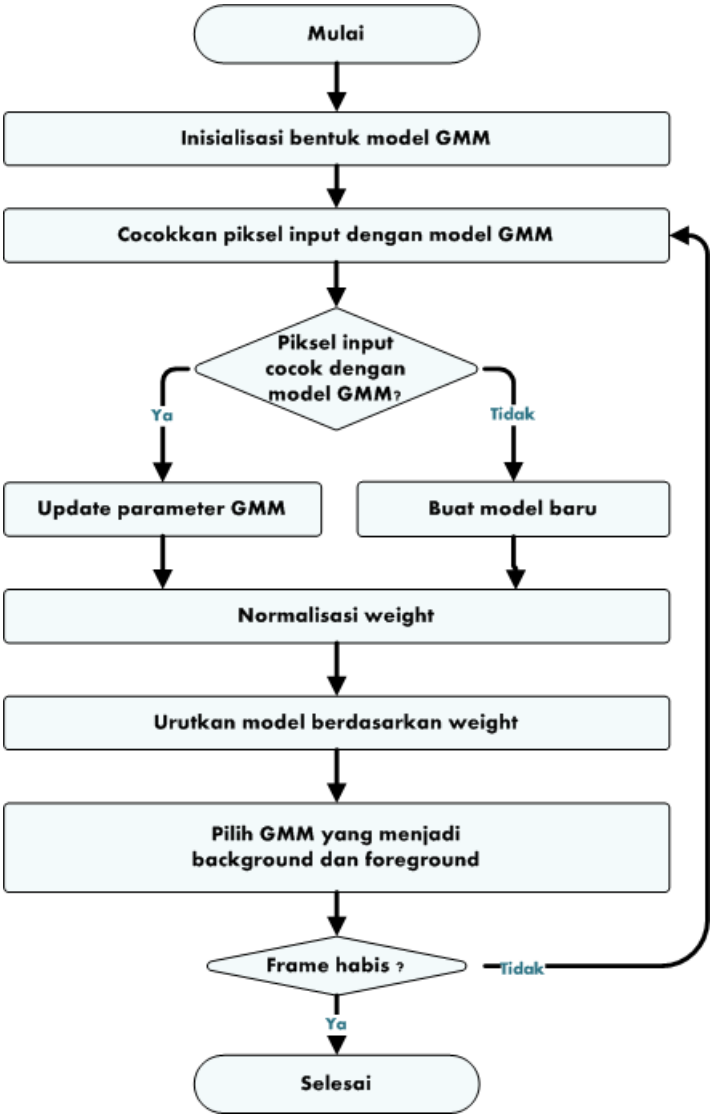
Menurut (Amaluddin, 2015) terdapat beberapa tahapan dalam pemilihan distribusi yang mencerminkan *background*. Tahapan-tahapan tersebut adalah :

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.

   $$\mu_k - 2.5 * \sigma_k < X_t < \mu_k + 2.5 * \sigma_k \tag{2.4}$$

   dimana $X_t$ adalah nilai intensitas dari suatu piksel $(i,j)$ pada *frame* ke- $t$, $\mu_k$ adalah nilai *mean* pada piksel $(i,j)$ dari gaussian ke-$k$, dan $\sigma_k$ sebagai standar deviasi pada piksel $(i,j)$ dari gaussian ke-$k$.




Gambar 2.4: Diagram Alir Metode GMM.

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 $\rho$ diberikan pada 2.6. Nilai *standard deviation* di-*update* menggunakan persamaan 2.8. Setiap persamaan tersebut berlaku pada setiap piksel $(i, j)$.

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

$$\rho = \frac{\alpha}{\omega_{k,t}} \tag{2.6}$$

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

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

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* $t$, $\alpha$ 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.

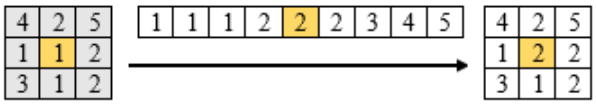
3. Pemilihan Distribusi *Background*

   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.

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

## 2.7 *Smoothing*

*Smoothing* merupakan proses mengurangi *noise* pada citra. Pada penelitian ini proses *smoothing* menggunakan metode median *filter*. Prinsip kerja dari metode median *filter* adalah dengan cara mengganti nilai intensitas dari tiap piksel dengan nilai median dari piksel-piksel tetangganya. Pola jangkauan piksel-piksel tetangga disebut dengan *window* yang mana telah ditentukan sebelumnya. Nilai median dihitung dengan melakukan *sorting* pada bagian *window* dan piksel itu sendiri. Hasil *output* dari *smoothing* adalah *frame* dengan intensitas yang diperoleh dari nilai-nilai median tersebut. Proses tersebut dapat digambarkan pada Gambar 2.5.




Gambar 2.5: Proses *smoothing* dengan median *filter*.

## 2.8 *Shadow Removal*

*Shadow removal* adalah proses dimana bayangan yang terdeteksi di *foreground* akan dihilangkan. Pada *shadow removal output* dari GMM yang berupa citra *grayscale* yang hanya memiliki 3 jenis intensitas yaitu 0 untuk obyek diam, 127 untuk obyek bayangan, dan 255 untuk obyek bergerak, akan diubah menjadi hanya 2 jenis intensitas atau biasa disebut biner yaitu 0 untuk obyek diam dan bayangan, dan 1 untuk obyek bergerak.

## 2.9 Deteksi Obyek

Pada penelitian ini, algoritma deteksi obyek yang digunakan adalah *pixel connected* yaitu dengan memeriksa semua piksel tetangganya dengan radius tertentu di dalam ROI (*Region of Interest*) yang sebelumnya telah ditentukan. Terdapat dua proses pada deteksi obyek yaitu mencari salah satu piksel dari suatu obyek yang bergerak dan mencari seluruh piksel dari obyek tersebut secara rekursif.

## 2.10 Logika *Fuzzy*

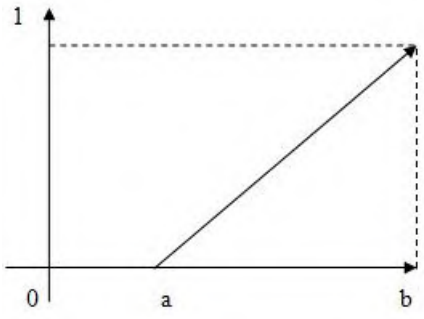
Logika *fuzzy* pertama dikenalkan oleh Prof. Lotfi A. Zadeh pada tahun 1965. Logika *fuzzy* merupakan suatu metode pengambilan keputusan berbasis aturan yang digunakan untuk memecahkan masalah *grayness* pada sistem yang sulit dimodelkan atau memiliki ambiguitas. Tujuan awal dari logika *fuzzy* adalah untuk memungkinkan nilai-nilai kebenaran menjadi nilai di interval $[1, 0]$ (Buckley dan Eslami, 2002).

Dasar logika *fuzzy* adalah teori himpunan *fuzzy*. Pada himpunan *crisp* nilai keanggotaan suatu *item* $x$ dalam suatu himpunan $A$ yang dituliskan dengan $[x]$, dimana memiliki dua buah kemungkinan nilai yaitu 1 yang memiliki arti bahwa suatu *item* menjadi anggota dalam himpunan $A$ dan 0 yang memiliki arti bahwa suatu *item* tidak menjadi anggota dalam himpunan $A$. Namun penggunaan himpunan *crisp* tidak cocok pada beberapa hal karena perubahan kecil saja pada suatu nilai mengakibatkan perbedaan kategori yang cukup signfikan. Untuk mengantisipasi hal tersebut maka digunakan himpunan *fuzzy*. Himpunan *fuzzy* merupakan suatu grup yang mewakili suatu kondisi atau keadaan tertentu dalam suatu variabel *fuzzy*. Variabel *fuzzy* merupakan variabel yang hendak dibahas dalam suatu sistem *fuzzy*.

## 2.11 Fungsi Keanggotaan *Fuzzy*

Fungsi keanggotaan *fuzzy* adalah suatu kurva yang menunjukkan pemetaan titik-titik *input* data ke dalam derajat keanggotaannya yang nilainya berkisar antara 0 hingga 1. Berikut beberapa contoh fungsi keanggotaan *fuzzy*.

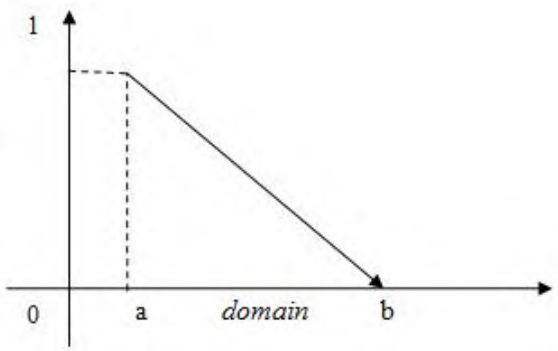
1. Fungsi keanggotaan linear naik
   Contoh dari fungsi keanggotaan linear naik diberikan pada Gambar 2.6 dimana nilai keanggotaannya diberikan pada persamaan 2.10.




Gambar 2.6: Fungsi keanggotaan linear naik.

$$\mu[x,a,b] = \begin{cases} 0, & x \leq a \\ \frac{(x-a)}{(b-a)}, & a < x < b \\ 1, & x \geq b \end{cases} \tag{2.10}$$

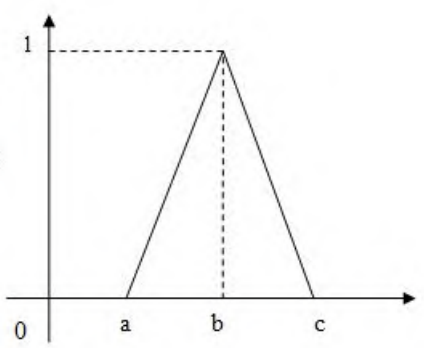
2. Fungsi keanggotaan linear turun
   Contoh dari fungsi keanggotaan linear turun diberikan pada Gambar 2.7 dimana nilai keanggotaannya diberikan pada persamaan 2.11.




Gambar 2.7: Fungsi keanggotaan linear turun.

$$\mu[x,a,b] = \begin{cases} 1, & x \leq a \\ \frac{(b-x)}{(b-a)}, & a < x < b \\ 0, & x \geq b \end{cases} \tag{2.11}$$

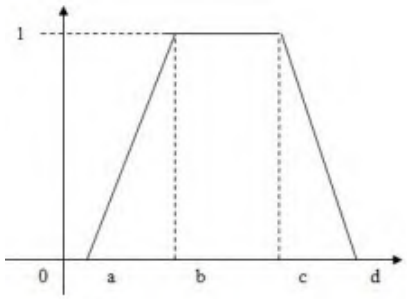
3. Fungsi keanggotaan segitiga
   Contoh dari fungsi keanggotaan segitiga diberikan pada Gambar 2.8 dimana nilai keanggotaannya diberikan pada persamaan 2.12.




Gambar 2.8: Fungsi keanggotaan segitiga.

$$\mu[x,a,b,c] = \begin{cases} 0, & x \leq a \text{ atau } x \geq c \\ \frac{(x-a)}{(b-a)}, & a < x < b \\ \frac{(c-x)}{(c-b)}, & b \leq x < c \end{cases} \tag{2.12}$$

4. Fungsi keanggotaan trapesium
   Contoh dari fungsi keanggotaan trapesium diberikan pada Gambar 2.9 dimana nilai keanggotaannya diberikan pada persamaan 2.13.




Gambar 2.9: Fungsi keanggotaan trapesium.

$$\mu[x,a,b,c,d] = \begin{cases} 0, & x \leq a \\ \frac{(x-a)}{(b-a)}, & a < x < b \\ 1, & b \geq x \leq c \\ \frac{(d-x)}{(d-c)}, & c < x < d \\ 0, & x \geq d \end{cases} \tag{2.13}$$

## 2.12 Sistem Inferensi Mamdani

Sistem inferensi *fuzzy* merupakan suatu kerangka komputasi yang didasarkan pada teori himpunan *fuzzy* dengan aturan pada *fuzzy* yang berbentuk If-Then. Pada penelitian ini sistem inferensi yang digunakan adalah sistem inferensi Mamdani. Ide utama dari sistem inferensi Mamdani adalah untuk menggambarkan kondisi proses dengan variabel linguistik dan menggunakan variabel tersebut sebagai masukan untuk mengontrol aturan. (Zimmerman, 2001).

Metode Mamdani diperkenalkan oleh Ebrahim Mamdani pada tahun 1975. Terdapat 4 tahapan dalam metode ini. Tahap pertama adalah fuzzyfikasi yaitu pembentukan himpunan *fuzzy*. Tahap kedua adalah aplikasi fungsi implikasi berbasis aturan, dimana fungsi implikasi pada metode Mamdani adalah Min. Tahap ketiga adalah komposisi aturan, dan tahap terakhir adalah defuzzyfikasi.

Pada penelitian ini metode yang digunakan pada saat mengkomposisi aturan adalah metode max. Pada metode ini, solusi himpunan *fuzzy* diperoleh dengan cara mengambil nilai maksimum aturan. Sedangkan metode yang digunakan pada defuzzyfikasi adalah COG (*Centre of Gravity*) yaitu dengan menentukan pusat gravitasi atau pusat massa.

# BAB III
# METODE PENELITIAN

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

## 3.1 Obyek Penelitian

Pada penelitian ini obyek penelitian yang digunakan adalah kendaraan bergerak dalam data berupa video rekaman. Terdapat 5 video yang diambil sebagai data dimana 4 video berdurasi pendek dengan durasi 1 menit dan 1 video berdurasi panjang dengan durasi 4 menit. Video berdurasi pendek diambil di Jalan Pemuda, Jalan Wonokromo, Jalan Raya Diponegoro dan Jalan Kedung Cowek. Sedangkan video berdurasi panjang diambil di Jalan Kedung Cowek.

## 3.2 Peralatan

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

## 3.3 Tahapan Penelitian

1. Studi Literatur

   Pada tahap ini, dikumpulkan berbagai informasi tentang pengolahan video digital, median *filtering*, metode *Gaussian Mixture Model*, *shadow removal*, deteksi obyek, logika *fuzzy* 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, diambil data berupa rekaman video untuk keperluan analisis metode. Rekaman video diambil dari jembatan penyebrangan di jalan protokol satu arah di Surabaya pada saat pagi dan siang hari.

3. Analisis Sistem

   Pada tahap ini akan dilakukan analisis data berupa rekaman video untuk menentukan parameter-parameter yang dibutuhkan oleh beberapa metode

yang ada pada sistem. Berikut adalah prosedur klasifikasi kendaraan bergerak dengan logika *fuzzy* berbasis pengolahan citra:

(a) *Input* data berupa rekaman video.

(b) *Input* parameter untuk *smoothing*, GMM, dan deteksi obyek.

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

(d) *Input* nilai keanggotaan *fuzzy* pada setiap jenis kendaraan.

(e) Ekstrak data berupa rekaman video menjadi *frame-frame*.

(f) *Background Subtraction* setiap *frame* menggunakan metode GMM dimana *output* dari GMM merupakan citra *grayscale* dengan 3 tingkat keabuan, yaitu 0 untuk *background*, 127 untuk *shadow*, dan 255 untuk *foreground*.

(g) *Smoothing* dengan tujuan untuk meminimalisir *noise*.

(h) *Shadow removal* yaitu tahap dimana seluruh piksel yang mempunyai nilai 127 (*shadow*) dari citra *output* GMM diubah menjadi 0 (*background*).

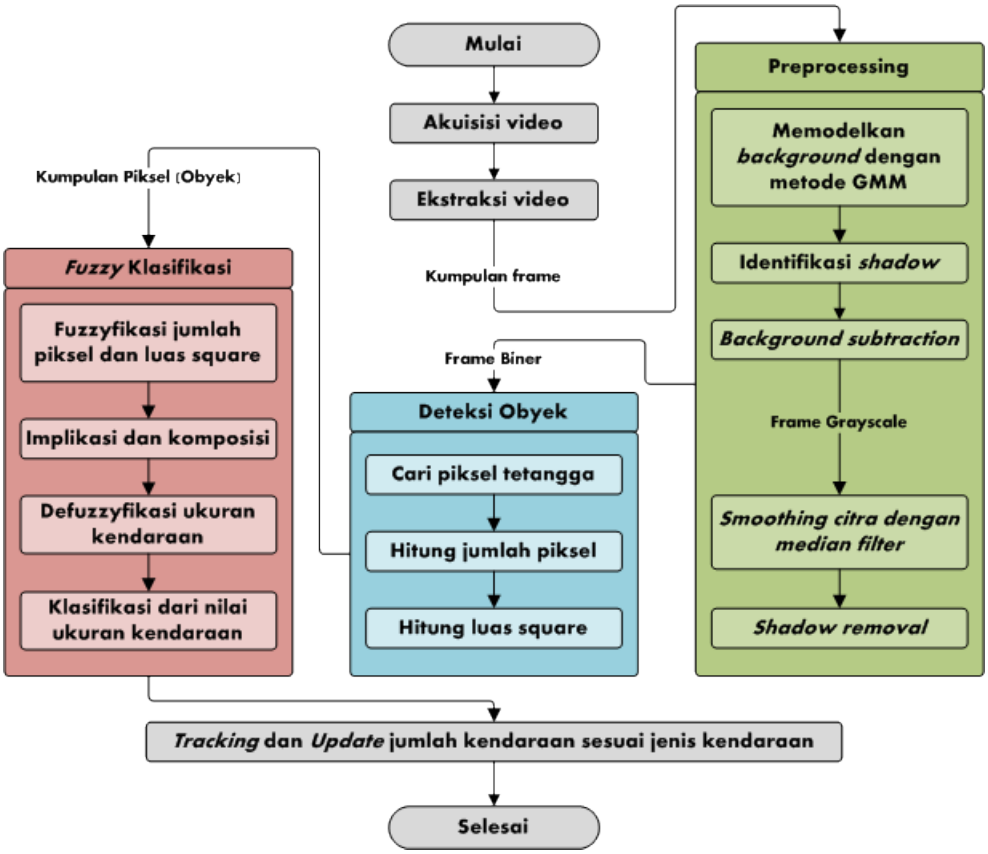
(i) Deteksi obyek kendaraan yang akan dilakukan di dalam ROI.

(j) Setiap obyek yang terdeteksi akan diklasifikasi sesuai jenis kendaraan berdasarkan ukuran kendaraan. Untuk memperoleh ukuran kendaraan akan digunakan logika *fuzzy* berdasarkan jumlah piksel dan luas *square* dari obyek. Sistem inferensi yang digunakan adalah metode Mamdani, dimana pada sistem inferensi tersebut terdapat empat tahap, yaitu fuzzyfikasi attribut dari obyek berupa jumlah piksel dan luas *square*, aplikasi fungsi implikasi dimana pada metode Mamdani fungsi implikasi yang digunakan adalah min, komposisi aturan dimana pada penelitian ini metode yang digunakan adalah max, dan tahap terakhir adalah defuzzyfikasi dimana pada penelitian ini metode yang digunakan adalah COG (*Centre of Gravity*).

(k) *Tracking object* pada kendaraan yang telah diklasifikasi.

(l) *Counting* sesuai jenis kendaraan setelah obyek melewati ROI.

Prosedur diatas digambarkan dengan diagram pada Gambar 3.1.




Gambar 3.1: Blok diagram klasifikasi kendaraan.

4. Percobaan dan Analisis Hasil

   Dengan mengubah parameter-parameter yang dibutuhkan, maka akan dianalisis bagaimana keakuratan dari klasifikasi jenis kendaraan tersebut. Adapun keakuratan tersebut dihitung dengan menggunakan persamaan 3.1.

$$tingkat\,keakuratan\,klasifikasi = \frac{KB}{KB + KS} \times 100\% \qquad (3.1)$$

   dengan KB adalah klasifikasi benar dan KS adalah klasifikasi salah.

5. Publikasi Artikel

6. Penyusunan Tesis

# BAB IV
# PERANCANGAN DAN IMPLEMENTASI SISTEM

Pada bab ini dijelaskan bagaimana melakukan perancangan sistem pada tesis ini, diantaranya adalah proses akuisisi video, proses *background subtraction* menggunakan metode *Gaussian Mixture Model*, proses *smoothing* dengan metode median *filter*, proses *shadow removal*, proses pendeteksian obyek, proses klasifikasi dengan logika *fuzzy*, proses *tracking*, dan proses *counting*. Selain proses tersebut perancangan sistem juga meliputi perancangan antar muka untuk memudahkan penelitian dalam melakukan uji coba dan analisis. Pada bab ini juga akan dilakukan implementasi terhadap sistem yang telah dirancang.

## 4.1 Analisis Sistem

Sesuai dengan tujuan dari tesis ini, sistem yang dirancang harus mampu mengklasifikasikan kendaraan sesuai dengan ukuran dari kendaraan tersebut.

### *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 penyebrangan. Parameter masukan adalah parameter dari beberapa metode dari serangkain proses yang dimasukkan ke dalam sistem oleh *user*. Parameter masukan yang dibutuhkan pada sistem ini adalah parameter *Region of Interest* (ROI), parameter median *filter*, parameter pendeteksian obyek, dan parameter logika *fuzzy*.

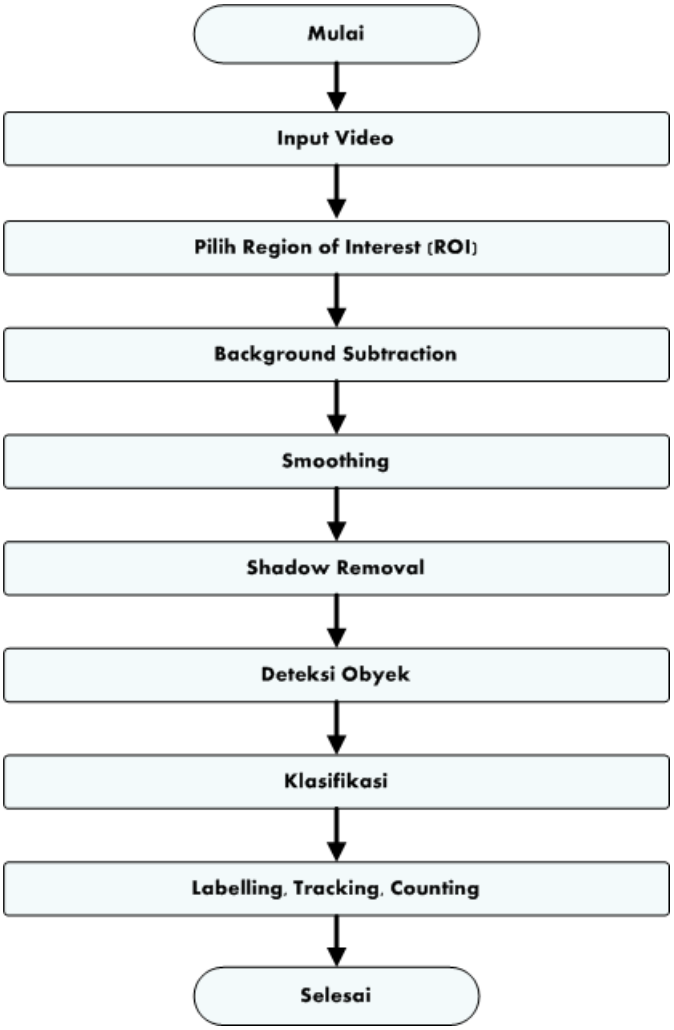
Setelah data masukan dan semua parameter diperoleh, sistem akan melakukan ekstraksi video menjadi rangkaian *frame*. Setiap *frame* yang berhasil diekstrak akan dicari *background*-nya menggunakan metode *Gaussian Mixture Model*. *Background* adalah *image* dari obyek yang tidak bergerak, seperti jalan raya, trotoar, dan rumah. *Background* yang didapatkan akan dibandingkan dengan *frame* tersebut sehingga akan menghasilkan *image* baru yaitu *foreground*. *Foreground* adalah *image* dari obyek yang bergerak. *Foreground* berbentuk *image grayscale* dengan 3 tingkat keabuan, yang masing-masing tingkat keabuan menyatakan obyek tidak bergerak, obyek bergerak, dan bayangan obyek bergerak.

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

Setelah mendapatkan *image* biner, sistem akan melakukan proses pendeteksian obyek. Pendeteksian obyek dilakukan dengan memeriksa satu persatu piksel di dalam *Region of Interest*. Jika piksel yang diperiksa merupakan piksel dari obyek yang bergerak, maka akan dicari piksel tetangga lainnya yang merupakan obyek yang sama. Setiap obyek yang terdeteksi akan dihitung jumlah piksel dan batas piksel-pikselnya untuk keperluan klasifikasi kendaraan bergerak.

Langkah selanjutnya adalah klasifikasi kendaraan bergerak dengan menggunakan logika *fuzzy* Mamdani. Untuk mengklasifikasikan obyek bergerak diperlukan parameter masukan berupa jumlah piksel obyek dan luas area obyek. Nilai luas area obyek diperoleh dari batas-batas piksel obyek tersebut. Setiap aturan yang dihasilkan dari kedua himpunan *fuzzy* tersebut akan dioperasikan ke dalam fungsi implikasi, dimana fungsi implikasi dari Mamdani adalah min. Hasil fungsi implikasi dari tiap aturan selanjutnya akan dikomposisikan dengan menggunakan metode max, sehingga kemudian didapatkan daerah hasil. Selanjutnya adalah defuzzyfikasi daerah hasil dengan menggunakan metode COG (*Centre of Gravity*). Hasil keluaran dari defuzzyfikasi adalah bobot ukuran kendaraan yang akan menentukan jenis kendaraan berdasarkan ukurannya.

Kendaraan yang telah diklasifikasi selanjutnya akan diberi label dan di-*track*. Hal ini bertujuan untuk mengetahui apakah kendaraan tersebut ada pada *frame* sebelumnya. Jika kendaraan tersebut ada pada *frame* sebelumnya maka kendaraan tersebut akan menggantikan kendaraan yang sama pada *frame* sebelumnya, jika tidak maka kendaraan tersebut akan ditandai sebagai kendaraan yang baru terdeteksi. Kendaraan yang akan meninggalkan ROI akan dihitung berdasarkan jenis kendaraannya. Secara garis besar tahapan pada sistem dijelaskan pada gambar Gambar 4.1.




Gambar 4.1: Tahapan dalam klasifikasi.

### *4.1.2 Analisis Kebutuhan Sistem*

Program akan dibuat dalam bentuk *desktop* menggunakan bahasa pemrograman JAVA dengan IDE Eclipse. *Library* yang dibutuhkan untuk membangun sistem adalah OpenCV dan Xuggler. OpenCV adalah sebuah *library* untuk membantu pengolahan citra. Sedangkan Xuggler adalah *library* untuk membantu pengolahan video. Sistem membutuhkan data rekaman video digital, sehingga diperlukan sebuah kamera digital untuk merekam kendaraan bergerak. Tabel kebutuhan sistem disajikan pada Tabel 4.1.

## 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

Tabel 4.1: Tabel kebutuhan sistem

| No | Perangkat Lunak | Keterangan |
|---|---|---|
| 1 | Bahasa Pemrograman Java | Sebagai bahasa pada sistem yang akan dirancang |
| 2 | IDE Eclipse | Sebagai *editor* untuk membuat program dengan bahasa pemrograman Java |
| 3 | *Library* OpenCV dan Xuggler | Sebagai *library* pendukung pada pembuatan program |

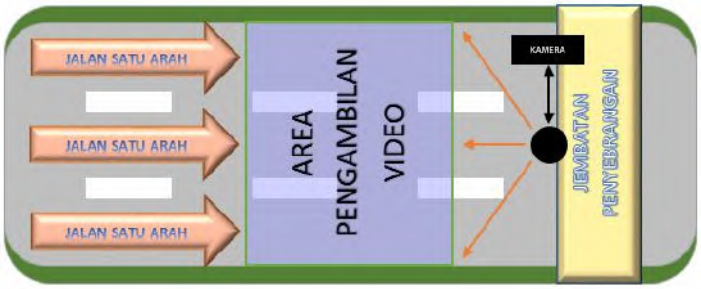
mengenai proses-proses yang disebutkan diatas. Dan terakhir adalah desain antarmuka sistem yang dibangun sebagai sarana untuk memproses data masukan dan parameter masukan yang ada dan melihat hasil yang diperoleh.

### *4.2.1 Perancangan Data Sistem*

Data yang digunakan untuk mengklasifikasi kendaraan bergerak terdiri dari data masukan, data proses, dan data keluaran. Data masukan merupakan video rekaman *offline* di beberapa ruas jalan yang dilakukan secara mandiri oleh peneliti. Data proses adalah data masukan yang dibutuhkan oleh metode-metode dari serangkaian proses yang telah disebutkan sebelumnya. Sedangkan data keluaran adalah informasi mengenai hasil klasifikasi kendaraan bergerak yang diperoleh sistem.

#### 4.2.1.1 Data Masukan

Untuk memperoleh data masukan berupa video *offline*, akan dilakukan akuisisi video terlebih dahulu. Adapun pengambilan rekaman video dilakukan di jalan protokol satu arah di Surabaya. Rekaman video diambil dari atas jembatan penyebrangan orang. Video harus mencakup semua bagian ruas jalan, sehingga sistem mampu memproses seluruh obyek yang terdeteksi pada jalan tersebut. Gambar akuisisi video yang dilakukan dalam penelitian ini disajikan pada Gambar 4.2.




Gambar 4.2: Perancangan akuisisi video.

Hasil rekaman video tersebut akan dijadikan data masukan dari sistem. Namun perlu dipertimbangkan resolusi piksel dari rekaman tersebut, semakin kecil resolusi piksel semakin menghemat waktu komputasi dan semakin besar resolusi piksel semakin jelas obyek dapat terlihat. FPS (*frame per second*) dari kamera juga perlu dipertimbangkan karena semakin kecil FPS semakin menghemat waktu komputasi dan semakin besar FPS semakin bagus model *background* yang didapatkan karena perubahan pencahayaan antar *frame* tidak signifikan. Kamera digital yang digunakan pada sistem untuk proses akuisisi data memiliki $\pm 30$ FPS.

#### 4.2.1.2 Data Proses

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

Tabel 4.2: Tabel data proses

| No | Tahapan | *Input* | *Output* |
|---|---|---|---|
| 1 | Ekstraksi Video | Video | *Image* RGB |
| 2 | *Generate Foreground* | *Image* RGB | *Image grayscale* |
| 3 | *Smoothing* | *Image grayscale* | *Image grayscale smoothing* |
| 4 | *Shadow Removal* | *Image grayscale smoothing* | *Image* biner |
| 5 | Deteksi Obyek | *Image* biner | Kumpulan Obyek |
| 6 | Klasifikasi Kendaraan | Kumpulan Obyek | Jenis kendaraan |
| 7 | Menghitung Kendaraan | Jenis kendaraan | Jumlah kendaraan per jenis kendaraan |

#### 4.2.1.3 Data Keluaran

Data keluaran dari sistem yang dibangun berupa tabel hasil dari klasifikasi tiap kendaraan, jumlah kendaraan tiap jenis, dan jumlah total seluruh kendaraan. Selain itu sistem juga menghasilkan keluaran berupa *image* dari masing-masing proses, yaitu GMM, *smoothing*, *shadow removal*, deteksi obyek, dan klasifikasi.

Tabel 4.3: Tabel Parameter *Region of Interest*

| No | Variabel | Tipe Data | Keterangan |
|---|---|---|---|
| 1 | X1 | Int | Koordinat titik x pada garis 1 |
| 2 | Y1 | Int | Koordinat titik y pada garis 1 |
| 3 | Width1 | Int | Panjang garis 1 |
| 4 | X2 | Int | Koordinat titik x pada garis 2 |
| 5 | Y2 | Int | Koordinat titik y pada garis 2 |
| 6 | Width2 | Int | Panjang garis 2 |

Tabel 4.4: Tabel Parameter Median *Filter*

| No | Variabel | Tipe Data | Keterangan |
|---|---|---|---|
| 1 | *Scale* | Int | Jangkauan tetangga piksel yang akan dicari nilai mediannya |

Tabel 4.5: Tabel Parameter Algoritma Deteksi Obyek

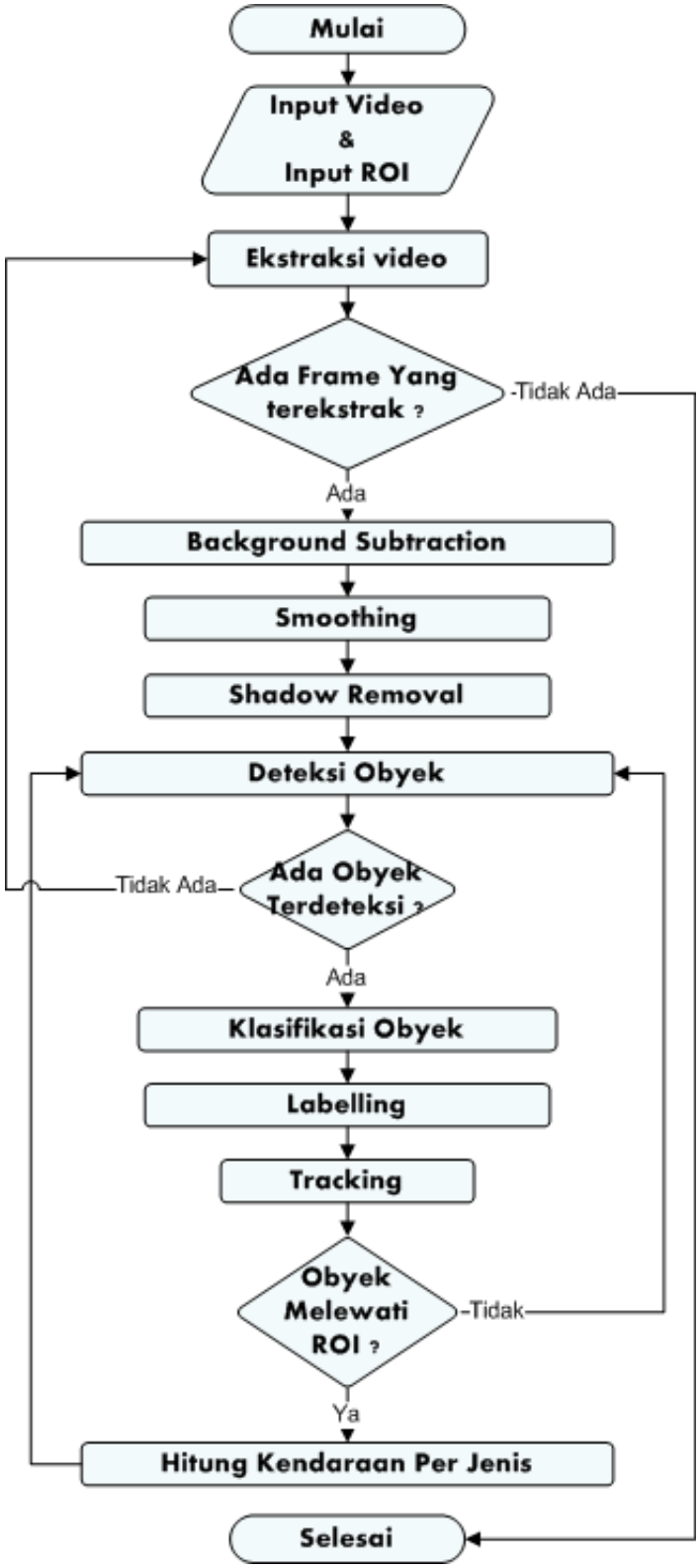
| No | Variabel | Tipe Data | Keterangan |
|---|---|---|---|
| 1 | *Scale* | Int | Jangkauan toleransi piksel tetangga dari obyek |

Tabel 4.6: Tabel Parameter Logika *Fuzzy*

| No | Variabel | Tipe Data | Keterangan |
|---|---|---|---|
| 1 | Himpunan Piksel | Double | Himpunan *input* dari jumlah piksel obyek |
| 2 | Himpunan Area | Double | Himpunan *input* dari luas area obyek |
| 3 | Himpunan *Weight* | Int | Himpunan *output* bobot ukuran obyek |
| 4 | Aturan | Double | Aturan *output* dari himpunan piksel dan himpunan area |
| 5 | Batas Jenis Kendaraan | Double | Batas *weight* tiap jenis kendaraan berdasarkan ukuran |

### *4.2.2 Perancangan Algoritma Sistem*

Diagram alir sistem untuk klasifikasi kendaraan bergerak diberikan pada Gambar 4.3.




Gambar 4.3: Diagram alir proses pada sistem.

#### 4.2.2.1 Perancangan Proses *Background Subtraction*

Data *input* berupa video diekstrak menjadi *frame*. Setiap *frame* yang terekstrak akan dicari model *background*-nya kemudian akan dibangkitkan *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 *means*. 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 $\alpha$ (*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 : $\omega_k$ yaitu bobot dari setiap piksel pada gaussian ke-$k$ dengan nilai $\frac{1}{3}$ dimana 3 merupakan jumlah dari gaussian; $\mu_k$ yaitu *mean* dari setiap piksel pada gaussian ke-$k$ dimana masing-masing piksel dari setiap gaussian mempunyai nilai random antara 0 sampai 255; $\sigma_k$ yaitu standar deviasi dari setiap piksel pada gaussian ke-$k$ dengan nilai 6. Contoh inisialisasi awal parameter GMM diberikan pada Gambar 4.4.

Inisialisasi Awal Bobot ($\omega$)

| 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 ($\mu$)

| 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 ($\sigma$)

| 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 4.4: Contoh inisialisasi awal parameter GMM.

Berikutnya akan diberikan contoh bagaimana proses GMM pada sistem yang akan dibangun. Misalkan *frame* yang terekstrak adalah seperti yang disajikan pada Gambar 4.5. Lalu dengan menggunakan pertidaksamaan 2.4 diperoleh pertidaksamaan 4.1

$$|X_{i,j,t} - \mu_{i,j,k}| < 2.5 * \sigma_{i,j,k} \tag{4.1}$$

dimana nilai intensitas $X$ pada piksel $(i, j)$ dari *frame input* yang memenuhi pertidaksamaan tersebut berarti masuk dalam distribusi. Dari inisialisasi awal $\sigma$ sebesar 6 maka diperoleh nilai dari ruas kanan pada pertidaksamaan 4.1 sebesar 15. Sedangkan ruas kiri merupakan selisih antara intensitas *frame input* dan intensitas mean pada *frame* sebelumnya yang masing-masing diberikan pada Gambar 4.5 dan Gambar 4.4 dimana hasil dari ruas kiri disajikan pada Gambar 4.6.

Frame Input

| | | |
|---|---|---|
| 150 | 17 | 26 |
| 200 | 191 | 43 |
| 187 | 66 | 223 |

Gambar 4.5: Contoh matriks intensitas dari *frame* yang terkestrak.

Selisih Intensitas Frame Input Terhadap Mean ($\mu_{diff}$)

Gaussian 1

| | | |
|---|---|---|
| 6 | 15 | 3 |
| 2 | 176 | 2 |
| 15 | 145 | 111 |

Gaussian 2

| | | |
|---|---|---|
| 51 | 5 | 14 |
| 168 | 9 | 1 |
| 45 | 24 | 1 |

Gaussian 3

| | | |
|---|---|---|
| 4 | 8 | 7 |
| 12 | 39 | 99 |
| 136 | 9 | 60 |

Gambar 4.6: Matriks selisih intensitas *frame input* dan *mean*.

Untuk proses selanjutnya hanya akan diberikan contoh pada piksel $(1, 1)$. Langkah berikutnya adalah *update* bobot dari masing-masing gaussian dengan menggunakan persamaan 2.5. Dari Gambar 4.6 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* paramater GMM tersebut disajikan pada Gambar 4.7. Bobot yang di-*update* kemudian akan dinormalisasi agar jumlah dari bobot tiap komponen gaussian tepat 1. Hasil normalisasi bobot disajikan pada Gambar 4.8.

Update Bobot ($\omega$)

| Gaussian 1 | Gaussian 2 | Gaussian 3 |
|---|---|---|
| 0.34 | 0.33 | 0.34 |

Update Mean ($\mu$)

| Gaussian 1 | Gaussian 2 | Gaussian 3 |
|---|---|---|
| 155.823 | 99 | 146.118 |

Update Standar Deviasi ($\sigma$)

| Gaussian 1 | Gaussian 2 | Gaussian 3 |
|---|---|---|
| 5.995 | 6 | 5.948 |

Gambar 4.7: Hasil *update* parameter GMM.

Normalisasi Bobot ($\omega$)

| Gaussian 1 | Gaussian 2 | Gaussian 3 |
|---|---|---|
| $\frac{0.34}{0.34+0.33+0.34} = 0.337$ | $\frac{0.33}{0.34+0.33+0.34} = 0.326$ | $\frac{0.34}{0.34+0.33+0.34} = 0.337$ |

Gambar 4.8: Hasil normalisasi bobot.

Untuk memilih model mana yang mencerminkan *background*, terlebih dahulu bobot akan diurutkan berdasarkan $\frac{\omega}{\sigma^2}$. Menurut Gambar 4.9 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*.

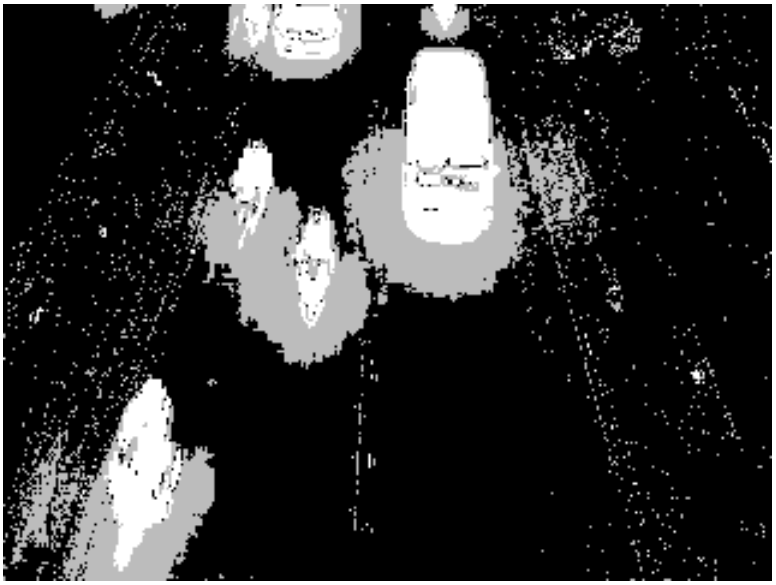
Perhitungan $\omega/\sigma^2$

| 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 4.9: Hasil perhitungan $\frac{\omega}{\sigma^2}$.

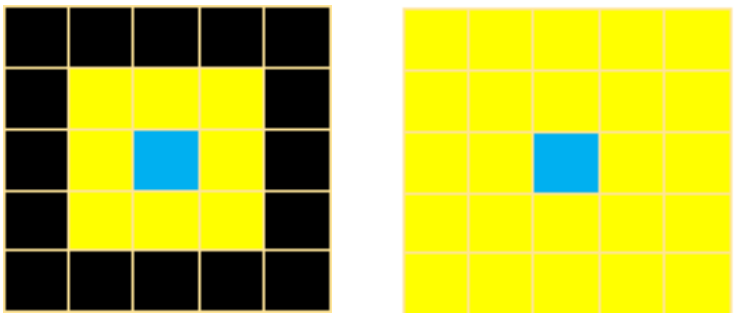
#### 4.2.2.2 Perancangan Proses *Smoothing*

*Smoothing* adalah proses untuk memperhalus *image* yang dilakukan agar *noise* pada *image foreground* hasil dari proses *background subtraction* berkurang. Gambar 4.10 menyajikan *image output* dari proses *background subtraction* dimana pada *image* tersebut terdapat banyak sekali *noise*.



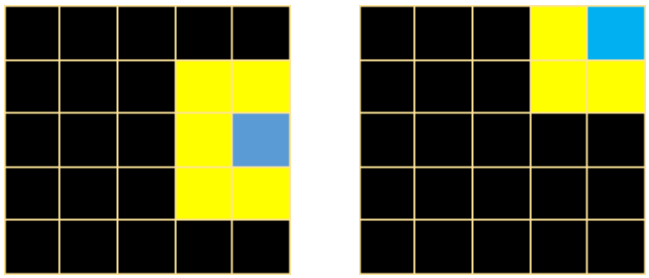
Gambar 4.10: Contoh hasil dari proses background subtraction.

Dalam penelitian ini metode *smoothing* yang digunakan adalah median *filter*. Median *filter* membutuhkan parameter masukan berupa *scale*, yaitu parameter untuk menentukan seberapa jauh jangkauan tetangga yang akan diproses untuk menentukan nilai median. Median *filter* pada sistem ini menggunakan median *filter* 8 ketetanggaan. Gambar 4.11 menjelaskan bagaimana *scale* pada proses *smoothing*. Warna biru menjelaskan piksel yang sedang diproses, sementara warna kuning menjelaskan area dari *scale*.



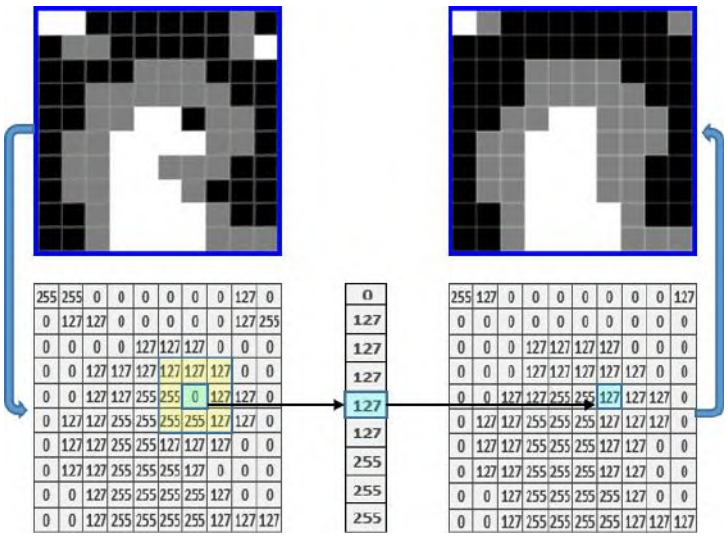
Gambar 4.11: *Scale* dengan 8 ketetanggaan pada proses *smoothing*. Kiri : *scale* = 1. Kanan : *scale* = 2.

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 4.12.



Gambar 4.12: *Scale* dengan 8 ketetanggaan pada piksel *border*. Kiri : piksel *border* kanan akan mencari nilai median dari 6 piksel berdasarkan *scale*. Kanan : piksel *border* kanan atas akan mencari nilai median dari 4 piksel berdasarkan *scale*.

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, abu-abu 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 4.13.



Gambar 4.13: Proses *smoothing* dengan median *filter*.

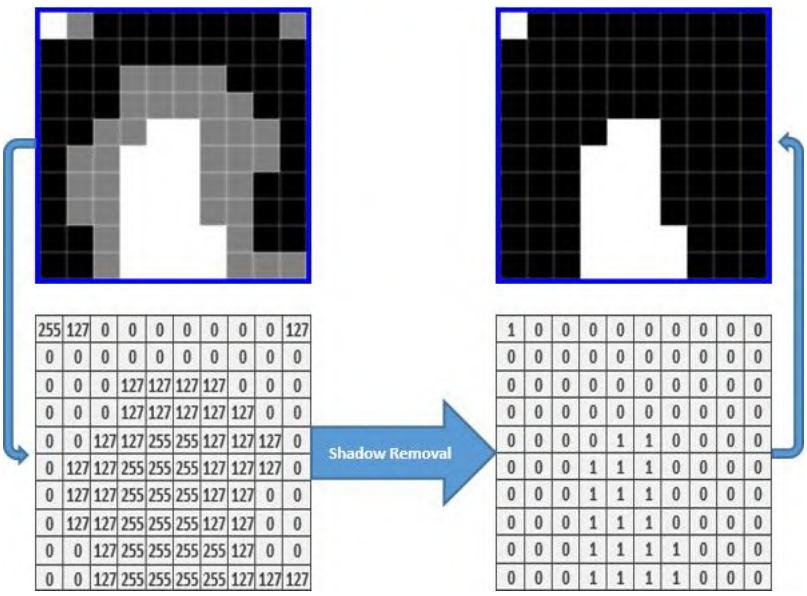
#### 4.2.2.3 Perancangan Proses *Shadow Removal*

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 4.2.

$$I_{biner}(x,y) = \begin{cases} 0, & I_{gray}(x,y) = 0 \quad \text{atau} \quad I_{gray}(x,y) = 127 \\ 1, & I_{gray}(x,y) = 255 \end{cases} \tag{4.2}$$

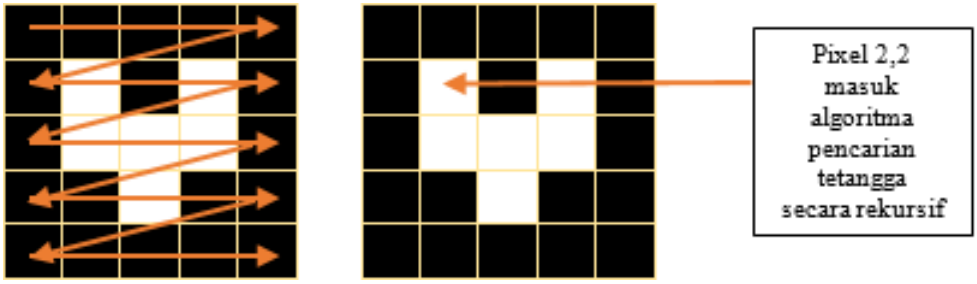
Proses *shadow removal* berlaku pada seluruh piksel dari *image*. Proses *shadow removal* akan mempermudah proses dari deteksi obyek bergerak karena hasil dari proses ini adalah *image* biner yang hanya mempunyai dua nilai, yaitu 0 atau hitam yang merepresentasikan *background* dan 1 atau putih yang merepresentasikan *foreground* atau obyek bergerak. contoh proses dari *shadow removal* dijelaskan pada Gambar 4.14.




Gambar 4.14: Proses *shadow removal*.

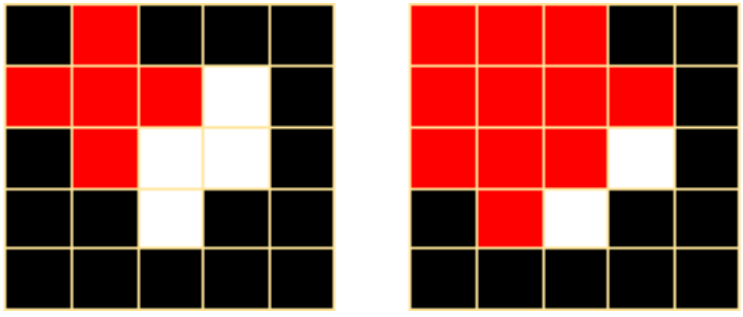
#### 4.2.2.4 Perancangan Proses Deteksi Obyek Bergerak

Pada penelitian ini proses pendeteksian obyek menggunakan algoritma pencarian tetangga secara rekursif. Data masukan pada proses ini adalah *image* biner hasil dari proses *shadow removal*. Proses deteksi obyek dibagi menjadi 2 tahap antara lain mencari salah satu piksel dari suatu obyek yang bergerak dan mencari seluruh piksel dari obyek tersebut. Kedua tahap tersebut dijelaskan pada Gambar 4.15.



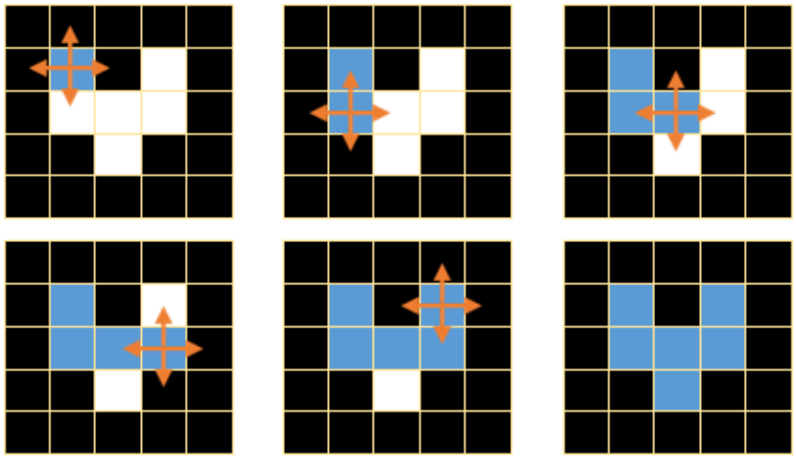

Gambar 4.15: Tahapan pada deteksi obyek. Kiri : arah pencarian piksel suatu obyek. Kanan : memulai proses pencarian piksel lainnya pada obyek.

Parameter masukan pada pendeteksian obyek adalah *scale* yaitu jangkauan toleransi obyek. *Scale* pada deteksi obyek menggunakan 4 ketetanggaan. *Scale* hanya berlaku pada piksel-piksel di dalam ROI. Contoh *scale* pada deteksi obyek diberikan pada Gambar 4.16. Warna merah menjelaskan area dari jangkauan toleransi obyek.



Gambar 4.16: *Scale* pada deteksi obyek. Kiri : *scale* deteksi obyek = 1. Kanan : *scale* deteksi obyek = 2.

Pada algoritma pencarian piksel tetangga, piksel akan mencari seluruh tetangganya sesuai dengan jangkauan toleransi obyek. Pencarian piksel akan dilakukan secara rekursif. Misalkan *scale* adalah 1, maka proses pencarian akan berlangsung seperti yang diberikan pada Gambar 4.17.

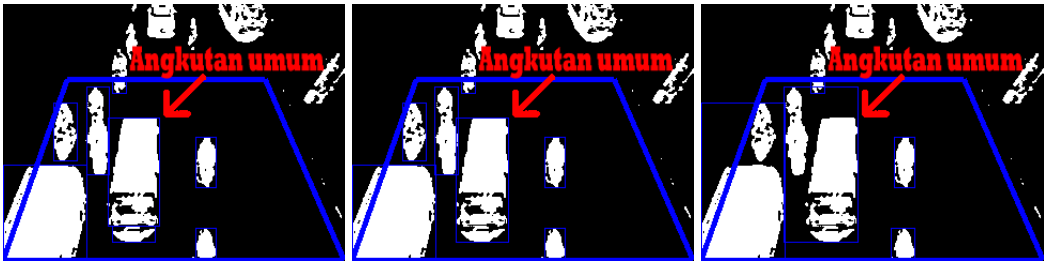


Gambar 4.17: Proses pencarian tetangga pada deteksi obyek.

Gambar 4.17 menjelaskan piksel yang ditemukan pertama akan menjadi titik pusat. Kemudian akan ditandai bahwa piksel tersebut sudah melalui proses pencarian tetangga dari obyek bergerak. Penandaan tersebut dalam gambar diatas disajikan dengan warna biru. Lalu titik pusat akan mencari tetangganya dengan *scale* 1, dengan kata lain titik pusat akan mencari di 4 titik (atas, kanan, bawah, kiri dari titik pusat). Jika piksel tetangga tersebut juga merupakan piksel dari obyek bergerak dan belum ditandai maka titik pusat akan berpindah ke piksel tersebut lalu ditandai. Algoritma tersebut terus berjalan sampai semua piksel obyek tersebut ter-*cover* seperti yang ditunjukkan pada gambar Gambar 4.17 kanan bawah.

Penggunaan parameter *scale* berdasarkan pada *preprocessing*. Jika obyek hasil *prepocessing* utuh (tidak terpisah) maka *scale* yang digunakan lebih kecil dari pada obyek dengan hasil *prepocessing* yang tidak utuh. Penggunaan parameter *scale* juga berdasarkan pada volume kendaraan pada data video. Jika *scale* terlalu besar maka dua obyek yang saling berdekatan akan terdeteksi sebagai satu obyek. Perbedaan hasil tersebut disajikan pada Gambar 4.18 dan Gambar 4.19.


Gambar 4.18: Contoh *image* yang akan dideteksi obyek.




Gambar 4.19: Perbedaan hasil berdasarkan *scale*. Kiri : deteksi dengan *scale* = 1. Tengah : deteksi dengan *scale* = 3. Kanan : deteksi dengan *scale* = 7.

Dari gambar tersebut terlihat deteksi obyek dengan *scale* 1 menghasilkan deteksi obyek yang salah pada mobil angkutan umum dimana obyek tersebut dideteksi menjadi dua bagian yaitu bagian depan mobil dan badan mobil. Hal itu terjadi karena obyek tersebut memiliki hasil *preprocessing* yang tidak utuh, yaitu bagian depan mobil terpisah dari bagian badan mobil. Sedangkan deteksi dengan *scale* 7 menghasilkan deteksi obyek yang salah karena mobil angkutan umum dianggap satu obyek dengan dua sepeda motor yang melintas di sebelahnya. Deteksi dengan *scale* 3 menghasilkan deteksi obyek yang benar karena nilai *scale* sanggup menyambung bagian obyek yang terpisah tetapi tetap tidak menyebabkan obyek tersebut tersambung dengan obyek lainnya.

Untuk menentukan parameter *scale* terbaik dilakukan pengujian deteksi obyek dengan rentang *scale* antara 1 sampai 8 ke semua data video berdurasi 30 detik. Pencarian parameter *scale* ini bertujuan untuk mencari nilai *scale* terbaik dari setiap jalan yang akan digunakan sebagai masukan uji coba deteksi obyek pada bab selanjutnya. Deteksi obyek yang dilihat dan dinilai kebenarannya adalah deteksi obyek pada saat sistem menghitung klasifikasi kendaraan, yaitu ketika obyek akan meninggalkan ROI. Deteksi obyek yang benar adalah deteksi obyek yang hanya mencakup piksel dari obyek tersebut, dimana obyek yang dideteksi adalah kendaraan. Hal tersebut berarti jika obyek yang dideteksi adalah non kendaraan seperti bayangan, jalan-raya, atau obyek lainnya yang bukan kendaraan, maka deteksi dianggap salah. Selain itu, jika satu obyek terdeteksi sebagai dua obyek, maka deteksi obyek juga dianggap salah. Hasil dari pengujian tersebut disajikan pada Tabel 5.1.

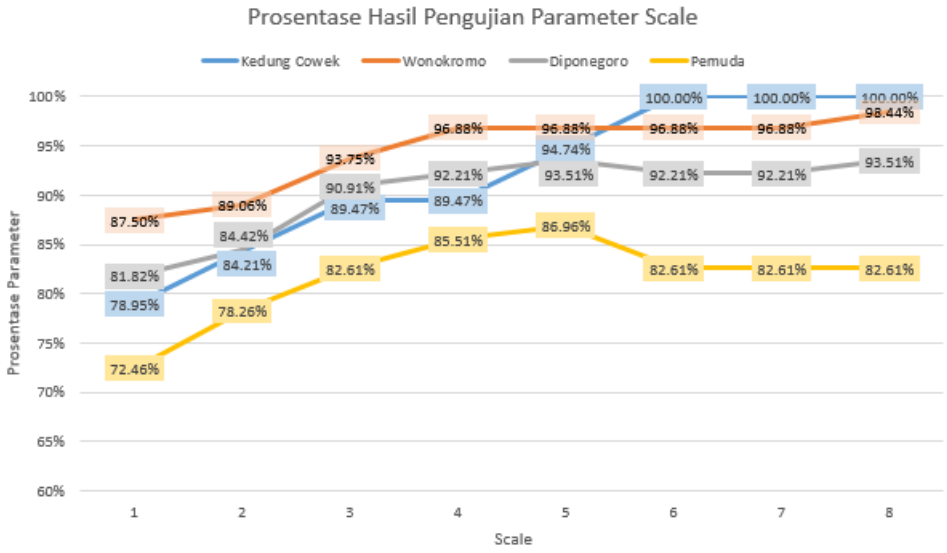
Dari hasil tersebut dilakukan perhitungan prosentase parameter *scale* deteksi obyek pada *scale* 1 sampai 8. Perhitungan prosentase parameter *scale* deteksi obyek berdasarkan pada persamaan 4.3.

$$prosentase\,deteksi\,obyek = \frac{DB}{DB + DS} \times 100\% \tag{4.3}$$

dengan DB adalah deteksi benar dan DS adalah deteksi salah. Prosentase parameter *scale* dari hasil pengujian diberikan pada Gambar 4.20.

Tabel 4.7: Tabel jumlah deteksi obyek benar tiap jalan

| Nama Jalan | *Scale* | | | | | | | | Total Obyek |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| Kedung Cowek | 15 | 16 | 17 | 17 | 18 | 19 | 19 | 19 | 19 |
| Wonokromo | 56 | 57 | 60 | 62 | 62 | 62 | 62 | 63 | 62 |
| Raya Diponegoro | 63 | 65 | 70 | 71 | 72 | 71 | 71 | 72 | 77 |
| Pemuda | 50 | 54 | 57 | 59 | 60 | 57 | 57 | 57 | 69 |

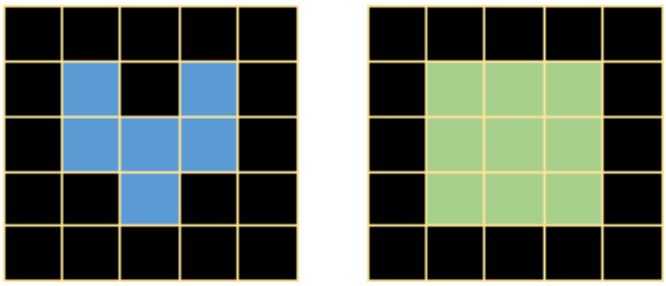



Gambar 4.20: Prosentase deteksi obyek pada *scale* 1 sampai 8.

Dari hasil pencarian parameter *scale* terbaik pada masing-masing video tersebut, didapatkan parameter terbaik pada masing-masing jalan. Prosentase parameter *scale* tertinggi pada Jalan Kedung Cowek adalah 6. Prosentase parameter *scale* tertinggi pada Jalan Wonokromo adalah 8. Prosentase parameter *scale* tertinggi pada Jalan Raya Diponegoro adalah 5. Prosentase parameter *scale* tertinggi pada Jalan Pemuda adalah 5.

Hasil prosentase dari parameter *scale* tersebut juga menjelaskan bahwa banyak terdapat obyek dari hasil *prepocessing* yang belum utuh. Hal tersebut dapat dilihat pada hasil pengujian deteksi obyek dengan *scale* 1 atau 2 pada masing-masing video. Terlihat bahwa *scale* 1 dan 2 mendapatkan prosentase lebih rendah dari pada *scale* lebih dari 2.

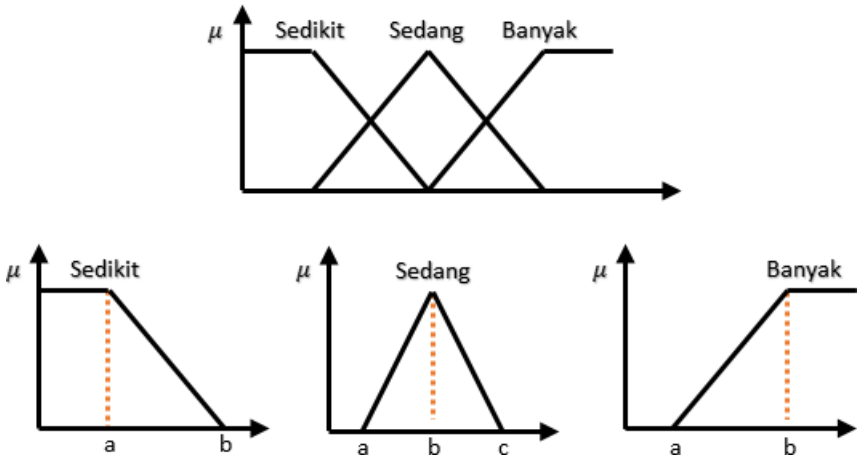
Dalam pendeteksian obyek, sistem harus mampu menghitung banyak piksel yang terjangkau dan luas area dari obyek sebagai *output* dari proses ini. Kedua hal tersebut adalah data masukan pada proses selanjutnya yaitu klasifikasi kendaraan bergerak. Dari hasil pada contoh deteksi obyek diatas, maka dapat ditentukan jumlah piksel obyek sebanyak 6 dan luas area obyek sebanyak 9 seperti yang disajikan pada Gambar 4.21.



Gambar 4.21: *Output* dari proses deteksi obyek. Kiri : jumlah piksel pada obyek adalah 6. Kanan : luas area pada obyek adalah 9.

#### 4.2.2.5 Perancangan Proses Klasifikasi Kendaraan Bergerak

Klasifikasi pada penelitian ini menggunakan logika *fuzzy*. Metode sistem inferensi *fuzzy* yang digunakan dalam penelitian ini adalah metode Mamdani. Pada proses ini diperlukan parameter masukan himpunan *fuzzy* dari jumlah piksel obyek, luas area obyek, dan bobot ukuran obyek. Selain himpunan *fuzzy*, proses ini juga membutuhkan batas bobot ukuran antara jenis kendaraan dan aturan tiap pasangan dari himpunan *fuzzy* jumlah piksel dan himpunan *fuzzy* luas area. Data masukan pada proses klasifikasi adalah berupa jumlah piksel dan luas area dari obyek yang didapatkan dari pendeteksian obyek.




Gambar 4.22: Variabel pada himpunan *fuzzy*.

Pada sistem, ketiga himpunan *fuzzy* mempunyai 3 variabel yang sama yaitu sedikit, sedang, dan banyak. Variabel sedikit mempunyai fungsi keanggotaan turun. Variabel sedang mempunyai fungsi keanggotaan segitiga. Variabel banyak mempunya fungsi keanggotaan naik. Variabel-variabel tersebut dijelaskan pada Gambar 4.22. fungsi keanggotaan pada masing-masing variabel secara berturut-turut dijelaskan pada persamaan 4.4, 4.5, dan 4.6.

$$\mu_{sedikit} = \begin{cases} 1, & x \leq a \\ \frac{(b-x)}{(b-a)}, & a < x < b \\ 0, & x \geq b \end{cases} \tag{4.4}$$

$$\mu_{sedang} = \begin{cases} 0, & x \leq a \quad atau \quad x \geq c \\ \frac{(x-a)}{(b-a)}, & a < x < b \\ \frac{(c-x)}{(c-b)}, & b \leq x < c \end{cases} \tag{4.5}$$

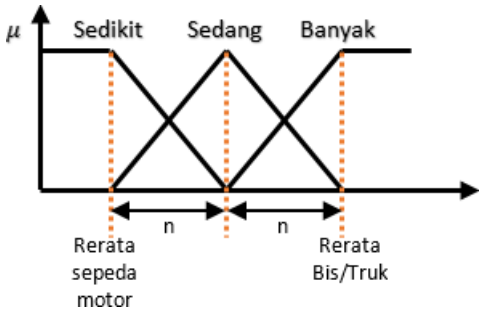
$$\mu_{banyak} = \begin{cases} 0, & x \leq a \\ \frac{(x-a)}{(b-a)}, & a < x < b \\ 1, & x \geq b \end{cases} \tag{4.6}$$

Untuk menentukan himpunan *fuzzy input* yaitu himpunan *fuzzy* jumlah piksel dan himpunan *fuzzy* luas area untuk ujicoba pada bab selanjutnya maka akan dilakukan perhitungan rata-rata jumlah piksel dan rata-rata luas pada video dengan durasi 30 detik pada setiap jalan. Rata-rata jumlah piksel dan rata-rata luas area pada masing-masing jalan bisa berbeda. Hal tersebut dipengaruhi oleh tinggi jembatan penyebrangan dan sudut kamera terhadap jalan raya pada saat pengambilan video. Dengan melakukan hal tersebut pada setiap data video maka nilai-nilai tersebut didapatkan dan disajikan pada Tabel 4.8. Himpunan *fuzzy* juga dapat di-*update* berdasarkan pada eksperimen yang dilakukan dengan bantuan *toolbox Fuzzy Logic Designer* pada *software* MATLAB.

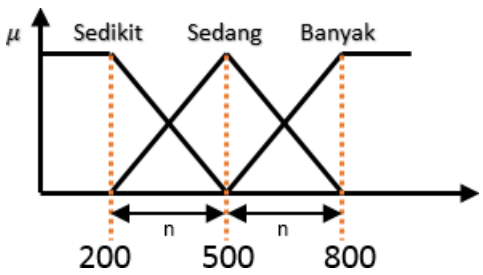
Cara penentuan himpunan *fuzzy input* diinisialisasi berdasarkan rata-rata jumlah piksel dan rata-rata luas area dari jenis kendaraan sepeda motor dan bis / truk seperti yang disajikan pada Gambar 4.23. Namun seperti yang terlihat pada tabel rata-rata jumlah piksel, tidak semua data video mempunyai jenis kendaraan bis atau truk, sehingga himpunan *fuzzy* akan ditentukan berdasarkan eksperimen seperti yang telah disebutkan sebelumnya. Pada penelitian ini nilai maksimal pada himpunan *fuzzy* jumlah piksel adalah 20000 dalam satuan piksel. Sedangkan nilai himpunan *fuzzy* luas area dibatasi 30000 dalam satuan piksel. Himpunan *fuzzy output* berupa

Tabel 4.8: Tabel rata-rata jumlah piksel dan luas area tiap jalan

| Jenis Kendaraan | Jalan Pengambilan Data | | | |
|---|---|---|---|---|
| | Kedung Cowek | Wonokromo | Raya Diponegoro | Pemuda |
| Jumlah Piksel Motor | 705.6 | 759.82 | 948.64 | 762.57 |
| Luas Area Motor | 1303.44 | 1386.54 | 1684.09 | 1454.01 |
| Jumlah Piksel Mobil | 4457.4 | 4557.97 | 5505.85 | 4414.05 |
| Luas Area Mobil | 6335.8 | 7801.37 | 8618.15 | 6917.79 |
| Jumlah Piksel Mini Bis / Mini Truk | 6656 | - | - | 6777 |
| Luas Area Mini Bis / Mini Truk | 10953 | - | - | 8778 |
| Jumlah Piksel Bis / Truk | 12197 | - | - | - |
| Luas Area Bis / Truk | 17808 | - | - | - |




Gambar 4.23: Inisialisasi himpunan *fuzzy input*.

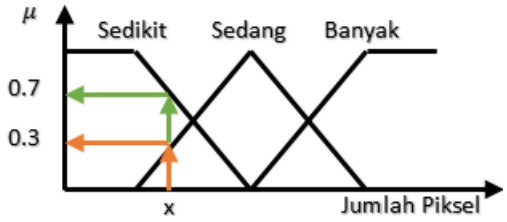



Gambar 4.24: Inisialisasi himpunan *fuzzy* bobot ukuran.

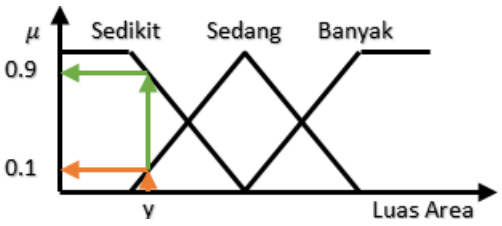
| | | Luas Area | | |
|---|---|---|---|---|
| | | Sedikit | Sedang | Banyak |
| Jumlah Piksel | Sedikit | Sedikit | Sedikit | Sedang |
| | Sedang | Sedikit | Sedang | Banyak |
| | Banyak | Sedang | Banyak | Banyak |

Gambar 4.25: Aturan *fuzzy*.

bobot ukuran diinisialisasikan seperti yang ditunjukkan pada Gambar 4.24 dengan nilai maksimal 1000 dimana aturan *fuzzy* ditunjukkan pada Gambar 4.25.




Gambar 4.26: Contoh pencarian nilai keanggotaan dengan jumlah piksel = $x$.

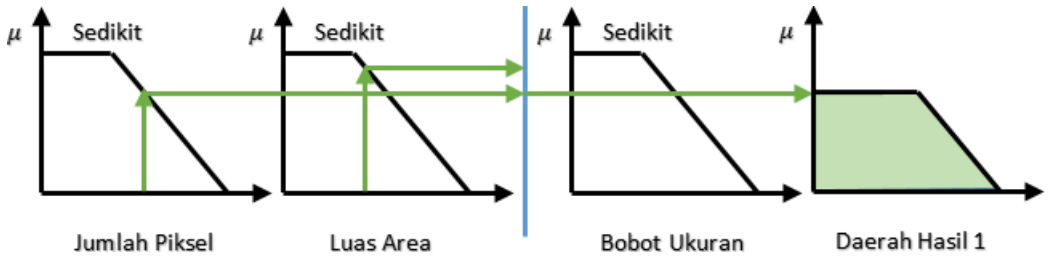



Gambar 4.27: Contoh pencarian nilai keanggotaan dengan luas area = $y$.

Proses klasifikasi dengan logika *fuzzy* dibagi menjadi 4 tahap. Tahap pertama adalah mencari nilai keanggotaan pada setiap variabel himpunan dari jumlah piksel dan luas area. Gambar 4.26 dan Gambar 4.27 adalah contoh bagaimana sistem mencari nilai keanggotaan pada masing-masing himpunan dimana nilai dari jumlah piksel sebesar $x$ dan nilai dari luas area sebesar $y$. Dari Gambar 4.26 dapat dilihat nilai $x$ masuk ke dalam *range* dari 2 variabel himpunan *fuzzy* jumlah piksel. yaitu variabel sedikit dan variabel sedang, sehingga masing-masing dari variabel tersebut dicari nilai keanggotaannya. Hal tersebut juga berlaku pada himpunan *fuzzy* luas area.

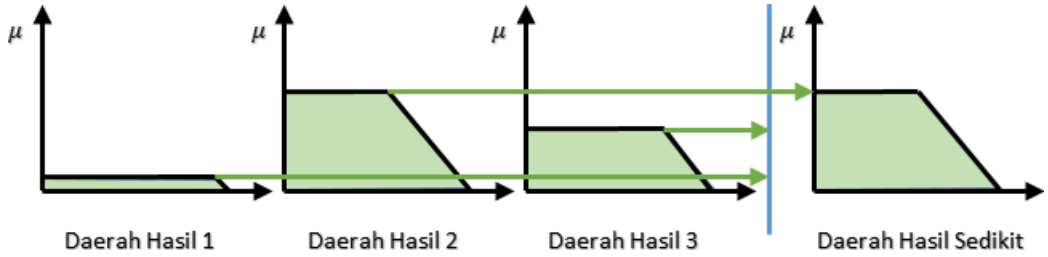
Tahap kedua pada klasifikasi adalah aplikasi fungsi implikasi, yaitu proses dimana setiap nilai keanggotan dari masing-masing variabel himpunan *fuzzy* jumlah

piksel diimplikasikan dengan semua nilai keanggotan dari tiap variabel pada himpunan *fuzzy* luas area berdasarkan aturan pada masing-masing pasangan. Pada sistem inferensi *fuzzy* dengan metode Mamdani, fungsi implikasi menggunakan metode min. Dari Gambar 4.26 dan Gambar 4.27 dapat dilihat ada 4 kombinasi fungsi implikasi yang mungkin, yaitu jumlah piksel sedikit dengan luas area sedikit, jumlah piksel sedikit dengan luas area sedang, jumlah piksel sedang dengan luas area sedikit, dan jumlah piksel sedang dengan luas area sedang. Salah satu contoh dari fungsi implikasi diberikan pada Gambar 4.28.

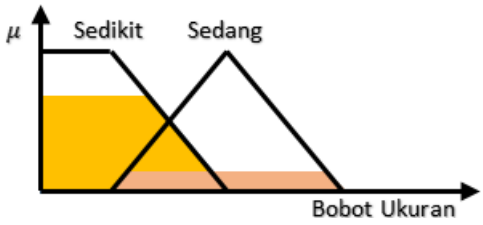



Gambar 4.28: Contoh fungsi implikasi antara jumlah piksel sedikit dan luas area sedikit dengan aturan bobot ukuran sedikit.

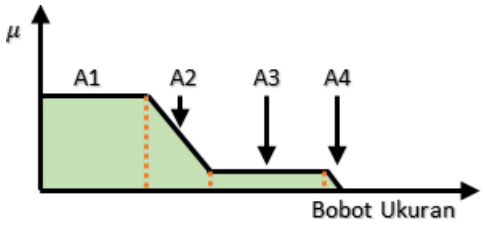
Masing-masing kombinasi fungsi implikasi mempunyai daerah hasil salah satu variabel pada himpunan *fuzzy* bobot ukuran seperti pada Gambar 4.28. Setiap kombinasi tersebut memungkinkan untuk mempunyai aturan yang sama, sehingga sistem dapat mempunyai daerah hasil suatu variabel lebih dari satu. Tahap ketiga dari sistem inferensi *fuzzy* adalah komposisi aturan dimana tiap variabel bobot ukuran yang mempunyai daerah hasil lebih dari satu akan dikomposisikan sehingga masing-masing variabel bobot ukuran hanya mempunyai satu daerah hasil. Contoh dari komposisi aturan pada variabel disajikan pada Gambar 4.29. Metode komposisi aturan pada sistem inferensi *fuzzy* Mamdani adalah max. Selanjutnya setiap daerah hasil pada masing-masing variabel juga akan dikomposisikan. Contoh dari komposisi aturan pada variabel disajikan pada Gambar 4.30 dan Gambar 4.31.




Gambar 4.29: Contoh komposisi aturan variabel sedikit.




Gambar 4.30: Contoh Daerah hasil masing-masing variabel.




Gambar 4.31: Contoh komposisi aturan setiap variabel.

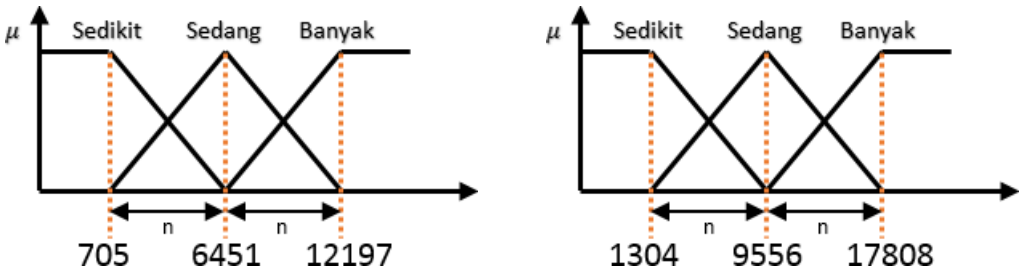
Tahap terakhir pada proses klasifikasi dengan logika *fuzzy* adalah defuzzyfikasi. Pada penelitian metode yang digunakan pada defuzzyfikasi adalah COG (*Centre of Gravity*). Ide dari COG adalah mencari nilai *output* dari titik tengah pada daerah hasil. Persamaan dari COG diberikan pada persamaan 4.7.

$$Nilai\,Bobot\,Ukuran = \frac{\int_A A\mu(A)dA}{\int_A \mu(A)dA} \tag{4.7}$$

dimana $A$ adalah bagian-bagian dari daerah hasil, dan $\mu(A)$ adalah fungsi keanggotaan pada bagian-bagian dari daerah hasil. Nilai dari bobot ukuran yang diperoleh selanjutnya akan dibandingkan dengan parameter masukan yaitu batas bobot ukuran pada masing-masing kendaraan untuk memperoleh jenis kendaraan.

Dari paparan tentang klasifikasi dengan logika *fuzzy* pada kendaraan bergerak diatas, akan diberikan sebuah contoh proses dari *fuzzy* dimana contoh diberikan pada Jalan Kedung Cowek. Menurut Tabel 4.8 Jalan Kedung Cowek memiliki rerata jumlah piksel motor 705.6 dan luas area motor 1303.44, sedangkan rerata jumlah piksel bis/truk 12197 dan luas area bis/truk 17808. Selanjutnya dengan membulatkan setiap rerata tersebut dapat dibentuk himpunan *fuzzy* seperti yang disajikan pada Gambar 4.32.

Dari himpunan yang sudah terbentuk, dimisalkan terdapat sebuah obyek yang terdeteksi dengan jumlah piksel 700 dan luas area 2000 piksel. Langkah pertama akan dicari nilai keanggotaan masing-masing himpunan. Karena jumlah piksel obyek sebesar 700 maka variabel himpunan *fuzzy* jumlah piksel yang ada pada *range* tersebut hanya variabel sedikit dimana jika dikenakan pada persamaan 4.4

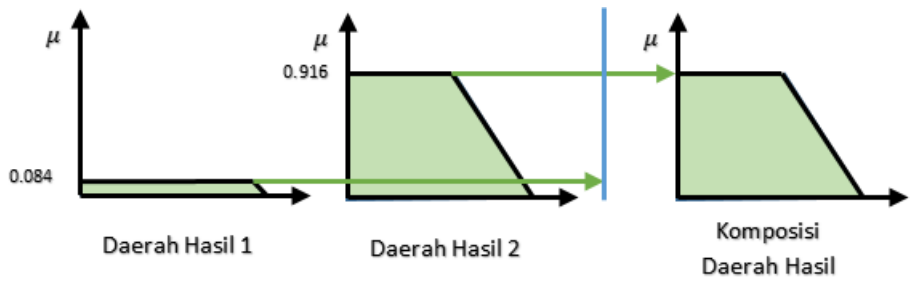



Gambar 4.32: Contoh klasifikasi : pembentukan himpunan *fuzzy*. Kiri : himpunan jumlah piksel. Kanan : himpunan luas area.

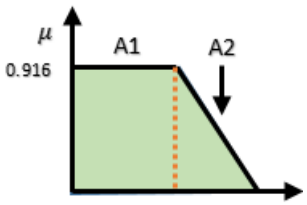
maka nilai keanggotaan variabel jumlah piksel sedikit adalah 1. Sedangkan luas area sebesar 2000, maka variabel himpunan *fuzzy* luas area yang ada pada *range* tersebut adalah variabel sedikit dan sedang. Kemudian akan dicari nilai anggota dari masing-masing variabel dimana secara berturut-turut akan dikenakan persamaan 4.4 dan 4.5. Hasilnya nilai keanggoataan luas area pada variabel sedikit adalah 0.916 dan variabel sedang adalah 0.084.

Langkah selanjutnya adalah fungsi implikasi dari tiap pasangan dimana hanya ada dua pasangan yaitu himpunan *fuzzy* jumlah piksel variabel sedikit (1) dengan himpunan *fuzzy* luas area variabel sedikit dimana aturannya sedikit (0.916). Himpunan *fuzzy* jumlah piksel variabel sedikit (1) dengan himpunan *fuzzy* luas area variabel sedang (0.084) dimana aturannya sedikit. Implikasi pada Mamdani adalah min sehingga hasil pada pasangan pertama adalah 0.916 dan pada pasangan kedua adalah 0.084 dimana variabel *output* dari kedua pasangan adalah sedikit.

Langkah selanjutnya adalah komposisi aturan yang diberikan pada Gambar 4.33. Daerah hasil dari komposisi aturan selanjutnya akan dibagi berdasarkan fungsinya, seperti yang disajikan pada Gambar 4.34. Dengan menggunakan persamaan 4.7 akan diperoleh nilai bobot ukuran sebesar 325.724. Jenis kendaraan pada obyek tersebut didapatkan dengan membandingkan nilai tersebut dengan batas-batas bobot kendaraan dimana batas dari bobot kendaraan dapat terus di-*update* berdasarkan hasil eksperimen.




Gambar 4.33: Contoh klasifikasi : komposisi aturan setiap variabel.




Gambar 4.34: Contoh klasifikasi : pembagian daerah hasil.

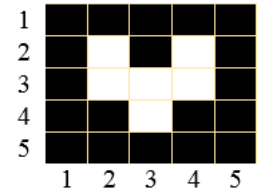
#### 4.2.2.6 Perancangan Proses *Labelling*, *Tracking*, dan *Counting*

Untuk memudahkan analisis dalam penelitian ini, sistem akan melakukan proses-proses berikut. Kendaraan akan diberi label berdasarkan jenis kendaraan. Setiap kendaraan yang akan meninggalkan ROI akan dihitung lalu disimpan ke dalam memori CPU. Data yang tersimpan tersebut ditampilkan dalam data keluaran berupa tabel hasil dari klasifikasi tiap kendaraan. Untuk memastikan obyek tidak terhitung dan tersimpan lebih dari satu kali, maka proses *tracking object* antar *frame* perlu dilakukan. *Tracking* adalah suatu proses yang bertujuan untuk mencari obyek yang sama pada *frame* sebelumnya. Dua obyek dikatakan sama jika memenuhi pertidaksamaan 4.8 dan pertidaksamaan 4.9.

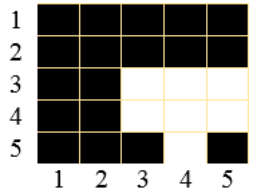
$$L_{obyek1} \leqslant \frac{L_{obyek2} + R_{obyek2}}{2} \leqslant R_{obyek1} \tag{4.8}$$

$$T_{obyek2} \leqslant T_{obyek1} \leqslant B_{obyek2} \tag{4.9}$$

dimana $obyek1$ adalah obyek yang sedang diproses, $obyek2$ adalah sebuah obyek pada *frame* sebelumnya, $L$ adalah batas kiri obyek, $R$ adalah batas kanan obyek, $T$ adalah batas atas obyek, dan $B$ adalah batas bawah obyek. Jika kedua obyek teridentifikasi sama maka *id* pada obyek sebelumnya akan berpindah ke obyek yang sedang diproses. Proses tersebut akan terus berlaku hingga obyek meninggalkan ROI. Batas-batas obyek dijelaskan pada Gambar 4.35.




| | |
|---|---|
| Batas Kiri | 2 |
| Batas Kanan | 4 |
| Batas Atas | 2 |
| Batas Bawah | 4 |




| | |
|---|---|
| Batas Kiri | 3 |
| Batas Kanan | 5 |
| Batas Atas | 3 |
| Batas Bawah | 5 |

Gambar 4.35: Masing-masing batas pada obyek yang berbeda. Kiri : obyek pada $frame_{t-1}$. Kanan : obyek pada $frame_t$.

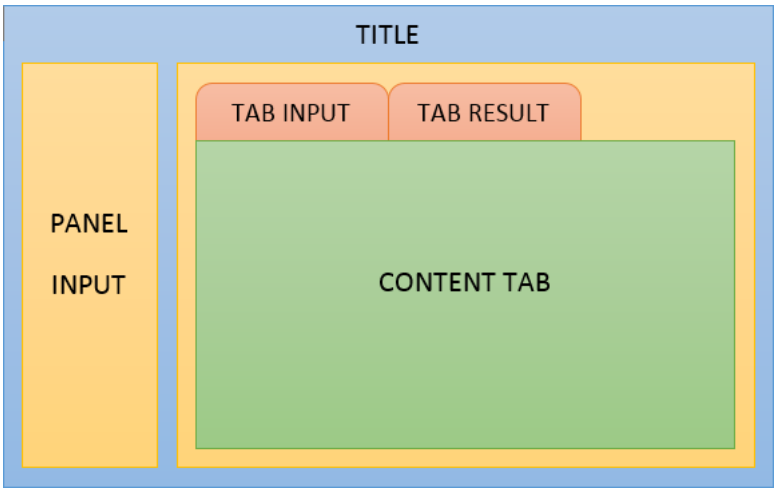
Dari Gambar 4.35 akan ditentukan apakah kedua obyek tersebut adalah obyek sama. Karena kedua obyek memenuhi pertidaksamaan 4.8 dan pertidaksamaan 4.9, maka kedua obyek tersebut adalah obyek yang sama. Proses *tracking* diharapkan dapat meningkatkan akurasi pada perhitungan kendaraan bergerak.

Proses *counting* akan dilakukan jika memenuhi 2 syarat. syarat pertama obyek yang diproses ada pada *frame* sebelumnya. Syarat kedua adalah jika obyek yang sedang diproses adalah $obyek1$ dan obyek pada *frame* sebelumnya adalah $obyek2$ maka akan memenuhi pertidaksamaan 4.10. Jika batas bawah dari ROI adalah 5 maka obyek pada Gambar 4.35 akan dihitung dikarenakan memenuhi kedua syarat yang telah disebutkan.

$$B_{obyek2} < Tepi\ Bawah\ ROI \leqslant B_{obyek1} \tag{4.10}$$

### *4.2.3 Perancangan Antar Muka Sistem*

Untuk lebih mempermudah eksperimen, dibutuhkan antar muka dari sistem yang telah dirancang. Pada subbab ini akan dijelaskan bagaimana desain antar muka dari program. Desain antar muka sistem diberikan pada Gambar 4.36.




Gambar 4.36: Desain antar muka sistem.

## 4.3 Implementasi Sistem

### *4.3.1 Implementasi Akuisisi Video*

Akuisisi video dilakukan di beberapa jembatan penyebrangan di jalan protokol satu arah di Surabaya. Kondisi jalan pada saat akuisisi video beragam dari yang lengang hingga yang padat karena pada penelitian ini membahas tentang bagaimana perbedaan tingkat akurasi sistem pada kondisi jalan yang berbeda. Gambar 4.37 menjelaskan bagaimana posisi kamera pada saat akuisisi video.

Gambar 4.37: Posisi kamera saat akuisisi video.

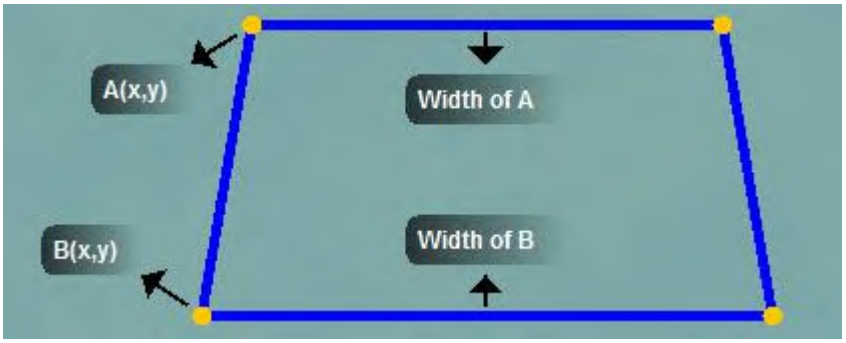
### *4.3.2 Implementasi Input Video*

Pada *input* video, user diharuskan memasukkan video. Setelah *user* memasukkan video, sistem akan menghasilkan *image* awal dari video tersebut yang nantinya akan digunakan untuk kebutuhan pemilihan area ROI. Selain *image* awal dari video, sistem juga akan menampilkan info-info dari video.

### *4.3.3 Implementasi Pemilihan Area ROI*

ROI (*Region of Interest*) dalam sistem berfungsi untuk memperkecil area pengolahan pada video masukan, hal tersebut bertujuan untuk lebih menghemat waktu komputasi dan menghindari *noise* atau obyek yang tidak diinginkan seperti pohon yang bergerak atau orang yang berjalan.

Untuk memilih area ROI, sistem akan menampilkan *image* awal yang dihasilkan pada *input* video di tab *input*. Terdapat 6 nilai pada *input* ROI seperti yang dijelaskan pada Gambar 4.38.




Gambar 4.38: Parameter pemilihan ROI.

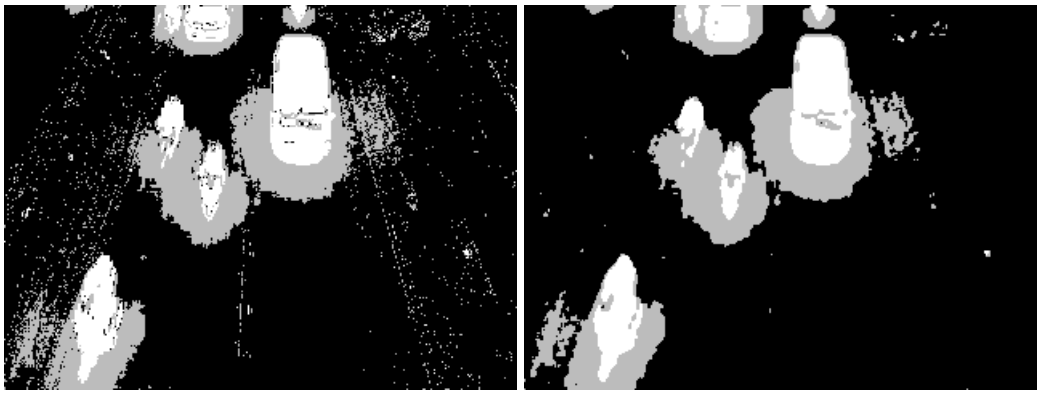
### 4.3.4 Implementasi Proses Background Subtraction

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*. Contoh hasil dari proses *background subtraction* pada program diberikan pada Gambar 4.39.

Gambar 4.39: Hasil *background subtraction*.

### 4.3.5 Implementasi Proses Smoothing

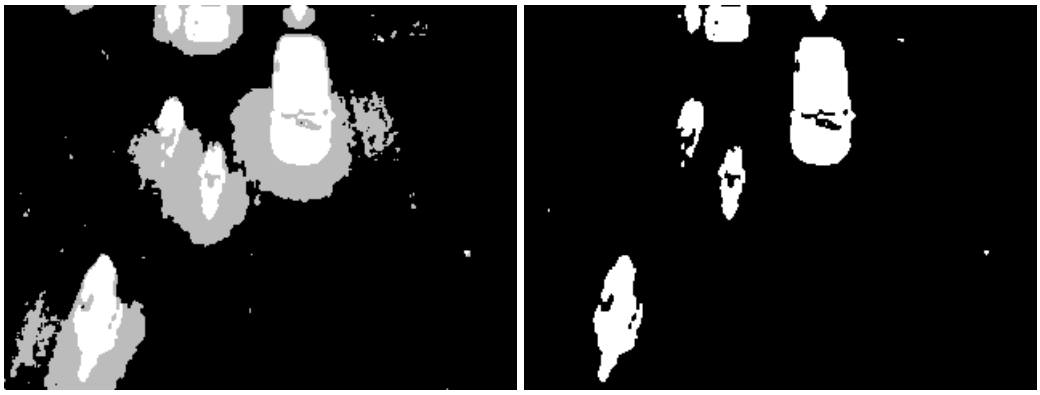
Pada penelitian ini metode *smoothing* yang digunakan adalah median *filter*. *Filtering* yang digunakan pada sistem adalah 8 ketetanggaan. Data masukan pada proses *smoothing* adalah *image grayscale* hasil dari proses *background subtraction*. Tipe data dari hasil proses *smoothing* juga merupakan *image grayscale*. Contoh hasil dari proses *smoothing* diberikan pada Gambar 4.40.



Gambar 4.40: Hasil *smoothing*.

### 4.3.6 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 4.41.



Gambar 4.41: Hasil *shadow removal*.

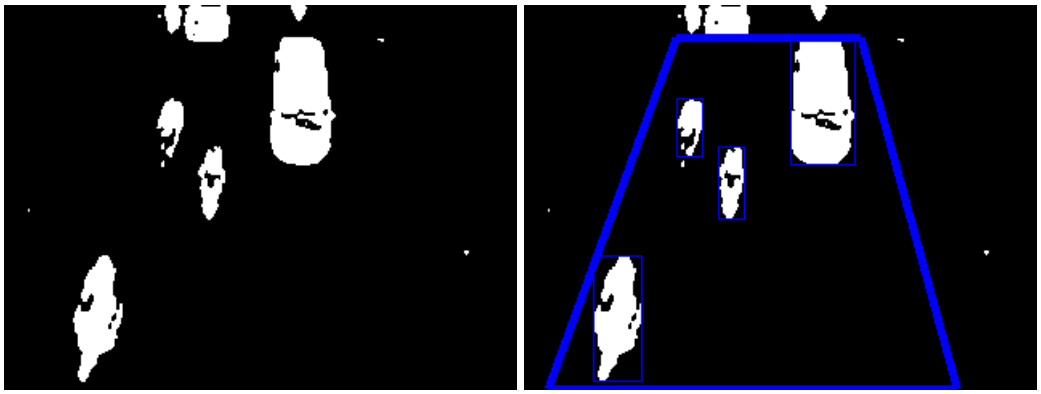
### 4.3.7 Implementasi Proses Deteksi Obyek Bergerak

Dari proses sebelumnya, sistem telah memiliki ROI (*Region of Interest*). Pada sistem, proses deteksi obyek dilakukan di dalam ROI, dengan tujuan untuk menghindari pendeteksian obyek yang tidak diinginkan, seperti pohon dan manusia yang bisa dikategorikan sebagai obyek yang bergerak. ROI juga bertujuan untuk menghemat waktu komputasi, Karena piksel yang akan diproses lebih sedikit.

Data masukan pada proses deteksi obyek pada sistem adalah *image* biner hasil dari proses *shadow removal*. Parameter masukan pada proses ini adalah *scale* atau jangkauan toleransi tetangga. Penggunaan parameter *scale* pada deteksi obyek harus sesuai pada kondisi *image*. Untuk mengurangi kesalahan obyek sama yang terdeteksi secara terpisah maka *scale* harus diperbesar. Sebaliknya untuk mengurangi kesalahan obyek yang berbeda yang terdeteksi sebagai obyek sama, maka *scale* harus diperkecil.

Setiap piksel obyek yang diproses, sistem akan meng-*update* jumlah piksel obyek dan batas piksel dari obyek tersebut. Jika obyek telah selesai diproses, sistem akan mempunyai jumlah piksel dari obyek tersebut. Kemudian dari batas piksel dari obyek tersebut akan dihitung luas area dari obyek. Keluaran dari proses deteksi obyek adalah obyek-obyek yang bergerak yang masing-masing obyek memiliki

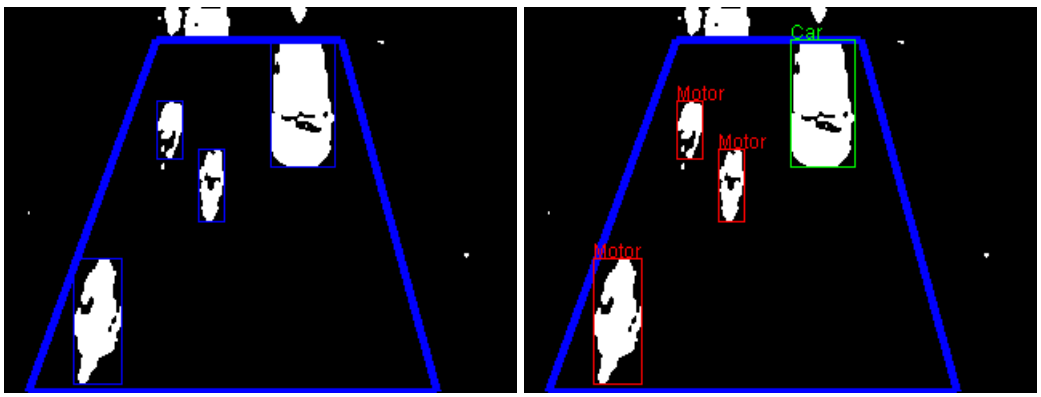
*property* jumlah piksel dan luas area obyek. Contoh hasil dari proses deteksi obyek diberikan pada Gambar 4.42.



Gambar 4.42: Hasil deteksi obyek.

### *4.3.8 Implementasi Proses Klasifikasi Kendaraan Bergerak*

Proses klasifikasi adalah proses yang merupakan tujuan dari sistem, yaitu mengklasifikasikan jenis kendaraan berdasarkan ukurannya. Data masukan pada proses ini adalah obyek-obyek bergerak yang terdeteksi pada sebuah *image*. Setiap obyek yang terdeteksi mempunyai nilai jumlah piksel dan nilai luas area. Kedua parameter tersebut akan menjadi parameter logika *fuzzy*. Selain itu terdapat beberapa parameter dari logika *fuzzy* yaitu himpunan *fuzzy input* diantaranya himpunan *fuzzy* jumlah piksel dan himpunan *fuzzy* luas area, himpunan *fuzzy output* yaitu himpunan *fuzzy* bobot ukuran, dan aturan dari sistem *fuzzy*. Kendaraan yang telah diklasifikasi selanjutnya akan diberi label sesuai dengan jenisnya. Contoh hasil dari proses klasifikasi diberikan pada Gambar 4.43. Contoh hasil akhir dari sistem diberikan pada Gambar 4.44.




Gambar 4.43: Hasil klasifikasi.


Gambar 4.44: Hasil proses dari sistem.

### *4.3.9 Implementasi Antar Muka Sistem*

Antar muka sistem yang telah dirancang akan dibuat berdasarkan *layout* pada perancangan antar muka sistem. Antar muka dibagi menjadi 3 halaman, yaitu halaman utama, halaman untuk *input*, dan halaman untuk hasil. Berikut adalah paparan dari ketiga halaman tersebut.

1. Perancangan Halaman Utama

   Halaman utama adalah halaman yang dilihat ketika program pertama kali dijalankan. Dari sini *user* dapat langsung memasukkan data masukan berupa video pada panel *input* di sisi kiri. Dari panel *input*, *user* juga dapat memasukkan parameter dari metode GMM, *smoothing*, dan deteksi obyek. Panel *input* tidak hanya terdapat pada halaman utama program, namun pada semua halaman, sehingga *user* tetap dapat mengganti nilai parameter meskipun telah meninggalkan halaman utama. Desain halaman utama disajikan pada Gambar 4.45.


Gambar 4.45: Antar muka halaman utama.

2. Perancangan Halaman *Input*

   Halaman *input* adalah halaman untuk memasukkan parameter *Region of Interest* dan parameter-parameter logika *fuzzy*. User dapat menentukan area ROI dengan dua cara, yaitu dengan cara *drag mouse* pada *image* atau dengan memasukkan koordinat dan lebar dari ROI yang tersedia di sebelah kanan *image*. Sementara *input* parameter *fuzzy* yaitu himpunan *fuzzy*, aturan *fuzzy*, dan klasifikasi bobot tersedia di sisi bawah dan kanan dari halaman *input*. Desain halaman *input* disajikan pada Gambar 4.46.

Gambar 4.46: Antar muka halaman *input*.

3. Perancangan Halaman Hasil

   Halaman hasil merupakan halaman dimana sistem menampilkan hasil keluaran dari proses yang dilakukan, yaitu *frame* yang telah diproses, tabel klasifikasi tiap kendaraan, jumlah tiap jenis kendaraan, dan jumlah total kendaraan. Selain dari itu halaman hasil juga menampilkan info dari data masukan video. Desain halaman hasil disajikan pada Gambar 4.47.

Gambar 4.47: Antar muka halaman hasil.

# BAB V
# 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 obyek menggunakan parameter terbaik yang diperoleh pada bab sebelumnya. Tujuan dari uji coba ini adalah untuk melihat tingkat keakurasian deteksi obyek dari sistem yang telah dibangun. Tahap berikutnya adalah uji coba terhadap klasifikasi kendaraan bergerak berdasarkan rata-rata jumlah piksel dan luas area tiap video yang diperoleh pada eksperimen bab sebelumnya. Uji coba ini bertujuan untuk melihat dan menganalisis konsistensi program berdasarkan kepadatan jalan dalam mengklasifikasi kendaraan bergerak.

Kedua uji coba tersebut dilakukan dengan data masukan berupa video. Akuisisi video dilakukan di 4 tempat dengan volume kendaraan yang berbeda, antara lain Jalan Pemuda, Jalan Wonokromo, Jalan Raya Diponegoro dan Jalan Kedung Cowek. Video yang di ambil dari Jalan Pemuda dan Jalan Diponegoro lebih padat dari pada lainnya. Sedangkan Jalan Kedung Cowek cenderung lebih lengang dari pada Jalan Wonokromo.

Dari keempat tempat pengambilan video, Jalan Kedung Cowek mempunyai semua variasi kendaraan berdasarkan ukuran, hal ini dikarenakan Jalan Kedung Cowek bukan termasuk jalan dalam kota dan merupakan jalan antar pulau sehingga banyak terdapat bis dan truk angkutan melintas di jalan ini. Sementara itu tempat pengambilan video lainnya seperti Jalan Pemuda dan Jalan Raya Diponegoro yang merupakan jalan dalam kota adalah jalan sangat padat, namun bis dan truk jarang melintas. Oleh karena itu untuk menganalisis lebih pada konsistensi program dalam melakukan klasifikasi berdasarkan jenis kendaraan akan dilakukan uji coba klasifikasi kendaraan bergerak di jalan yang memiliki variasi kendaraan beragam yaitu Jalan Kedung Cowek dengan video berdurasi lebih panjang dari uji coba sebelumnya.

## 5.1 Uji Coba Deteksi obyek

Pengujian dilakukan dengan memasukkan nilai parameter deteksi obyek terbaik dari hasil pengujian parameter pada bab sebelumnya pada masing-masing video. Data masukan pada uji coba ini adalah video dengan durasi kurang lebih 1 menit.

Hasil dari uji coba ini disajikan pada Tabel 5.1.

Tabel 5.1: Tabel jumlah deteksi obyek benar tiap jalan

| Nama Jalan | Deteksi Benar | Jumlah Obyek | Prosentase |
|---|---|---|---|
| Kedung Cowek (*scale* 6) | 39 | 39 | 100% |
| Wonokromo (*scale* 8) | 126 | 129 | 97.67% |
| Raya Diponegoro (*scale* 5) | 142 | 154 | 92.21% |
| Pemuda (*scale* 5) | 117 | 138 | 84.78% |

Dari Tabel 5.1 terlihat bahwa prosentase deteksi obyek paling tinggi adalah Jalan Kedung Cowek dan prosentase deteksi obyek paling rendah adalah Jalan Pemuda. Hal tersebut dikarenakan Jalan Kedung Cowek sangat lengang dibandingkan dengan jalan lainnya. Sementara itu Jalan Pemuda cenderung sangat ramai dibandingkan dengan jalan lainnya.

Terdapat beberapa faktor yang dapat mempengaruhi deteksi obyek diantaranya sebagai berikut :

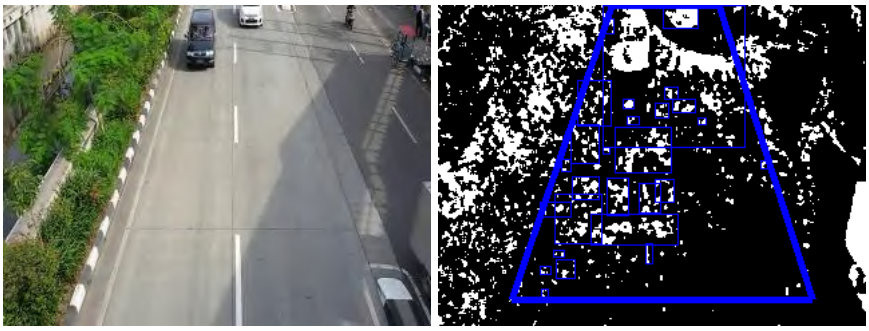
1. Volume lalu lintas

   Kepadatan volume lalu lintas seringkali membuat kendaraan saling berdekatan dan berhimpitan, akibatnya beberapa kendaraan terdeteksi sebagai 1 obyek. Hasil deteksi obyek dengan faktor tersebut disajikan pada Gambar 5.1.

Gambar 5.1: Faktor volume lalu lintas.

2. Kondisi cuaca

   Berubahnya intensitas cahaya pada *background* secara signifikan juga dapat mengakibatkan kesalahan deteksi obyek. Hal tersebut dikarenakan intensitas pada beberapa piksel *background* dapat dianggap sebuah obyek. Hasil deteksi obyek dengan faktor tersebut disajikan pada Gambar 5.2.



Gambar 5.2: Faktor kondisi cuaca.

3. Pantulan cahaya

   Pantulan cahaya terhadap kamera juga dapat menyebabkan salah deteksi karena pantulan cahaya merubah intensitas beberapa piksel menjadi lebih cerah. Hasil deteksi obyek dengan faktor tersebut disajikan pada Gambar 5.3.

Gambar 5.3: Faktor pantulan cahaya.

4. Warna kendaraan

   Warna dari kendaraan yang hampir sama dengan *background* yang didekatnya seperti jalan raya (warna abu-abu) akan dianggap sebagai *background*. Hasil deteksi obyek dengan faktor tersebut disajikan pada Gambar 5.4.

Gambar 5.4: Faktor warna kendaraan.

5. Bayangan

   Bayangan yang tidak terdeteksi ketika proses *Background Subtraction* dengan GMM akan menjadi masalah jika bayangan dari obyek berdekatan atau bahkan terhubung dengan obyek lain. Hasil deteksi obyek dengan faktor tersebut disajikan pada Gambar 5.5.

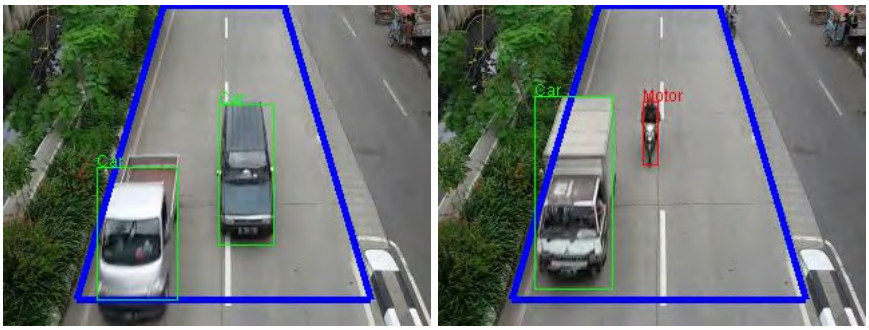
Gambar 5.5: Faktor bayangan.

## 5.2 Uji Coba Klasifikasi Kendaraan Bergerak

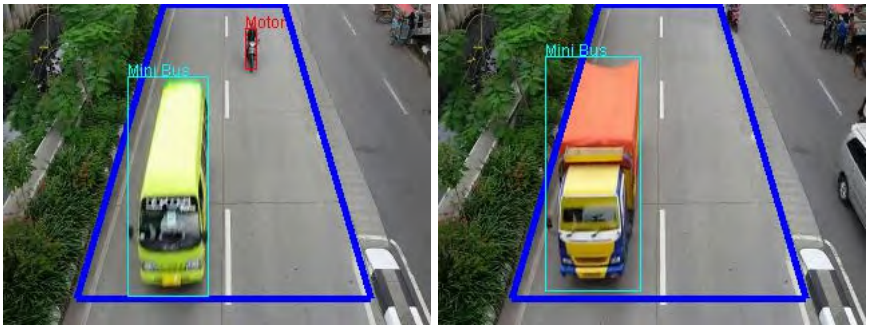
Dalam penelitian ini obyek berupa kendaraan bergerak akan diklasifikasikan menjadi 4 jenis berdasarkan ukuran dari kendaraan tersebut. Keempat jenis tersebut antara lain : sepeda motor, mobil, mini bis / mini truk, dan bis / truk. Kendaraan dengan ukuran jenis sepeda motor diberikan pada Gambar 5.6. Kendaraan dengan ukuran jenis mobil diberikan pada Gambar 5.7. Kendaraan dengan ukuran jenis mini bis / mini truk diberikan pada Gambar 5.8. Kendaraan dengan ukuran jenis bis / truk diberikan pada Gambar 5.9.

Pada program yang telah dirancang sebelumnya, klasifikasi setiap kendaraan dihitung dan dicatat dalam jarak yang sama antara kendaraan dan jembatan penyebrangan, yaitu ketika kendaraan akan meninggalkan ROI. Hal tersebut perlu


Gambar 5.6: Kendaraan dengan ukuran jenis sepeda motor.



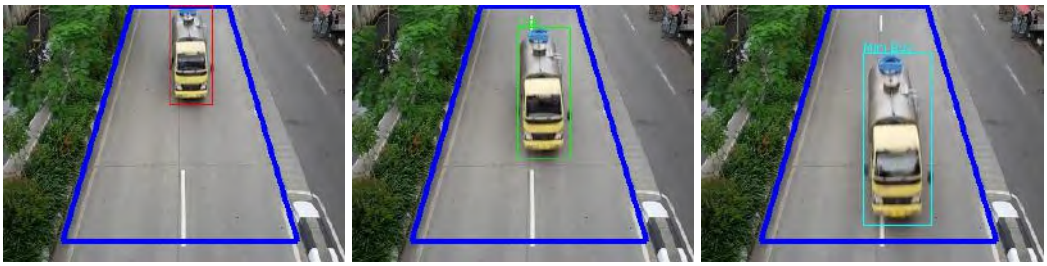

Gambar 5.7: Kendaraan dengan ukuran jenis mobil.




Gambar 5.8: Kendaraan dengan ukuran jenis mini bis / mini truk.


Gambar 5.9: Kendaraan dengan ukuran jenis bis / truk .

dilakukan karena jarak yang berbeda akan menyebabkan klasifikasi yang berbeda pula. Jarak yang jauh dari jembatan penyebrangan cenderung akan mengklasifikasikan kendaraan dengan ukuran lebih kecil dari pada jarak yang dekat dari jembatan penyebrangan. Contoh dari penjelasan tersebut disajikan pada Gambar 5.10.



Gambar 5.10: Perbedaan klasifikasi terhadap jarak. Kiri : mini truk dianggap motor. Tengah : mini truk dianggap mobil. Kanan : klasifikasi tepat.

Dalam uji coba ini klasifikasi yang dilihat dan dinilai kebenarannya adalah ketika kendaraan akan meninggalakan ROI. Klasifikasi dianggap benar jika hasil klasifikasi dalam program sama dengan jenis kendaraan sebenarnya. Sementara klasifikasi dianggap salah jika obyek yang telah diklasifikasi tidak tercatat dan terhitung dalam program. Klasifikasi juga dianggap salah ketika obyek terklasifikasi 2 jenis kendaraan.

Selanjutnya klasifikasi dari setiap video diuji. Data masukan pada uji coba ini adalah video dengan durasi 1 menit. Berdasarkan dari pengujian parameter *scale* pada bab sebelumnya nilai *scale* yang digunakan pada uji coba kali ini adalah nilai *scale* yang mendapatkan tingkat keakuratan tertinggi pada masing-masing jalan. Selain itu inisialisasi himpunan *fuzzy* juga berdasarkan eksperimen untuk mencari rerata jumlah piksel dan rerata luas area pada bab sebelumnya. Hasil dari uji coba ini diberikan pada Tabel 5.3. Sementara Tabel 5.2 menyajikan jumlah kendaraan sebenarnya tiap jenis dari tiap jalan.

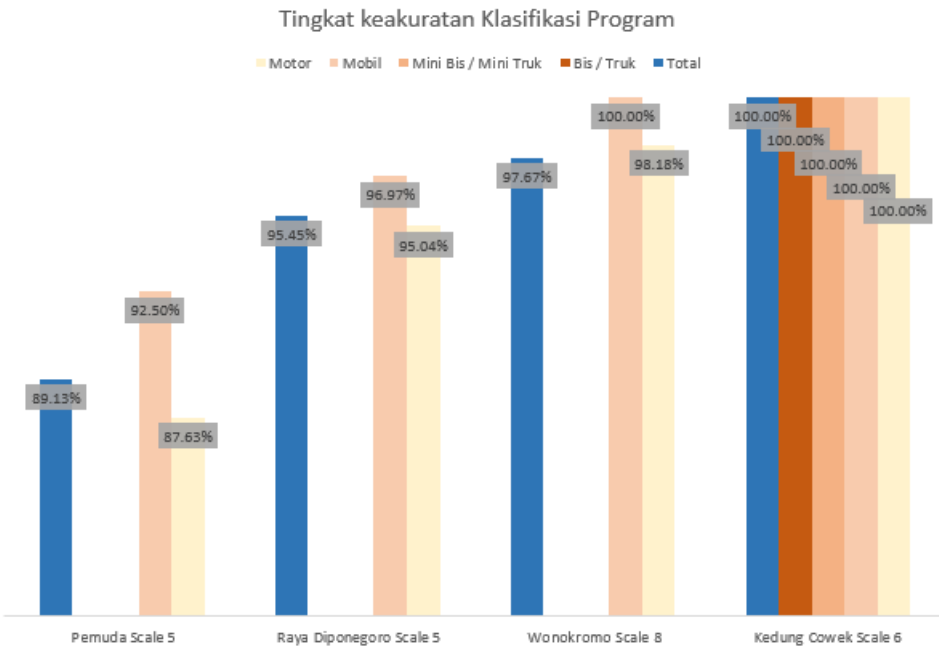
Tabel 5.2: Tabel total kendaraan sebenarnya

| Nama Jalan | Jenis Kendaraan | | | | Total Obyek |
|---|---|---|---|---|---|
| | Motor | Mobil | Mini Bis / Mini Truk | Bis / Truk | |
| Kedung Cowek | 25 | 9 | 4 | 1 | 39 |
| Wonokromo | 110 | 18 | 1 | 0 | 129 |
| Raya Diponegoro | 121 | 33 | 0 | 0 | 154 |
| Pemuda | 97 | 40 | 1 | 0 | 138 |

Tabel 5.3: Tabel jumlah klasifikasi benar dari program

| Nama Jalan | Jenis Kendaraan | | | | Total Obyek |
|---|---|---|---|---|---|
| | Motor | Mobil | Mini Bis / Mini Truk | Bis / Truk | |
| Kedung Cowek (*scale* 6) | 25 | 9 | 4 | 1 | 39 |
| Wonokromo (*scale* 8) | 108 | 18 | 1 | 0 | 126 |
| Raya Diponegoro (*scale* 5) | 115 | 32 | 0 | 0 | 147 |
| Pemuda (*scale* 5) | 85 | 37 | 1 | 0 | 123 |

Dari tabel Tabel 5.3 akan dilakukan perhitungan tingkat keakuratan program dalam mengklasifikasi kendaraan bergerak. Perhitungan tingkat keakuratan program berdasarkan pada persamaan 3.1. Hasil dari perhitungan tingkat akurasi program dalam mengklasifikasi kendaraan bergerak dapat dilihat pada Gambar 5.11.




Gambar 5.11: Tingkat keakuratan klasifikasi kendaraan pada program.

Dari hasil uji coba pada subbab ini Jalan Kedung Cowek mendapatkan tingkat akurasi 100%. Hal ini dikarenakan karena tingkat akurasi deteksi obyek pada video Jalan Kedung Cowek dengan *scale* tersebut sangat tinggi. Jalan Wonokromo juga mendapatkan tingkat akurasi sebesar 97.67%, hal tersebut dikarenakan kendaraan

yang melintas di jalan tersebut jarang yang berhimpit. Berbeda dengan Jalan Kedung Cowek dan Jalan Wonokromo, pada Jalan Raya Diponegoro tingkat akurasi klasifikasinya sebesar 95.45%. sedangkan Jalan Raya Pemuda tingkat akurasi klasifikasinya sebesar 89.13%. Hal tersebut dikarenakan jalan Raya Diponegoro dan Jalan Pemuda merupakan jalan dengan tingkat kepadatan yang tinggi.

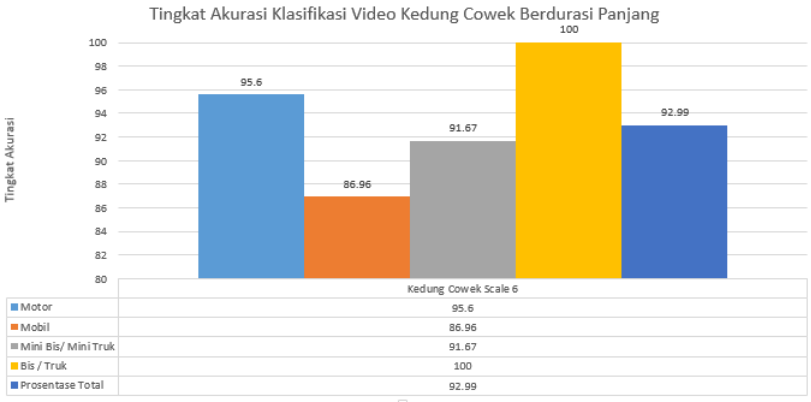
Hasil perhitungan tiap jenis kendaraan yang diperoleh diatas dapat digunakan sebagai acuan instansi terkait dalam membuat kebijakan baru dengan tujuan mengurangi kemacetan, seperti merubah sistem lampu lalu lintas pada persimpangan yang terkait dengan jalan yang diteliti.

## 5.3 Uji Coba Klasifikasi Berdurasi Panjang

Untuk lebih mengetahui sejauh mana konsistensi program dalam mengklasifikasikan kendaraan bergerak, pengujian dilanjutkan dengan data masukan video berdurasi 4 menit, dengan jalan yang mempunyai kendaraan beragam yaitu Jalan Kedung Cowek. Hasil dari uji coba ini adalah dapat dilihat pada Tabel 5.4. Sementara tingkat akurasi program dapat dilihat pada Gambar 5.12.

Tabel 5.4: Tabel hasil klasifikasi durasi panjang.

| Nama Jalan | Jenis Kendaraan | | | |
|---|---|---|---|---|
| | Motor | Mobil | Mini Bis / Mini Truk | Bis / Truk |
| Kedung Cowek *scale* 6 | 87 | 40 | 11 | 8 |
| Kedung Cowek Jumlah Sebenarnya | 91 | 46 | 12 | 8 |

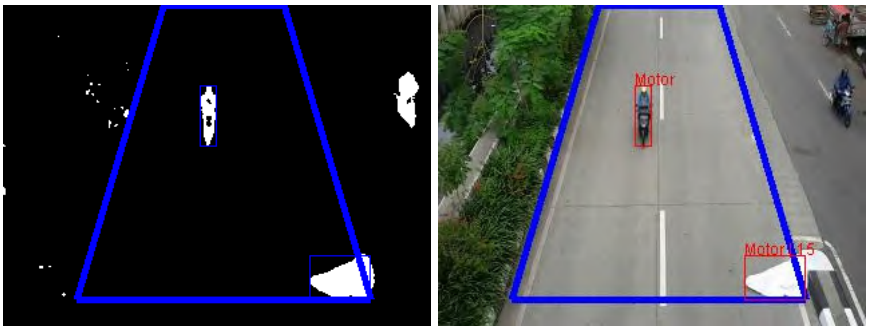



Gambar 5.12: Tingkat Keakuratan klasifikasi program pada video berdurasi panjang.

Dari hasil diatas dapat dilihat kendaraan jenis mobil mempunyai prosentase keakuratan yang paling rendah. Hal tersebut diakibatkan dikarenakan faktor warna mobil pada video tersebut banyak memiliki warna yang menyerupai *background*. Faktor tersebut juga dialami kendaraan jenis mini bis atau mini truk dimana bagian belakang atau angkutan dari mini truk memiliki warna yang hampir sama dengan *background*. Sementara itu bis dan truk dalam video memiliki tingkat keakuratan sempurna. hal tersebut dikarenakan rata-rata jumlah piksel dan rata-rata luas area sangat jauh dari kendaraan lainnya. Selain itu piksel dari bis dan truk juga hampir semua utuh, tidak ada yang terpisah, itu karena ukuran bis dan truk yang besar. Sedangkan kesalahan kendaraan jenis sepeda motor dalam hasil tersebut kebanyakan dikarenakan karena tertutup oleh kendaraan yang lebih besar seperti bis dan truk.

Terdapat beberapa faktor yang dapat mempengaruhi klasifikasi kendaraan bergerak diantaranya sebagai berikut :

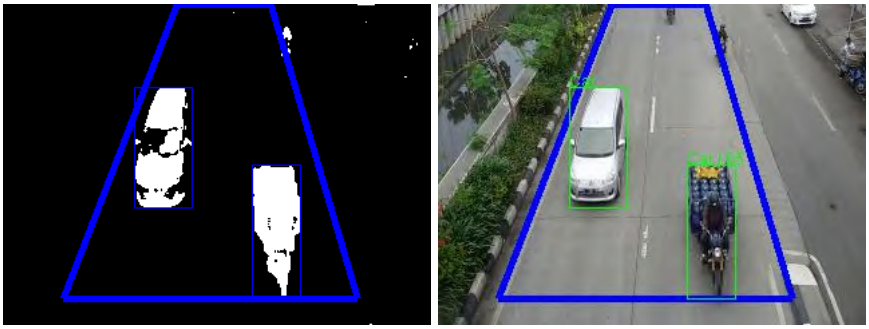
1. Faktor obyek yang tidak diinginkan melintas ROI
   Obyek *foreground* lainnya seperti orang yang lebih memilih berjalan di jalan dari pada jembatan penyebrangan atau benda lain yang dideteksi sebagai *foreground* karena bergerak. Hasil klasifikasi dengan faktor tersebut disajikan pada Gambar 5.13.




Gambar 5.13: Faktor obyek selain kendaraan.

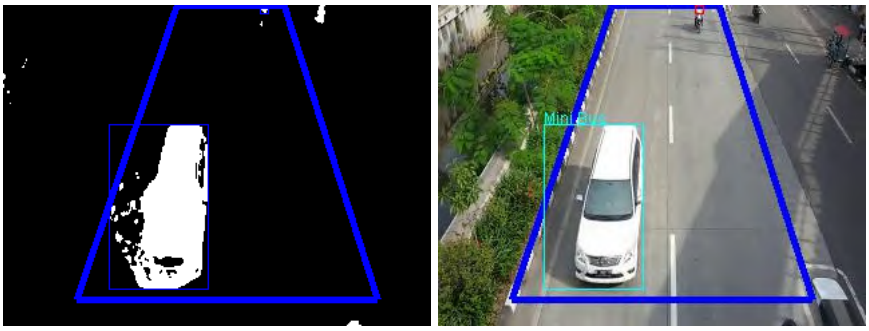
2. Kesalahan deteksi obyek
   Kesalahan yang disebabkan oleh deteksi obyek juga akan mempengaruhi klasifikasi kendaraan bergerak. Hal tersebut dikarenakan perhitungan klasifikasi berdasarkan jumlah piksel dan luas area obyek yang diperoleh dari hasil deteksi obyek.

3. Kendaraan yang memiliki ukuran tidak wajar
   Kendaraan seperti sepeda motor roda tiga dengan *box* dibelakang yang berukuran lebih mirip mobil atau sepeda motor yang mengangkut benda yang besar. Hasil klasifikasi dengan faktor tersebut disajikan pada Gambar 5.14.



Gambar 5.14: Faktor ukuran yang tidak wajar.

4. Bayangan
   Bayangan yang tidak terdeteksi ketika proses *Background Subtraction* dengan GMM akan menjadi masalah ketika perhitungan klasifikasi karena jumlah piksel dan luas area obyek pasti terlihat lebih besar dari yang semestinya. Hasil klasifikasi dengan faktor tersebut disajikan pada Gambar 5.15.



Gambar 5.15: Faktor bayangan dalam klasifikasi.

# LAMPIRAN

# LAMPIRAN A
# Kode Program Deteksi Obyek

```
package com.charisma.thesis.process;
import java.util.ArrayList;
import java.util.List;
import org.opencv.core.Mat;
import org.opencv.core.Size;
import com.charisma.thesis.model.Frame;
import com.charisma.thesis.model.Vehicle;
import com.charisma.thesis.utils.Global;

public class ImageObjectDetection {

	private int threshold;
	private Mat image;
	private Mat temp = null;
	private double grad1;
	private double grad2;
	private Vehicle vehicle;
	private static int index = 1;

	public ImageObjectDetection() {
		this.threshold = Global.Input.detectionScale;
	}


	public Frame getVehicle(Mat image){
		Frame frame = new Frame();
		this.image = image;
		temp = Mat.zeros(new Size(image.width(), image.height()), image.type());
		List<Vehicle> vehicles = new ArrayList<>();

		grad1 = (Global.ROI.bounds[0] - Global.ROI.bounds[3])*1.0 / (Global.ROI.bounds[1] - Global.ROI.bounds[4])*1.0;
		grad2 = (Global.ROI.bounds[0] + Global.ROI.bounds[2] - Global.ROI.bounds[3] - Global.ROI.bounds[5])*1.0 /
				(Global.ROI.bounds[1] - Global.ROI.bounds[4])*1.0;

		for (int i = Global.ROI.bounds[1]; i <= Global.ROI.bounds[4]; i++) {
			int start = (int) (Global.ROI.bounds[0] + Math.floor(grad1*(i-Global.ROI.bounds[1])));
			int end = (int) (Global.ROI.bounds[0] + Global.ROI.bounds[2] + Math.floor(grad2*(i-Global.ROI.bounds[1])));

			for (int j = start; j <= (end < Global.ROI.image.getWidth(null) ? end : Global.ROI.image.getWidth(null) ) ; j++) {
				if (image.get(i, j)[0] == 255 && temp.get(i, j)[0] == 0) {
					temp.put(i, j, new double[] { 1});
					vehicle = new Vehicle();
					checkMinMax(i, j);
					getArround(i, j);

					if (vehicle.getPixBottom() - vehicle.getPixTop() >= 5
						&& vehicle.getPixRight() - vehicle.getPixLeft() >= 5) {
						vehicles.add(vehicle);
					}
				}
			}
		}

		frame.setVehicles(vehicles);
		return frame;
	}
```

```
	private void getArround(int i, int j){
		for (int x = i-threshold; x <= i+threshold; x++) {
			int change = threshold-Math.abs(i-x);
			int start = (int) (Global.ROI.bounds[0] + Math.floor(grad1*(x-Global.ROI.bounds[1])));
			int end = (int) (Global.ROI.bounds[0] + Global.ROI.bounds[2] + Math.floor(grad2*(x-Global.ROI.bounds[1])));
			for (int y = j-change; y <= j+change; y++) {
				if(y >= start && x >= Global.ROI.bounds[1] && y <= end && x <= Global.ROI.bounds[4]){
					if(image.get(x, y)[0] == 255  &&   temp.get(x, y)[0] == 0 ){
						temp.put(x, y, new double[]{1});
						index++;
						checkMinMax(x, y);
						getArround(x, y);
					}
				}
			}
		}
		return;
	}

	private void checkMinMax(int y, int x) {
		vehicle.setSizePixel(vehicle.getSizePixel() + 1);
		vehicle.setPixTop(-1 == vehicle.getPixTop() || y < vehicle.getPixTop() ? y : vehicle.getPixTop());
		vehicle.setPixBottom(-1 == vehicle.getPixBottom() || y > vehicle.getPixBottom() ? y : vehicle.getPixBottom());
		vehicle.setPixLeft(-1 == vehicle.getPixLeft() || x < vehicle.getPixLeft() ? x : vehicle.getPixLeft());
		vehicle.setPixRight(-1 == vehicle.getPixRight() || x > vehicle.getPixRight() ? x : vehicle.getPixRight());
	}
}
```

# LAMPIRAN B
# Kode Program Klasifikasi Kendaraan Bergerak

```
package com.charisma.thesis.process;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import com.charisma.thesis.model.CogSet;
import com.charisma.thesis.model.DomainSet;
import com.charisma.thesis.model.Set;
import com.charisma.thesis.utils.Global;

public class VehicleClassification {
	private int pixel = 0;
	private int area = 0;
	private Set setPixel = new Set("Pixel");
	private Set setArea = new Set("Area");
	private Set setWeight = new Set("Weight");
	private double[][] alphaParent;

	public VehicleClassification() {initialize();}

	private void initialize(){
		double[][][] val = Global.fuzzy.value;
		List<DomainSet> pixelSets = new ArrayList<DomainSet>();
		DomainSet pixel1 = new DomainSet("Low", 0, val[0][0][0], val[0][0][1], val[0][0][2]);
		DomainSet pixel2 = new DomainSet("Medium", 1, val[0][1][0], val[0][1][1], val[0][1][2]);
		DomainSet pixel3 = new DomainSet("High", 2, val[0][2][0], val[0][2][1], val[0][2][2]);
		pixelSets.add(pixel1);
		pixelSets.add(pixel2);
		pixelSets.add(pixel3);
		setPixel.setDomainSets(pixelSets);

		List<DomainSet> areaSets = new ArrayList<DomainSet>();
		DomainSet area1 = new DomainSet("Low", 0, val[1][0][0], val[1][0][1], val[1][0][2]);
		DomainSet area2 = new DomainSet("Medium", 1, val[1][1][0], val[1][1][1], val[1][1][2]);
		DomainSet area3 = new DomainSet("High", 2, val[1][2][0], val[1][2][1], val[1][2][2]);
		areaSets.add(area1);
		areaSets.add(area2);
		areaSets.add(area3);
		setArea.setDomainSets(areaSets);

		List<DomainSet> weightSets = new ArrayList<DomainSet>();
		DomainSet weight1 = new DomainSet("Low", 0, val[2][0][0], val[2][0][1], val[2][0][2]);
		DomainSet weight2 = new DomainSet("Medium", 1, val[2][1][0], val[2][1][1], val[2][1][2]);
		DomainSet weight3 = new DomainSet("High", 2, val[2][2][0], val[2][2][1], val[2][2][2]);
		weightSets.add(weight1);
		weightSets.add(weight2);
		weightSets.add(weight3);
		setWeight.setDomainSets(weightSets);
	}

	private int getClassification(double weight) {
		double[] classify = Global.fuzzy.classify;
		if(weight >= classify[0] && weight <= classify[1])
			return 0;
		else if(weight > classify[1] && weight <= classify[2])
			return 1;
		else if(weight > classify[2] && weight <= classify[3])
			return 2;
		else if(weight > classify[3] && weight <= classify[4])
			return 3;
		else
			return -1;
	}
```

```
public Object[] classification (int pixel, int area) {
	this.pixel = pixel;
	this.area = area;
	this.alphaParent = new double[][]{{0, 0, 0}, {0, 0, 0}};

	for (int i = 0; i < setPixel.getDomainSets().size(); i++) {
		for (int j = 0; j < setArea.getDomainSets().size(); j++) {
			if (pixel > setPixel.getDomainSets().get(i).getMin() &&
				pixel < setPixel.getDomainSets().get(i).getMax() &&
				area > setArea.getDomainSets().get(j).getMin() &&
				area < setArea.getDomainSets().get(j).getMax()){
				updateAlphaPredicate(setPixel.getDomainSets().get(i), setArea.getDomainSets().get(j));
	}		}		}

	List<CogSet> allCogSet = new ArrayList<CogSet>();
	for (int i = 0; i < setWeight.getDomainSets().size(); i++) {
		if( setWeight.getDomainSets().get(i).getAlpha() > 0){
			List<CogSet> sets = getCogSet(i);
			allCogSet = allCogSet.size() == 0 ? sets : updateCogSets(allCogSet, sets);
	}		}

	double sumMoment = 0, sumWidth = 0;
	for (CogSet cogSet : allCogSet) {
		sumMoment += getMoment(cogSet);
		sumWidth += getWidth(cogSet);
	}

	for (DomainSet set : setWeight.getDomainSets()) {
		set.setAlpha(0);
	}

	double weight = sumMoment/sumWidth;
	return new Object[]{weight, getClassification(weight)};
}

private double getWidth(CogSet cogSet) {
	if(cogSet.getGrad() == 0)
		return cogSet.getAlpha()*(cogSet.getEnd()-cogSet.getStart());
	else
		return (cogSet.getyStart()+cogSet.getyEnd())*(cogSet.getEnd()-cogSet.getStart())/2;
}

private double getMoment(CogSet cogSet) {
	if(cogSet.getGrad() == 0){
		return getMomentZero(cogSet.getAlpha(), cogSet.getEnd()) -
		getMomentZero(cogSet.getAlpha(), cogSet.getStart());
	} else if(cogSet.getGrad() > 0){
		return getMomentUpHill(cogSet.getMin(), cogSet.getMid(), cogSet.getEnd()) -
		getMomentUpHill(cogSet.getMin(), cogSet.getMid(), cogSet.getStart());
	} else {
		return getMomentDownHill(cogSet.getMid(), cogSet.getMax(), cogSet.getEnd()) -
		getMomentDownHill(cogSet.getMid(), cogSet.getMax(), cogSet.getStart());
	}
}

private double getMomentZero(double alpha, double x){
	return (alpha/2)*Math.pow(x, 2);
}

private double getMomentUpHill(double min, double mid, double x){
	return (((1/(mid-min))/3)*(Math.pow(x, 3))) - (((min/(mid-min))/2)*Math.pow(x, 2));
}
private double getMomentDownHill(double mid, double max, double x){
	return (((1/(max-mid))/3)*(Math.pow(x, 3)))*(-1) + (((max/(max-mid))/2)*Math.pow(x, 2));
}
```

```
private List<CogSet> updateCogSets(List<CogSet> cogSets, List<CogSet> sets) {
	List<CogSet> updateCog = new ArrayList<CogSet>();
	CogSet cog1 = new CogSet();
	CogSet cog2 = new CogSet();
	boolean cog1Turn = true;

	int index1 = 0;
	int index2 = 0;
	int lastCross1 = 0;
	int lastCross2 = 0;
	while (index1 < cogSets.size() || index2 < sets.size() ) {
		cog1 = cogSets.get(index1);
		cog2 = sets.get(index2);

		Object[] oCross = checkCogCross(cog1, cog2);
		if ((oCross[0] instanceof Boolean)){
			if (cog1Turn) {
				if (index1 < cogSets.size()-1) {
					updateCog.add(cog1);
					index1++;
				} else {
					if(index2 < sets.size()-1){
						if (index1-lastCross1 < 2) updateCog.add(cog1);
						index2++;
					} else {
						updateCog.add(cog1);
						break;
				}	}
			} else {
				if (index2 < sets.size()-1) {
					updateCog.add(cog2);
					index2++;
				} else {
					if(index1 < cogSets.size()-1) {
						if (index2-lastCross2 < 2) updateCog.add(cog2);
						index1++;
					} else {
						updateCog.add(cog2);
						break;
			}	}	}
		} else {
			double[] cross = (double[]) oCross[0];
			if (cog1Turn){
				cog1.setEnd(cross[0]);
				cog1.setyEnd(cross[1]);
				cog2.setStart(cross[0]);
				cog2.setyStart(cross[1]);
				updateCog.add(cog1);
				updateCog.add(cog2);
			} else {
				cog1.setStart(cross[0]);
				cog1.setyStart(cross[1]);
				cog2.setEnd(cross[0]);
				cog2.setyEnd(cross[1]);
				updateCog.add(cog2);
				updateCog.add(cog1);
			}
			lastCross1 = index1;
			lastCross2 = index2;
			if(! (index1 < cogSets.size()-1) && ! (index2 < sets.size()-1)) break;
			if(index1 < cogSets.size()-1) index1++;
			if(index2 < sets.size()-1) index2++;
			cog1Turn = !cog1Turn;
	}	};
	return updateCog;
}
```

```
private Object[] checkCogCross(CogSet cog1, CogSet cog2){
		double x1 = cog1.getStart();
		double y1 = cog1.getyStart();
		double x2 = cog1.getEnd();
		double y2 = cog1.getyEnd();
		double x3 = cog2.getStart();;
		double y3 = cog2.getyStart();
		double x4 = cog2.getEnd();
		double y4 = cog2.getyEnd();

		double m1 = (y2-y1)/(x2-x1);
		double m2 = (y4-y3)/(x4-x3);

		double c1 = y1-(m1*x1);
		double c2 = y3-(m2*x3);

		double xRes = (c2-c1)/(m1-m2);
		double yRes = m1*xRes+c1;

		Object[] object = new Object[2];
		if (Double.isNaN(xRes) || Double.isNaN(yRes) ||
				xRes < x1 || xRes > x2 || xRes < x3 || xRes > x4){
				object[0] = false;
		} else {
				object[0] = new double[]{xRes, yRes};
				object[1] = m1 <= m2;
		}
		return object;
}

private void updateAlphaPredicate(DomainSet pixelSet, DomainSet areaSet) {
		double alphaPixel = getAlpha(pixelSet.getNoSet(), pixelSet.getMin(),
					pixelSet.getMid(), pixelSet.getMax(), pixel, setPixel.getDomainSets().size());
		double alphaArea = getAlpha(areaSet.getNoSet(), areaSet.getMin(),
					areaSet.getMid(), areaSet.getMax(), area, setArea.getDomainSets().size());

		alphaParent[0][pixelSet.getNoSet()] = alphaPixel;
		alphaParent[1][areaSet.getNoSet()] = alphaArea;

		double alphaWeight =
				setWeight.getDomainSets().get(Global.fuzzy.rules[pixelSet.getNoSet()][areaSet.getNoSet()]).getAlpha();
		double alphaMin = Math.min(alphaPixel, alphaArea);
		int index = Global.fuzzy.rules[pixelSet.getNoSet()][areaSet.getNoSet()];
		if (alphaWeight < alphaMin){
				setWeight.getDomainSets().get(index).setAlpha(alphaMin);
		}

		double alphaWeightPixel = setWeight.getDomainSets().get(pixelSet.getNoSet()).getAlpha();
		if (alphaWeightPixel < alphaParent[0][pixelSet.getNoSet()]){
				setWeight.getDomainSets().get(pixelSet.getNoSet()).setAlpha(alphaParent[0][pixelSet.getNoSet()]);
		}

		double alphaWeightArea = setWeight.getDomainSets().get(areaSet.getNoSet()).getAlpha();
		if (alphaWeightArea < alphaParent[1][areaSet.getNoSet()]){
				setWeight.getDomainSets().get(areaSet.getNoSet()).setAlpha(alphaParent[1][areaSet.getNoSet()]);
		}
}

private double getValueUpHill(double min, double mid, double alpha){
		return (double)((alpha * (mid - min)) + min);
}

private double getValueDownHill(double mid, double max, double alpha){
		return (double)(max - (alpha * (max - mid)));
}
```

```
	private double getAlpha(int noSet, double min, double mid, double max, int value, int sizeSet){

		if( 0 == noSet )
			return (mid > value ? 1 :
				(double)(max - value)/(max - mid));

		else if(sizeSet-1 == noSet)
			return (mid < value ? 1 :
				(double)(value - min)/(mid - min));

		else
			return (mid > value ?
				(double)(value - min)/(mid - min):
				(double)(max - value)/(max - mid));
	}

	private List<CogSet> getCogSet(int noSet){
		int sizeSet = setWeight.getDomainSets().size() - 1;
		double min = setWeight.getDomainSets().get(noSet).getMin();
		double mid = setWeight.getDomainSets().get(noSet).getMid();
		double max = setWeight.getDomainSets().get(noSet).getMax();
		double alpha = setWeight.getDomainSets().get(noSet).getAlpha();
		List<CogSet> cogSets = new ArrayList<CogSet>();

		if( 0 == noSet){
			double value = getValueDownHill(mid, max, alpha);
			if(alpha < 1){
				CogSet set1 = new CogSet(noSet, mid, value, alpha, 0, alpha, alpha, min, mid, max);
				cogSets.add(set1);
			}
			CogSet set2 = new CogSet(noSet, value, max, alpha, -1, alpha, 0.0, min, mid, max);
			cogSets.add(set2);
		} else if(sizeSet == noSet) {
			double value = getValueUpHill(min, mid, alpha);
			CogSet set1 = new CogSet(noSet, min, value, alpha, 1, 0.0, alpha, min, mid, max);
			cogSets.add(set1);
			if(alpha < 1){
				CogSet set2 = new CogSet(noSet, value, mid, alpha, 0, alpha, alpha, min, mid, max);
				cogSets.add(set2);
			}
		} else {
			double value1 = getValueUpHill(min, mid, alpha);
			double value2 = getValueDownHill(mid, max, alpha);
			CogSet set1 = new CogSet(noSet, min, value1, alpha, 1, 0.0, alpha, min, mid, max);
			cogSets.add(set1);
			if(alpha < 1){
				CogSet set2 = new CogSet(noSet, value1, value2, alpha, 0, alpha, alpha, min, mid, max);
				cogSets.add(set2);
			}
			CogSet set3 = new CogSet(noSet, value2, max, alpha, -1, alpha, 0.0, min, mid, max);
			cogSets.add(set3);

		}

		return cogSets;
	}


}
```

# BAB VI
# KESIMPULAN DAN SARAN

Bab ini berisi tentang beberapa kesimpulan yang dihasilkan berdasarkan 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 terhadap sistem klasifikasi jenis kendaraan dengan logika *fuzzy*, maka dapat diambil beberapa kesimpulan sebagai berikut:

1. Penelitian ini berhasil mendeteksi obyek hasil *output* dari *background subtraction* dengan melakukan proses *smoothing* dan *shadow removal* terlebih dahulu untuk mengurangi *noise* dan bayangan pada *image*. Deteksi obyek dilakukan dengan melakukan pencarian piksel suatu obyek kemudian dilanjutkan dengan mencari piksel-piksel yang lain dari obyek tersebut secara rekursif.

2. Parameter *scale* terbaik dari tiap video berbeda-beda berdasarkan kondisi dari tempat pengambilan video. Jalan Kedung Cowek memiliki tingkat akurasi sebesar 100% dengan *scale* 6. Jalan Wonokromo memiliki tingkat akurasi sebesar 97.67% dengan *scale* 8. Jalan Raya Diponegoro memiliki tingkat akurasi sebesar 92.21% dengan *scale* 5. Jalan Pemuda memiliki tingkat akurasi sebesar 84.78% dengan *scale* 5.

3. Penelitian ini berhasil mengklasifikasi dalam 4 jenis kendaraan berdasarkan ukuran yaitu motor, mobil, mini bis / mini truk, dan bis / truk dengan beberapa tahapan : *background subtraction* dengan GMM, *smoothing* dengan median *filter*, *shadow removal*, deteksi obyek, dan klasifikasi kendaraan dengan logika *fuzzy*.

4. Tingkat akurasi tertinggi pada klasifikasi kendaraan bergerak masing-masing jalan, antara lain : Jalan Kedung Cowek sebesar 100%, Jalan Wonokromo sebesar 97.67%, Jalan Raya Diponegoro sebesar 95.45%, Jalan Pemuda sebesar 89.13%. Selain itu sistem yang dibangun berhasil mengklasifikasikan

kendaraan dengan video berdurasi 4 menit di jalan kedung cowek sebesar 93.63%.

## 6.2 Saran

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

1. Program belum dapat menghilangkan bayangan obyek secara keseluruhan sehingga dapat mempengaruhi klasifikasi. Untuk penelitian berikutnya diharapkan untuk mengkhususkan pada proses penghilangan bayangan dari obyek.

2. Untuk penelitian klasifikasi kendaraan bergerak selanjutnya diharapkan dikaitkan dengan algoritma *robust* berdasarkan kondisi video, seperti tinggi jembatan, sudut pengambilan video, dan kondisi cuaca. Hal tersebut bertujuan untuk lebih meminimalkan parameter *input*.

3. Program belum terhubung dengan *database*. Sehingga data penghitungan hanya disimpan sementara oleh memori CPU. Pada penelitian berikutnya program dapat dihubungkan dengan *database*. Sehingga data dapat disimpan dan diolah untuk kepentingan lebih lanjut.

# DAFTAR PUSTAKA


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

Amaluddin, F., Muslim, M. A., Naba, A., (2015). "Klasifikasi Kendaraan Menggunakan Gaussian Mixture Model (GMM) dan Fuzzy Cluster Means (FCM)", *Jurnal EECCIS*, Vol. 9, No. 1.

Buckley J. J., Eslami E., (2002), *An Introduction to Fuzzy Logic and Fuzzy Sets*, Physica-Verlag Heidelberg, New York, USA.

Jailson A. de Brito Jr., Luis Edmundo Prado de Campos, (2007), "Automatic Vehicle Classification Using Learning-based Computer Vision and Fuzzy Logic", *Workshop de Visao Computacional*, Sao Jose do Rio Preto, SP.

Paulraj, M. P., Adom, A. H.., Sundararaj, S., Rahim, Norasmadi Bin Abdul, (2013), "Moving Vehicle Recognition and Classification Based on Time Domain Approach", *Malaysian Technical Universities Conference on Engineering and Technology 2012*, Part 1- Electronic and Electrical Engineering, Perlis.

Rafael C. Gonzales, Richard E. Woods, (2001), *Digital Image Processing*, Prentice Hall, New Jersey.

Rahim, N. A., Paulraj, M. P., Adom, A. H., (2013), "Adaptive Boosting with SVM Classifier for Moving Vehicle Classification", *Procedia Engineering*, Perlis, hal. 411 419.

Y. Benezith, P. M. Jodoin, B. Emile, H. Lauren, C. Rosenberger, (2012), "Comparative Study of Background Subtraction Algorithm", *Journal of Electronic Imaging*, Vol. 19, hal. 1-30.

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

Sung-Wook, Kim, Kwang-soo, Kim, dan Joo-hyung, Lee, (2001), "Application of Fuzzy Logic to Vehicle Classification Algorithm in Loop/Piezo-Sensor Fusion System", *Asian Journal of Control*, Vol. 3, No. 1, hal. 64-68.

Sutoyo. T., Mulyanto, E., Suhartono, V., Nurhayati, O.D. dan Wijanarto, (2009), *Teori Pengolahan Citra Digital*, Penerbit ANDI, Yogyakarta.

Zimmermann H. J., (2001), *Fuzzy Set Theory-and Its Applications Fourth Edition*, Springer Science+Business Media, New York, USA.

# BIODATA PENULIS

Penulis bernama lengkap Bayu Charisma Putra, dilahirkan di sebuah pulau yang bernama Masalembu Kab. Sumenep pada tanggal 3 September 1991. Penulis telah menempuh pendidikan formal yaitu di SD Muhammadiyah 2 Sidoarjo pada tahun 1997 dan lulus pada tahun 2003 selanjutnya SMP Negeri 16 Surabaya pada tahun 2003-2006, SMA Negeri 13 Surabaya pada tahun 2006-2009. Setelah itu penulis melanjutkan pendidikan sarjana di Jurusan Matematika FMIPA, Universitas Airlangga melalui jalur prestasi pada tahun 2009 dan lulus pada tahun 2013. Di Matematika Universitas Airlangga penulis tertarik mengambil bidang minat Ilmu Komputer, dengan judul skripsi ”Aplikasi *Text Mining* Menggunakan Algoritma TF-IDF dan VSM pada *E-library* Fakultas Sains dan Teknologi Universitas Airlangga”. Penulis sempat bekerja pada tahun 2013 selama 2,5 tahun di sebuah perusahaan swasta yang merupakan vendor IT perbankan. Pada tahun 2014, penulis melanjutkan kuliah di S2 Matematika ITS Surabaya. Di jurusan Matematika ITS penulis mengambil bidang minat Ilmu Komputer, dengan judul penelitian ”Klasifikasi Kendaraan Bergerak dengan Logika *Fuzzy* Berbasis Pengolahan Citra”. Informasi, kritik, dan saran yang berhubungan dengan Tesis ini dapat ditujukan ke alamat e-mail: `bayu.charisma.putra@gmail.com`