IMPLEMENTASI METODE KMEANS-GMM BERBASIS *HU MOMENTS* UNTUK KLASIFIKASI KENDARAAN PADA VIDEO LALU LINTAS

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Gigih Trianung Anggoro
NIM: 115090601111006

PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

IMPLEMENTASI METODE KMEANS-GMM BERBASIS *HU MOMENTS* UNTUK KLASIFIKASI KENDARAAN PADA VIDEO LALU LINTAS

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Gigih Trianung Anggoro
NIM: 115090601111006

Skripsi ini telah diuji dan dinyatakan lulus pada
14 April 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Imam Cholissodin, S.Si., M.Kom.	Suprapto, S.T, M.T.
NIK: 201201 850719 1 001	NIP: 19710727 196603 1 001
/*jika terdapat NIK saja*/	/*jika tidak terdapat NIP, NIK, atau keduanya*/

Mengetahui
Ketua Program Studi Informatika

Issa Arwani, S.Kom, M.Sc.
NIP: 19830922 201212 1 003
/*jika terdapat NIP saja*/

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 14 April 2016

Gigih Trianung Anggoro

NIM: 115090601111006

KATA PENGANTAR

Alhamdulillahi rabbil ‘alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayah-Nya, penulis dapat menyelesaikan laporan skripsi yang berjudul “Implementasi Metode KMEANS-GMM Berbasis *Hu Moments* untuk Klasifikasi Kendaraan pada Video Lalu Lintas”. Dalam pelaksanaan dan penulisan laporan skripsi ini penulis mendapatkan banyak bantuan dari berbagai pihak baik secara moril maupun materil. Dalam kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Bapak Imam Cholissodin, S.Si., M.Kom dan Bapak Suprapto, S.T, M.T selaku dosen pembimbing skripsi yang telah dengan sabar membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
2. Bapak Ir. Sutrisno, M.T, Bapak Ir. Heru Nurwasito, M.Kom, Bapak Himawat Aryadita, S.T, M.Sc, dan Bapak Edy Santoso, S.Si, M.Kom selaku Ketua, Wakil Ketua 1, Wakil Ketua 2, dan Wakil Ketua 3 Fakultas Ilmu Komputer Universitas Brawijaya.
3. Bapak Drs. Marji, M.T dan Bapak Issa Arwani , S.Kom, M.Sc selaku Ketua dan Sekretaris Program Studi Informatika/Ilmu Komputer Universitas Brawijaya.
4. Ibu Puji Sulasih dan Ayah Edy Purwito, S.P selaku orang tua penulis yang telah memberikan kasih sayang, dukungan, dan do’a kepada penulis untuk menyelesaikan skripsi ini.
5. Seluruh Dosen Program Studi Informatika/Ilmu Komputer atas ilmu yang telah diberikan kepada penulis selama menempuh studi di Fakultas Ilmu Komputer Universitas Brawijaya.
6. Seluruh Civitas Akademika Program Studi Informatika/Ilmu Komputer Fakultas Ilmu Komputer Universitas Brawijaya yang telah memberi bantuan dan dukungan kepada penulis.
7. Wisnu Wijaya, Luthfi Hidayat, Pratiwi Kurniasari, Aisya Ami Wardhani, Abu Bakar, M. Choirul Rahmadan, Stevanie Amanda, Latifah Maulida Rahma, Adzhana Hasfi, Haris Fachruddin Al Mahi, Fakhry Ihsan firdaus, Singgih Rochmad Saputra, Yusuf Aji Wibowo dan seluruh teman-teman ilkom 2011 atas segala dukungan dan do’a kepada penulis sehingga dapat menyelesaikan skripsi ini.
8. Seluruh pihak yang membantu kelancaran atas skripsi ini yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa dalam penyusunan laporan skripsi ini masih banyak kekurangan baik format penulisan maupun isinya. Oleh karena itu, saran dan kritik membangun dari para pembaca senantiasa penulis harapkan guna perbaikan bagi laporan skripsi selanjutnya. Semoga laporan skripsi ini dapat memberikan manfaat bagi semua pihak, Amin.

Malang, 14 April 2016

Gigih Trianung Anggoro
gigihtrianung@gmail.com

ABSTRAK

Teknologi merupakan hal penting pada masa sekarang ini yang memiliki banyak pengaruh besar dalam kehidupan sehari-hari manusia. Salah satu teknologi yang sangat berpengaruh adalah komputer. komputer dapat membantu berbagai pekerjaan manusia seperti salah satu contohnya adalah mengelompokkan dan menghitung jumlah kendaraan dijalan raya dengan menggunakan pengolahan citra digital. Metode yang dapat digunakan untuk mengelompokkan dan menghitung kendaraan dijalan raya adalah KMEANS-GMM. KMEANS-GMM merupakan gabungan metode antara metode *clustering* KMEANS dengan metode *background subtraction* GMM. Tahapan untuk mengelompokkan dan menghitung jumlah kendaraan dijalan raya menggunakan KMEANS-GMM adalah melakukan perhitungan KMEANS, kemudian melakukan perhitungan *background subtraction* menggunakan GMM, kemudian mengekstraksi fitur kendaraan, terakhir mengelompokkan dan menghitung jumlah kendaraan berdasarkan hasil dari ekstraksi fitur. Cara untuk mengelompokkan atau membedakan kendaraan satu dengan yang lainnya adalah dengan menyimpan data hasil dari ekstraksi fitur dari berbagai jenis kendaraan yang disebut sabagai *data training*, kemudian pada saat aplikasi dijalankan setiap kendaraan yang melintas akan dibandingkan dengan *data training*, nilai terkecil dari perbandingan dengan *data training* akan menentukan jenis kendaraan yang sedang melintas. Hasil pengujian akurasi metode KMEANS-GMM menggunakan 10 jenis data uji menunjukkan nilai rata-rata akurasi sebesar 84.8%. Sehingga dapat disimpulkan bahwa metode KMEANS-GMM mampu mengelompokkan dan menghitung jumlah kendaraan dengan baik.

Kata kunci : kendaraan, *clustering*, *background subtraction*, KMEANS, GMM, ekstraksi fitur

ABSTRACT

Technology is an important thing at this era that has a lot of great influence on human daily life. One of the most influential technology is computer. Computer can help a lot of human job as one example is classifying and counting the number of vehicles on the highway using digital image processing method. The method that can be used to classify and count the number of vehicles on the highway is KMEANS-GMM method. KMEANS-GMM is a combination of the methods of KMEANS clustering and GMM background subtraction. The stages to classify and count the number of vehicles on the highway using KMEANS-GMM method is calculating the KMEANS clustering, calculating background subtraction using GMM, extracting features of the vehicle, and the last is classifying and counting the number of vehicles based on the results of features extraction. The procedure to classify or differentiate each vehicle is by storing data extraction features of various types of vehicles, we can call that as training data, and then when the program is running, every passing vehicles are compared with the training data, the smallest value of the comparison data will determine the type of vehicle that was passing. The accuracy of the test results from KMEANS-GMM method using 10 types of test data produces an average accuracy of 84.8%. So it can be concluded that the KMEANS-GMM method is able to classify and count the number of vehicles.

Keywords : vehicle, clustering, background subtraction, KMEANS, GMM, features extraction

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE PROGRAM

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Indonesia merupakan negara dengan jumlah penduduk sebanyak 237.641.326 jiwa menurut data resmi hasil sensus penduduk 2010 (BPS, 2010). Jumlah penduduk mempengaruhi jumlah pertumbuhan kendaraan bermotor di Indonesia, berdasarkan katalog yang diterbitkan Badan Pusat Statistik Indonesia tahun 2013 tercatat jumlah unit mobil penumpang sebesar 11.484.514, bis 2.286.309, mobil barang 5.615.494, dan sepeda motor 84.732.652, kemudian pertumbuhan pertahun mobil penumpang sebesar 9,77%, bis 1,42%, mobil barang 5,70%, dan sepeda motor 12,57% (BPS, 2013). Berdasarkan *The Global Competitiveness Report* 2013-2014 yang menyebutkan bahwa secara keseluruhan infrastruktur Indonesia menempati peringkat 82 dari 148 negara, masih di bawah Brunei Darussalam yang berada di posisi 39 dan Malaysia menempati posisi ke-25. Hal ini menunjukkan perkembangan infrastruktur di Indonesia lambat apabila dibandingkan dengan negara tetangga seperti Brunei Darussalam dan Malaysia. Infrastruktur yang dibahas dalam *The Global Competitiveness Report* 2013-2014 mencakup transportasi, telekomunikasi dan sumber daya energi (TWEF, 2013). Apabila masalah infrastruktur transportasi seperti jalan raya dan masalah pertumbuhan kendaraan bermotor terus bertambah pertahunnya, maka Indonesia akan memiliki serangkaian masalah yang terus bertambah tiap tahunnya.

Masalah insfrastruktur jalan raya di Indonesia kemudian diikuti masalah pertumbuhan kendaraan tiap tahun akan menyebabkan permasalahan-permasalahan baru. Permasalahan baru yang pertama adalah permasalahan menumpuknya kendaran di jalan raya (kemacetan). Kemacetan merupakan masalah yang umum dialami di Indonesia yang disebabkan pertumbuhan kendaraan lebih pesat dibandingkan dengan pertumbuhan infrastruktur jalan raya. Permasalahan kedua adalah pelanggaran lalu lintas, seiring bertambahnya jumlah kendaraan dan permasalahan kemacetan maka banyak pihak yang memilih untuk melanggar aturan lalu lintas seperti menerobos jalur Transjakarta yang sering terjadi di kota Jakarta (Kartika, 2013). Pelanggaran lalu lintas lainnya adalah mengemudi dengan kecepatan tinggi yang membahayakan pengguna jalan raya. Permasalahan ketiga adalah kecelakaan lalu lintas yang sering terjadi karena semakin banyaknya kendaraan di jalan raya dan tidak patuhnya pengguna jalan akan rambu-rambu lalu lintas. Masalah terakhir akibat pertambahnya jumlah kendaraan dan minimnya infrastruktur jalan raya adalah sulitnya melakukan pengawasan lalu lintas oleh pihak petugas kepolisian lalu lintas karena semakin bertambahnya pengguna jalan raya. Dengan permasalahan-permasalahan terkait, maka perlu dibuat sistem cerdas yang mampu melakukan pengawasan video lalu lintas secara otomatis.

Penelitian-penelitian tentang pengawasan video lalu lintas pernah dilakukan oleh peneliti-peneliti sebelumnya. Penelitian yang dilakukan Thou-Ho Chen dkk mengenai masalah kuantifikasi dan klasifikasi kendaraan di jalan raya menggunakan metode *Blob Analysis* menghasilkan nilai rata-rata klasifikasi dan kuantifikasi sebesar 91.7% untuk 3 skema lalu lintas. Namun penelitian yang dilakukan oleh Thou-Ho Chen dkk hanya dapat mengklasifikasikan 2 objek di jalan raya yaitu mobil dan sepeda motor (Chen, 2007). Penelitian tentang pengawasan video lalu lintas juga dilakukan oleh Saroj K. Meher dkk dengan menggunakan metode *Principal Component Analysis* (PCA) dan *Scale-Invariant Feature Transform* (SIFT). Pada penelitian Saroj K. Meher dkk metode PCA digunakan untuk mendapatkan arah dari bayangan objek yang dideteksi kemudian dipisahkan dari wilayah kendaraan, sedangkan SIFT digunakan untuk proses klasifikasi objek kendaraan yang berhasil dideteksi. Penelitian dari Saroj K. Meher dkk menghasilkan peningkatan akurasi dengan melakukan penghapusan bayangan pada objek deteksi. Akurasi yang didapat dari 10% data training menghasilkan 94.12% dengan adanya bayangan objek kemudian akurasi meningkat menjadi 97.35% dengan penghapusan bayangan. Akan tetapi, penelitian yang dilakukan oleh Saroj K. Meher dkk tidak menyebutkan jenis-jenis kendaraan yang berhasil diklasifikasikan seperti mobil, truk, sepeda motor, dan lain-lain, hanya menyebutkan hasil presentase dari keseluruhan klasifikasi (Meher, 2014).

Penelitian tentang pengawasan video lalu lintas juga dilakukan oleh Yingjie Xia dkk dengan menggunakan metode *Expectation-Maximization* (EM) digabungkan dengan *Gaussian Mixture Model* (GMM) untuk melakukan perhitungan jumlah kendaraan di jalan raya. Namun penelitian yang dilakukan Yingjie Xia dkk hanya untuk perhitungan jumlah kendaraan di jalan raya, tidak untuk klasifikasi jenis kendaraan (Xia, 2014). Penelitian lain yang berkaitan dengan pengawasan video lalu lintas dilakukan oleh H. Asaidi dkk menggunakan metode kombinasi KMEANS-*Gaussian Mixture Model* (GMM) dengan berbasis eliminasi bayangan (*Shadow Elimination*) dan *Hu Moments* untuk klasifikasi kendaraan dengan hasil akurasi klasifikasi 96.96%. Penelitian yang dilakukan H. Asaidi dkk dapat mengklasifikasikan 3 objek dijalan raya yaitu mobil, van dan truk (Asaidi, 2014). Berdasarkan penelitian yang disebutkan, penelitian yang dilakukan H. Asaidi dkk memberikan keandalan yang lebih baik dibandingkan penelitian yang lain karena nilai akurasi tetap bernilai besar yaitu 96.96% walaupun dengan jenis objek deteksi yang lebih banyak dibandingkan dengan penelitian yang lain. Penelitian yang lain dapat mengklasifikasikan objek menjadi 2 jenis sedangkan penelitian H. Asaidi dkk dapat mengklasifikasikan 3 jenis objek. Oleh karena itu penulis menggunakan judul skripsi "*Implementasi Metode KMEANS-GMM Berbasis Hu Moments untuk Klasifikasi Kendaraan pada Video Lalu Lintas*" yang diharapkan dapat melakukan pengawasan lalu lintas secara otomatis dan hasil yang optimal.

1.2 Rumusan masalah

Berdasarkan uraian latar belakang yang sudah dijabarkan, maka dapat dirumuskan masalah sebagai berikut :

1. Mengimplementasikan metode KMEANS-GMM berbasis *Hu Moments* untuk klasifikasi kendaraan pada video lalu lintas.
2. Menguji tingkat akurasi dari hasil implementasi metode KMEANS-GMM berbasis *Hu Moments* untuk klasifikasi kendaraan pada video lalu lintas.

1.3 Tujuan

Berdasarkan rumusan masalah yang sudah dijabarkan maka tujuan dari penelitian ini adalah sebagai berikut :

1. Mengimplementasi metode KMEANS-GMM berbasis *Hu Moments* untuk klasifikasi kendaraan pada video lalu lintas.
2. Mengetahui tingkat akurasi dari hasil implementasi metode KMEANS-GMM berbasis *Hu Moments* untuk klasifikasi kendaraan pada video lalu lintas.

1.4 Manfaat

Penelitian ini memiliki manfaat sebagai berikut :

1. Memberikan kemudahan berbagai pihak pengawas lalu lintas dalam melakukan pengawasan yang dibantu dengan sistem cerdas yang berkemampuan mengklasifikasi kendaraan pada video lalu lintas secara otomatis.
2. Membantu berbagai pihak pengawas lalu lintas melakukan perhitungan kendaraan di jalan raya menggunakan sistem cerdas berdasarkan video lalu lintas.
3. Membantu berbagai pihak pengawas lalu lintas memantau kepadatan kendaraan di jalan raya sehingga dapat mengantisipasi terjadinya kemacetan.

1.5 Batasan masalah

Agar permasalahan yang dirumuskan dapat lebih terfokus, maka penelitian ini dibatasi dalam hal :

1. Data video lalu lintas yang digunakan merupakan data video yang di ubah ke dalam bentuk gambar dengan format ".jpg".
2. Kondisi video berada pada sore hari yang tidak memiliki bayangan objek lain seperti pohon dan lain-lain akibat pantulan dari cahaya matahari.
3. Klasifikasi objek yang ada pada lalu lintas dibedakan menjadi 2 jenis yaitu sepeda motor dan mobil.
4. Metode yang digunakan dalam mengimplementasikan menggunakan metode KMEANS-GMM yang berbasis *Hu Moments*.

1.6 Sistematika pembahasan

Pembuatan tugas akhir ini dilakukan dengan sistematika penulisan sebagai berikut :

1. BAB 1 PENDAHULUAN
   Memuat latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan.
2. BAB 2 TINJAUAN PUSTAKA
   Berisi tentang uraian dasar teori dan referensi yang menunjang dalam penelitian ini. Teori yang terdapat pada bab ini antara lain adalah deteksi kendaraan, *Background Subtraction*, *KMEANS*, *Gaussian Mixture Model* (GMM), ekstraksi fitur, dan klasifikasi kendaraan.
3. BAB 3 METODOLOGI PENELITIAN DAN PERANCANGAN
   Membahas tentang metode, analisis kebutuhan, perancangan, dan langkah-langkah yang dilakukan untuk menyelesaikan penelitian ini. Sub bab yang terdapat dalam bab ini antara lain adalah studi literatur, pengumpulan data, analisa kebutuhan, perancangan sistem, implementasi sistem, pengujian, analisis, dan kesimpulan.
4. BAB 4 IMPLEMENTASI
   Membahas segala hal yang berkaitan dengan implementasi dari metode GMM berbasis *hu moments* pada kasus klasifikasi kendaraan pada video lalu lintas, meliputi *user interface* dan *source code*.
5. BAB 5 PENGUJIAN DAN ANALISIS
   Memuat penjelasan mengenai proses pengujian dan hasil yang akan dilakukan berdasarkan sistem klasifikasi kendaraan pada video lalu lintas yang telah dibangun dengan menggunakan metode GMM berbasis *hu moments*.
6. BAB 6 PENUTUP
   Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak yang dikembangkan dalam skripsi ini serta saran-saran untuk pengembangan lebih lanjut.

BAB 2 TINJAUAN PUSTAKA

Bab ini membahas kajian pustaka dan dasar teori yang digunakan untuk menunjang penulisan skripsi mengenai Implementasi Metode GMM Berbasis *Hu Moments* untuk Klasifikasi Kendaraan pada Video Lalu Lintas. Kajian pustaka membahas penelitian sebelumnya dan pengunaan metode pada kasus yang berbeda. Dasar teori membahas teori penunjang yang berkaitan dengan skripsi ini diantaranya adalah deteksi kendaraan, *Background Subtraction*, *KMEANS*, *Gaussian Mixture Model* (GMM), ekstraksi fitur, dan klasifikasi kendaraan.

2.1 Kajian pustaka

Kajian pustaka pada skripsi ini akan membahas beberapa penelitian sebelumnya yang relevan dengan judul skripsi yang dibuat. Beberapa penelitian tersebut dilakukan untuk melakukan proses deteksi kendaraan di jalan raya. Beberapa metode yang digunakan untuk mendeteksi kendaraan adalah Blob Analysis, PCA, SHIFT, EM, dan GMM, akan tetapi dalam skripsi ini metode yang digunakan adalah GMM.

Terdapat empat penelitian yang digunakan oleh penulis sebagai referensi dalam pengerjaan skripsi yang dilakukan. Penelitian pertama dilakukan oleh Thou-Ho Chen, Yu-Feng Lin, dan Tsong-Yi Chen yang berfokus pada masalah kuantifikasi(perhitungan banyak kendaraan) dan klasifikasi kendaraan dijalan raya. Penelitian kedua dilakukan oleh Saroj K. Meher dan M.N. Murty yang bertujuan untuk meningkatkan akurasi klasifikasi dengan melakukan proses eliminasi bayangan objek kendaraan. Penelitian ketiga dilakukan oleh Yingjie Xia, Xingmin Shi, Guanghua Song, Qiaolei Geng, dan Yuncai Liu yang berfokus pada perhitungan jumlah kendaraan di jalan raya. Penelitian keempat dilakukan oleh H.Asaidi, A. Aarab, dan M.Bellouki yang bertujuan untuk melakukan proses kuantifikasi dan klasifikasi kendaraan di jalan raya. Uraian terhadap keempat penelitian tersebut ditunjukkan pada tabel 2.1.

Penelitian pertama adalah penelitian yang dilakukan oleh Thou-Ho Chen dkk. Penelitian yang berjudul *Intelligent Vehicle Counting Method Based on Blob Analysis in Traffic Surveillance* bertujuan untuk menghitung total dan mengklasifikasi kendaraan pada video pengawasan lalu lintas dengan memanfaatkan hasil pemrosesan citra digital pada data video. Tahapan dalam penelitian ini melalui beberapa tahap diantaranya *vehicle segmentation*, *background updating*, *feature extraction*, *vehicle classification*, *vehicle tracking*, dan *vehicle counting*. *Vehicle segmentation* merupakan proses membedakan objek kendaraan(*foreground*) dengan *background*, pada tahap ini melakukan proses penghapusan *noise* pada *foreground*. Tahap selanjutnya adalah *background updating* yang bertujuan untuk memperkuat atau membuat *background* lebih handal dan sesuai dengan *background* di tiap *frame-frame* video.

Tabel 2.1 Kajian pustaka penelitian pengawasan lalu lintas

No	Tahun/Judul	Objek	Metode	Hasil
1	[2007]*Intelligent Vehicle Counting Method Based on Blob Analysis in Traffic Surveillance*(Chen, 2007).	Perhitungan jumlah kendaraan di jalan raya - Video kendaraan dengan format 320x240 pixels dan 30 frame per detik.	Metode *Blob Analysis* Langkah-langkah : 1. masukkan data video 2. Melakukan proses segmentasi objek bergerak menggunakan *background subtraction* 3. Melakukan *background updating* 4. Melakukan proses ekstraksi fitur dan klasifikasi menggunakan metode *Blob Analysis* 5. Melakukan *vehicle tracking* 6. Melakukan *vehicle-flow analysis* dan *counting*.	Keluaran dari penelitian ini berupa hasil klasifikasi kendaraan berupa *car*, *bike* dan hasil *counting*.

Sumber : Chen (2007) Meher (2014) Xia (2014) Asaidi (2014)

Tabel 2.1 Kajian pustaka penelitian pengawasan lalu lintas (lanjutan)

No	Tahun/Judul	Objek	Metode	Hasil
2	[2013]*Efficient Method of Moving Shadow Detection and Vehicle Classification*(Meher, 2014).	Penigkatan tingkat akurasi metode untuk deteksi kendaraan dijalan raya 12 data set dari database yang berbeda	Metode *Principal Component Analysis* (PCA) Langkah-langlah : 1. Ekstraksi area objek bergrak menggunakan *background subtraction* 2. Segmentasi menggunakan *mean-shift algorithm* 3. Proses analisa menggunakan PCA 4. Penghapusan area bayangan 5. Proses klasifikasi kendaraan menggunakan algoritma SIFT.	Keluaran dari penelitian ini berupa tingkat akurasi dari 10% data yang di uji coba nilai akurasi yang semula 94.12% untuk objek dengan bayangan, meningkat menjadi 97.35% dengan eliminasi bayangan.
3	[2014]*Towards Improving Quality of Video-Based Vehicle Counting Method for Traffic Flow Estimation*(Xia, 2014).	Peningkatan kualitas perhitungan kendaraan di jalan raya : - Data video lalu lintas	Metode EM-GMM Langkah-langkah : 1. Melakukan *Image Acquisition* 2. Melakukan *Background Modeling* 3. Melakukan ekstraksi kendaraan	Keluaran dari penelitian ini berupa peningka-tan nilai akurasi dari perhitungan kendaraan yang mencapai 98%.

Sumber : Chen (2007) Meher (2014) Xia (2014) Asaidi (2014)

Tabel 2.1 Kajian pustaka penelitian pengawasan lalu lintas (lanjutan)

No	Tahun/Judul	Objek	Metode	Hasil
3			3. Melakukan *Vehicle Restoration* 4. Melakukan *Occlusion Estimation* 5. Melakukan *Vehicle Counting*.	
4	[2014]*Shadow Elimination and Vehicles Classification Approaches in Traffic Video Surveillance Context*(Asaidi, 2014).	Klasifikasi kendaraan pada video kendaraan lalu lintas : - Data video dari kamera statis dari beberapa jalan raya diantaranya, *Taiwan's highways*, *Italy highway*, *highway* 1 dan *France highway* dengan masing nilai frame 2000, 3339, 439, 2792, dan 5380 frame.	Metode GMM berbasis eliminasi bayangan dan *Hu Moments* Langkah-langkah : 1. Melakukan *vehicle detection* menggunakan *background subtraction* 2. Melakukan *shadow elimination* 3. Melakukan *features extraction* 4. Melakukan vehicles classification.	Keluaran dari penelitian ini hasil klasifikasi kendaraan dengan kategori *car*, *van*, dan *truck* dengan hasil rata-rata akurasi sebesar 96.96%.

Sumber : Chen (2007) Meher (2014) Xia (2014) Asaidi (2014)

Tahap selanjutnya adalah *feature extraction*, pada tahap ini digunakan untuk membedakan objek yang berbeda pada saat yang sama dan mengenali objek yang sama pada waktu yang berbeda. Tahap selanjutnya adalah *vehicle classification*, pada tahap ini mengambil dari hasil proses *feature extraction* kemudian dibedakan objek menjadi 2 yaitu *car* dan *bike*. Tahap selanjutnya adalah *vehicle tracking*, pada tahap ini dua objek yang spasial nya berdekatan dalam frame saling terhubung. Dan pada tahap ini *euclidian distance* digunakan untuk mengukur jarak centroid objek-objek

tersebut. Tahap terakhir adalah *vehicle counting*, pada tahap ini dipasang 2 *base line* yang digunakan untuk merekam frame dengan objek kendaraan yang melewati *base line* tersebut. Sehingga didapatkan hasil *counting* untuk objek kendaraan berdasarkan data video (Chen, 2007).

Penelitian kedua adalah penelitian yang dilakukan oleh Saroj K.Meher dkk. Penelitian yang berjudul *Efficient Method of Moving Shadow Detection and Vehicle Classification* bertujuan untuk mendeteksi bayangan objek bergerak dan menghapus bayangan tersebut dari area foreground yang sudah diekstraksi dari frame video agar meningkatkan tingkat akurasi deteksi dan klasifikasi benda bergerak. Tahapan dalam penelitian ini melalui beberapa tahap yaitu *shadow region detection*, *region segmentation*, *searching direction of shadow*, *shadow region separation*, *vehicle classification*. *shadow region detection* merupakan proses untuk mendeteksi area bayangan dari objek dengan menggunakan proses *background substraction*. Tahap selanjutnya adalah *region segmentation*, pada tahap ini hasil ekstraksi frame berisi area *foreground* yang disegmentasi menggunakan algoritma *mean-shift* (MS). Pada dasarnya algoritma *mean-shift* (MS) merupakan teknik untuk menemukan kemungkinan pusat kluster berdasarkan densitas gradient proses segmentasi. Tahap selanjutnya *searching direction of shadow*, tahap ini bertujuan untuk meminimalkan ruang pencarian untuk daerah bayangan menggunakan pendekatan *principal component analysis* (PCA). Tahap selanjutnya adalah *shadow region separation*, pada tahap ini pencarian area *homogeneous* (sejenis) terbuat dari dua arah yang ditentukan melalui prosedur pencarian arah. Kemudian area *homogeneous* yang terdeteksi dipisahkan dari area *foreground*. Tahap terakhir adalah *vehicle classification*, pada tahap ini proses klasifikasi kendaraan di jalan raya dilakukan dengan menggunakan algoritma SIFT berdasarkan kecocokan fitur yang sudah diekstraksi (Meher, 2014).

Penelitian ketiga adalah penelitian yang dilakukan oleh Yingjie Xia dkk. Penelitian yang berjudul *Toward Improving Quality of Video-Based Vehicle Counting Method for Traffic Flow Estimation* bertujuan untuk meng-estimasi arus lalu lintas dengan cara melakukan *counting* terhadap kendaraan di jalan raya. Tahapan dalam penelitian ini melalui beberapa tahap yaitu *configuration of virtual loops*, *vehicle extraction*, dan *vehicle tracking*. *configuration of virtual loops* merupakan tahap memasang wilayah area deteksi pada bagian bawah dari video yang bertujuan untuk mengeliminasi kesalahan deteksi akibat penumpukan kendaraan. Tahap selanjutnya adalah *vehicle extraction*, tahap ini menggunakan pendekatan *background subtraction* dan di sempurnakan dengan metode GMM agar memperoleh hasil ekstraksi fitur yang sempurna. Tahap selanjutnya adalah *vehicle tracking*, pada tahap ini penyelesaian masalah deteksi kendaraan saat ada penumpukan kendaraan (Xia, 2014).

Penelitian keempat adalah penelitian yang dilakukan oleh H. Asaidi dkk. Penelitian yang berjudul *Shadow Elimination and Vehicles Classification Approaches in Traffic Video Surveillance Context* bertujuan untuk mendeteksi kendaraan dengan

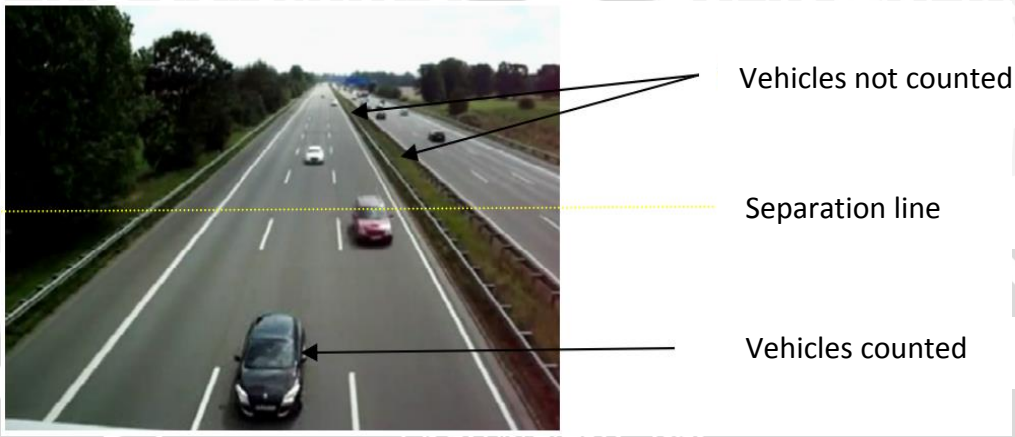
menghilangkan unsur bayangan dari kendaraan tersebut dan melakukan klasifikasi berdasarkan hasil pengolahan citra digital yang dilakukan. Tahapan dalam penelitian ini melalui beberapa tahap yaitu *detection*, *background subtraction*, *vehicle tracking*, *occlusion elimination*, *features extraction* dan *vehicle classification*. *Detection* merupakan tahap yang terdiri dari ekstraksi objek bergerak dari data video. Tahap selanjutnya adalah *background subtraction*, *background subtraction* merupakan teknik untuk menghapus komponen yang tidak bergerak dari frame-frame video dan GMM merupakan metode yang popular yang digunakan dalam memonitor lalu lintas. Tahap selanjutnya adalah *vehicle tracking*, pada tahap ini terdiri dari proses mengekstrak informasi sementara tentang kendaraan seperti lintasan, postur, kecepatan, dan arah yang bertujuan untuk membangun korespondensi antara kendaraan dalam frame yang berurutan. Tahap selanjutnya adalah *occlusion elimination*, pada tahap ini terdapat proses pendeteksian dan penseleksian objek kendaraan yang saling tumpang tindih. Tahap selanjutnya adalah *features extraction*, pada tahap ini terdapat proses ekstrak fitur kendaraan seperti bentuk, panjang, lebar, tinggi, dll yang bertujuan untuk mengklasifikasi dan menentukan parameter lalu lintas. Tahap selanjutnya adalah *vehicle classification*, pada tahap ini hasil dari ekstraksi ciri akan digunakan untuk mengklasifikasikan objek kendaraan berdasarkan jenis nya (Asaidi, 2014).

Berdasarkan penelitian yang telah ada, maka penulis akan melakukan sebuah penelitian yang berkaitan dengan deteksi kendaraan berdasarkan data video lalu lintas menggunakan metode KMEANS-GMM. Pada penelitian ini objek yang akan dideteksi adalah objek kendaraan, objek yang dideteksi sama dengan penelitian ke-1, ke-2, ke-3, dan ke-4. Objek yang telah dideteksi akan di klasifikasikan menjadi 2 jenis kendaraan yaitu mobil dan sepeda motor. Pada penelitian ini menggunakan metode KMEANS-GMM berbasis *hu moments* seperti yang dilakukan pada penelitian ke-4. Dasar teori yang diperlukan untuk penelitian ini adalah deteksi kendaraan, *Background Subtraction*, *KMEANS*, *Gaussian Mixture Model* (GMM), ekstraksi fitur, dan klasifikasi kendaraan.

2.2 Deteksi kendaraan

Deteksi kendaraan merupakan proses ekstraksi objek kendaraan bergerak pada *video sequence*. Ada beberapa metode yang dapat digunakan untuk mendeteksi objek kendaraan pada *video sequence* diantaranya yaitu, *thresholding*, *multi-resolution processing*, *edge detection*, *background subtraction*, dan *inter-frame differencing*. Metode yang paling sering digunakan dalam mendeteksi objek kendaraan adalah *background subtraction*. Metode *background subtraction* memiliki akurasi yang cukup baik untuk aplikasi monitor lalu lintas seperti *human motion capture* dan *video surveillance*. Deteksi kendaraan dapat dilakukan berdasarkan video dari kamera statis yang di pasang dijalan raya, kemudian frame frame video yang di dapat akan di proses menggunakan *background subtraction* untuk memperoleh objek kendaraan. Gambar

2.1 menunjukkan proses deteksi kendaraan di jalan raya pada sebuah citra digital menggunakan garis pemisah untuk melakukan proses perhitungan kendaraan, ketika kendaraan melewati garis pemisah, hal itu menandakan kendaraan sudah melalui proses perhitungan (Asaidi, 2014).

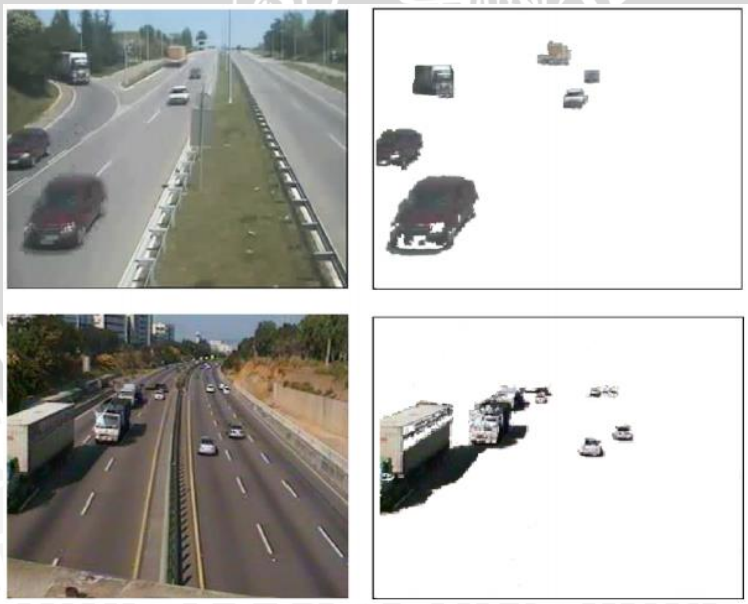


Gambar 2.1 Contoh deteksi kendaraan

Sumber : Asaidi (2014)

2.3 *Background subtraction*

Background subtraction merupakan teknik untuk menghapus komponen objek tidak bergerak dalam sebuah frame-frame video dan mengestimasi perubahan antara gambar *background* dan gambar *foreground*.



Gambar 2.2 Contoh background subtraction

Sumber : Asaidi (2014)

Asumsi dasar untuk mengimplementasikan *background subtraction* adalah frame-frame video didapat dari kamera yang tidak bergerak(statis). Ada beberapa permasalah yang dialami dalam metode *background subtraction* yaitu perubahan pencahayaan, perubahan geometri *background*, dan gerakan berulang dari objek kontektual seperti daun atau pohon. Untuk menangani masalah tersebut maka diperlukan sebuah pendekatan yang dapat beradaptasi terhadap perubahan pencahayaan, geometri *background*, dan objek repetitive. Salah satu cara yang dapat dilakukan untuk beradaptasi dengan perubahan adalah melakukan pembelajaran terhadap perubahan model *background* kemudian menerapkan perubahan pada model *background* yang baru. Gambar 2.2 menunjukkan hasil dari proses *background subtraction*, pada bagian kiri gambar terlihat masih ada jalan raya beserta objek kendaraan, sedangkan pada bagian kanan gambar terlihat jalan raya sudah mengalami proses *background subtraction* sehingga hanya terlihat objek kendaraan yang ada pada gambar (Asaidi, 2014).

2.4 KMEANS

KMEANS merupakan salah satu metode data clustering non hirarki yang berusaha mempartisi data yang ada ke dalam bentuk satu atau lebih kelompok(*cluster*). Metode KMEANS mempartisi data ke dalam kelompok(*cluster*), sehingga data yang memiliki karakteristik yang sama akan dikelompokkan ke dalam satu kelompok(*cluster*) yang sama dan data yang mempunyai karakteristik berbeda akan dikelompokkan ke dalam kelompok(*cluster*) lain. Tujuan dari mengelompokkan data adalah untuk meminimalkan *objective function* yang diset dalam proses *clustering*, yang pada umumnya berusaha meminimalkan variasi dalam *cluster* dan memaksimalkan variasi antar cluster (Agusta, 2007).

Tahapan metode KMEANS secara umum dilakukan dengan algoritma dasar sebagai berikut (Ong, 2013):

1. Pilih jumlah *cluster k*.
2. Inisialisasi *k* pusat *cluster* ini bisa dilakukan dengan berbagai cara. Namun yang paling sering dilakukan adalah dengan cara random. Pusat-pusat *cluster* diberi nilai awal dengan angka-angka random.
3. Alokasi semua data / objek ke *cluster* terdekat. Kedekatan dua objek ditentukan berdasarkan jarak kedua objek tersebut. Demikian juga kedekatan suatu data ke *cluster* tertentu ditentukan jarak antara data dengan pusat *cluster*. Dalam tahap ini perlu dihitung jarak tiap data ke tiap pusat *cluster*. Jarak paling dekat antara satu data dengan satu *cluster* tertentu akan menentukan suatu data masuk dalam *cluster*.
4. Hitung kembali pusat *cluster* dengan keanggotaan *cluster* yang baru. Pusat *cluster* adalah rata-rata dari semua data / objek dalam *cluster* tertentu. Jika

dikehendaki bisa juga menggunakan nilai tengah (*median*) dari *cluster* tersebut. Jadi rata-rata (*mean*) bukan satu-satunya ukuran yang bisa dipakai.

5. Tugaskan lagi setiap objek memakai pusat *cluster* yang baru. Jika pusat *cluster* tidak berubah lagi maka proses *clustering* selesai. Atau kembali ke langkah nomor 3 sampai pusat *cluster* tidak mengalami perubahan.

Ada beberapa alternatif penerapan KMEANS dengan beberapa pengembangan teori-teori perhitungan terkait yang telah diusulkan (Agusta, 2007) :

1. *Distance space* untuk menghitung jarak di antara suatu data dan *centroid*
2. Metode pengalokasian data kembali ke dalam setiap *cluster*
3. *Objective function* yang digunakan.

Pada sub bab 2.4.1 akan menjabarkan tentang *distance space* untuk menghitung jarak di antara suatu data dan *centroid*.

2.4.1 *Distance space* menghitung jarak antara data dan centroid

Beberapa *distance space* telah diimplementasikan dalam menghitung jarak (*distance*) antara data dengan *centroid*, beberapa contoh implementasi menghitung jarak kedekatan dengan *centroid* adalah L_1 (*Manhattan*/*City Block*) *distance space*, L_2 (*Euclidean*) *distance space*, dan L_p (*Minkowski*) *distance space*. Jarak antara dua titik x_1 dan x_2 pada *Manhattan*/*City Block distance space* dihitung dengan menggunakan persamaan (2.1). Sedangkan untuk L_2 (*Euclidean*) *distance space*, jarak antara dua titik dihitung dengan menggunakan persamaan (2.2) (Agusta, 2007).

$$D_{L_1}(x_2, x_1) = |x_2 - x_1|_1 = \sum_{j=1}^{p} |x_{2j} - x_{1j}| \quad (2.1)$$

$$D_{L_2}(x_2, x_1) = |x_2 - x_1|_2 = \sqrt{\sum_{j=1}^{p} (x_{2j} - x_{1j})^2} \quad (2.2)$$

Keterangan persamaan (2.1) dan (2.2) :

- P : Dimensi data
- | . | : Nilai absolut

Untuk L_p (*Minkowski*) *distance space* merupakan generalisasi dari beberapa distance space yang ada seperti L_1 (*Manhattan*/*City Block*) *distance space* dan L_2 (*Euclidean*) *distance space* juga telah diimplementasikan. Tapi secara umum *distance space* yang sering digunakan adalah *Manhattan* dan *Euclidean*. *Euclidean* sering digunakan karena perhitungan jarak dalam *distance space* ini merupakan jarak terpendek yang bisa didapatkan antara dua titik yang diperhitungkan, sedangkan *Manhattan* sering digunakan karena kemampuannya dalam mendeteksi keadaan khusus seperti keberadaan *ouliers* dengan lebih baik (Agusta, 2007).

2.5 *Gaussian Mixture Model* (GMM)

Pada umumya model *background* dibuat dan diperbarui dengan menggunakan pemodelan berdasarkan *singular modal distribution*, seperti *singular Gaussian distribution* yang tidak dapat memberikan pendekatan yang lebih baik terhadap data dengan distribusi multi-modal. GMM adalah descriptor untuk distribusi multi-modal, model GMM terdiri dari komponen K mengikuti distribusi Gaussian tunggal (Xia, 2014). GMM sering digunakan untuk model *background* yang komplek dan diterapkan pada permasalahan pengawasan lalu lintas. GMM adalah salah satu algoritma dasar untuk pemodelan latar belakang karena efektifitas dan efisiensi untuk mendeteksi gerakan dari video sekuens. Algoritma GMM mempertimbangkan setiap pixel dari setiap frame video yang diproses. Apabila pixel dari frame video disajikan seperti berikut {X1, X2, …, Xt}, maka distribusi Gaussian akan ditampilkan seperti persamaan (2.3) dan (2.4) (Asaidi, 2014).

$$P(X_t) = \sum_{i=1}^{k} \omega_{i,t}\eta(X_t, \mu_{i,t}\Sigma_{i,t}) \quad (2.3)$$

$$\eta(X_t, \mu_{i,t}\Sigma_{i,t}) = \frac{1}{(2\pi)^{n/2}|\Sigma_{i,t}|^{1/2}} \exp\left(\left(-\frac{1}{2}\right)(X_t - \mu_t)^T\left(\Sigma_{i,t}\right)^{-1}(X_t - \mu_t)\right) \quad (2.4)$$

Keterangan persamaan (2.3) dan (2.4) :

- X_t : nilai data pixel pada waktu ke t
- k : Total dari banyaknya distribusi
- ω : Nilai estimasi *prior weights*
- t : *Time*(waktu)
- μ : Nilai rata-rata
- Σ : Matrik kovarian
- η : *Gaussian probability density function*
- n : Dimensi data(pada kasus ini saya menggunakan 1 dimensi data).

Dimensi data pada GMM terdiri dari 1 dimensi(*gray*), 2 dimensi (warna yang sudah dinormalisasi atau *intensity-plus-range*), 3 dimensi(nilai warna *red*,*green*,*blue*), dan n dimensi(bergantung pada total n kolom vektor data yang digunakan)

Untuk mendapatkan nilai kovarian(Σ) maka digunakan persamaan (2.5) (Asaidi, 2014).

$$\Sigma_{i,t} = \sigma_{i,t}^2 \mathrm{I} \quad (2.5)$$

Keterangan persamaan (2.5) :

- Σ : Nilai kovarian
- σ^2 : Nilai *variance* data
- ɪ : Matrik identitas
- i : Nilai cluster
- t : *Time*(waktu)

Pada algoritma GMM setelah nilai dari peluang P(Xt) didapatkan, maka selanjutnya akan dilakukan proses *update equation* dan *classification*. *Update equation* dilakukan untuk mencari nilai pixel yang match pada setiap distribusi, sedangkan *classification* dilakukan untuk mengelompokan antara *foreground* dan *background*. Sub bab 2.4.1 dan 2.4.2 akan menjabarkan tentang *update equation* dan *classification*.

2.5.1 *Update equation*

Update equation merupakan prosedur untuk mendeteksi *pixel foreground* berdasarkan nilai kecocokan pada distribusi *Gaussian*. Proses untuk *update equation* dimulai saat sistem dijalankan, distribusi *k Gaussian* untuk setiap *pixel* diinisialisasi dengan nilai rata-rata yang di definisikan sebelumnya, nilai *variance* yang tinggi, dan nilai *prior weights* yang rendah. Saat *pixel* yang baru diobservasi dalam objek urutan gambar, nilai vektor warna dari pixel tersebut di cek menggunakan distribusi *k Gaussian* sampai di temukan nilai yang cocok. Kecocokan nilai didefinisikan pada saat nilai pixel berada dalam lamda(λ)=2.5 standard deviasi dari distribusi. Nilai *prior weights* akan di *update* sesuai dengan ke cocokan pixel dalam λ. Persamaan (2.6) digunakan untuk melakukan *update* nilai *prior weights*. Untuk parameter rata-rata(μ) dan variance(σ^2) akan di update untuk distribusi yang tidak mengalami kecocokan pada nilai pixel nya. Persamaan (2.7), (2.8), dan (2.9) digunakan untuk melakukan *update* parameter rata-rata(μ) dan variance(σ^2) (Asaidi, 2014).

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

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

$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t-\mu)^T(X_t-\mu) \quad (2.8)$$

$$\rho = \alpha\eta(X_t|\mu_k,\sigma_k) \quad (2.9)$$

Keterangan persamaan (2.6), (2.7), (2.8), dan (2.9) :

- α : *Learning rate*
- $M_{k,t}$: Bernilai 1 untuk model yang cocok(*match*) dan 0 untuk lainnya.

Persamaan (2.10) dan (2.11) digunakan untuk menghitung nilai λ.

$$\lambda = 2.5 * std \quad (2.10)$$

$$std = \sqrt{\sigma^2} \quad (2.11)$$

Keterangan persamaan (2.10) dan (2.11) :

- std : Standar deviasi.

2.5.2 *Classification*

Classification merupakan prosedur untuk mengelompokkan *pixel* baru yang di cek berdasarkan distribusi *k Gaussian*. *Pixel* baru yang di proses akan dikelompokkan kedalam kategori *background* atau *foreground* berdasarkan nilai ω/σ yang sudah di urutkan. Nilai ω/σ yang sudah di urutkan berfungsi untuk mendapatkan nilai peluang terbesar dalam proses pengelompokan *background* dan *foreground*. Untuk *pixel background* yang sudah diproses akan berkorespondensi dengan distribusi *Gaussian* dan menghasilkan nilai *prior weights* yang besar dan nilai *variance* yang kecil. Persamaan (2.12) digunakan untuk mengelompokan *pixel background*. Nilai distribusi *B* pertama akan didefinisikan sebagai *background* (Asaidi, 2014).

$$\mathrm{B} = argmin_b\left(\sum_{k=1}^{b} \omega_k > T\right) \quad (2.12)$$

Keterangan persamaan (2.12) :

- T : *Threshold*
- b : total data

2.6 Ekstraksi fitur

Mengestimasi parameter jalan raya seperti volume, kepadatan, rata-rata kecepatan dan arus lalu lintas ini sangat penting untuk manajemen dan analisa lalu lintas. Banyak pendekatan yang berbeda yang dapat di lakukan untuk mengestimasi parameter jalan raya yang berbasis *image processing*, salah satu contohnya adalah ekstraksi fitur. Ekstraksi fitur adalah proses ekstraksi ciri-ciri dari objek di jalan raya seperti bentuk, panjang, lebar, tinggi, dan tekstur yang kemudian digunakan untuk mengelompokkan kendaraan ke dalam beberapa kategori. Objek yang didapat saat melakukan ekstraksi fitur akan memiliki perbedaan kenampakan berdasarkan 3 parameter yaitu jarak dari kamera, eksentrisitas dari sumbu, dan bidang miring dalam tiga arah. Oleh karena itu untuk meminimalkan perbedaan kenampakan objek maka proses kalibrasi kamera untuk normalisasi fitur harus diterapkan dahulu di awal (Asaidi, 2014).

Ada banyak teknik yang dikembangkan untuk menggambarkan kendaraan di jalan raya dengan memanfaatkan dimensi dari kendaraan seperti panjang, lebar, dan tinggi. Namun tidak menutup kemungkinan ada kesalahan dalam perhitungan dimensi kendaraan, karena dimensi kendaraan dapat berubah-ubah sesuai letak kendaraan di jalan raya. Oleh karena itu metode *seven invariants moment* digunakan untuk mengekstrak fitur dimensi kendaraan di jalan raya. Metode *sevent invariants moment* dapat menghilangkan efek homothetic dan redundansi pada saat proses perhitungan. *Hu moments* merupakan metode yang pertama kali digunakan untuk masalah pengenalan dalam *computer vision* dengan menggunakan metode *sevent invariants moment*. *Hu* berasal dari kombinasi relatif dan absolut nilai *moment invariant* dengan skala, posisi, dan orientasi berdasarkan teori aljabar *invariant* yang berhubungan

dengan ekspresi aljabar dan tetap berubah dalam transformasi linear pada umumnya. Metode ekstraksi fitur *seven invariants moment* dapat dihitung dengan menggunakan persamaan (2.13), (2.14), (2.15), (2.16), (2.17), (2.18), (2.19), (2.20), (2.21), dan (2.22) (Asaidi, 2014).

$$\phi_1 = \mu_{2,0} + \mu_{0,2} \tag{2.13}$$

$$\phi_2 = (\mu_{2,0} - \mu_{0,2})^2 + 4\mu_{1,1}^2 \tag{2.14}$$

$$\phi_3 = (\mu_{3,0} - 3\mu_{1,2})^2 + (3\mu_{2,1} - \mu_{0,3})^2 \tag{2.15}$$

$$\phi_4 = (\mu_{3,0} + \mu_{1,2})^2 + (\mu_{2,1} + \mu_{0,3})^2 \tag{2.16}$$

$$\phi_5 = (\mu_{3,0} - 3\mu_{1,2})(\mu_{3,0} + \mu_{1,2})\left[(\mu_{3,0} + \mu_{1,2})^2 - 3(\mu_{2,1} + \mu_{0,3})^2\right]$$

$$\phi_5 = +(3\mu_{2,1} - \mu_{0,3})(\mu_{2,1} + \mu_{0,3})\left[3(\mu_{3,0} + \mu_{1,2})^2 - (\mu_{2,1} + \mu_{0,3})^2\right] \tag{2.17}$$

$$\phi_6 = (\mu_{2,0} - \mu_{0,2})\left[(\mu_{3,0} + \mu_{1,2})^2 - (\mu_{2,1} + \mu_{0,3})^2\right]$$

$$\phi_6 = +4\mu_{1,1}(\mu_{3,0} + \mu_{1,2})(\mu_{2,1} + \mu_{0,3}) \tag{2.18}$$

$$\phi_7 = (3\mu_{2,1} - \mu_{0,3})(\mu_{3,0} + \mu_{1,2})\left[(\mu_{3,0} + \mu_{1,2})^2 - 3(\mu_{2,1} + \mu_{0,3})^2\right]$$

$$\phi_7 = -(\mu_{3,0} - 3\mu_{1,2})(\mu_{2,1} + \mu_{0,3})\left[3(\mu_{3,0} + \mu_{1,2})^2 - (\mu_{2,1} + \mu_{0,3})^2\right] \tag{2.19}$$

$$\mu_{pq} = \sum_{(x,y)\in V}(x - x_0)^p\,(y - y_0)^q f(x,y) \tag{2.20}$$

$$x_0 = \frac{m_{10}}{m_{00}}, y_0 = \frac{m_{01}}{m_{00}} \tag{2.21}$$

$$m_{pq} = \sum_{(x,y)\in V} x^p y^q f(x,y) \tag{2.22}$$

Keterangan persamaan (2.13), (2.14), (2.15), (2.16), (2.17), (2.18), (2.19), (2.20), (2.21), dan (2.22) :

- V : area kendaraan
- F(x,y) : intensitas *pixel* pada koordinat x dan y.

2.7 Klasifikasi kendaraan

Klasifikasi kendaraan adalah proses pengelompokkan kendaraan kedalam beberapa kategori seperti *truck*, *bus*, *car*, *bike*, dan lain-lain. Metode klasifiksai dalam kontek deteksi kendaraan berdasarkan kenampakan citra dapat mempelajari dan menentukan keputusan berdasarkan batas antara dua kelas. Dalam pengolahan citra digital untuk melakukan klasifikasi kendaraan maka diperlukan proses *image preprocessing* seperti ekstraksi fitur kemudian hasil nya di integrasikan dengan berbagai metode seperti *neural networks*, *bayessian classifier*, *ecludian distance*, dan lain-lain, persamaan (2.21) menunjukkan persamaan untuk melakukan klasifikasi kendaraan menggunakan 7-D *Euclidean features space* (Asaidi, 2014). Salah satu

contoh metode untuk klasifikasi pada objek citra digital adalah *neural networks*, pada umumnya metode *neural networks* memiliki banyak parameter untuk penyesuaian dan hasil training cenderung menyatu ke local optimum. Para peneliti mulai beranjak beralih menuju pengklasifikasian yang proses training datanya cenderung menuju ke global optimum selama proses training data, seperti *SVMs* and *AdaBoost* (Wen, 2014). Pada dasarnya metode-metode yang digunakan untuk klasifikasi kendaraan diperoleh dari analisa gambar video kemudian di ekstrak untuk mendapatkan informasi objek kendaraan yang akan di kelompokkan (Wang, 2011).

$$d(v,c) = \sqrt{\sum_{i=1}^{7}(v_i - c_i)^2} \quad (2.21)$$

Keterangan persamaan (2.21) :

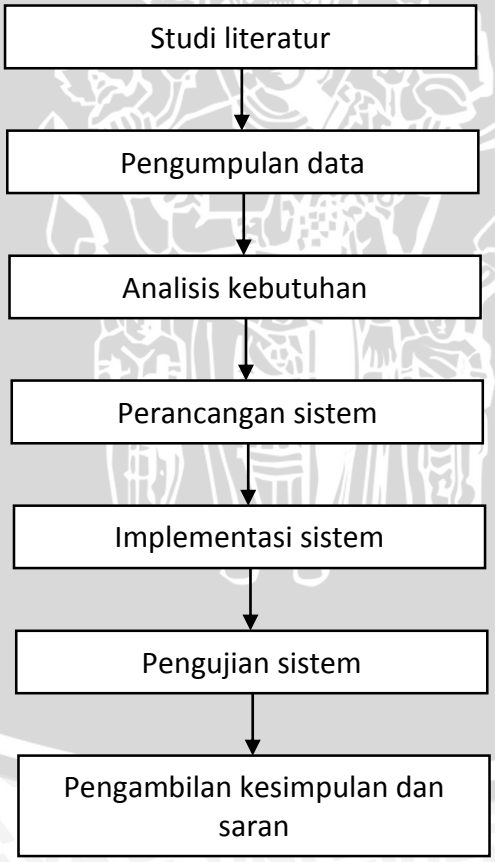
- v : target kendaraan
- c : kendaraan pada data training.

BAB 3 METODOLOGI PENELITIAN DAN PERANCANGAN

Bab metodologi penelitian dan perancangan menjelaskan langkah-langkah yang ditempuh selama penyusunan skripsi dan proses pembuatan rancangan sistem. Langkah-langkah metodologi penelitian yang ditempuh yaitu studi literatur, pengumpulan data, analisa kebutuhan, perancangan sistem, implementasi sistem, pengujian sistem, dan pengambilan kesimpulan. Sedangkan perancangan berisi tentang diagram alir sistem dan perhitungan manual.

3.1 Metodologi penelitian

Sub bab metodologi penelitian menjelaskan langkah-langkah yang ditempuh selama penyusunan skripsi. Langkah-langkah metodologi penelitian yang ditempuh yaitu studi literatur, pengumpulan data, analisa kebutuhan, perancangan sistem, implementasi sistem, pengujian sistem, dan pengambilan kesimpulan. Gambar 3.1 menjelaskan tentang langkah-langkah penelitian yang dilakukan untuk membuat sistem pengawasan lalu lintas menggunakan metode GMM.



Gambar 3.1 Diagram alir metodologi penelitian

3.1.1 Studi literatur

Studi literatur dilakukan dengan cara mengumpulkan dan mempelajari sumber (literatur-literatur) yang berkaitan dengan deteksi kendaraan pada video lalu lintas, metode KMEANS, GMM, ekstaksi fitur berbasis *Hu moments*, klasifikasi kendaraan menggunakan *Euclidean distance space*, pengolahan citra digital, pemrograman dengan menggunakan Bahasa *java*, dan proses pengujian sistem. Sumber literatur dapat berupa buku cetak, buku elektronik (*e-book*), paper, jurnal, karya ilmiah, dan penjelasan dari dosen pembimbing skripsi.

3.1.2 Pengumpulan data

Pengumpulan data dilakukan dengan cara melakukan *survey* ke jalan raya wilayah kota Malang. Data yang di dapatkan dari survey berupa data video lalu lintas kendaraan yang di rekam pada sebuah kamera. Data video diambil dari kamera statis yang sudah di atur sedemikian rupa agar dapat merekam aktifitas lalu lintas kendaraan di jalan raya. Kemudian data video kendaraan akan di ekstrak ke dalam bentuk format frame-frame ".jpg" dan akan di proses ke dalam sistem deteksi kendaraan menggunakan metode GMM.

3.1.3 Analisis kebutuhan

Analisis kebutuhan bertujuan untuk menentukan kebutuhan yang diperlukan dalam mengimplementasikan sistem pendeteksi kendaraan. Analisis kebutuhan diterapkan sesuai dengan lokasi penelitian, variabel penelitian, dan mempersiapkan kebutuhan penelitian. Berikut ini keseluruhan kebutuhan yang digunakan dalam mengimplementasikan sistem pendeteksi kendaraan :

1. Kebutuhan *hardware* :
   - Laptop dengan *processor* Intel(R) Pentium(R) CPU B940 @2.00GHz
   - Laptop dengan 2.00GB RAM
2. Kebutuhan *software* :
   - Sistem operasi Windows 8.1 Pro
   - Netbeans IDE 8.0.1
   - *Java Development Kit* (JDK) 8
3. Data yang dibutuhkan:
   - Data video lalu lintas kendaraan

3.1.4 Perancangan sistem

Perancangan sistem merupakan tahapan yang menjelaskan desain dari sistem secara keseluruhan, baik dari segi model maupun arsitektur yang akan digunakan.

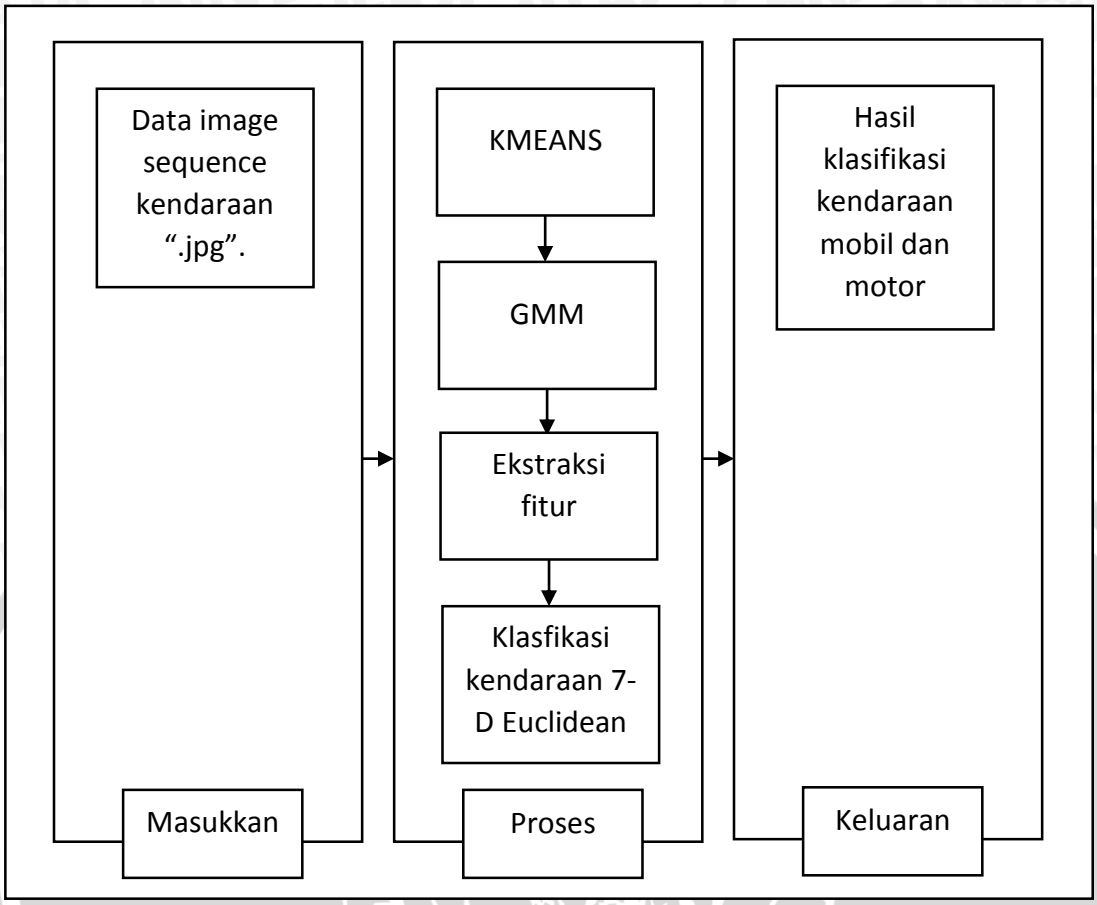
Perancangan sistem dibuat berdasarkan hasil yang telah diperoleh dalam tahap pengumpulan data dan analisis kebutuhan. Perancangan sistem dilakukan agar proses pengimplementasian sistem menjadi lebih mudah. Sub bab 3.4.1 dan 3.4.2 akan menjabarkan tentang diagram blok sistem dan arsitektur sistem pendeteksi kendaraan.

3.1.4.1 Diagram blok sistem

Diagram blok sistem menjelaskan cara kerja sistem yang akan dibuat mulai dari masukkan sampai keluaran yang dihasilkan sistem. Secara umum sistem yang akan dibangun menggunakan algoritma GMM yang bertujuan untuk melakukan proses *background subtraction* pada *image sequence* dengan format ".jpg" yang dimasukkan. *Background subtraction* berarti memisahkan objek kendaraan(*foreground*) dengan jalan raya(*background*). Setelah proses *background subtraction* selesai dilakukan maka proses ekstraksi fitur berbasis *Hu moments* dilakukan untuk mendapatkan nilai dari 7 fitur kendaraan. Berdasarkan 7 nilai fitur kendaraan, maka klasifikasi kendaraan menggunakan 7-D Euclidean dilakukan untuk mendapatkan hasil klasifikasi kendaraan menjadi 2 objek. 2 objek klasifikasi tersebut adalah objek kendaraan mobil dan motor. Gambar 3.2 menggambarkan diagram blok secara umum dari sistem yang akan dibangun.

3.1.4.2 Arsitektur sistem

Arsitektur sistem merupakan gambaran mengenai sistem yang akan di buat yang berisi tentang hubungan sistem dengan pengguna. Pada sistem pendeteksi kendaraan pengguna akan menjalankan sistem dan memantau kinerja dari sistem. Data masukkan sistem deteksi kendaraan berasal dari luar, kemudian diambil ke dalam sistem untuk di proses. Sistem akan mengeksekusi data masukkan menggunakan algoritma GMM kemudian sistem akan memberikan keluaran berupa hasil klasifikasi kendaraan. Gambar 3.3 menjabarkan tentang arsitektur sistem pendeteksi kendaraan menggunakan algoritma GMM.

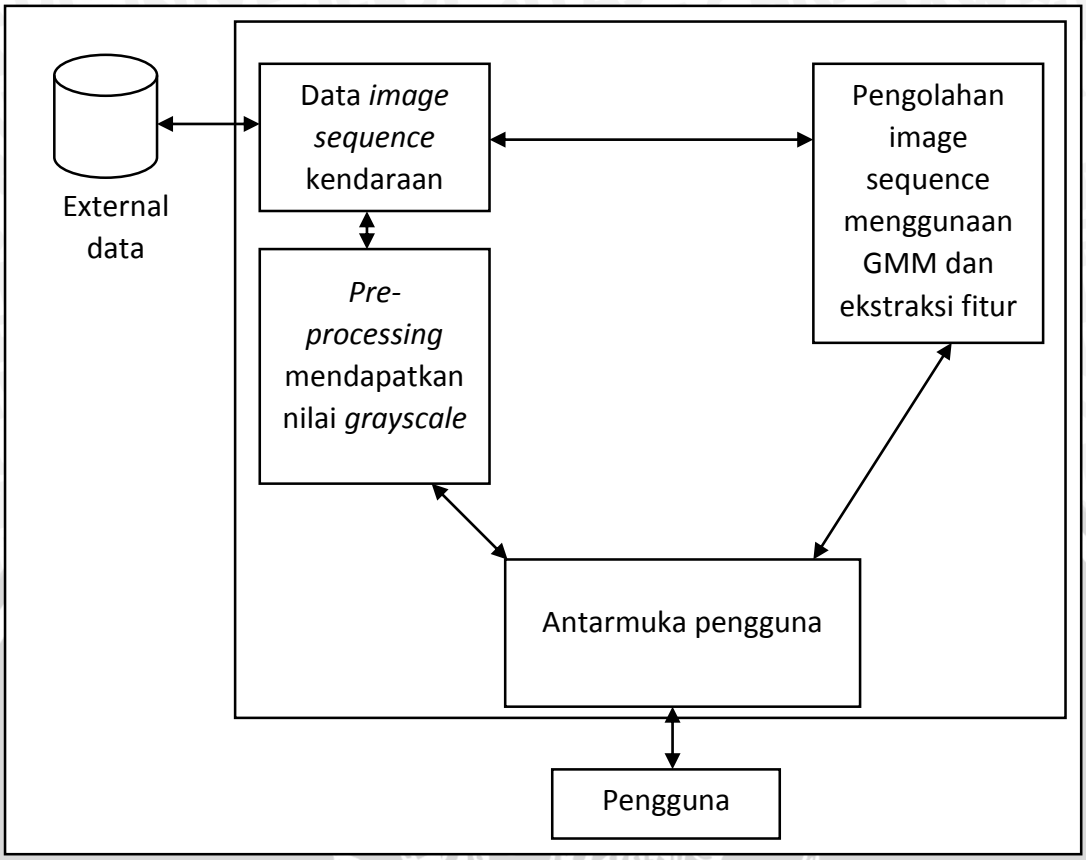


Gambar 3.2 Diagram blok deteksi kendaraan menggunakan algoritma GMM

3.1.5 Implementasi sistem

Implementasi dalam penelitian ini dilakukan dengan mengacu kepada perancangan sistem. implementasi sisem dilakukan dengan menggunakan Bahasa pemrograman *java*, *Java Development Kit* (JDK) 8, dan *tools* pendukung lainnya. Masukan data dari sistem ini adalah image sequence format ".jpg" yang diproses menggunakan algoritma GMM. Keluaran untuk penelitian ini adalah hasil klasifikasi kendaraan menjadi 2 objek yaitu mobil dan motor. Tahapan – tahapan yang ada dalam implementasi meliputi :

1. Pembuatan antarmuka.
2. Perhitungan metode KMEANS.
3. Perhitungan metode GMM.
4. Perhitungan ekstraksi fitur *Hu moments*.
5. Perhitungan klasifikasi 7-D *Euclidean*.
6. Keluaran berupa hasil klasifikasi kendaraan mobil dan motor.



Gambar 3.3 Arsitektur sistem pendeteksi kendaraan algoritma GMM

3.1.6 Pengujian sistem

Pengujian dalam penelitian ini dilakukan agar dapat membuktikan bahwa sistem yang dibangun telah mampu bekerja sesuai dengan spesifikasi dan kebutuhan yang sudah direncanakan. Selain itu juga akan dicari pengujian mana yang menghasilkan tingkat akurasi paling baik. Pengujian sistem yang dilakukan meliputi pengujian terhadap parameter pendeteksi kendaraan seperti *threshold* dan *learning rate*. Selain itu untuk metode-metode KMEANS, GMM, *Hu moments*, 7-D *Euclidean* akan diuji berdasarkan hasil klasifikasi yang muncul.

3.1.7 Pengambilan kesimpulan dan saran

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi, dan pengujian sistem telah selesai dilakukan yang didasarkan pada kesesuaian antara teori dan praktik. Kesimpulan diambil berdasarkan hasil pengujian sistem dan analisa dari penggunaan metode GMM dengan tujuan untuk menjawab rumusan masalah yang telah ditetapkan sebelumnya. Tahap terakhir dari penulisan

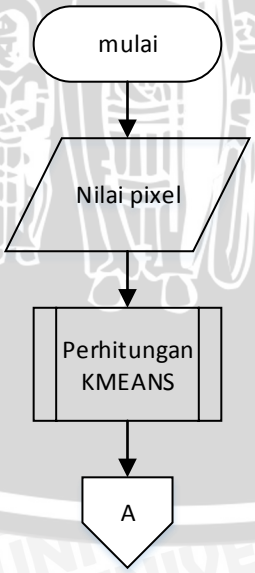
adalah saran untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan, serta untuk memberikan pertimbangan atas pengembangan penelitian selanjutnya.

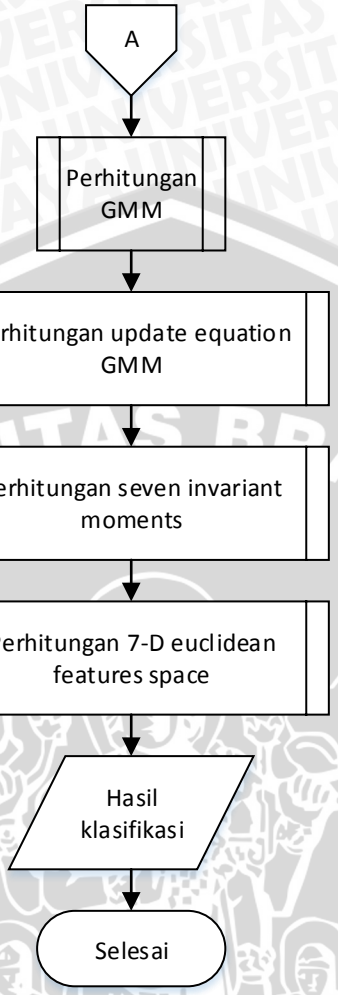
3.2 Perancangan

Sub bab perancangan menjelaskan tahapan proses desain sistem secara keseluruhan. Perancangan sistem dibuat berdasarkan hasil yang telah diperoleh dalam tahap pengumpulan data dan analisis kebutuhan. Perancangan sistem dilakukan agar proses pengimplementasian sistem menjadi lebih mudah. Tahapan perancangan berisi tentang diagram alir sistem, perhitungan manual, skenario pengujian, dan perancangan *user interface*.

3.2.1 Diagram alir sistem

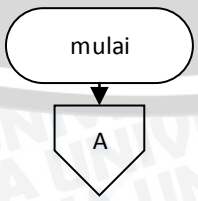
Diagram alir sistem berisi tahapan sistem klasifikasi kendaraan mulai dari masukkan sampai keluaran. Perancangan diagram alir sistem bertujuan untuk mengetahu proses secara keseluruhan dan jalannya mekanisme pengklasifikasian kendaraan. Diagram alir sistem akan dibagi menjadi 2 jenis diagram yaitu diagram alir proses klasifikasi kendaraan dan diagram alir proses training data kendaraan. Gambar 3.4 menjabarkan tentang proses pengklasifikasian kendaraan di jalan raya mengguanakan metode GMM berbasis *Hu Moments*, sedangkan gambar 3.5 menjabarkan tentang proses training data kendaraan di jalan raya mengguanakan metode GMM berbasis *Hu Moments*.

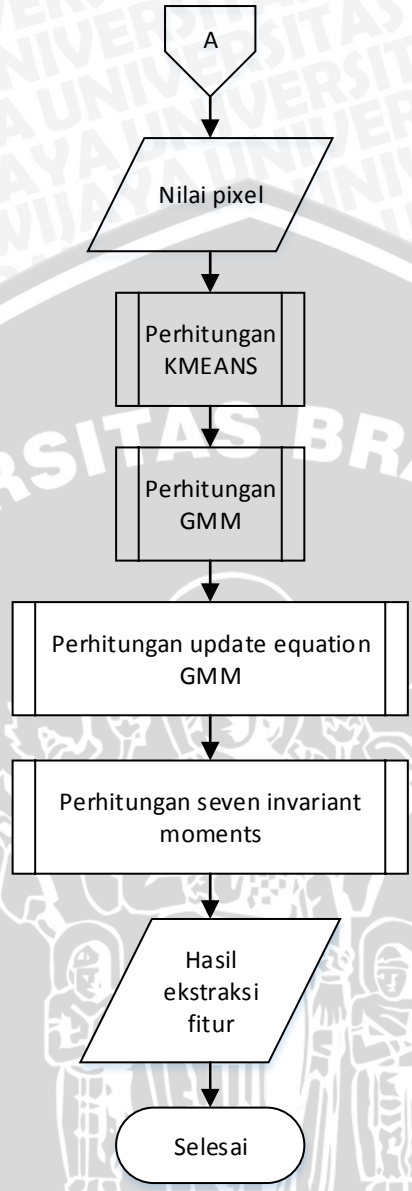




Gambar 3.4 Diagram alir proses klasifikasi kendaraan

Pada penelitian ini, untuk proses klasifikasi kendaraan data masukkan berupa nilai *pixel* dari *image sequence*, kemudian terdapat 5 tahap sub bab perancangan. Tahap pertama adalah perhitungan KMEANS, tahap kedua adalah perhitungan GMM, tahap ketiga adalah perhitungan *update equation* GMM, tahap keempat adalah perhitungan *seven invariant moments*, tahap kelima adalah perhitungan 7-D *euclidean features space*. Keluaran dari penelitian ini berupa hasil klasifikasi berupa motor atau mobil beserta jumlah dari masing-masing motor dan mobil.



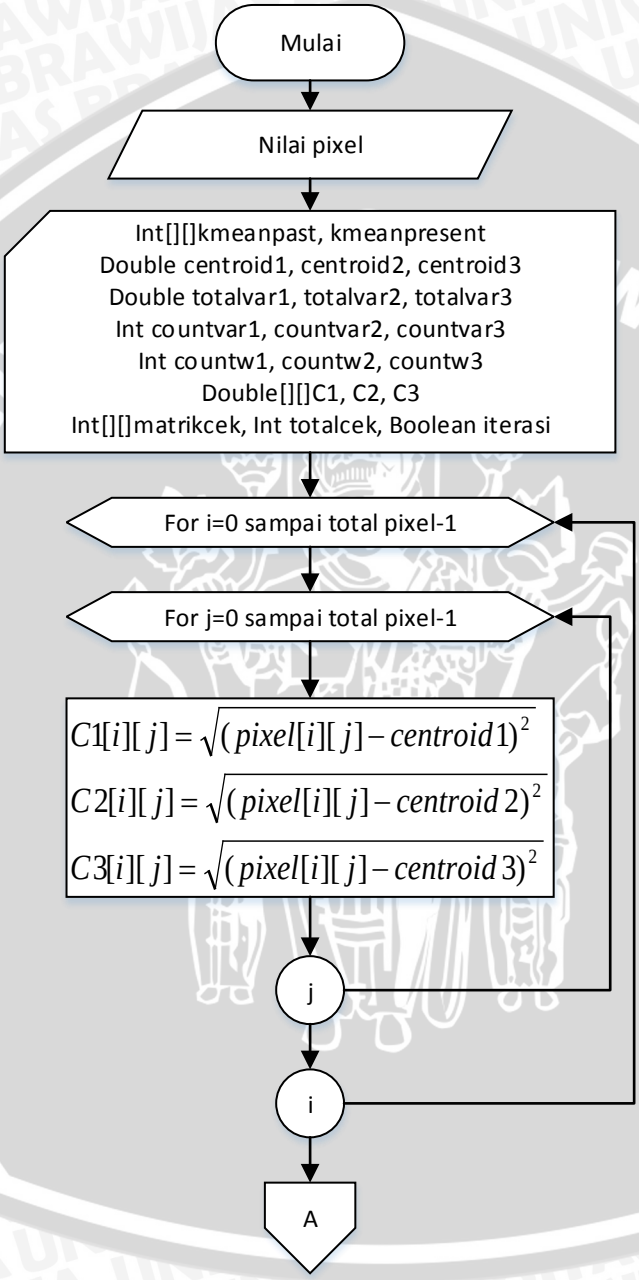


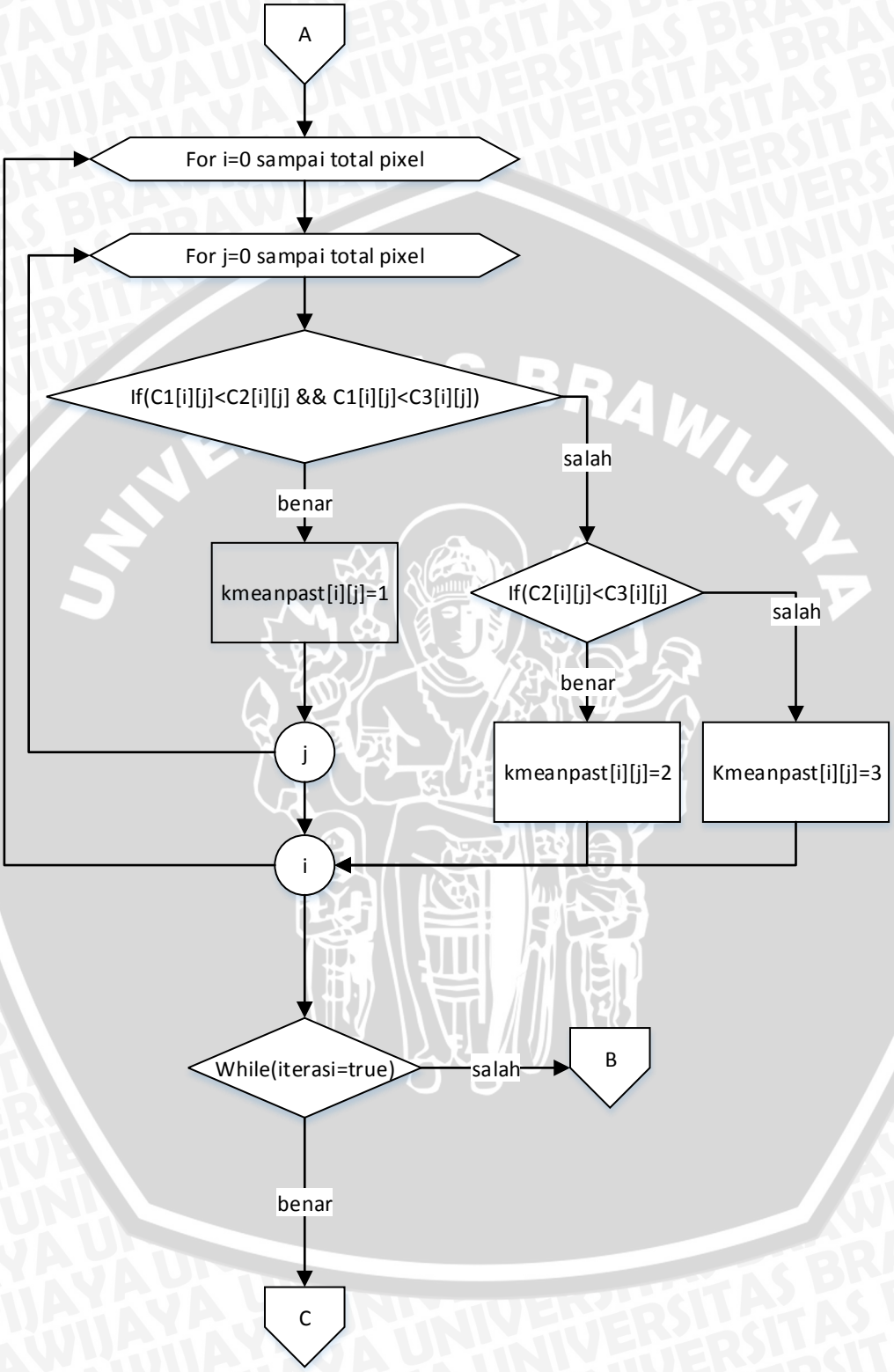
Gambar 3.5 Diagram alir proses training data kendaraan

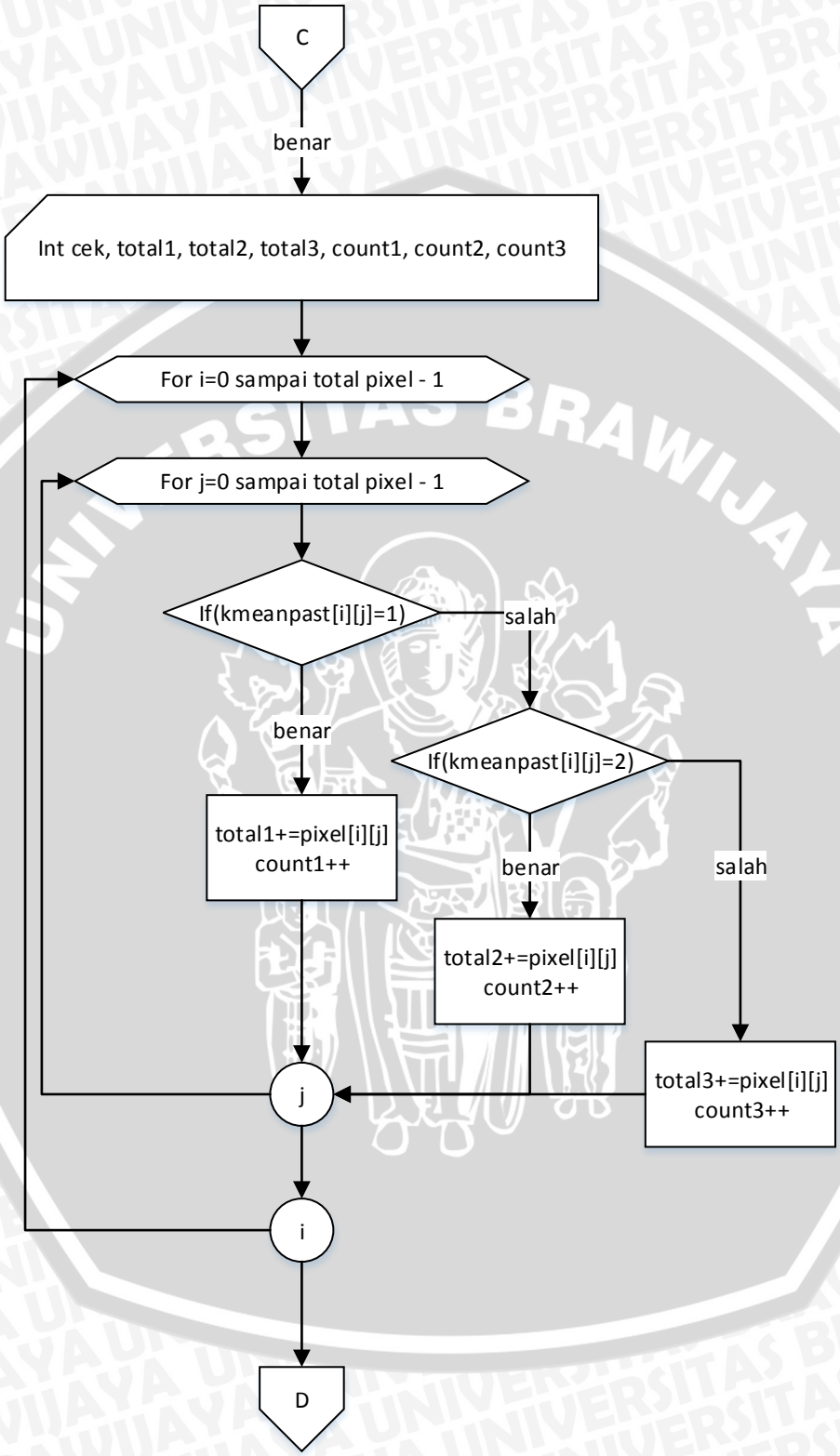
Pada penelitian ini, untuk proses training kendaraan data masukkan berupa nilai *pixel* dari *image sequence*, kemudian terdapat 4 tahap sub bab perancangan. Tahap pertama adalah perhitungan KMEANS, tahap kedua adalah perhitungan GMM, tahap ketiga adalah perhitungan *update equation* GMM, tahap keempat adalah perhitungan *seven invariant moments*. Keluaran dari penelitian ini adalah hasil training data berupa nilai perhitungan ekstraksi fitur tiap jenis sampel data kendaraan yaitu motor dan mobil.

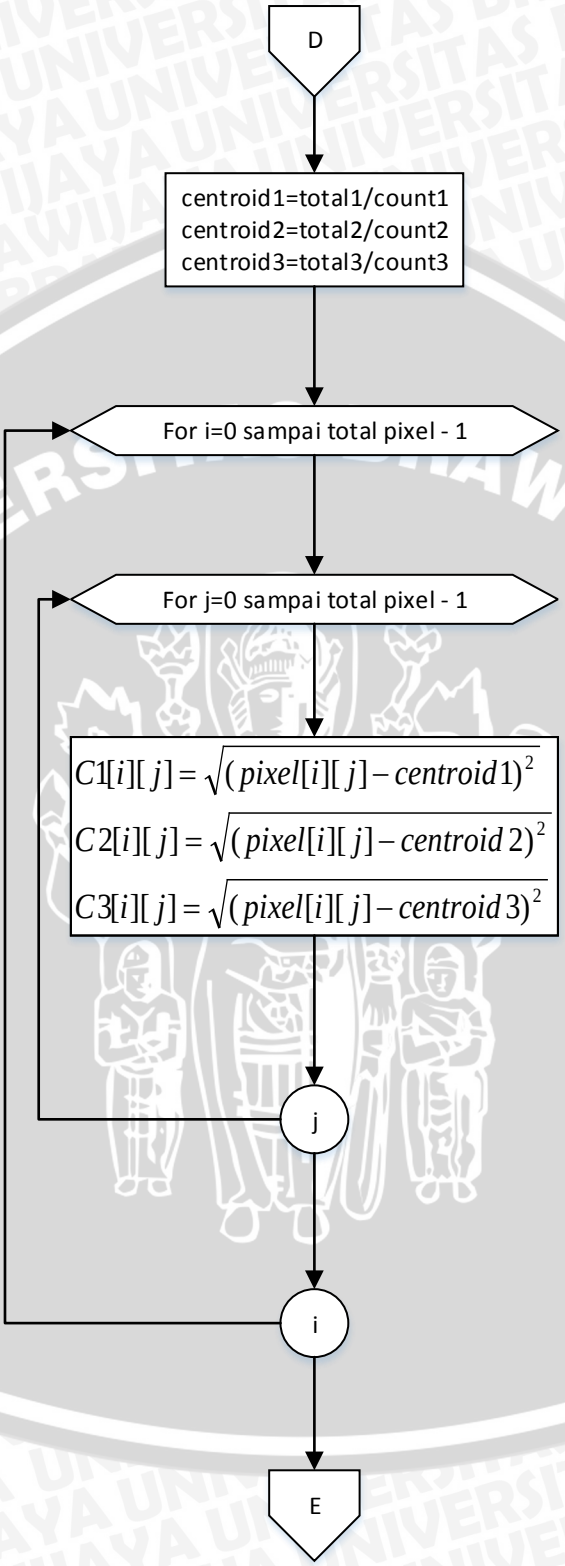
3.2.1.1 Proses perhitungan KMEANS

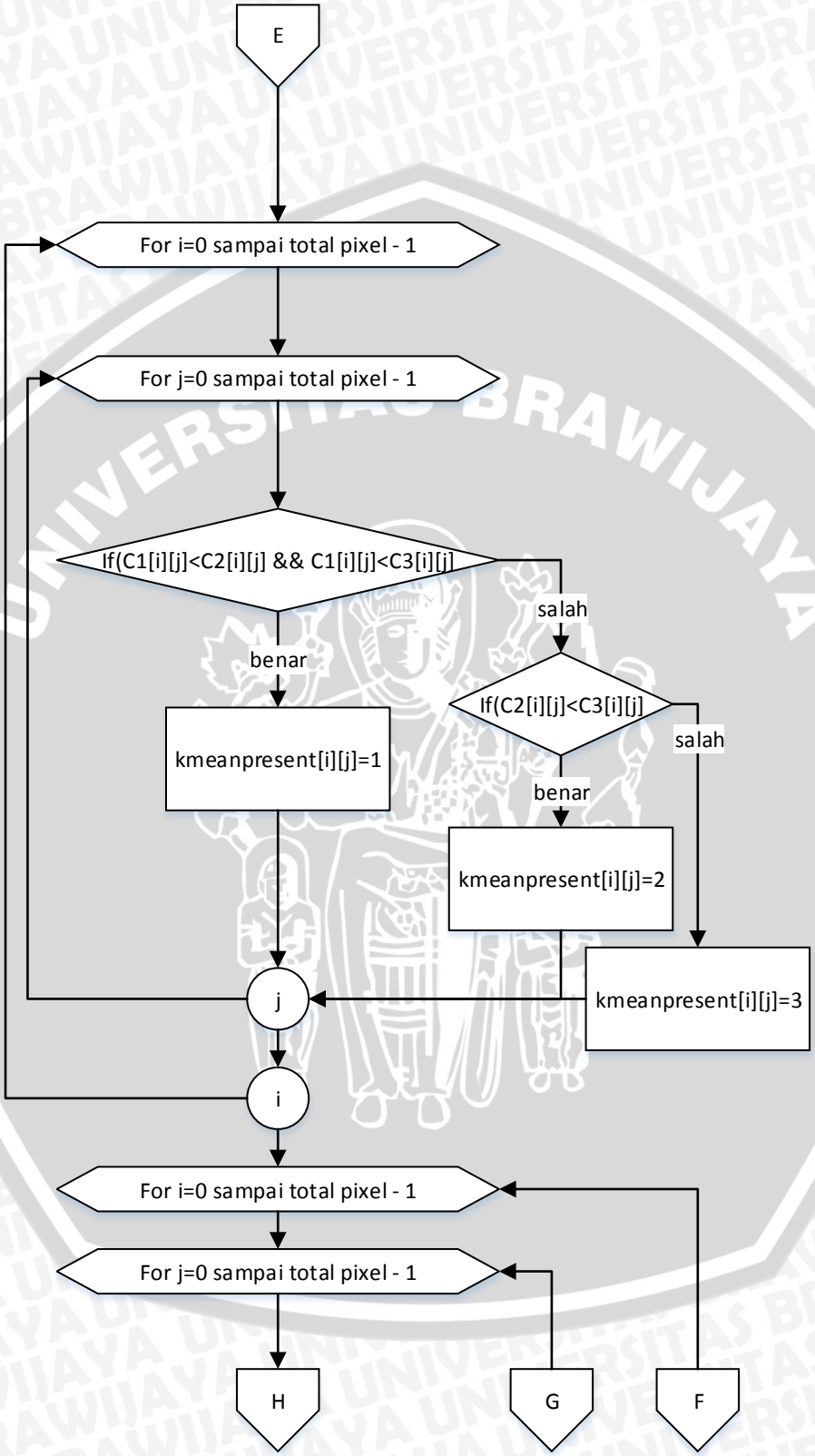
Proses perhitungan KMEANS merupakan perhitungan untuk menentukan cluster dari data *pixel* yang dimasukkan. Gambar 3.6 menjabarkan langkah-langkah dari proses KMEANS dalam bentuk diagram alir.

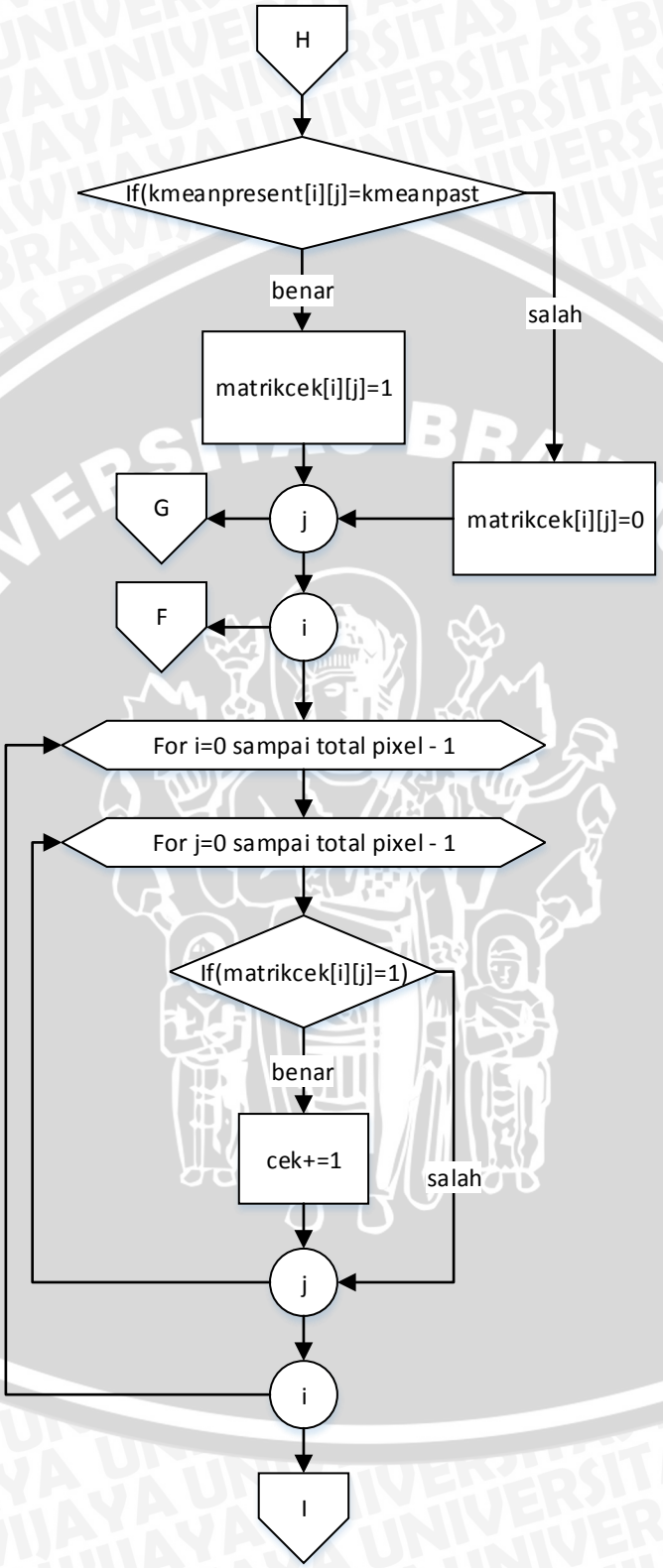


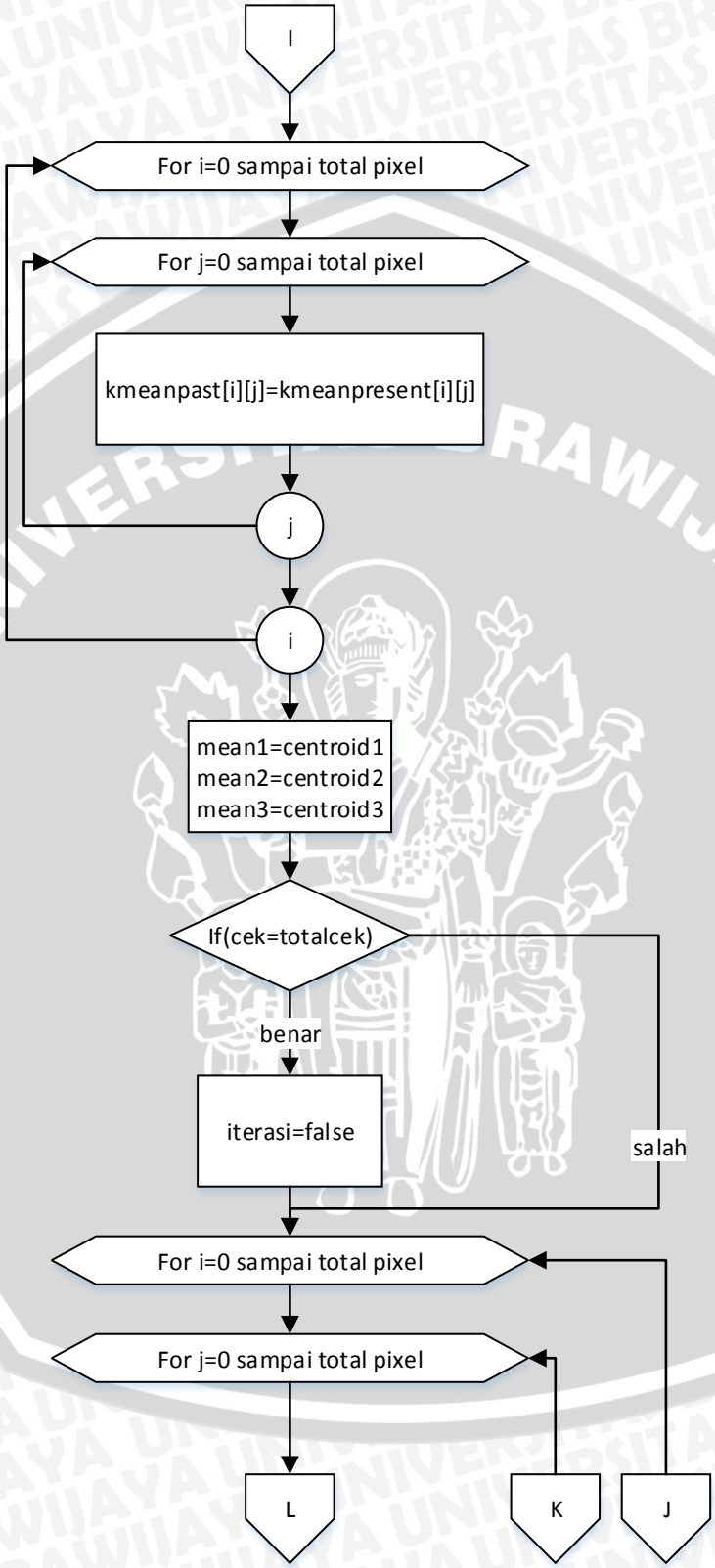

A
For i=0 sampai total pixel
For j=0 sampai total pixel
If(C1[i][j]<C2[i][j] && C1[i][j]<C3[i][j])
salah
benar
kmeanpast[i][j]=1
If(C2[i][j]<C3[i][j]
salah
benar
j
kmeanpast[i][j]=2
Kmeanpast[i][j]=3
i
While(iterasi=true)
salah
B
benar
C

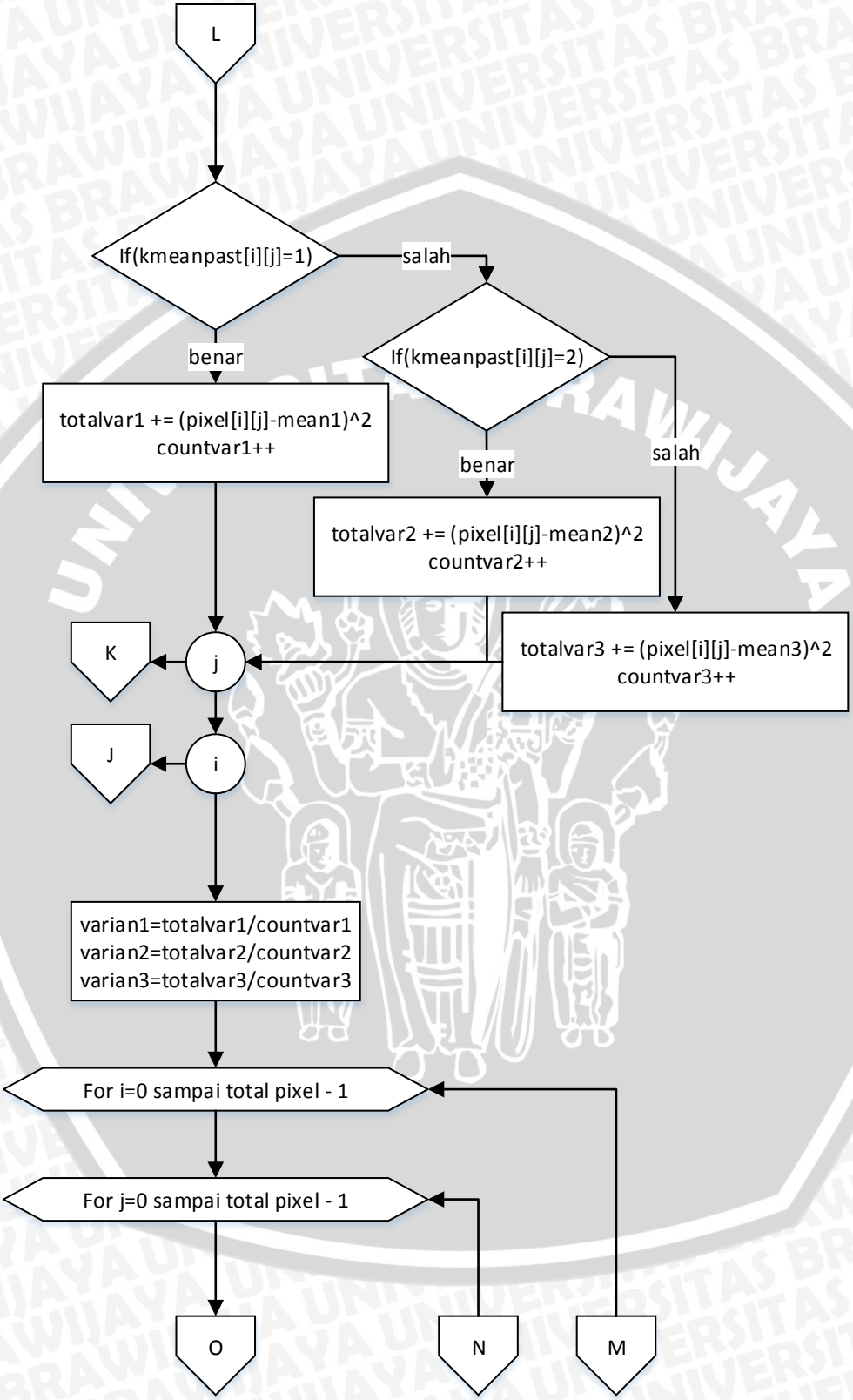

C
benar
Int cek, total1, total2, total3, count1, count2, count3
For i=0 sampai total pixel - 1
For j=0 sampai total pixel - 1
If(kmeanpast[i][j]=1)
salah
benar
If(kmeanpast[i][j]=2)
total1+=pixel[i][j]
count1++
benar
salah
total2+=pixel[i][j]
count2++
total3+=pixel[i][j]
count3++
j
i
D

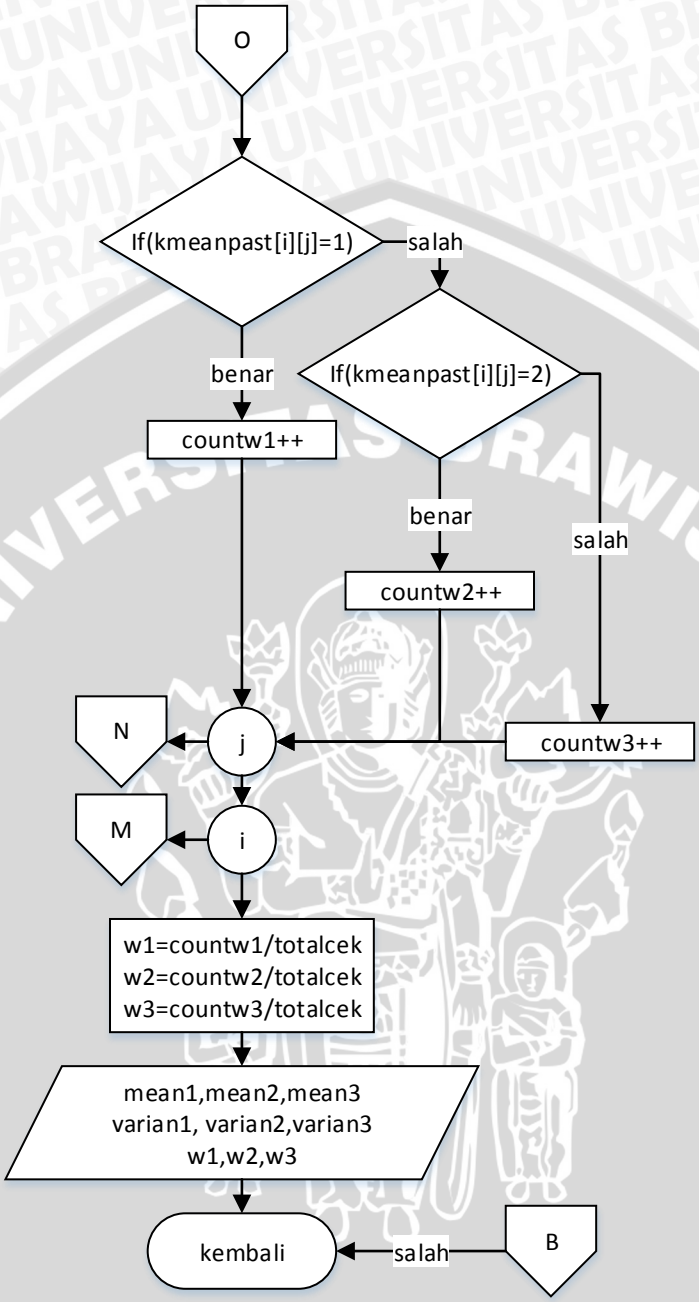

D
centroid1=total1/count1
centroid2=total2/count2
centroid3=total3/count3
For i=0 sampai total pixel - 1
For j=0 sampai total pixel - 1
$C1[i][j] = \sqrt{(pixel[i][j] - centroid1)^2}$
$C2[i][j] = \sqrt{(pixel[i][j] - centroid2)^2}$
$C3[i][j] = \sqrt{(pixel[i][j] - centroid3)^2}$
j
i
E


E
For i=0 sampai total pixel - 1
For j=0 sampai total pixel - 1
If(C1[i][j]<C2[i][j] && C1[i][j]<C3[i][j]
salah
benar
If(C2[i][j]<C3[i][j]
salah
benar
kmeanpresent[i][j]=1
kmeanpresent[i][j]=2
kmeanpresent[i][j]=3
j
i
For i=0 sampai total pixel - 1
For j=0 sampai total pixel - 1
H
G
F


H
If(kmeanpresent[i][j]=kmeanpast
benar
salah
matrikcek[i][j]=1
matrikcek[i][j]=0
G
j
F
i
For i=0 sampai total pixel - 1
For j=0 sampai total pixel - 1
If(matrikcek[i][j]=1)
benar
cek+=1
salah
j
i
I


I
For i=0 sampai total pixel
For j=0 sampai total pixel
kmeanpast[i][j]=kmeanpresent[i][j]
j
i
mean1=centroid1
mean2=centroid2
mean3=centroid3
If(cek=totalcek)
benar
iterasi=false
salah
For i=0 sampai total pixel
For j=0 sampai total pixel
L
K
J


L
If(kmeanpast[i][j]=1)
salah
benar
If(kmeanpast[i][j]=2)
totalvar1 += (pixel[i][j]-mean1)^2
countvar1++
benar
salah
totalvar2 += (pixel[i][j]-mean2)^2
countvar2++
totalvar3 += (pixel[i][j]-mean3)^2
countvar3++
K
j
J
i
varian1=totalvar1/countvar1
varian2=totalvar2/countvar2
varian3=totalvar3/countvar3
For i=0 sampai total pixel - 1
For j=0 sampai total pixel - 1
O
N
M



Gambar 3.6 Diagram alir proses KMEANS

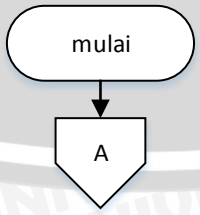
Berikut adalah penjabaran diagram alir proses perhitungan KMEANS pada gambar 3.6 :

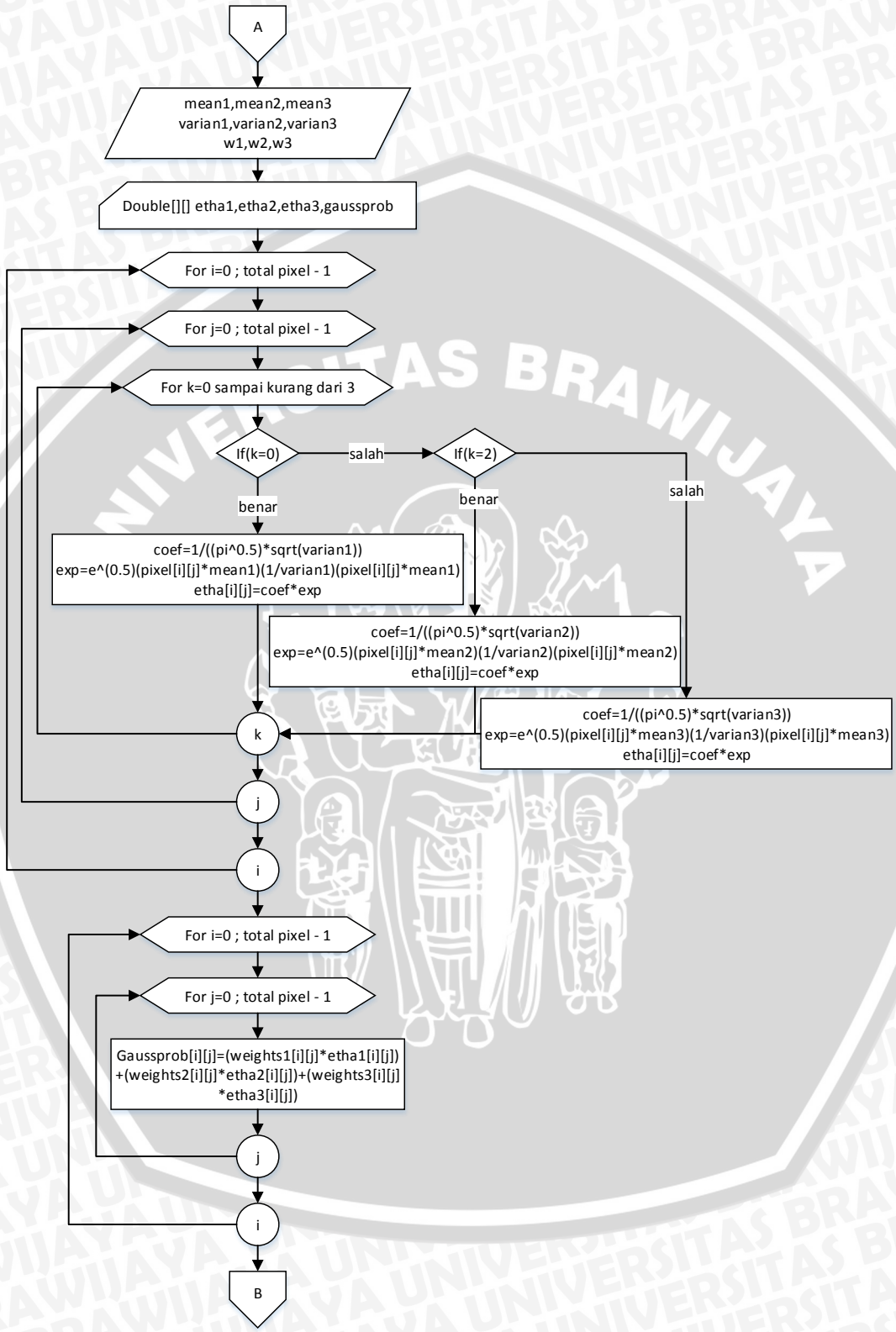
1. Data masukkan berupa nilai *pixel*.
2. Inisialisasi variabel kmeanpast, kmeanpresent, centroid1, centroid2, centroid3, totalvar1, totalvar2, totalvar3, countvar1, countvar2, countvar3, countw1, countw2, countw3, C1, C2, C3, matrikcek, totalcek, iterasi.

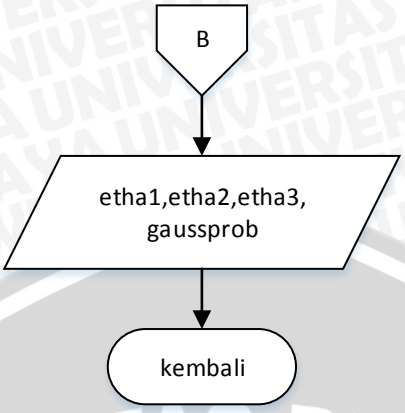
3. Perulangan untuk menghitung nilai kedekatan tiap *pixel* terhadap masing-masing *centroid* menggunakan persamaan 2.2.
4. Perulangan untuk memberi nilai pada variabel kmeanpast.
5. Masuk ke iterasi while yang berfungsi untuk mencari cluster pada data *pixel image sequence*.
6. Proses inisialisasi variabel cek, total1, total2, total3, count1, count2, count3.
7. Perulangan untuk memberi nilai pada variabel total1, total2, total3, count1, count2, count3.
8. Proses perhitungan variabel centroid1, centroid2, centroid3.
9. Perulangan untuk menghitung nilai kedekatan tiap *pixel* terhadap masing-masing *centroid* menggunakan persamaan (2.2).
10. Perulangan untuk memberi nilai pada variabel kmeanpresent.
11. Perulangan untuk memberi nilai pada variabel matrikcek.
12. Perulangan untuk memberi nilai pada variabel cek.
13. Perulangan untuk memberi nilai pada variabel kmeanpast menggunakan nilai dari kmeanpresent.
14. Proses perhitungan variabel mean1, mean2, mean3.
15. Kondisi percabangan untuk memberi nilai pada variabel iterasi.
16. Perulangan untuk memberi nilai pada variabel totalvar1, totalvar2, totalvar3, countvar1, countvar2, countvar3.
17. Proses perhitungan variable varian1, varian2, varian3.
18. Perulangan untuk memberi nilai pada variabel countw1, countw2, countw3.
19. Proses perhitungan variabel w1, w2, w3.
20. Data keluaran berupa variabel mean1, mean2, mean3, varian1, varian2, varian3, w1, w2, w3.

3.2.1.2 Proses perhitungan GMM

Proses perhitungan GMM merupakan perhitungan untuk mendapatkan nilai *gauss probability* yang dapat digunakan untuk mengelompokkan *background* dengan *foreground*. Proses GMM di lakukan setelah proses KMEANS selesai dilakukan karena hasil keluaran dari KMEANS digunakan untuk melakukan perhitungan pada GMM. Gambar 3.7 menunjukkan diagram alir proses perhitungan GMM.




A
mean1,mean2,mean3
varian1,varian2,varian3
w1,w2,w3
Double[][] etha1,etha2,etha3,gaussprob
For i=0 ; total pixel - 1
For j=0 ; total pixel - 1
For k=0 sampai kurang dari 3
If(k=0)
salah
If(k=2)
benar
benar
salah
coef=1/((pi^0.5)*sqrt(varian1))
exp=e^(0.5)(pixel[i][j]*mean1)(1/varian1)(pixel[i][j]*mean1)
etha[i][j]=coef*exp
coef=1/((pi^0.5)*sqrt(varian2))
exp=e^(0.5)(pixel[i][j]*mean2)(1/varian2)(pixel[i][j]*mean2)
etha[i][j]=coef*exp
coef=1/((pi^0.5)*sqrt(varian3))
exp=e^(0.5)(pixel[i][j]*mean3)(1/varian3)(pixel[i][j]*mean3)
etha[i][j]=coef*exp
k
j
i
For i=0 ; total pixel - 1
For j=0 ; total pixel - 1
Gaussprob[i][j]=(weights1[i][j]*etha1[i][j])
+(weights2[i][j]*etha2[i][j])+(weights3[i][j]
*etha3[i][j])
j
i
B



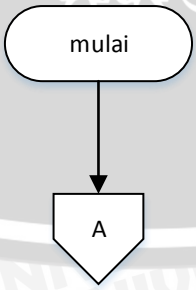
Gambar 3.7 Diagram alir proses GMM

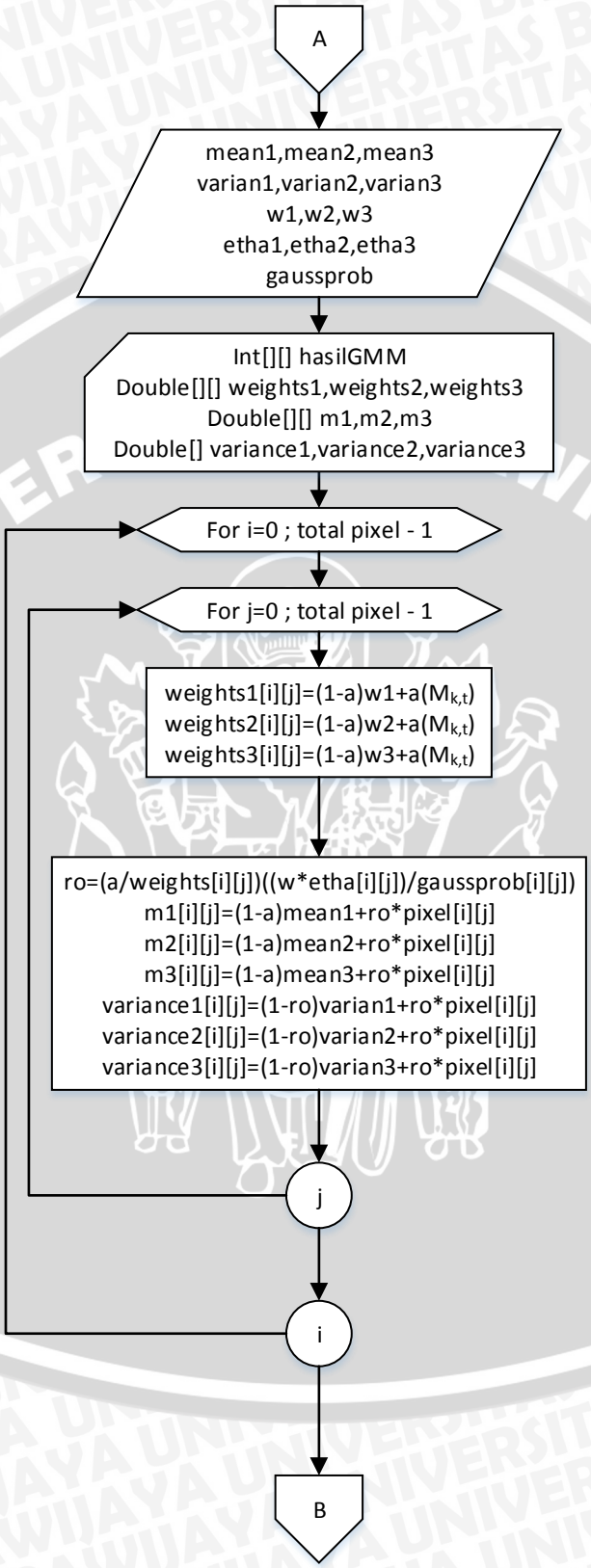
Berikut adalah penjabaran diagram alir proses perhitungan GMM pada gambar 3.7 :

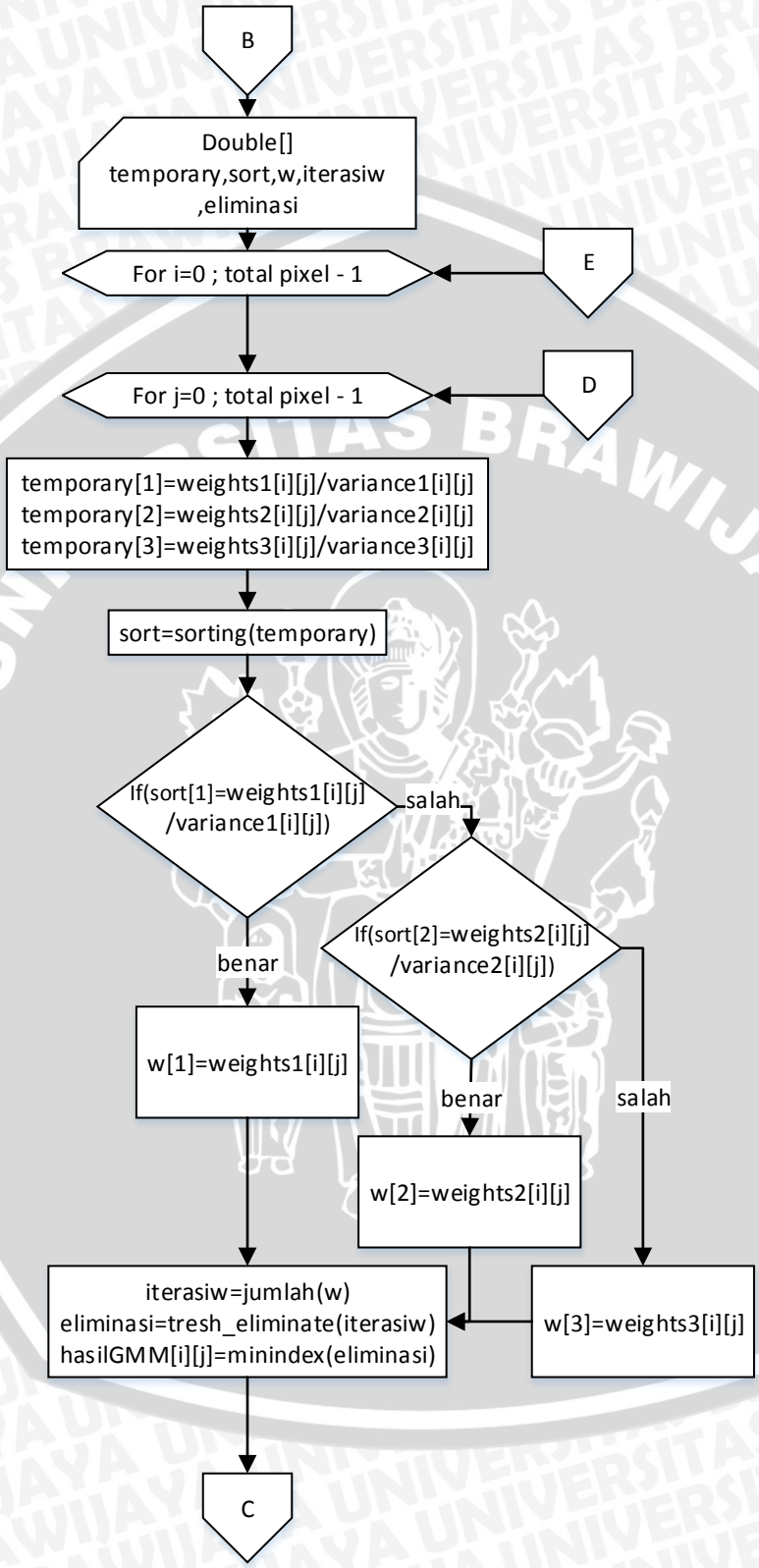
1. Data masukkan berupa variabel mean1, mean2, mean3, varian1, varian2, varian3, w1, w2, w3.
2. Proses inisialisasi variabel etha1, etha2, etha3, gaussprob.
3. Perulangan untuk menghitung nilai dari variabel etha1, etha2, etha3 menggunakan persamaan (2.4).
4. Perulangan untuk menghitung nilai dari variabel gaussprob menggunakan persamaan (2.3).
5. Data keluaran berupa variabel etha1, etha2, etha3, gaussprob.

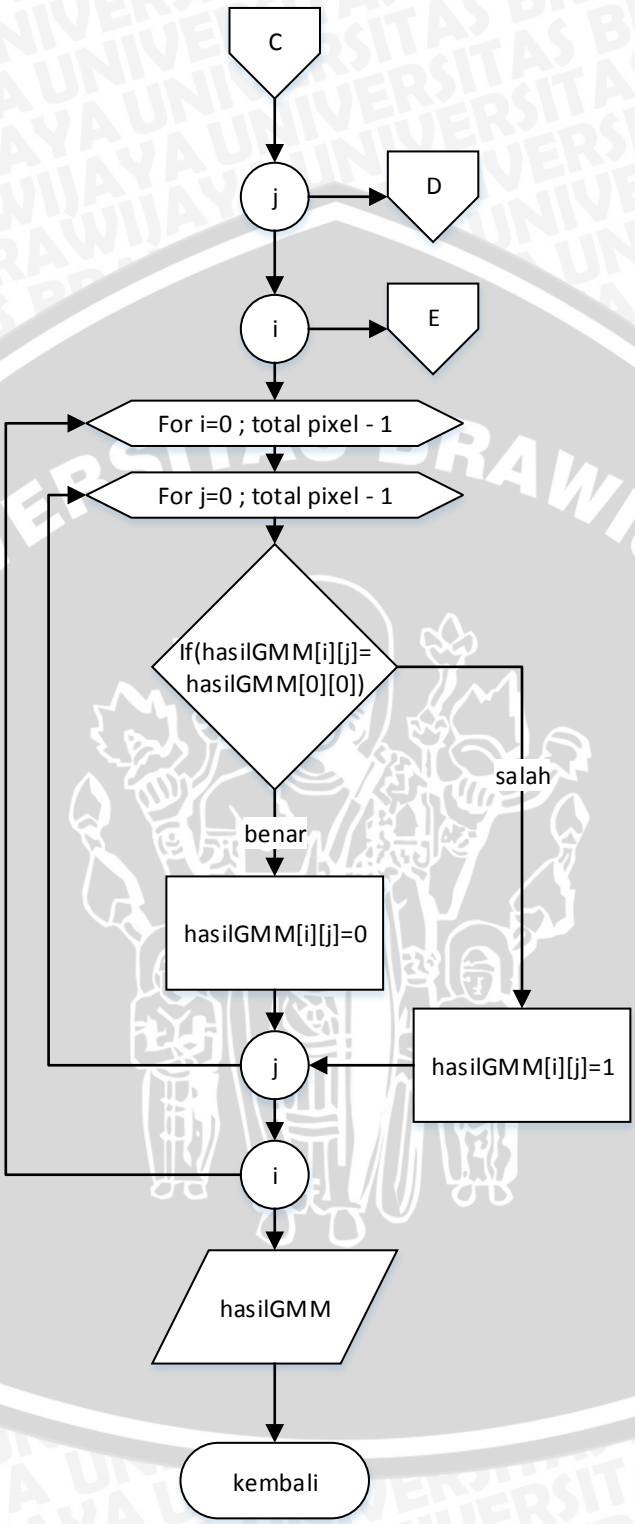
3.2.1.3 Proses perhitungan *update equation* GMM

Proses perhitungan *update equation* GMM merupakan perhitungan untuk melakukan perbaruan pada variable GMM seperti *prior weights*, *mean*, dan *variance*. Pada proses *update equation* GMM juga dilakukan proses untuk mengelompokkan pixel *background* dan *foreground*. Gambar 3.8 menunjukkan diagram alir proses *update equation* GMM.




A
mean1,mean2,mean3
varian1,varian2,varian3
w1,w2,w3
etha1,etha2,etha3
gaussprob
Int[][] hasilGMM
Double[][] weights1,weights2,weights3
Double[][] m1,m2,m3
Double[] variance1,variance2,variance3
For i=0 ; total pixel - 1
For j=0 ; total pixel - 1
weights1[i][j]=(1-a)w1+a(M_k,t)
weights2[i][j]=(1-a)w2+a(M_k,t)
weights3[i][j]=(1-a)w3+a(M_k,t)
ro=(a/weights[i][j])((w*etha[i][j])/gaussprob[i][j])
m1[i][j]=(1-a)mean1+ro*pixel[i][j]
m2[i][j]=(1-a)mean2+ro*pixel[i][j]
m3[i][j]=(1-a)mean3+ro*pixel[i][j]
variance1[i][j]=(1-ro)varian1+ro*pixel[i][j]
variance2[i][j]=(1-ro)varian2+ro*pixel[i][j]
variance3[i][j]=(1-ro)varian3+ro*pixel[i][j]
j
i
B


B
Double[]
temporary,sort,w,iterasiw
,eliminasi
For i=0 ; total pixel - 1
E
For j=0 ; total pixel - 1
D
temporary[1]=weights1[i][j]/variance1[i][j]
temporary[2]=weights2[i][j]/variance2[i][j]
temporary[3]=weights3[i][j]/variance3[i][j]
sort=sorting(temporary)
If(sort[1]=weights1[i][j]
/variance1[i][j])
salah
benar
If(sort[2]=weights2[i][j]
/variance2[i][j])
benar
salah
w[1]=weights1[i][j]
w[2]=weights2[i][j]
iterasiw=jumlah(w)
eliminasi=tresh_eliminate(iterasiw)
hasilGMM[i][j]=minindex(eliminasi)
w[3]=weights3[i][j]
C



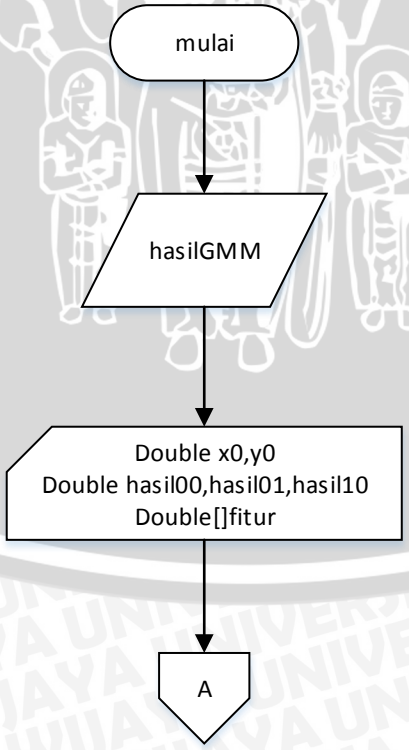
Gambar 3.8 Diagram alir proses perhitungan *update equation* GMM

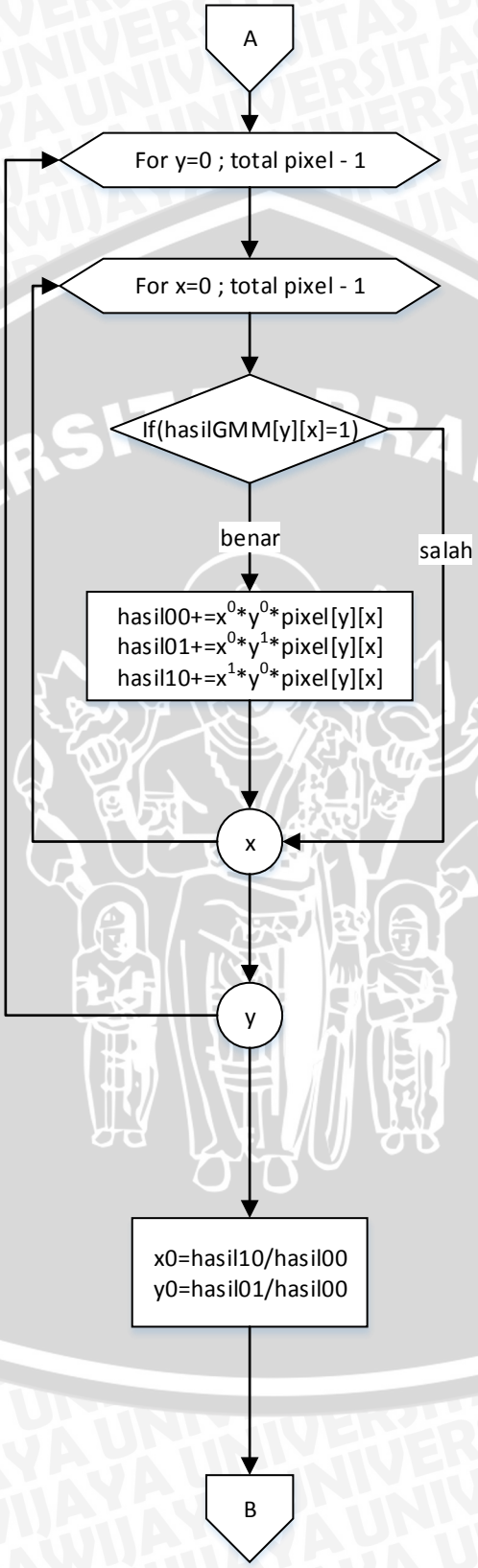
Berikut adalah penjabaran diagram alir proses perhitungan *update equation* GMM pada gambar 3.8 :

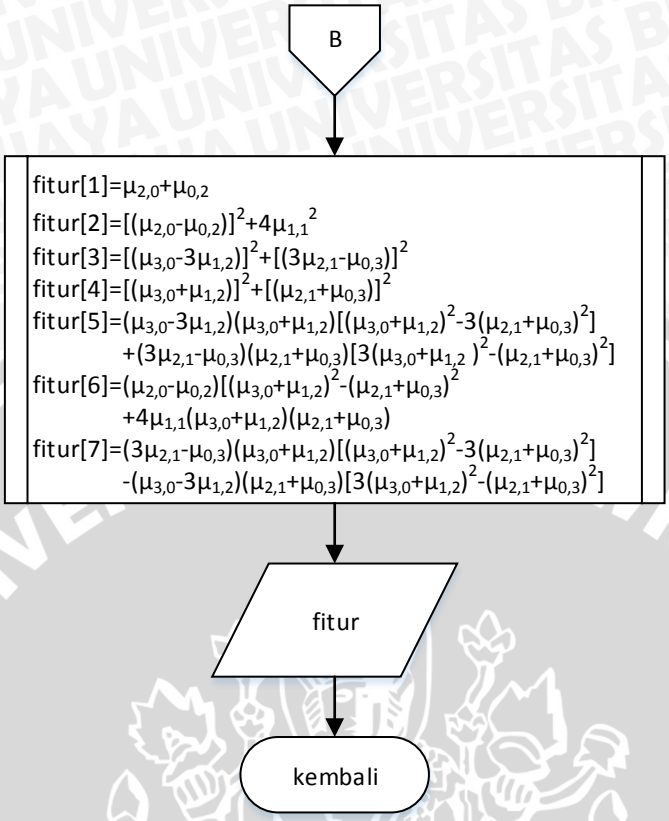
1. Data masukkan berupa variabel mean1, mean2, mean3, varian1, varian2, varian3, w1, w2, w3, etha1, etha2, etha3, gaussprob.
2. Proses inisialisasi variabel hasilGMM, weights1, weights2, weights3, m1, m2, m3, variance1, variance2, variance3.
3. Perulangan untuk menghitung variabel weights1, weights2, weights3, m1, m2, m3, variance1, variance2, variance3 menggunakan persamaan (2.6), (2.7), (2.8), dan (2.9).
4. Proses inisialisasi variabel temporary, sort, w, iterasiw, eliminasi.
5. Perulangan untuk menghitung variabel hasilGMM menggunakan persamaan (2.12).
6. Perulangan untuk mengolah variabel hasilGMM menjadi dua nilai antara 0 atau 1.
7. Data keluaran berupa variabel hasilGMM.

3.2.1.4 Proses perhitungan *seven invariant moments*

Proses perhitungan *seven invariant moments* merupakan perhitungan ekstraksi fitur dari objek *foreground*. Objek foreground diperoleh dari hasil pengolahan berurutan mulai dari KMEANS, GMM, sampai perhitungan *update equation* GMM. Gambar 3.9 menunjukkan diagram alir proses perhitungan *seven invariant moments*.




A
For y=0 ; total pixel - 1
For x=0 ; total pixel - 1
If(hasilGMM[y][x]=1)
benar
salah
hasil00+=x^0*y^0*pixel[y][x]
hasil01+=x^0*y^1*pixel[y][x]
hasil10+=x^1*y^0*pixel[y][x]
x
y
x0=hasil10/hasil00
y0=hasil01/hasil00
B



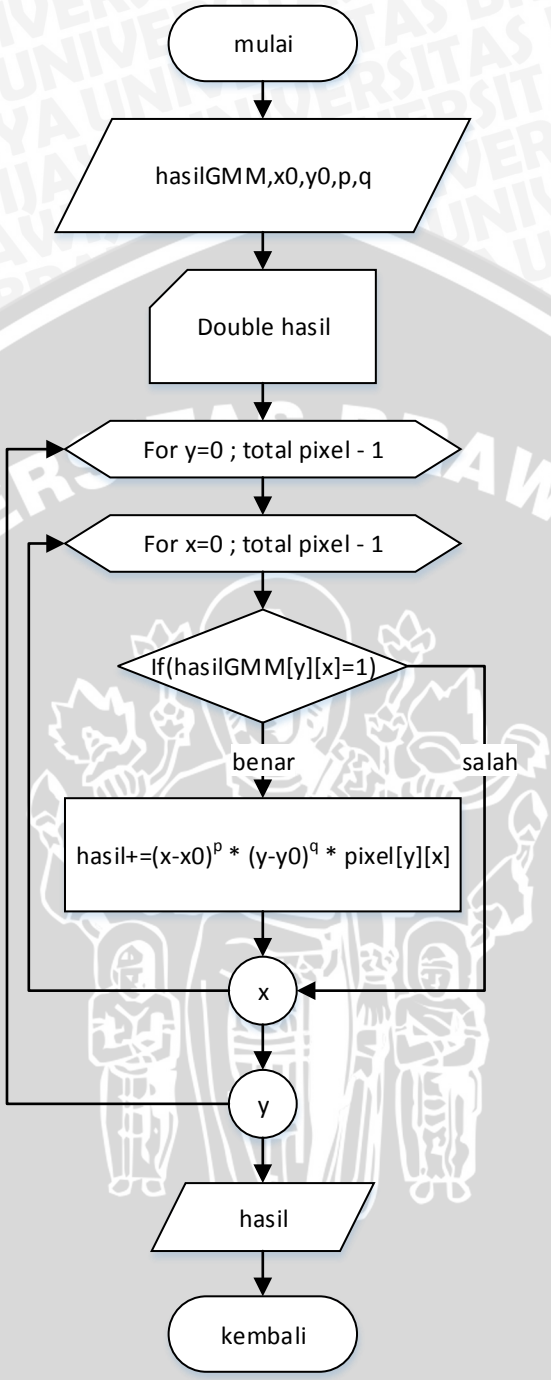
Gambar 3.9 Diagram alir proses perhitungan *seven invariant moments*

Berikut adalah penjabaran diagram alir proses perhitungan *seven invariant moments* pada gambar 3.9 :

1. Data masukkan berupa variabel hasilGMM.
2. Proses inisialisasi variabel x0, y0, hasil00, hasil01, hasil10, fitur.
3. Perulangan untuk menghitung variabel x0, y0, hasil00, hasil01, hasil10 menggunakan persamaan (2.21) dan (2.22).
4. Proses perhitungan variabel fitur menggunakan persamaan (2.13), (2.14), (2.15), (2.16), (2.17), (2.18), dan (2.19).
5. Data keluaran berupa variabel fitur.

- **Proses perhitungan *μiu***

Proses perhitungan *μiu* merupakan perhitungan nilai *μiu* pada indeks p,q yang telah dimasukkan oleh proses perhitungan *seven invariant moments*. Proses perhitungan μiu merupakan sub proses dari perhitungan *seven invariant moments*. Gambar 3.10 menunjukkan diagram alir proses perhitungan *μiu*.



Gambar 3.10 Diagram alir proses perhitungan *μiu*

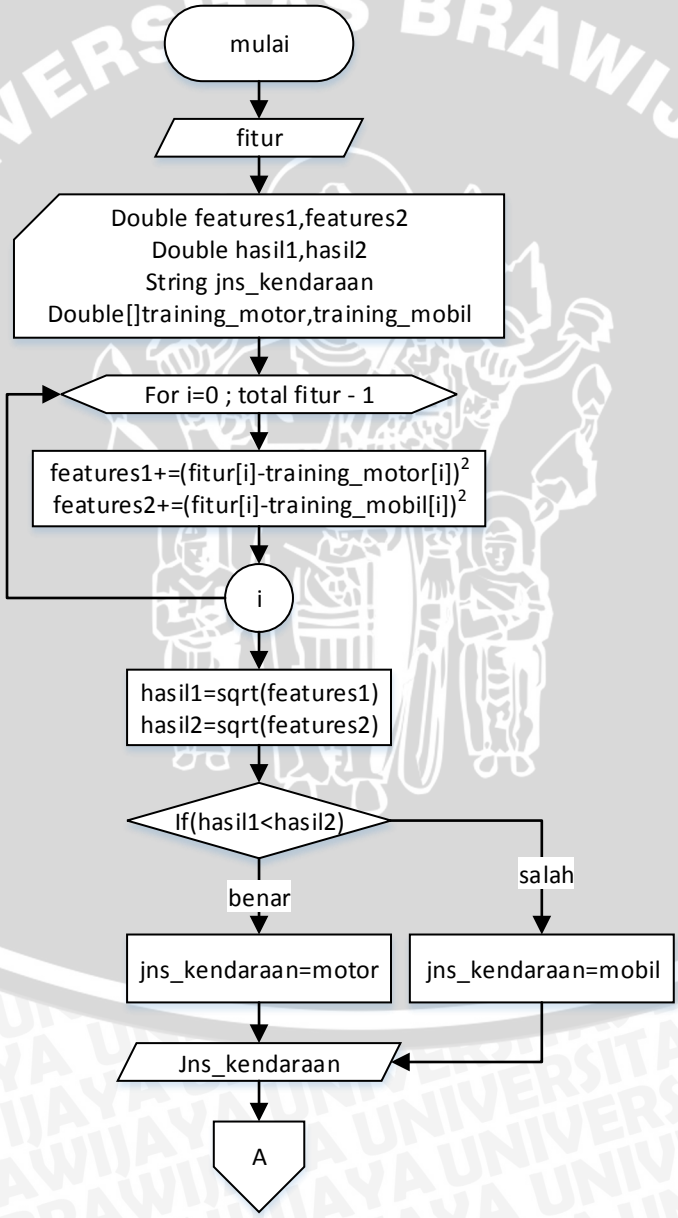
Berikut adalah penjabaran diagram alir proses perhitungan *μiu* gambar 3.10 :

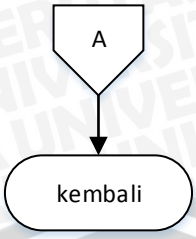
1. Data masukkan berupa variabel hailGMM, x0, y0, p, q.
2. Proses inisialisasi variabel hasil.

3. Perulangan untuk menghitung nilai dari variabel hasil menggunakan persamaan (2.18).
4. Ouput data berupa variabel hasil.

3.2.1.5 Proses perhitungan 7-D *eucidean features space*

Proses perhitungan 7-D *euclidean features space* merupakan perhitungan klasifikasi kendaraan. Hasil dari ekstraksi fitur akan di kalkulasi pada proses 7-D *euclidean features space* sehingga menghasilkan klasifikasi kendaraan mobil atau motor. Gambar 3.11 menunjukkan diagram alir proses perhitungan 7-D e*uclidean features space*.





Gambar 3.11 Diagram alir proses perhitungan 7-D e*uclidean features space*

Berikut adalah penjabaran diagram alir proses perhitungan 7-D *euclidean features space* pada Gambar 3.11 :

1. Data masukkan berupa variabel fitur.
2. Proses inisialisasi variabel features1, features2, hasil1, hasil2, jns_kendaraan, training_motor, training_mobil.
3. Perulangan untuk menghitung variabel features1 dan features2.
4. Proses perhitungan variabel hasil1 dan hasil2 menggunakan persamaan (2.21).
5. Kondisi percabagan untuk memproses variabel jns_kendaraan dengan nilai "motor" atau "mobil".
6. Data keluaran berupa variabel jns_kendaraan.

3.2.2 Perhitungan manual

Proses perhitungan manual pada penelitian ini menggunakan dataset berupa nilai *gray* pada tiap *pixel* frame gambar lalu lintas. Data tersebut merupakan data yang skalanya sudah diubah menjadi dimensi *pixel* 10x10 dari data asli. Data asli yang digunakan memiliki dimensi *pixel* 250x250. Tahapan pada perhitungan manual terdiri dari perhitungan KMEANS, GMM, *update equation*, ekstraksi fitur, dan klasifikasi kendaraan. Pada proses perhitungan KMEANS data *pixel* akan dikelompokkan menjadi 3 *cluster* sesuai dengan jurnal yang diacu yaitu penelitian yang dilakukan H.Asaidi dkk, dengan menggunakan 3 cluster akan menghasilkan hasil yang optimal dan beban komputasi juga ringan, kemudian proses perhitungan GMM akan melakukan perhitungan *gauss probability* menjadi 3 distribusi, setelah itu akan melakukan *update equation* pada variabel GMM, kemudian menghitung nilai ekstraksi fitur berdasarkan hasil dari GMM, langkah terakhir adalah klasifikasi jenis objek menjadi dua yaitu mobil atau motor. Table 3.1 menunjukkan nilai gray pada sebuah frame gambar lalu lintas dengan dimensi *pixel* 10x10.

Table 3.1 Dataset pixel gray

gray									
63	71	77	79	83	85	82	80	77	71

Tabel 3.1 Dataset pixel gray (lanjutan)

gray									
59	65	70	74	78	79	79	75	71	66
87	105	113	124	133	137	133	88	81	77
153	70	70	77	77	166	78	74	69	63
58	65	67	73	78	80	66	74	71	64
55	57	62	67	71	73	72	70	65	59
41	40	55	65	68	70	68	67	55	54
40	47	52	58	61	62	54	59	56	50
42	47	54	58	61	64	58	61	57	52
38	43	47	51	55	56	57	54	50	46

3.2.2.1 Perhitungan manual KMEANS

Proses perhitungan KMEANS bertujuan untuk mengelompokkan *dataset pixel gray* pada frame ke-1 ($X_{t=1}$) menjadi 3 cluster. Untuk mendapatkan 3 cluster pada *dataset pixel gray* maka proses KMEANS akan melakukan iterasi sampai nilai cluster stabil. Nilai cluster stabil artinya cluster yang terbentuk pada proses perhitungan KMEANS tidak mengalami perpindahan saat iterasi. Berikut adalah tahapan-tahapan proses perhitungan manual KMEANS :

1. Menentukan nilai awal centroid.
   Nilai awal Centroid1, Centroid2, dan Centroid3 didapat dari dataset pixel gray pada indeks (0,0)(0,1)(0,2).
   Centroid1 = 63
   Centroid2 = 71
   Centroid3 = 77
2. Menghitung jarak antara *dataset pixel gray* dengan masing-masing *centroid* menggunakan persamaan (2.2).
   $D_{L_2} = \sqrt{(X_1(0,0) - centroid1)^2}$
   $D_{L_2} = \sqrt{(63 - 63)^2}$
   $D_{L_2} = 0$
   $D_{L_2} = \sqrt{(X_1(0,0) - centroid2)^2}$
   $D_{L_2} = \sqrt{(63 - 71)^2}$
   $D_{L_2} = 8$
   $D_{L_2} = \sqrt{(X_1(0,0) - centroid3)^2}$
   $D_{L_2} = \sqrt{(63 - 77)^2}$
   $D_{L_2} = 14$
   Tabel 3.2 menunjukkan hasil perhitungan jarak antara *dataset pixel gray* dengan Centroid1.

Tabel 3.2 Jarak *dataset pixel gray* dengan Centroid1

C1									
0	8	14	16	20	22	19	17	14	8
4	2	7	11	15	16	16	12	8	3
24	42	50	61	70	74	70	25	18	14
90	7	7	14	14	103	15	11	6	0
5	2	4	10	15	17	3	11	8	1
8	6	1	4	8	10	9	7	2	4
22	23	8	2	5	7	5	4	8	9
23	16	11	5	2	1	9	4	7	13
21	16	9	5	2	1	5	2	6	11
25	20	16	12	8	7	6	9	13	17

3. Menentukan nilai cluster berdasarkan nilai terkecil dari hasil perhitungan jarak.
   C1 indeks(0,0) = 0
   C2 indeks(0,0) = 8
   C3 indeks(0,0) = 14
   Nilai terkecil antara C1, C2, dan C3 adalah 0, maka nilai cluster untuk *dataset pixel gray* indeks (0,0) adalah 1. Jika bernilai 1 maka *pixel* tersebut masuk ke cluster 1, jika bernilai 2 maka *pixel* tersebut masuk ke cluster 2, dan jika bernilai 3 maka *pixel* tersebut masuk ke cluster 3. Tabel 3.3 menunjukkan hasil perhitungan cluster pada iterasi ke-0.

Tabel 3.3 Hasil perhitungan cluster iterasi ke-0

Hasil iterasi ke 0									
1	2	3	3	3	3	3	3	3	2
1	1	2	3	3	3	3	3	2	1
3	3	3	3	3	3	3	3	3	3
3	2	2	3	3	3	3	3	2	1
1	1	2	2	3	3	1	3	2	1
1	1	1	2	2	2	2	2	1	1
1	1	1	1	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

4. Memperbarui nilai centroid berdasarkan hasil perhitungan cluster.
   Untuk mendapatkan nilai pada masing-masing centroid maka nilai *gray* untuk tiap cluster harus dijumlah kemudian dibagi dengan jumlah tiap cluster yang terbentuk.
   Centroid1 = 2767 / 50 = 55.34

Centroid2 = 1329 / 19 = 69.94737
Centroid3 = 2883 / 31 = 93

5. Mengulangi perhitungan jarak dengan centroid sampai hasil cluster stabil.
   Mengulangi proses perhitungan jarak dengan centroid dan memperbarui nilai centroid sampai menemukan hasil cluster yang stabil. Hasil cluster stabil ditandai dengan cluster baru yang terbentuk mempunyai nilai cluster yang sama dengan nilai cluster yang sebelumnya. Tabel 3.4 menunjukkan hasil cluster yang stabil berada pada iterasi ke-3, jika bernilai 1 maka *pixel* tersebut berada pada cluster 1, jika bernilai 2 maka *pixel* tersebut berada pada cluster 2, dan jika bernilai 3 maka *pixel* tersebut berada pada cluster 3.

   Hasil perhitungan centroid pada iterasi ke 3 :
   Centroid1 = 40
   Centroid2 = 52
   Centroid3 = 8
   Hasil perhitungan centroid pada iterasi ke 4 :
   Centroid1 = 40
   Centroid2 = 52
   Centroid3 = 8
   Perbandingan centroid iterasi ke 3 dengan ke 4 :
   Centroid1 = (40-40) = 0
   Centroid2 = (52-52) = 0
   Centroid3 = (8-8) = 0
   Karena nilai perbandingan nilai centroid menghasilkan nilai 0 maka proses KMEANS berhenti dan hasil pengelompokkan dinyatakan sudah stabil sesuai dengan tahapan KMEANS pada tahap kelima.

Tabel 3.4 Hasil perhitungan cluster iterasi ke-3

Hasil iterasi ke 3									
2	2	2	2	2	2	2	2	2	2
1	2	2	2	2	2	2	2	2	2
2	3	3	3	3	3	3	2	2	2
3	2	2	2	2	3	2	2	2	2
1	2	2	2	2	2	2	2	2	2
1	1	1	2	2	2	2	2	2	1
1	1	1	2	2	2	2	2	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	2	1	1	1	1
1	1	1	1	1	1	1	1	1	1

6. Menghitung nilai *prior weights*, *mean*, dan *variance*.

Untuk menghitung nilai *prior weghts* maka jumlah data dari masing-masing cluster dibagi dengan total dari cluster. Kemudian untuk menghitung nilai *mean* maka total data *gray* pada tiap cluster dijumlah kemudian dibagi dengan jumlah tiap cluster. Kemudian terakhir menghitung nilai *variance* tiap cluster menggunakan persamaan (3.1).

$$\sigma^2 = \sum \frac{(x-\mu)^2}{N} \quad (3.1)$$

Keterangan persamaan (3.1) :

- σ^2 : *Variance*
- x : Data
- μ : Rata-rata(*mean*)
- N : Total data

Prior weights cluster 1(ω_1) = 40 / 100 = 0.4
Prior weights cluster 2(ω_2) = 52 / 100 = 0.52
Prior weights cluster 3(ω_3) = 8 / 100 = 0.08

Mean cluster 1(μ_1) = ($X_{t=1}$(1,0)+ $X_{t=1}$(4,0)+…+ $X_{t=1}$(9,9)- μ_1) / 40
Mean cluster 1(μ_1) =(59+ 58+…+ 46) / 40
Mean cluster 1(μ_1) = 2121 / 40 = 53.025
Mean cluster 2(μ_2) = 3794 / 52 = 72.96154
Mean cluster 3(μ_3) = 1064 / 8 = 133

Variance cluster 1($\sigma^2_{i=1,t=1}$) = (($X_{t=1}$(1,0)- μ_1)+ ($X_{t=1}$(4,0)- μ_1)+…+ ($X_{t=1}$(9,9)- μ_1)) / 40
Variance cluster 1($\sigma^2_{i=1,t=1}$) =(35.700625+24.750625+…+49.350625) / 40
Variance cluster 1 ($\sigma^2_{i=1,t=1}$) = 1756.975 / 40 = 43.924375
Variance cluster 2($\sigma^2_{i=2,t=1}$) = 2133.923 / 52 = 41.03698225
Variance cluster 3($\sigma^2_{i=3,t=1}$) = 2770 / 8 = 346.25

3.2.2.2 Perhitungan manual GMM

Proses perhitungan GMM bertujuan untuk menghitung nilai dari distribusi gaussian. Untuk menghitung distribusi gaussian maka memerlukan beberapa variabel seperti *prior weights*, *mean* dan *variance* dari hasil proses perhitungan KMEANS. Mengacu pada persamaan(2.5) yang menyebutkan tentang perhitungan matrik kovarian(Σ) dengan mengalikan nilai variance(σ^2) dengan matrik indentitas(I), karena nilai variance(σ^2) berupa nilai skalar maka nilai matrik indentitas(I) berdimensi 1x1, maka persamaan(2.5) ketika digunakan pada cluster 1 dan frame 1 :

$$\Sigma_{i=1,t=1} = \sigma^2_{i=1,t=1} \cdot 1$$

$$\Sigma_{i=1,t=1} = \sigma^2_{i=1,t=1}$$

$\Sigma_{i=1,t=1}$ = 43.924375

Berikut adalah tahapan-tahapan proses perhitungan manual GMM :

1. Menghitung nilai *probability density fuction*.
   Untuk menghitung nilai *probability density function* pada masing-masing cluster maka digunakan persamaan (2.4) Tabel 3.5 menunjukkan hasil perhitungan *probability density fuction* pada cluster 1.

   Perhitungan *probability density function*(η) cluster 1 indeks(0,0) :

   $\eta(X_{t=1}, \mu_{i=1,t=1}, \Sigma_{i=1,t=1}) = \frac{1}{((2*3.14)^{1/2})(43.924375^{1/2})}$

   $\eta(X_{t=1}, \mu_{i=1,t=1}, \Sigma_{i=1,t=1}) = * 2.71^{(-\frac{1}{2}*(63-53.025)*(43.924375)^{-1}*(63-53.025))}$

   $\eta(X_{t=1}, \mu_{i=1,t=1}, \Sigma_{i=1,t=1}) = 0.19393664$

Tabel 3.5 Hasil perhitungan *probability density function* cluster 1

$\eta(X_t, \mu_{i,t}, \Sigma_{i,t})_{i=1}$									
0.0193 93664	0.0015 21448	8.67 E-05	2.78 E-05	2.18 E-06	5.3107 4E-07	4.26 E-06	1.52 E-05	8.67 E-05	0.0015 21448
0.0400 92715	0.0117 66116	0.00 2265	0.00 0402	4.97 E-05	2.7801 3E-05	2.78 E-05	0.00 0247	0.00 1521	0.0088 57043
1.1832 8E-07	2.6591 8E-15	9.94 E-20	7.52 E-27	1.45 E-33	8.2764 1E-37	1.45 E-33	5.4E-08	8.14 E-06	8.6683 7E-05
2.3316 2E-51	0.0022 64843	0.00 2265	8.67 E-05	8.67 E-05	4.8069 8E-65	4.97 E-05	0.00 0402	0.00 3296	0.0193 93664
0.0454 14921	0.0117 66116	0.00 6517	0.00 0641	4.97 E-05	1.5215 9E-05	0.00 8857	0.00 0402	0.00 1521	0.0152 78832
0.0575 80297	0.0502 8568	0.02 4063	0.00 6517	0.00 1521	0.0006 41259	0.00 0999	0.00 2265	0.01 1766	0.0400 92715
0.0116 06485	0.0087 2694	0.05 758	0.01 1766	0.00 4687	0.0022 64843	0.00 4687	0.00 6517	0.05 758	0.0595 46701
0.0087 2694	0.0398 19818	0.05 9479	0.04 5415	0.02 9183	0.0240 62583	0.05 9547	0.04 0093	0.05 4426	0.0542 4
0.0150 88711	0.0398 19818	0.05 9547	0.04 5415	0.02 9183	0.0152 78832	0.04 5415	0.02 9183	0.05 0286	0.0594 78956
0.0150 88711	0.0398 19818	0.05 9547	0.04 5415	0.02 9183	0.0152 78832	0.04 5415	0.02 9183	0.05 0286	0.0594 78956

2. Menghitung nilai *gauss probability*.
   Untuk menghitung nilai *gauss probability* maka digunakan persamaan (2.3). tabel 3.6 menunjukkan hasil perhitungan *gauss probability*.

   Perhitungan *gauss probability*(P) indeks(0,0) :

$P(X_{t=1}(0,0)) = (0.4*0.019393664)+(0.52*0.018588)+(0.08*1.8123E-05)$
$P(X_{t=1}(0,0)) = 0.017424723$

Tabel 3.6 Hasil perhitungan *gauss probability*

$P(X_t)$									
0.0174 24723	0.0315 15772	0.02 6601	0.02 0804	0.00 9534	0.0056 01113	0.01 201	0.01 7744	0.02 6601	0.0315 15772
0.0190 49906	0.0196 68142	0.03 0013	0.03 2133	0.02 381	0.0208 03938	0.02 0804	0.03 0897	0.03 1516	0.0214 87968
0.0030 14991	0.0005 52995	0.00 0963	0.00 1526	0.00 1715	0.0016 75988	0.00 1715	0.00 2151	0.01 4775	0.0266 00818
0.0009 62607	0.0300 13006	0.03 0013	0.02 6601	0.02 6601	0.0003 55918	0.02 381	0.03 2133	0.02 807	0.0174 24723
0.0202 84037	0.0196 68142	0.02 3612	0.03 2649	0.02 381	0.0177 44414	0.02 1488	0.03 2133	0.03 1516	0.0182 8552
0.0236 68002	0.0215 67482	0.01 7117	0.02 3612	0.03 1516	0.0326 49038	0.03 2428	0.03 0013	0.01 9668	0.0190 49906
0.0046 4273	0.0034 9084	0.02 3668	0.01 9668	0.02 5871	0.0300 13006	0.02 5871	0.02 3612	0.02 3668	0.0242 24229
0.0034 9084	0.0159 36754	0.02 3945	0.02 0284	0.01 734	0.0171 16938	0.02 4224	0.01 905	0.02 2743	0.0217 4862
0.0060 35769	0.0159 36754	0.02 4224	0.02 0284	0.01 734	0.0182 8552	0.02 0284	0.01 734	0.02 1567	0.0239 44933
0.0018 43258	0.0076 70251	0.01 5937	0.02 3071	0.02 3668	0.0227 43279	0.02 1567	0.02 4224	0.02 1749	0.0137 33818

3.2.2.3 Perhitungan manual *update equation*

Proses perhitungan update equation bertujuan untuk memperbarui nilai variabel GMM seperti *prior weights*, *mean*, dan *variance* pada *frame* selanjutnya. Variabel *prior weights*, *mean*, dan *variance* akan diperbarui tiap cluster. Berikut adalah tahapan-tahapan proses perhitungan manual *update equation* :

1. Membandingkan nilai pixel dengan lamda(λ).
   Pixel pada *frame* baru akan dibandingkan dengan nilai lamda(λ), untuk mendapatkan nilai lamda(λ) pada tiap cluster maka digunakan persamaan (2.10) dan (2.11). Setelah mendapatkan nilai lamda(λ) maka *pixel* baru akan dibandingkan dengan lamda(λ) tiap cluster, hasilnya akan bernilai 1 jika *pixel* tersebut lebih kecil atau sama dengan lamda(λ) dan bernilai 0 jika *pixel* lebih besar dari lamda(λ). Tabel 3.7 menunjukkan hasil perbandingan *pixel* dengan lamda(λ) pada cluster 1.

   λ cluster 1 = 2.5 * $\sqrt{43.924375}$ = 16.568867
   λ cluster 2 = 2.5 * $\sqrt{41.03698225}$ = 16.01503

λ cluster 3 = 2.5 * $\sqrt{346.25}$ = 46.51948516

hasil perbandingan akan bernilai 1 jika nilai pixel lebih kecil atau sama dengan nilai λ dan akan bernilai 0 jika nilai pixel lebih besar dari nilai λ.

Tabel 3.7 Hasil perbandingan pixel dengan lamda(λ) cluster 1

pixel dibandingkan dengan $\lambda_{i=1}$									
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

2. Memperbarui variabel *prior weights*(ω).

   Nilai *prior weights*(ω) diperbarui menggunakan persamaan (2.6). Hasil dari perbaruan nilai *prior weights*(ω) bergantung pada hasil dari perbandingan *pixel* dengan lamda(λ), jika nilai lamda(λ) 1 maka *pixel* tersebut diaggap "match" dan jika nilai lamda(λ) 0 maka *pixel* dianggap tidak "match". Tabel 3.8 menunjukkan hasil *update prior weights*(ω) cluster 1.

   Perhitungan *update prior weights*(ω) indeks(0,0) cluster 1 (tidak "match") :
   ω = (1-0.2) * 0.4 + 0.2 * 0 = 0.32

   Perhitungan *update prior weights*(ω) indeks(5,0) cluster 1 ("match") :
   ω = (1-0.2) * 0.4 + 0.2 * 1 = 0.52

Tabel 3.8 Hasil *update prior weights*(ω) cluster 1

$\omega_{k=1,t=2}$									
0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32
0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32
0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32
0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32
0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32
0.52	0.52	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32
0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32
0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32
0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32
0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32	0.32

3. Memperbarui variabel *mean*(μ).
   Nilai variabel *mean*(μ) diperbarui dengan menggunakan persamaan (2.7) dan (2.9). Hasil perbaruan variabel *mean*(μ) bergantung pada hasil perhitungan variabel lamda(λ). Tabel 3.9 menunjukkan hasil *update mean*(μ) cluster 1.

   Perhitungan ro(ρ) pada indeks(0,0) cluster 1 :
   ρ = (0.2 / 0.32) * (0.4 * 0.019393664 / 0.017424723) = 0.278249244

   Perhitugan *mean*(μ) pada indeks(0,0) cluster 1 :
   μ = (1-0.2) * 53.025 + 0.278249244 * 62 = 59.67145316

Tabel 3.9 Hasil *update mean*(μ) cluster 1

$\mu_{k=1,t=2}$									
59.671453	43.27689	42.48191504	42.44706	42.42474	42.42201483	42.42735	42.43694	42.48273	43.27689491
73.463069	52.14127	43.74058598	42.65477	42.46119	42.44639291	42.44639	42.56974	43.27689	49.22107188
42.420873	42.42	42.42	42.42	42.42	42.42	42.42	42.42054	42.43116	42.48272971
42.42	43.74059	43.74058598	42.48273	42.48436	42.42	42.46119	42.65164	44.44523	59.67145316
108.46899	60.81564	47.04310381	42.77845	42.46119	42.43715012	49.22107	42.65164	43.26483	55.78911992
53.025	53.025	63.15519874	47.31911	43.28896	42.78335788	42.97454	43.74059	52.14127	73.46306931
60.54447	61.79464	74.04682937	52.29083	45.50019	43.7594515	45.50019	47.0431	76.47966	75.60497529
65.544575	68.65546	74.71185951	74.88476	68.08643	64.20953088	75.60498	73.46307	75.92254	73.59439136
67.418819	71.77873	74.99043871	74.88476	68.08643	55.78911992	75.4445	67.66567	75.6446	74.71185951
65.544815	68.04303	71.15407684	74.16938	75.26325	76.52080007	75.6446	76.21951	73.59439	71.16028295

4. Memperbarui variabel *variance*(σ^2).
   Nilai variabel *variance*(σ^2) diperbarui dengan menggunakan persamaan (2.8). Hasil perbaruan variabel *variance*(σ^2) bergantung pada hasil perhitungan variabel lamda(λ) dan *mean*(μ). Tabel 3.10 menunjukkan hasil *update variance*(σ^2) cluster 1.

   Perhitungan *variance*(σ^2) pada indeks(0,0) cluster 1 :
   σ^2 = (1-0.278249244) * 43.924375 + 0.278249244 * (62-59.67) * (62-59.67)
   = 33.21115453

Tabel 3.10 Hasil *update variance*(σ^2) cluster 1

$\sigma^2_{k=1,t=2}$									
33.21 1155	52.6 7009	44.803 84366	44.4 0626	44.0 1582	43.966 30631	44.0 6635	44.2 0155	44.8 5922	52.670 08797
130.8 7443	62.0 8408	56.104 56303	47.0 6178	44.5 9752	44.356 09684	44.3 561	45.9 3651	52.6 7009	68.409 06369
43.94 5231	43.9 2438	43.924 375	43.9 2438	43.9 2438	43.924 375	43.9 2438	43.9 3601	44.1 2322	44.859 22497
43.92 4375	56.1 0456	56.104 56303	44.8 5922	44.9 7487	43.924 375	44.5 9752	46.8 6303	60.3 3202	33.211 15453
70.18 4878	615. 6804	68.375 30384	48.1 9343	44.5 9752	44.217 43702	68.4 0906	46.8 6303	52.0 2077	48.832 13226
43.92 4375	43.9 2438	34.555 33416	79.5 8853	53.3 4297	48.493 62182	50.0 7481	56.1 0456	62.0 8408	130.87 4433
638.3 6276	609. 1551	312.83 65462	65.4 6327	64.8 6591	57.094 82628	64.8 6591	68.3 753	272. 3018	303.78 15255
525.7 0825	460. 3119	336.97 56006	178. 9166	46.5 7229	30.203 17414	303. 7815	130. 8744	255. 1	363.63 08071
486.3 2054	400. 0152	314.10 8467	178. 9166	46.5 7229	48.832 13226	170. 7032	50.1 6769	220. 9454	336.97 56006
525.7 218	473. 517	411.72 25618	350. 77	292. 1957	245.61 99944	220. 9454	293. 6372	363. 6308	411.99 70947

5. Mengelompokkan *background* dan *foreground.*
   Setelah semua proses *update* variabel *prior weights*(ω), *mean*(μ), dan *variance*(σ^2), maka dilakukan proses pengelompokkan *background* dan *foreground*. Proses pengelompokkan *background* dan *foreground* memanfaatkan hasil *update* variabel *prior weights*(ω). Variabel *prior weights*(ω) akan diurutkan berdasarkan nilai ω/σ secara *ascending*. Setelah diurutkan maka nilai *prior weights*(ω) akan dilakukan proses perhitungan menggunakan persamaan (2.12). Tabel 3.11 menunjukkan hasil pengelompokkan *background* dan *foreground*.

   ω/σ indeks(0,0) cluster 1 = 0.32 / 33.21115453 = 0.009635317
   ω/σ indeks(0,0) cluster 2 = 0.416 / 74.50105734 = 0.005583813
   ω/σ indeks(0,0) cluster 3 = 0.064 / 346.6729298 = 0.000184612

   nilai ω/σ indeks(0,0) tiap cluster sebelum di urutkan :
   0.009635317, 0.005583813, 0.000184612

   nilai ω/σ indeks(0,0) tiap cluster setelah di urutkan :
   0.000184612, 0.005583813, 0.009635317

   Nilai *prior weights*(ω) indeks(0,0) tiap cluster sebelum diurutkan :
   0.32, 0.416, 0.064

Nilai *prior weights*(ω) indeks(0,0) tiap cluster setelah diurutkan berdasarkan nilai ω/σ :
0.064, 0.416, 0.32

Proses pengelompokkan *background* dan *foreground* menggunakan nilai *prior weights*(ω) indeks(0,0) tiap cluster yang sudah diurutkan berdasarkan nilai ω/σ :
T = 0.7
Jika 0.064 > 0.7 maka nilai B = 1,
Jika (0.064+0.416) > 0.7 maka nilai B = 2,
Jika (0.064+0.416+0.32) > 0.7 maka nilai B = 3.
Dari 3 kondisi diatas nilai minimum yang memenuhi adalah kondisi ke-3 maka nilai B pada indeks(0,0) adalah 3.

Tabel 3.11 Hasil pengelompokkan *background* dan *foreground*

B									
3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3
2	2	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3

6. Merubah hasil pengelompokkan *background* dan *foreground* menjadi bentuk biner.
   Hasil dari pengelompokkan *background* dan *foreground* dirubah menjadi bentuk biner (bernilai 1 dan 0). Cara merubah ke bentuk biner yaitu dengan membandingkan nilai semua indeks dengan indeks(0,0), jika nilai indeks tersebut sama dengan indeks(0,0) maka hasilnya 0 dan jika tidak sama dengan indeks(0,0) maka hasilnya 1. Tabel 3.12 menunjukkan hasil perubahan ke bentuk biner.

Tabel 3.12 Hasil perubahan biner

Biner									
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0

Tabel 3.12 Hasil perubahan biner (lanjutan)

Biner									
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

3.2.2.4 Perhitungan manual ekstraksi fitur

Proses perhitungan ekstraksi fitur bertujuan untuk mendapatkan nilai fitur dari objek *foreground*. Perhitungan ekstraksi fitur bergantung pada hasil biner pada proses perhitungan *update equation*. Berikut adalah tahapan-tahapan proses perhitungan manual ekstraksi fitur :

1. Menghitung nilai titik pusat.
   Mencari nilai titik pusat(x_0,y_0) menggunakan persamaan (2.21) dan (2.22). Tabel 3.13 menunjukkan hasil perhitungan m_{10}.

   m_{10} indeks(0,0) = (0^1) * (0^0) * 0(karena pada koordinat ini nilai binernya 0)
   m_{10} indeks(0,0) = 0
   m_{10} indeks(5,0) = (5^1) * (0^0) * 252(karena pada koordinat ini nilai binernya 1)
   m_{10} indeks(0,0) = 1260

   x_0 = 2520 / 504 = 5
   y_0 = 252 / 504 = 0.5

Tabel 3.13 Hasil perhitungan m_{10}

(x,y)	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	1260	1260	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
m_{10}										

2. Melakukan ekstraksi fitur.
   Setelah mendapatkan titik pusat yaitu x_0 dan y_0, maka ekstraksi fitur dapat dilakukan dengan menggunakan persamaan persamaan (2.13), (2.14), (2.15),

(2.16), (2.17), (2.18), (2.19), dan (2.20). Tabel 3.14 menunjukkan hasil perhitungan $\mu_{0,2}$.

$\mu_{0,2}$ indeks(0,0) = $(0-5)^0 * (0-0.5)^2 * 0$(koordinat ini nilai binernya 0)
= 0
$\mu_{0,2}$ indeks(5,0) = $(5-5)^0 * (0-0.5)^2 * 252$(koordinat ini nilai binernya 1)
= 63

$\phi_1 = 171$
$\phi_2 = 15876$
$\phi_3 = 0$
$\phi_4 = 0$
$\phi_5 = 0$
$\phi_6 = 0$
$\phi_7 = 0$

Tabel 3.14 Hasil perhitungan $\mu_{0,2}$

(x,y)	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	63	63	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
$\mu_{0,2}$										

3.2.2.5 Perhitungan manual klasifikasi kendaraan

Proses perhitungan klasifikasi kendaraan bertujuan untuk mengelompokkan hasil ekstraksi fitur menjadi dua jenis kelompok yaitu motor dan mobil. Untuk dapat membedakan antara mobil dengan motor maka diperlukan data training mobil dan motor. Berikut adalah tahapan proses perhitungan manual klasifikasi kendaraan :

1. Menghitung 7-D *euclidean features space*.
   7-D euclidean features space merupakan perhitungan selisih hasil ekstraksi fitur dengan data training. Masing-masing data training motor dan mobil akan dihitung nilai selisihnya dengan hasil ekstraksi fitur. Persamaan (2.21) digunakan untuk menghitung 7-D *euclidean featrures space*.

   Perhitungan selisih training motor dengan hasil ekstraksi fitur :

$d(v,c)_1 = \sqrt{(171-0)^2 + (15876-0)^2 + (0-0)^2 + (0-0)^2 + (0-0)^2}$
$= \sqrt{+(0-0)^2 + (0-0)^2}$
$= 15876.92089$
$d(v,c)_2 = \sqrt{(171-1008)^2 + (15876-1016064)^2 + (0-0)^2 + (0-0)^2}$
$= \sqrt{(0-0)^2 + (0-0)^2 + (0-0)^2}$
$= 1000188.35$

2. Membandingkan hasil 7-D *euclidean features space*.
   Mencari nilai terkecil dari 2 hasil perhitungan 7-D *euclidean features space*.
   $d(v,c)_1 = 15876.92089$
   $d(v,c)_2 = 1000188.35$
   jika $d(v,c)_1 < d(v,c)_2$ maka hasil klasifikasi bernilai 1(motor),
   jika $d(v,c)_2 < d(v,c)_1$ maka hasil klasifikasi bernilai 2(mobil).
   Karena kondisi 1 memenuhi persyaratan maka hasil klasifikasi bernilai 1(motor).

3.2.3 Skenario pengujian

Skenario pengujian dilakukan untuk mendapatkan parameter yang dapat digunakan untuk memaksimalkan nilai akurasi yang optimal pada algoritma GMM. Skenario pengujian yang dilakukan antara lain :

1. Skenario pengujian terhadap variasi video.
2. Skenario pengujian terhadap parameter *learning rate* (α).
3. Skenario pengujian terhadap parameter *threshold* (T).
4. Skenario pengujian terhadap morfologi citra (erosi dan dilasi).

3.2.3.1 Skenario pengujian terhadap variasi video

Skenario pengujian terhadap variasi video berisi tentang penjabaran skema pengujian yang akan dilakukan terhadap variasi video. Data video akan dipecah ke dalam 10 jenis data *testing* yang memiliki nilai frame yang berbeda-beda. Tabel 3.15 menunjukkan skenario pengujian yang akan dilakukan terhadap variasi video.

Tabel 3.15 Skenario pengujian variasi video

Data video	Perhitungan sistem		Pengamatan manusia		Nilai akurasi(%)
	Jumlah motor	Jumlah mobil	Data aktual kendaraan	Kesalahan	
data_vid1_testing1					
data_vid1_testing2					
data_vid1_testing3					
data_vid1_testing4					
data_vid1_testing5					

Tabel 3.15 Skenario pengujian variasi video (lanjutan)

Data video	Perhitungan sistem		Pengamatan manusia		Nilai akurasi(%)
	Jumlah motor	Jumlah mobil	Data aktual kendaraan	Kesalahan	
data_vid2_testing1					
data_vid2_testing2					
data_vid2_testing3					
data_vid2_testing4					
data_vid2_testing5					
Rata-rata					

3.2.3.2 Skenario pengujian terhadap parameter *learning rate* (α)

Skenario pengujian terhadap parameter *learning rate* (α) berisi tentang penjabaran skema pengujian yang akan dilakukan terhadap parameter *learning rate* (α). Untuk skema pengujian parameter *learning rate* (α) akan dibagi menjadi 10 jenis variasi nilai *learning rate* (α) mulai dari 0.10 samapi 0.55. Tabel 3.16 menunjukkan skenario pengujian terhadap parameter *learning rate* (α).

Tabel 3.16 Skenario pengujian parameter *learning rate* (α)

Parameter *learning rate* (α)	Perhitungan sistem		Pengamatan manusia		Nilai akurasi(%)
	Jumlah motor	Jumlah mobil	Data aktual kendaraan	Kesalahan	
0.10					
0.15					
0.20					
0.25					
0.30					
0.35					
0.40					
0.45					
0.50					
0.55					
Rata-rata					

3.2.3.3 Skenario pengujian terhadap parameter *threshold* (T)

Skenario pengujian terhadap parameter *threshold* (T) berisi tentang penjabaran skema pengujian yang akan dilakukan terhadap parameter *threshold* (T). Untuk skema pengujian parameter *threshold* (T) akan dibagi menjadi 10 jenis variasi nilai *threshold*

(T) mulai dari 0.1 samapi 1.0. Tabel 3.17 menunjukkan skenario pengujian terhadap parameter *threshold* (T).

Tabel 3.17 Skenario pengujian parameter *threshold* (T)

Parameter *threshold* (T)	Perhitungan sistem		Pengamatan manusia		Nilai akurasi(%)
	Jumlah motor	Jumlah mobil	Data aktual kendaraan	Kesalahan	
0.1					
0.2					
0.3					
0.4					
0.5					
0.6					
0.7					
0.8					
0.9					
1.0					
Rata-rata					

3.2.3.4 Skenario pengujian terhadap morfologi citra (erosi dan dilasi)

Skenario pengujian terhadap morfologi citra (erosi dan dilasi) berisi tentang penjabaran skema pengujian yang akan dilakukan terhadap morfologi citra (erosi dan dilasi). Untuk skema pengujian morfologi citra (erosi dan dilasi) akan dibagi menjadi 10 jenis kombinasi morfologi citra (erosi dan dilasi). Tabel 3.18 menunjukkan skenario pengujian terhadap morfologi citra (erosi dan dilasi).

Tabel 3.18 Skenario pengujian morfologi citra (erosi dan dilasi)

Morfologi citra		Perhitungan sistem		Pengamatan manusia		Nilai akurasi(%)
Total erosi	Total dilasi	Jumlah motor	Jumlah mobil	Data aktual kendaraan	Kesalahan	
0	1					
0	2					
0	3					
0	4					
0	5					
1	1					
1	2					
1	3					
1	4					

Tabel 3.18 Skenario pengujian morfologi citra (erosi dan dilasi) (lanjutan)

Morfologi citra		Perhitungan sistem		Pengamatan manusia		Nilai akurasi(%)
Total erosi	Total dilasi	Jumlah motor	Jumlah mobil	Data aktual kendaraan	Kesalahan	
1	5					
Rata-rata						

3.2.4 Perancangan *user interface*

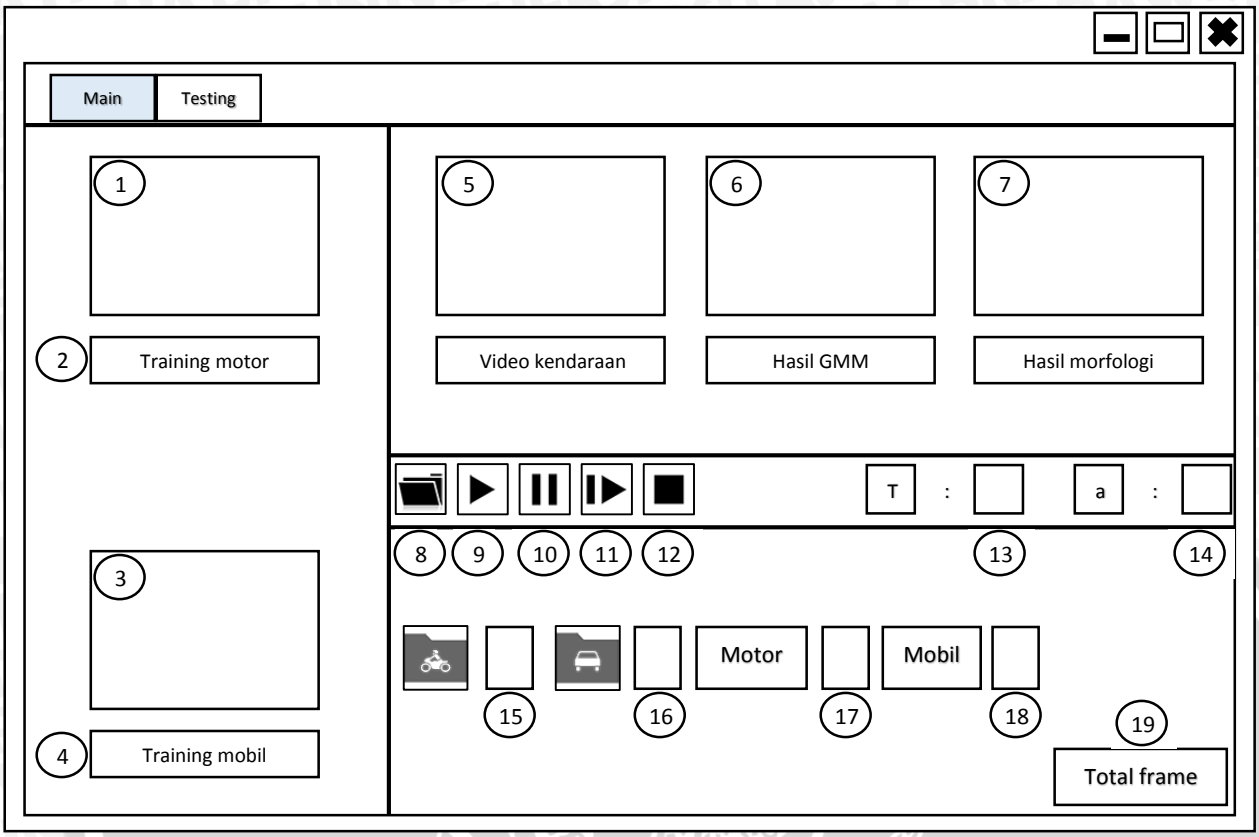
Perancangan *user interface* bertujuan untuk mewakili keadaan sebenarnya dari sistem yang akan dibangun. Sistem klasifikasi kendaraan dengan metode KMEANS-GMM terdiri dari 2 bagian *user interface* yaitu halaman *main* dan halaman *testing*. Halaman *main* merupakan tampilan utama dari sistem klasifikasi yang dibuat, sedangkan halaman *testing* merupakan tampilan yang digunakan untuk melakukan *testing* pada data kendaraan.

3.2.4.1 Perancangan *user interface* halaman *main*

Halaman *main* menampilkan hasil klasifikasi kendaraan yang berisi jumlah dari masing-masing jenis kendaraan yaitu motor dan mobil. Gambar 3.12 menunjukkan ilustrasi dari perancangan *user interface* halaman *main*.

Keterangan Gambar 3.12 :

1. *Frame* untuk menampilkan hasil dari proses training data kendaraan motor.
2. *Button* digunakan untuk melakukan proses training data kendraan motor.
3. *Frame* untuk menampilkan hasil dari proses traning data kendaraan mobil.
4. *Button* digunakan untuk melakukan proses training data kendaraan mobil.
5. *Frame* untuk menampilkan gambar berurutan dari data kendaraan di jalan raya.
6. *Frame* untuk menampilkan gambar berurutan data kendaraan di jalan raya yang sudah diproses menggunakan algorimta KMEANS-GMM.
7. *Frame* untuk menampilkan gambar berurutan data kendaraan di jalan raya yang sudah diproses menggunakan dilasi dan erosi.
8. *Button* yang berfungsi untuk membuka *file* yang berisi gambar berurutan data kendaraan di jalan raya dari direktori.

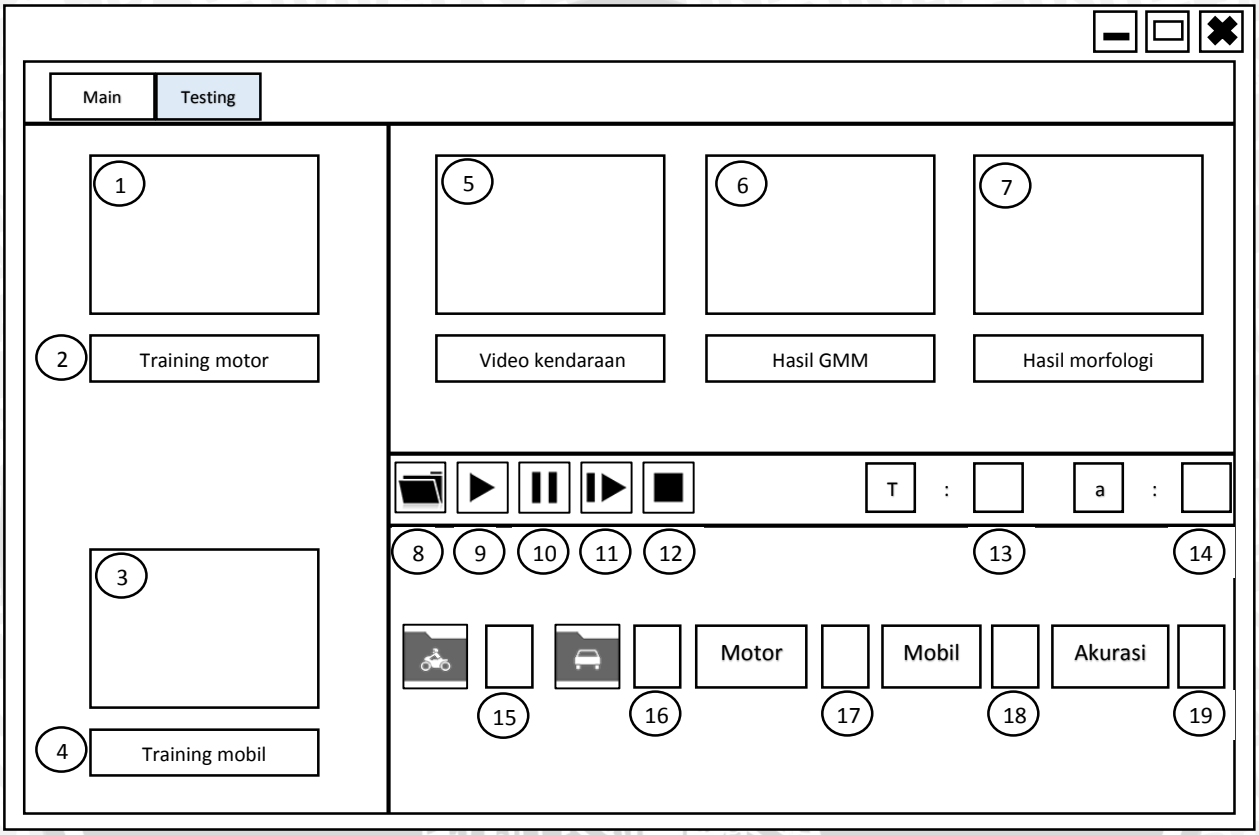


Gambar 3.12 Perancangan *user interface* halaman *main*

9. *Button* yang berfungsi untuk menjalankan gambar berurutan data kendaraan di jalan raya.
10. *Button* yang berfungsi untuk memberhentikan sementara dari gambar berurutan data kendaraan di jalan raya.
11. *Button* berfungsi untuk menjalankan kembali gambar berurutan data kendaraan di jalan raya setelah diberhentikan sementara.
12. *Button* berfungsi untuk menghentikan gambar berurutan data kendaraan di jalan raya.
13. *Textfield* untuk memasukkan angka *threshold* (T).
14. *Testfield* untuk memasukkan angka *learning rate* (α).
15. *Textfield* untuk menampilkan total angka kendaraan motor.
16. *Textfield* untuk menampilkan total angka kendaraan mobil.
17. *Textfield* untuk menampilkan data aktual kendaraan motor.
18. *Textfield* untuk menampilkan data aktual kendaraan mobil.
19. *Frame* untuk menampilkan total gambar yang sudah diproses.

3.2.4.2 Perancangan *user interface* halaman *testing*

Halaman *testing* menampilkan hasil klasifikasi kendaraan yang berisi jumlah dari masing-masing jenis kendaraan yaitu motor dan mobil beserta nilai akurasinya. Gambar 3.13 menunjukkan ilustrasi dari perancangan *user interface* halaman *testing*.



Gambar 3.13 Perancangan *user interface* halaman *testing*

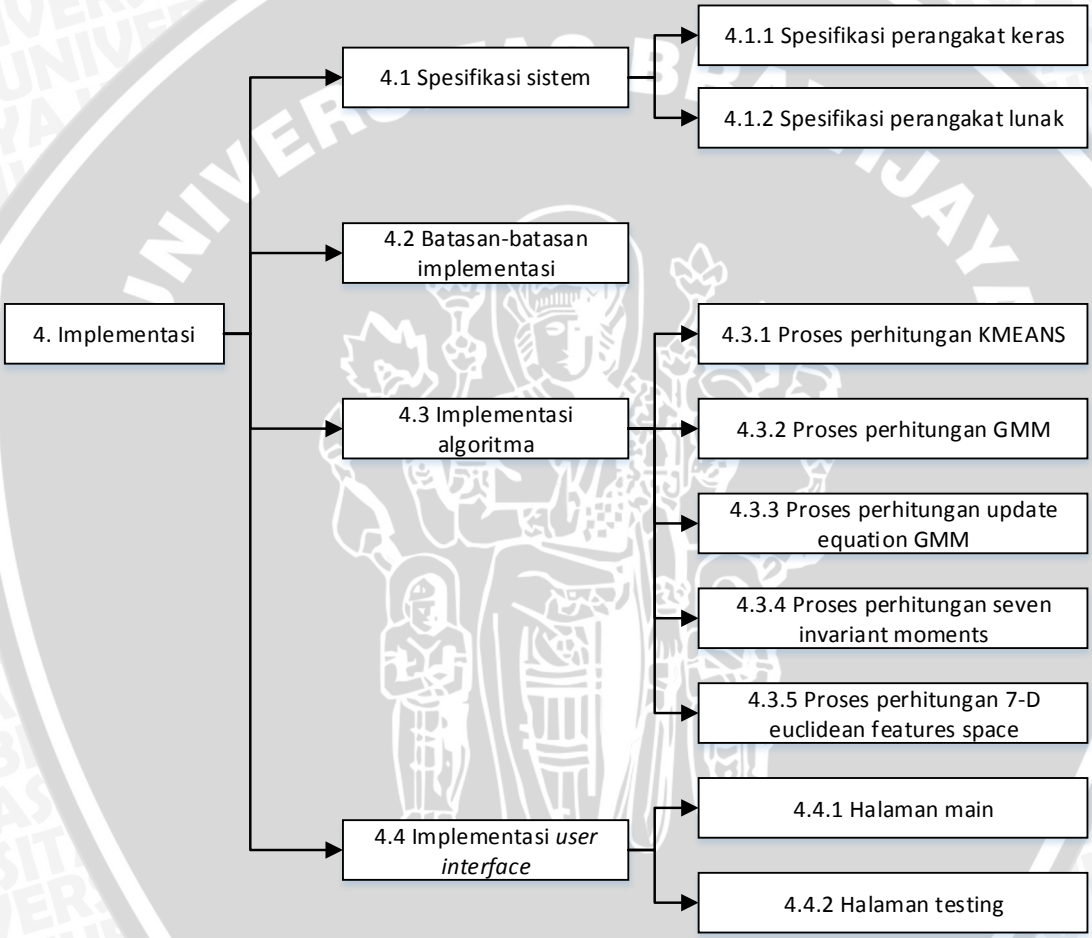
Keterangan gambar 3.13 :

1. *Frame* untuk menampilkan hasil dari proses training data kendaraan motor.
2. *Button* digunakan untuk melakukan proses training data kendraan motor.
3. *Frame* untuk menampilkan hasil dari proses traning data kendaraan mobil.
4. *Button* digunakan untuk melakukan proses training data kendaraan mobil.
5. *Frame* untuk menampilkan gambar berurutan dari data kendaraan di jalan raya.
6. *Frame* untuk menampilkan gambar berurutan data kendaraan di jalan raya yang sudah diproses menggunakan algorimta KMEANS-GMM.
7. *Frame* untuk menampilkan gambar berurutan data kendaraan di jalan raya yang sudah diproses menggunakan dilasi dan erosi.
8. *Button* yang berfungsi untuk membuka *file* yang berisi gambar berurutan data kendaraan di jalan raya dari direktori.
9. *Button* yang berfungsi untuk menjalankan gambar berurutan data kendaraan di jalan raya.

10. *Button* yang berfungsi untuk memberhentikan sementara dari gambar berurutan data kendaraan di jalan raya.
11. *Button* berfungsi untuk menjalankan kembali gambar berurutan data kendaraan di jalan raya setelah diberhentikan sementara.
12. *Button* berfungsi untuk menghentikan gambar berurutan data kendaraan di jalan raya.
13. *Textfield* untuk memasukkan angka *threshold* (T).
14. *Testfield* untuk memasukkan angka *learning rate* (α).
15. *Textfield* untuk menampilkan total angka kendaraan motor.
16. *Textfield* untuk menampilkan total angka kendaraan mobil.
17. *Textfield* untuk menampilkan data aktual kendaraan motor.
18. *Textfield* untuk menampilkan data aktual kendaraan mobil.
19. *Textfield* untuk menampilkan nilai akurasi.

BAB 4 IMPLEMENTASI

Pada bab implementasi akan dibahas mengenai implementasi perangkat lunak berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan perangkat lunak yang telah dibuat. Pembahasan terdiri dari penjelasan tentang spesifikasi sistem, batasan-batasan dalam implementasi , implementasi algoritma, dan implementasi *user interface*. Gambar 4.1 menggambarkan tentang tahap-tahap implementasi yang dilakukan mulai dari implementasi spesifikasi sistem sampai implementasi *user interface*.



Gambar 4.1 Pohon implementasi

4.1 Spesifikasi sistem

Hasil analisis kebutuhan yang telah diuraikan pada bab 3 menjadi acuan untuk melakukan implementasi menjadi sebuah sistem yang dapat berfungsi sesuai dengan

kebutuhan. Spesifikasi sistem dibagi menjadi 2 jenis yaitu spesifikasi perangkat keras dan spesifikasi perangakat lunak.

4.1.1 Spesifikasi perangkat keras

Spesifikasi perangkat keras meliputi spesifikasi minimal dari prosesor dan memori(RAM) komputer yang digunakan untuk mengimplementasikan sistem klasifikasi kendaraan. Tabel 4.1 menjabarkan tentang spesifikasi prosesor dan memori(RAM) komputer yang digunakan.

Tabel 4.1 Spesifikasi perangkat keras komputer

Nama komponen	Spesifikasi
Prosesor	Intel(R) Pentium(R) CPU B940 @2.00GHz
Memori(RAM)	2.00GB RAM

4.1.2 Spesifikasi perangkat lunak

Spesifikasi perangkat lunak meliputi spesifikasi sistem operasi dan perangkat lunak komputer yang digunakan dalam mengimplementasikan sistem klasifikasi kendaraan. Tabel 4.2 menjabarkan tentang spesifikasi sistem operasi dan perangkat lunak yang digunakan.

Tabel 4.2 Spesifikasi perangkat lunak komputer

Nama komponen	Spesifikasi
Sistem operasi	Sistem operasi Windows 8.1 Pro
Tools pemrograman	- Netbeans IDE 8.0.1 - *Java Development Kit* (JDK) 8

4.2 Batasan-batasan implementasi

Batasan-batasan dalam mengimplementasikan sistem klasifikasi kendaraan adalah sebagai berikut :

1. Sistem yang dibuat dapat menerima masukkan berupa *file folder* yang berisi data video lalu lintas yang sudah di ubah kedalam kumpulan gambar *frame* dengan format ".jpg".
2. Sistem hanya dapat melakukan klasifikasi objek kendaraan menjadi 2 jenis yaitu sepeda motor dan mobil.
3. Sistem hanya dapat melakukan klasifikasi objek kendaraan didalam area deteksi.
4. Sistem akan memberikan jumlah untuk masing-masing jenis kendaraan yaitu jumlah sepeda motor dan mobil.
5. Pembahasan ditekankan pada bagaimana mengimplemnetasikan metode KMEANS-GMM serta mengetahui tingkat kinerja metode KMEANS-GMM apabila diterapkan untuk klasifikasi kendaraan.

4.3 Implementasi algoritma

Aplikasi implementasi metode KMEANS-GMM berbasis *hu moments* untuk klasifikasi kendaraan pada video lalu lintas mempunyai beberapa proses utama. Proses-proses utama dari Aplikasi implementasi metode KMEANS-GMM berbasis *hu moments* untuk klasifikasi kendaraan pada video lalu lintas terdiri dari proses perhitungan KMEANS, proses perhitungan GMM, proses perhitungan *update equation* GMM, proses perhitungan *seven invariant moments*, dan proses perhitungan 7-D e*uclidean features space*. Implementasi pembuatan program aplikasi dibuat dengan menggunakan Bahasa pemrograman java.

4.3.1 Proses perhitungan KMEANS

Proses perhitungan KMEANS berisi pengimplementasian algoritma KMEANS ke dalam bahasa pemrograman java. Pada proses perhitungan KMEANS nilai *grayscale frame* pertama akan dihitung menggunakan algoritma KMEANS sehingga *frame* pertama akan dibagi menjadi 3 *cluster* dan menghasilkan nilai *prior weights*, *mean*, dan *variance* untuk masing-masing *cluster*. Untuk proses perhitungan KMEANS hanya dilakukan pada *frame* pertama pada kumpulan frame yang ada. Kode program 4.1 menjabarkan proses perhitungan KMEANS yang sudah diimpelentasikan dalam bentuk bahasa pemrograman java.

```
1   public void KMeans(int[][] src) {
2           double centroid1 = dataimg[0][0];
3           double centroid2 = dataimg[0][1];
4           double centroid3 = dataimg[0][2];

5           double totalvar1 = 0.0;
6           double totalvar2 = 0.0;
7           double totalvar3 = 0.0;
8           int countvar1 = 0;
9           int countvar2 = 0;
10          int countvar3 = 0;

11          int countw1 = 0;
12          int countw3 = 0;
13          int countw2 = 0;

14          double[][] C1 = new double[src.length][src.length];
15          double[][] C2 = new double[src.length][src.length];
16          double[][] C3 = new double[src.length][src.length];

17          int[][] matrikcek = new int[src.length][src.length];

18          int totalcek = (int) Math.pow(src.length, 2);

19          boolean iterasi = true;
```

```
20          for (int i = 0; i < src.length; i++) {
21              for (int j = 0; j < src.length; j++) {
22                  C1[i][j]  =  Math.sqrt(Math.pow((dataimg[i][j]
    - centroid1), 2));
23                  C2[i][j]  =  Math.sqrt(Math.pow((dataimg[i][j]
    - centroid2), 2));
24                  C3[i][j]  =  Math.sqrt(Math.pow((dataimg[i][j]
    - centroid3), 2));
25              }
26          }

27          for (int i = 0; i < src.length; i++) {
28              for (int j = 0; j < src.length; j++) {
29                  if  (C1[i][j]  <  C2[i][j]  &&  C1[i][j]  <
C3[i][j]) {
30                      kmeanpast[i][j] = 1;
31                  } else if (C2[i][j] < C3[i][j]) {
32                      kmeanpast[i][j] = 2;
33                  } else {
34                      kmeanpast[i][j] = 3;
35                  }
36              }
37          }

38          while (iterasi) {
39              int cek = 0;
40              int total1 = 0;
41              int total2 = 0;
42              int total3 = 0;
43              int count1 = 0;
44              int count2 = 0;
45              int count3 = 0;

46              for (int i = 0; i < src.length; i++) {
47                  for (int j = 0; j < src.length; j++) {
48                      if (kmeanpast[i][j] == 1) {
49                          total1 += dataimg[i][j];
50                          count1++;
51                      } else if (kmeanpast[i][j] == 2) {
52                          total2 += dataimg[i][j];
53                          count2++;
54                      } else if (kmeanpast[i][j] == 3) {
55                          total3 += dataimg[i][j];
56                          count3++;
57                      }
58                  }
59              }

60              centroid1 = (double) total1 / count1;
61              centroid2 = (double) total2 / count2;
62              centroid3 = (double) total3 / count3;

63              for (int i = 0; i < src.length; i++) {
64                  for (int j = 0; j < src.length; j++) {
```

```
65                      C1[i][j]                                        =
    Math.sqrt(Math.pow((dataimg[i][j] - centroid1), 2));
66                      C2[i][j]                                        =
    Math.sqrt(Math.pow((dataimg[i][j] - centroid2), 2));
67                      C3[i][j]                                        =
    Math.sqrt(Math.pow((dataimg[i][j] - centroid3), 2));
68                  }
69              }

70              for (int i = 0; i < src.length; i++) {
71                  for (int j = 0; j < src.length; j++) {
72                      if  (C1[i][j]  <  C2[i][j]  &&  C1[i][j]  <
    C3[i][j]) {
73                          kmeanpresent[i][j] = 1;
74                      } else if (C2[i][j] < C3[i][j]) {
75                          kmeanpresent[i][j] = 2;
76                      } else {
77                          kmeanpresent[i][j] = 3;
78                      }
79                  }
80              }

81              for (int i = 0; i < src.length; i++) {
82                  for (int j = 0; j < src.length; j++) {
83                      if          (kmeanpresent[i][j]          ==
    kmeanpast[i][j]) {
84                          matrikcek[i][j] = 1;
85                      } else {
86                          matrikcek[i][j] = 0;
87                      }
88                  }
89              }

90              for (int i = 0; i < src.length; i++) {
91                  for (int j = 0; j < src.length; j++) {
92                      if (matrikcek[i][j] == 1) {
93                          cek += 1;
94                      }
95                  }
96              }

97              for (int i = 0; i < src.length; i++) {
98                  for (int j = 0; j < src.length; j++) {
99                      kmeanpast[i][j] = kmeanpresent[i][j];
100                 }
101             }

102             mean1 = centroid1;
103             mean2 = centroid2;
104             mean3 = centroid3;

105             if (cek == totalcek) {
106                 iterasi = false;
107             }
```

```
108         }
109         for (int i = 0; i < src.length; i++) {
110             for (int j = 0; j < src.length; j++) {
111                 if (kmeanpast[i][j] == 1) {
112                     totalvar1  +=  Math.pow((dataimg[i][j]  -
    mean1), 2);
113                     countvar1++;
114                 } else if (kmeanpast[i][j] == 2) {
115                     totalvar2  +=  Math.pow((dataimg[i][j]  -
    mean2), 2);
116                     countvar2++;
117                 } else if (kmeanpast[i][j] == 3) {
118                     totalvar3  +=  Math.pow((dataimg[i][j]  -
    mean3), 2);
119                     countvar3++;
120                 }
121             }
122         }
123         varian1 = totalvar1 / countvar1;
124         varian2 = totalvar2 / countvar2;
125         varian3 = totalvar3 / countvar3;
126         for (int i = 0; i < src.length; i++) {
127             for (int j = 0; j < src.length; j++) {
128                 if (kmeanpast[i][j] == 1) {
129                     countw1++;
130                 } else if (kmeanpast[i][j] == 2) {
131                     countw2++;
132                 } else if (kmeanpast[i][j] == 3) {
133                     countw3++;
134                 }
135             }
136         }
137         w1 = (double) countw1 / totalcek;
138         w2 = (double) countw2 / totalcek;
139         w3 = (double) countw3 / totalcek;
140     }
```

Kode program 4.1 Daftar kode proses perhitungan KMEANS

Penjelasan kode program 4.1 :

1. Baris 2-4 berfungsi untuk mendapatkan nilai *centroid* dari masing-masing *cluster* di awal.
2. Baris 20-26 berfungsi untuk menghitung nilai kedekatan data *grayscale* dengan masing-masing *centroid* di awal.
3. Baris 27-37 berfungsi untuk mengelompokkan hasil perhitungan nilai kedekatan data *grayscale* dengan masing-masing *centroid* menjadi 3 *cluster* diawal.

4. Baris 38 berfungsi untuk melakukan perulangan, perulangan akan terus berlanjut sampai mendapatkan pengelompokkan *cluster* yang stabil.
5. Baris 46-62 berfungsi untuk menghitung nilai *centroid* dari masing-masing *cluster*.
6. Baris 63-69 berfungsi untuk menghitung nilai kedekatan data *grayscale* dengan masing-masing *centroid*.
7. Baris 70-80 berfungsi untuk mengelompokkan hasil perhitungan nilai kedekatan data *grayscale* dengan masing-masing *centroid* menjadi 3 *cluster*.
8. Baris 102-104 berfungsi untuk menghitung nilai *mean* pada masing-masing *cluster*.
9. Baris 109-125 berfungsi untuk menghitung nilai *variance* pada masing-masing *cluster*.
10. Baris 126-139 berfungsi untuk menghitung nilai *prior weights* pada masing-masing *cluster*.

4.3.2 Proses perhitungan GMM

Proses perhitungan GMM berisi pengimplementasian algoritma GMM ke dalam bahasa pemrograman java. Pada proses perhitungan GMM hasil perhitungan KMEANS yaitu *prior weights*, *mean*, dan *variance* digunakan untuk menghitung *probability density function* dan *gauss probability*. Untuk proses perhitungan GMM dilakukan pada frame pertama sampai frame terakhir. Kode program 4.2 menjabarkan proses perhitungan GMM yang sudah diimpelentasikan dalam bentuk bahasa pemrograman java.

```
1   public void PDF(int[][] src) {
2           for (int i = 0; i < src.length; i++) {
3               for (int j = 0; j < src.length; j++) {
4                   for (int k = 0; k < 3; k++) {
5                       if (k == 0) {
6                           double coef = 1 / ((Math.pow(2 *
    Math.PI, (1 / 2))) * (Math.sqrt(varian1)));
7                           double exp = Math.pow(Math.E, ((-0.5)
    * (dataimg[i][j] - mean1) * (1 / varian1) * (dataimg[i][j] -
    mean1)));
8                           etha1[i][j] = coef * exp;
9                       } else if (k == 1) {
10                          double coef = 1 / ((Math.pow(2 *
    Math.PI, (1 / 2))) * (Math.sqrt(varian2)));
11                          double exp = Math.pow(Math.E, ((-0.5)
    * (dataimg[i][j] - mean2) * (1 / varian2) * (dataimg[i][j] -
    mean2)));
12                          etha2[i][j] = coef * exp;
13                      } else if (k == 2) {
14                          double coef = 1 / ((Math.pow(2 *
    Math.PI, (1 / 2))) * (Math.sqrt(varian3)));
15                          double exp = Math.pow(Math.E, ((-0.5)
    * (dataimg[i][j] - mean3) * (1 / varian3) * (dataimg[i][j] -
    mean3)));
```

```
16                          etha3[i][j] = coef * exp;
17                      }
18                  }
19              }
20          }
21      }
22  public void Gaussianprobability(int[][] src) {
23          for (int i = 0; i < src.length; i++) {
24              for (int j = 0; j < src.length; j++) {
25                  gaussprob[i][j] = (w1 * etha1[i][j]) + (w2 *
    etha2[i][j]) + (w3 * etha3[i][j]);
26              }
27          }
28      }
```

Kode program 4.2 Daftar kode proses perhitungan GMM

Penjelasan kode program 4.2 :

1. Baris 1-21 berfungsi untuk menghitung nilai *probability density function*.
2. Baris 22-28 berfungsi untuk menghitung nilai *gauss probability*.

4.3.3 Proses perhitungan *update equation* GMM

Proses perhitungan *update equation* GMM berisi pengimplementasian algoritma *update equation* GMM ke dalam bahasa pemrograman java. Hasil *prior weights*, *mean*, *variance*, *probability density function*, dan *gauss probability* yang berasal dari perhitungan KMEANS dan GMM akan digunakan untuk melakukan proses *update equation* GMM. Tujuan dari proses update equation GMM adalah untuk memperbarui komponen-komponen dari GMM yaitu *prior weights*, *mean*, dan *variance* pada saat data *frame* memasuki *frame* kedua sampai frame terakhir. Kode program 4.3 menjabarkan proses perhitungan *update equation* GMM yang sudah diimpelentasikan dalam bentuk bahasa pemrograman java.

```
1   public void updateLamda(int[][] src) {
2           for (int i = 0; i < src.length; i++) {
3               for (int j = 0; j < src.length; j++) {
4                   for (int k = 0; k < 3; k++) {
5                       if (k == 0) {
6                           if    (dataimg[i][j]    <=    (2.5    *
    (Math.sqrt(variance1[i][j])))) {
7                               lamda1[i][j] = 1;
8                           } else {
9                               lamda1[i][j] = 0;
10                          }
11                      } else if (k == 1) {
12                          if    (dataimg[i][j]    <=    (2.5    *
    (Math.sqrt(variance2[i][j])))) {
13                              lamda2[i][j] = 1;
14                          } else {
15                              lamda2[i][j] = 0;
16                          }
```

```
17                      } else if (k == 2) {
18                          if    (dataimg[i][j]    <=    (2.5    *
    (Math.sqrt(variance3[i][j])))) {
19                              lamda3[i][j] = 1;
20                          } else {
21                              lamda3[i][j] = 0;
22                          }
23                      }
24                  }
25              }
26          }
27      }
28  public void UpdateEquationWeight(int[][] src, double a) {
29          for (int i = 0; i < src.length; i++) {
30              for (int j = 0; j < src.length; j++) {
31                  for (int k = 0; k < 3; k++) {
32                      if (k == 0) {
33                          if (lamda1[i][j] == 1) {
34                              weights1[i][j] = ((1 - a) * w1) +
    (a * 1);
35                          } else {
36                              weights1[i][j] = ((1 - a) * w1) +
    (a * 0);
37                          }
38                      } else if (k == 1) {
39                          if (lamda2[i][j] == 1) {
40                              weights2[i][j] = ((1 - a) * w2) +
    (a * 1);
41                          } else {
42                              weights2[i][j] = ((1 - a) * w2) +
    (a * 0);
43                          }
44                      } else if (k == 2) {
45                          if (lamda3[i][j] == 1) {
46                              weights3[i][j] = ((1 - a) * w3) +
    (a * 1);
47                          } else {
48                              weights3[i][j] = ((1 - a) * w3) +
    (a * 0);
49                          }
50                      }
51                  }
52              }
53          }
54      }
55  public void updateEquationMean(int[][] src, double a) {
56          for (int i = 0; i < src.length; i++) {
57              for (int j = 0; j < src.length; j++) {
58                  for (int k = 0; k < 3; k++) {
59                      if (k == 0) {
60                          if (lamda1[i][j] == 1) {
61                              m1[i][j] = mean1;
62                          } else {
63
```

```
                                double ro = (a / weights1[i][j])
64  * ((w1 * etha1[i][j]) / gaussprob[i][j]);
                                m1[i][j] = ((1 - a) * mean1) + (ro
65  * dataimg[i][j]);
66                          }
67                      } else if (k == 1) {
68                          if (lamda2[i][j] == 1) {
69                              m2[i][j] = mean2;
70                          } else {
                                double ro = (a / weights2[i][j])
71  * ((w2 * etha2[i][j]) / gaussprob[i][j]);
                                m2[i][j] = ((1 - a) * mean2) + (ro
72  * dataimg[i][j]);
73                          }
74                      } else if (k == 2) {
75                          if (lamda3[i][j] == 1) {
76                              m3[i][j] = mean3;
77                          } else {
                                double ro = (a / weights3[i][j])
78  * ((w3 * etha3[i][j]) / gaussprob[i][j]);
                                m3[i][j] = ((1 - a) * mean3) + (ro
79  * dataimg[i][j]);
80                          }
81                      }
82                  }
83              }
84          }
85      }
86  public void UpdateEquationVariance(int[][] src, double a) {
87          for (int i = 0; i < src.length; i++) {
88              for (int j = 0; j < src.length; j++) {
89                  for (int k = 0; k < 3; k++) {
90                      if (k == 0) {
91                          if (lamda1[i][j] == 1) {
92                              variance1[i][j] = varian1;
93                          } else {
                                double ro = (a / weights1[i][j])
94  * ((w1 * etha1[i][j]) / gaussprob[i][j]);
                                variance1[i][j] = ((1 - ro) *
    varian1) + ((ro) * (dataimg[i][j] - m1[i][j]) * (dataimg[i][j]
95  - m1[i][j]));
96                          }
97                      } else if (k == 1) {
98                          if (lamda2[i][j] == 1) {
99                              variance2[i][j] = varian2;
100                         } else {
                                double ro = (a / weights2[i][j])
101 * ((w2 * etha2[i][j]) / gaussprob[i][j]);
                                variance2[i][j] = ((1 - ro) *
    varian2) + ((ro) * (dataimg[i][j] - m2[i][j]) * (dataimg[i][j]
102 - m2[i][j]));
103                         }
104                     } else if (k == 2) {
105                         if (lamda3[i][j] == 1) {
```

```
106                             variance3[i][j] = varian3;
107                         } else {
                                double ro = (a / weights3[i][j])
108 * ((w3 * etha3[i][j]) / gaussprob[i][j]);
                                variance3[i][j] = ((1 - ro) *
    varian3) + ((ro) * (dataimg[i][j] - m3[i][j]) * (dataimg[i][j]
109 - m3[i][j]));
110                         }
111                     }
112                 }
113             }
114         }
115     }
116 public void ClasifyBackground(int[][] src, double T) {
117         double[] temporary = new double[3];
118         double[] sort = new double[3];
119         double[] w = new double[3];
120         double[] iterasiw = new double[3];
121         double[] eliminasi = new double[3];
122         for (int i = 0; i < src.length; i++) {
123             for (int j = 0; j < src.length; j++) {
124                 for (int k = 0; k < 3; k++) {
125                     if (k == 0) {
                            temporary[k] = weights1[i][j] /
126 variance1[i][j];
127                     } else if (k == 1) {
                            temporary[k] = weights2[i][j] /
128 variance2[i][j];
129                     } else if (k == 2) {
                            temporary[k] = weights3[i][j] /
130 variance3[i][j];
131                     }
132                 }
133                 sort = Sorting(temporary);
134                 for (int k = 0; k < sort.length; k++) {
                        if (sort[k] == (weights1[i][j] /
135 variance1[i][j])) {
136                         w[k] = weights1[i][j];
                        } else if (sort[k] == (weights2[i][j] /
137 variance2[i][j])) {
138                         w[k] = weights2[i][j];
                        } else if (sort[k] == (weights3[i][j] /
139 variance3[i][j])) {
140                         w[k] = weights3[i][j];
141                     }
142                 }
143                 iterasiw = Tambah(w);
144                 eliminasi = Elimination(iterasiw, T);
145                 argmin[i][j] = Getminindex(eliminasi);
146             }
147         }
    }
```

Kode program 4.3 Daftar kode proses perhitungan *update equation* GMM

Penjelasan kode program 4.3 :

1. Baris 1-27 berfungsi untuk menghitung nilai "match" dan "tidak match" pada tiap data *pixel*, *pixel* akan diberi tanda 1 jika "match" dan 0 jika "tidak match".
2. Baris 28-54 berfungsi untuk memperbarui nilai *prior weights* dari GMM, untuk memperbarui nilai *prior weights* harus memperhatikan nilai *pixel* "match" dan "tidak match".
3. Baris 55-84 berfungsi untuk memperbarui nilai *mean* dari GMM, nilai *mean* yang diperbarui adalah nilai *mean* yang memiliki nilai pixel yang "tidak match".
4. Baris 85-114 berfungsi untuk memperbarui nilai *variance* dari GMM, nilai *variance* yang diperbarui adalah nilai *variance* yang memiliki nilai pixel yang "tidak match".
5. Baris 115-147 berfungsi untuk mengelompokkan *pixel* menjadi bagian *pixel foreground* dan *background* dengan cara membandingkan nilai pada indeks (0,0), jika nilai pada indeks yang lain sama dengan indeks (0,0) maka akan masuk kelompok *background*.

4.3.4 Proses perhitungan *seven invariant moments*

Proses perhitungan *seven invariant moments* berisi pengimplementasian algoritma *seven invariant moments* ke dalam bahasa pemrograman java. Hasil pengelompokkan *background* dan *foreground* pada perhitungan *update equation* GMM akan menghasilkan area *pixel* dari objek kendaraan. Area pixel kendaraan akan dihitung nilai titik tengah koordinatnya, kemudian setelah mendapat nilai koordinat titik tengah dari objek kendaraan maka dapat dihitung 7 jenis fitur kendaraan. Kode program 4.4 menjabarkan proses perhitungan *seven invariant moments* yang sudah diimpelentasikan dalam bentuk bahasa pemrograman java.

```
1     public void centerofobject(int[][] src) {
2             double hasil00 = 0.0;
3             double hasil01 = 0.0;
4             double hasil10 = 0.0;
5             for (int y = 0; y < src.length; y++) {
6                 for (int x = 0; x < src.length; x++) {
7                     if (hasildataimg[y][x] == 252) {
8                         hasil00    +=    ((Math.pow(x,    0))    *
      (Math.pow(y, 0)) * ((double) hasildataimg[y][x]));
9                         hasil01    +=    ((Math.pow(x,    0))    *
      (Math.pow(y, 1)) * ((double) hasildataimg[y][x]));
10                        hasil10    +=    ((Math.pow(x,    1))    *
      (Math.pow(y, 0)) * ((double) hasildataimg[y][x]));
11                    }
12                }
13            }
14            x0 = hasil10 / hasil00;
15            y0 = hasil01 / hasil00;
16        }
17    public double miupq(int p, int q, int[][] src) {
```

```
18          double hasil = 0.0;
19          for (int y = 0; y < src.length; y++) {
20              for (int x = 0; x < src.length; x++) {
21                  if (hasildataimg[y][x] == 252) {
22                      hasil += ((Math.pow((x - x0), p)) *
    (Math.pow((y - y0), q)) * ((double) hasildataimg[y][x]));
23                  }
24              }
25          }
26          return hasil;
27      }
28  public void ekstraksifitur(int x2, int x1, int y2, int y1) {
29          fitur[0] = (miupq_proses(2, 0, x2, x1, y2, y1)) +
    (miupq_proses(0, 2, x2, x1, y2, y1));
30          fitur[1] = (Math.pow((miupq_proses(2, 0, x2, x1, y2,
    y1)) - (miupq_proses(0, 2, x2, x1, y2, y1))), 2)) +
    (Math.pow((4 * miupq_proses(1, 1, x2, x1, y2, y1)), 2));
31          fitur[2] = (Math.pow((miupq_proses(3, 0, x2, x1, y2,
    y1)) - (3 * miupq_proses(1, 2, x2, x1, y2, y1))), 2)) +
    (Math.pow(((3 * miupq_proses(2, 1, x2, x1, y2, y1)) -
    (miupq_proses(0, 3, x2, x1, y2, y1))), 2));
32          fitur[3] = (Math.pow((miupq_proses(3, 0, x2, x1, y2,
    y1)) + (miupq_proses(1, 2, x2, x1, y2, y1))), 2)) +
    (Math.pow((miupq_proses(2, 1, x2, x1, y2, y1)) +
    (miupq_proses(0, 3, x2, x1, y2, y1))), 2));
33          fitur[4] = (((miupq_proses(3, 0, x2, x1, y2, y1)) -
    (miupq_proses(1, 2, x2, x1, y2, y1))) * ((miupq_proses(3, 0,
    x2, x1, y2, y1)) + (miupq_proses(1, 2, x2, x1, y2, y1))) *
    ((Math.pow(((miupq_proses(3, 0, x2, x1, y2, y1)) +
    (miupq_proses(1, 2, x2, x1, y2, y1))), 2)) - (3 *
    (Math.pow((miupq_proses(2, 1, x2, x1, y2, y1)) -
    (miupq_proses(0, 3, x2, x1, y2, y1))), 2))))) +
    (((miupq_proses(3, 0, x2, x1, y2, y1)) - (3 * (miupq_proses(1,
    2, x2, x1, y2, y1)))) * ((miupq_proses(2, 1, x2, x1, y2, y1))
    + (miupq_proses(0, 3, x2, x1, y2, y1))) * ((3 *
    (Math.pow(((miupq_proses(3, 0, x2, x1, y2, y1)) +
    (miupq_proses(1, 2, x2, x1, y2, y1))), 2))) -
    (Math.pow((miupq_proses(2, 1, x2, x1, y2, y1)) +
    (miupq_proses(0, 3, x2, x1, y2, y1))), 2))));
34          fitur[5] = (((miupq_proses(2, 0, x2, x1, y2, y1)) -
    (miupq_proses(0, 2, x2, x1, y2, y1))) *
    ((Math.pow(((miupq_proses(3, 0, x2, x1, y2, y1)) +
    (miupq_proses(1, 2, x2, x1, y2, y1))), 2)) -
    (Math.pow((miupq_proses(2, 1, x2, x1, y2, y1)) +
    (miupq_proses(0, 3, x2, x1, y2, y1))), 2)))) + ((4 *
    (miupq_proses(1, 1, x2, x1, y2, y1))) * ((miupq_proses(3, 0,
    x2, x1, y2, y1)) + (miupq_proses(1, 2, x2, x1, y2, y1))) *
    ((miupq_proses(2, 1, x2, x1, y2, y1)) + (miupq_proses(0, 3,
    x2, x1, y2, y1))));
35          fitur[6] = (((miupq_proses(2, 1, x2, x1, y2, y1)) -
    (miupq_proses(0, 3, x2, x1, y2, y1))) * ((miupq_proses(3, 0,
    x2, x1, y2, y1)) + (miupq_proses(1, 2, x2, x1, y2, y1))) *
    ((Math.pow(((miupq_proses(3, 0, x2, x1, y2, y1)) +
    (miupq_proses(1, 2, x2, x1, y2, y1))), 2)) - (3 *
```

```
        (Math.pow(((miupq_proses(2,   1,   x2,   x1,   y2,   y1))   +
        (miupq_proses(0,   3,   x2,   x1,   y2,   y1))),   2)))))   -
        (((miupq_proses(3, 0, x2, x1, y2, y1)) - (3 * (miupq_proses(1,
        2, x2, x1, y2, y1)))) * ((miupq_proses(2, 1, x2, x1, y2, y1))
        -   (miupq_proses(0,   3,   x2,   x1,   y2,   y1)))   *   ((3   *
        (Math.pow(((miupq_proses(3,   0,   x2,   x1,   y2,   y1))   +
        (miupq_proses(1,   2,   x2,   x1,   y2,   y1))),   2)))   -
        (Math.pow(((miupq_proses(2,   1,   x2,   x1,   y2,   y1))   +
        (miupq_proses(0, 3, x2, x1, y2, y1))), 2))));
36      }
```

Kode program 4.4 Daftar kode proses perhitungan *seven invariant moments*

Penjelasan kode program 4.4 :

1. Baris 1-16 berfungsi untuk menghitung nilai tengah koordinat dari objek kendaraan.
2. Baris 17-36 berfungsi untuk menghitung 7 jenis fitur kendaraan.

4.3.5 Proses perhitungan 7-D *euclidean features space*

Proses perhitungan 7-D e*uclidean features space* berisi pengimplementasian algoritma *seven invariant moments* ke dalam bahasa pemrograman java. Untuk melakukan klasifikasi kendaraan menggunakan algoritma 7-D e*uclidean features space* maka terlebih dahulu harus dilakukan proses perhitungan 7 fitur untuk masing-masing data kendaraan motor dan mobil sebagai data sampel, kemudian hasil dari perhitungan *seven invariant moments* untuk kendaraan yang sedang diproses akan dibandingkan dengan kedua data sampel. Sehingga hasil dari perhitungan 7-D e*uclidean features space* akan menghasilkan dua jenis nilai yaitu sepeda motor dan mobil, jika nilai motor lebih kecil maka objek kendaraan yang sedang di proses akan masuk ke kategori motor, dan jika nilai mobil lebih kecil maka objek kendaraan yang sedang di proses akan masuk ke kategori mobil. Kode program 4.5 menjabarkan proses perhitungan 7-D e*uclidean features space* yang sudah diimpelentasikan dalam bentuk bahasa pemrograman java.

```
1   public void klasifikasi_Kendaraan() {
2           int hitung = 0;
3           double features_f1_data1 = 0.0;
4           double features_f1_data2 = 0.0;
5           double features_f1_data3 = 0.0;
6           double features_f1_data4 = 0.0;
7           double features_f1_data5 = 0.0;
8           double features_f2_data1 = 0.0;
9           double features_f2_data2 = 0.0;
10          double features_f2_data3 = 0.0;
11          double features_f2_data4 = 0.0;
12          double features_f2_data5 = 0.0;
13          double hasil_f1_data1 = 0.0;
14          double hasil_f1_data2 = 0.0;
15          double hasil_f1_data3 = 0.0;
16          double hasil_f1_data4 = 0.0;
```

```
17          double hasil_f1_data5 = 0.0;
18          double hasil_f2_data1 = 0.0;
19          double hasil_f2_data2 = 0.0;
20          double hasil_f2_data3 = 0.0;
21          double hasil_f2_data4 = 0.0;
22          double hasil_f2_data5 = 0.0;

23          for (int i = 0; i < fitur.length; i++) {
24              features_f1_data1    +=    Math.pow((fitur[i]    -
    fitursave1_data1[i]), 2);
25              features_f1_data2    +=    Math.pow((fitur[i]    -
    fitursave1_data2[i]), 2);
26              features_f1_data3    +=    Math.pow((fitur[i]    -
    fitursave1_data3[i]), 2);
27              features_f1_data4    +=    Math.pow((fitur[i]    -
    fitursave1_data4[i]), 2);
28              features_f1_data5    +=    Math.pow((fitur[i]    -
    fitursave1_data5[i]), 2);
29              features_f2_data1    +=    Math.pow((fitur[i]    -
    fitursave2_data1[i]), 2);
30              features_f2_data2    +=    Math.pow((fitur[i]    -
    fitursave2_data2[i]), 2);
31              features_f2_data3    +=    Math.pow((fitur[i]    -
    fitursave2_data3[i]), 2);
32              features_f2_data4    +=    Math.pow((fitur[i]    -
    fitursave2_data4[i]), 2);
33              features_f2_data5    +=    Math.pow((fitur[i]    -
    fitursave2_data5[i]), 2);
34          }

35          hasil_f1_data1 = Math.sqrt(features_f1_data1);
36          hasil_f1_data2 = Math.sqrt(features_f1_data2);
37          hasil_f1_data3 = Math.sqrt(features_f1_data3);
38          hasil_f1_data4 = Math.sqrt(features_f1_data4);
39          hasil_f1_data5 = Math.sqrt(features_f1_data5);
40          hasil_f2_data1 = Math.sqrt(features_f2_data1);
41          hasil_f2_data2 = Math.sqrt(features_f2_data2);
42          hasil_f2_data3 = Math.sqrt(features_f2_data3);
43          hasil_f2_data4 = Math.sqrt(features_f2_data4);
44          hasil_f2_data5 = Math.sqrt(features_f2_data5);

45          if (hasil_f2_data1<hasil_f1_data1) {
46              hitung++;
47          } if (hasil_f2_data2<hasil_f1_data2) {
48              hitung++;
49          } if (hasil_f2_data3<hasil_f1_data3) {
50              hitung++;
51          } if (hasil_f2_data4<hasil_f1_data4) {
52              hitung++;
53          } if (hasil_f2_data5<hasil_f1_data5) {
54              hitung++;
55          }

56          if (hitung>=1) {
```

```
57                  jns_kendaraan = 2;
58              } else {
59                  jns_kendaraan = 1;
60              }
61          }
```

Kode program 4.5 Daftar kode proses perhitungan 7-D *euclidean features space*

Penjelasan kode program 4.5 :

1. Baris 23-44 berfungsi untuk menghitung jarak kedekatan antara objek kendaraan yang sedang diproses dengan data sampel objek kendaraan.
2. Baris 45-60 berfungsi untuk mengelompokkan objek kendaraan yang sedang diproses menjadi dua jenis yaitu motor dan mobil, dan juga sekaligus menghitung total dari masing kategori mobil dan motor.

4.4 Implementasi *user interface*

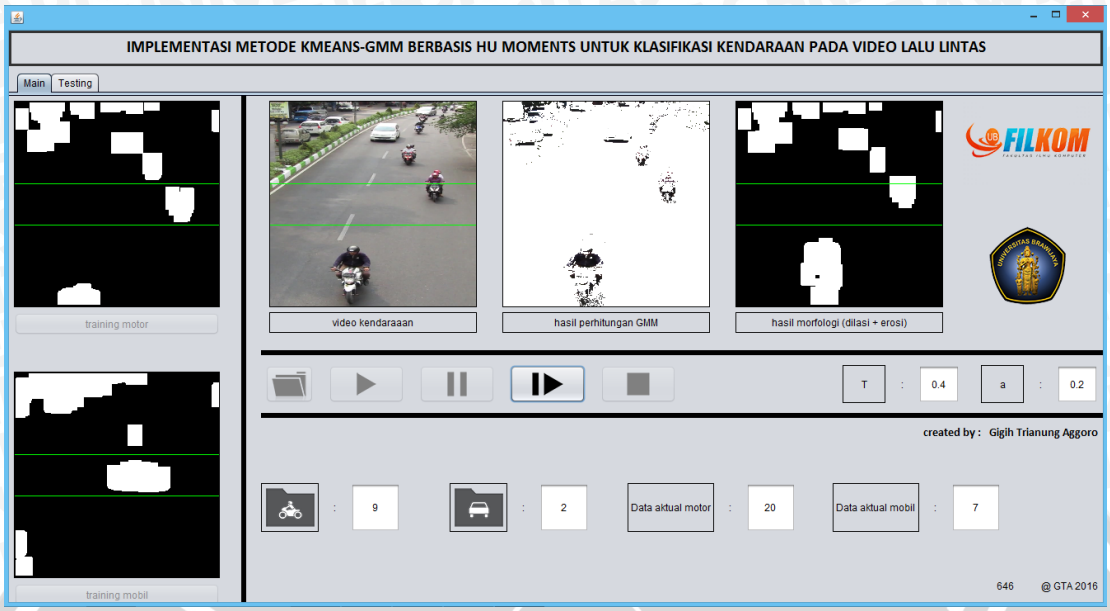
Implementasi *user interface* dari aplikasi klasifikasi kendaraan pada video lalu lintas dibuat agar mempermudah pengguna untuk berinteraksi dengan sistem. *User interface* dari aplikasi klasifikasi kendaraan pada video lalu lintas ini dibagi menjadi 2 tampilan, yaitu halaman *main* dan halaman *testing*.

4.4.1 Halaman *main*

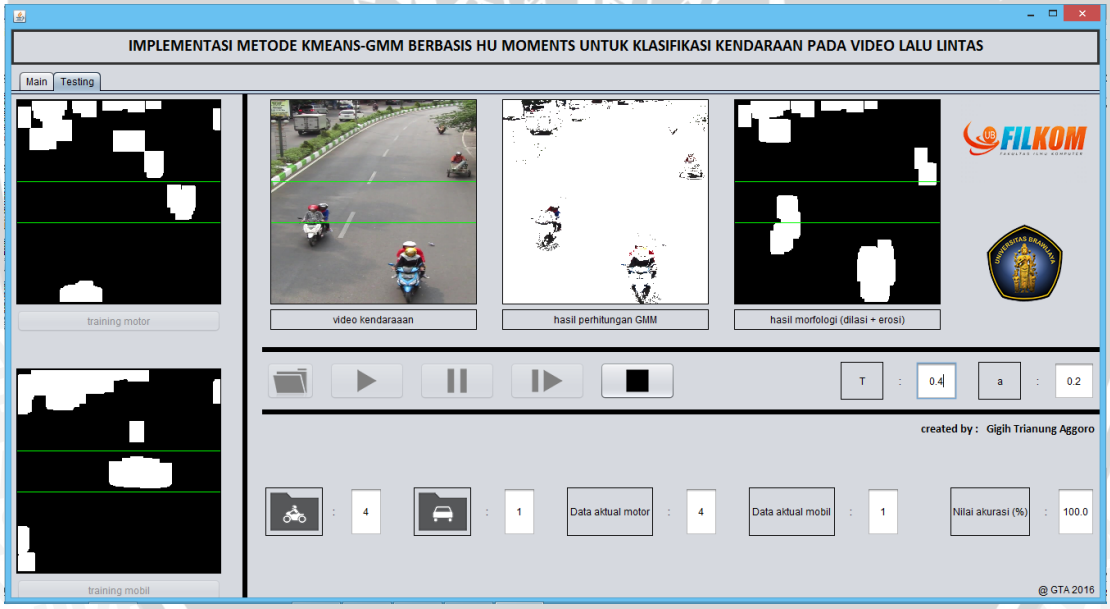
Halaman *main* berfungsi untuk menampilkan video kendaraan, hasil perhitungan GMM, hasil morfologi, proses *training* data kendaraan, jumlah motor, jumlah mobil, data atual motor, dan data aktual mobil. Pada halaman *main* terdapat panel tombol *open folder*, *play*, *pause*, *resume* dan *stop*. halaman *main* juga memiliki *textfield* yang digunakan untuk memasukkan data *threshold*(T) dan *learning rate*(a) sehingga dapat mengubah hasil dari perhitungan GMM. Gambar 4.2 merupakan gambar dari implementasi halaman *main*.

4.4.2 Halaman *testing*

Halaman *testing* berfungsi untuk menampilkan video kendaraan, hasil perhitungan GMM, hasil morfologi, proses *training* data kendaraan, jumlah motor, jumlah mobil, data aktual motor, data aktual mobil, dan nilai akurasi. Pada halaman *testing* terdapat panel tombol *open folder*, *play*, *pause*, *resume* dan *stop*. halaman *testing* juga memiliki *textfield* yang dapat digunakan untuk memasukkan data *threshold*(T) dan *learning rate*(a) sehingga dapat mengubah hasil dari perhitungan GMM. Gambar 4.3 merupakan gambar dari implementasi halaman *testing*.



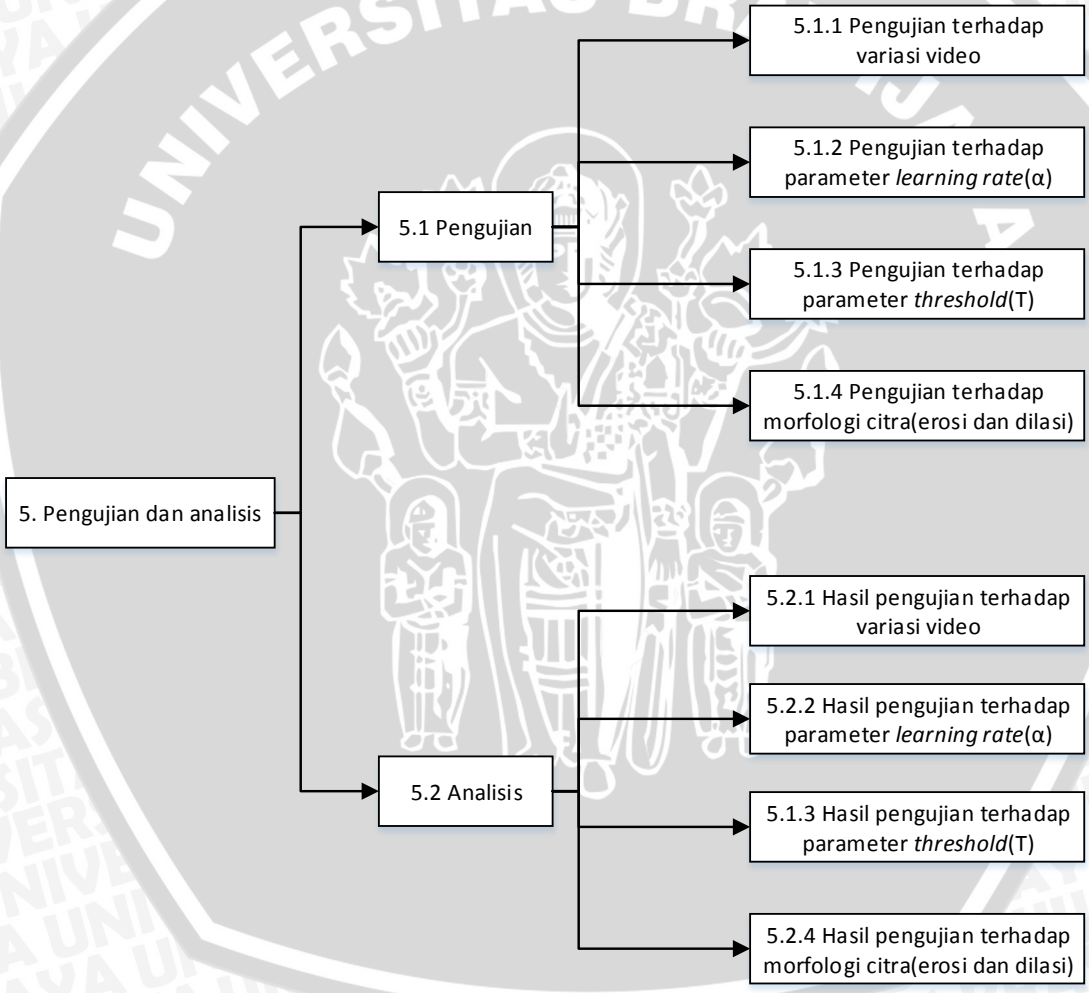
Gambar 4.2 Implementasi halaman *main*



Gambar 4.3 Implementasi halaman *testing*

BAB 5 PENGUJIAN DAN ANALISIS

Pada bab ini membahas mengenai tahapan pengujian dan analisis sistem klasifikasi kendaraan menggunakan metode KMEANS-GMM berbasis *hu moments* pada video lalu lintas. Proses pengujian dilakukan dengan cara pengujian akurasi yang dihasilkan oleh sistem dengan beberapa nilai konstanta tertentu. Untuk menghitung nilai akurasi pada sistem maka setiap kendaraan yang berhasil diklasifikasikan akan dihitung totalnya kemudian dibandingkan dengan pengamatan mata manusia. Analisis akan dilakukan setelah proses pengujian selesai dilakukan, analisis akan ditampilkan dalam sebuah grafik hasil pengujian akurasi. Gambar 5.1 menunjukkan skema urutan pengujian dan analisis yang dilakukan pada sistem klasifikasi kendaraan.



Gambar 5.1 Skema pengujian dan analisis

5.1 Pengujian

Sub bab pengujian membahas mengenai tahapan pengujian yang dilakukan pada sistem klasifikasi kendaraan menggunakan metode KMEANS-GMM berbasis *hu moments* pada video lalu lintas. Tujuan dari pengujian sistem adalah untuk mendapatkan parameter yang dapat digunakan untuk memperoleh nilai akurasi terbaik pada sistem. Tahapan pengujian yang dilakukan antara lain :

1. Pengujian terhadap variasi video.
2. Pengujian terhadap parameter *learning rate* (α).
3. Pengujian terhadap parameter *threshold* (T).
4. Pengujian terhadap morfologi citra (erosi dan dilasi).

5.1.1 Pengujian terhadap variasi video

Pengujian terhadap variasi video dilakukan untuk mengetahui data *testing* tertentu yang memiliki nilai akurasi terbaik. Data video dipecah kedalam 10 jenis data *testing* dengan nilai *frame* yang berbeda-beda. Pengujian ini bertujuan untuk mendapatkan data *testing* yang memiliki nilai paling optimal agar dapat digunakan untuk melakukan pengujian terhadap *threshold* (T), *learning rate* (α), dan morfologi citra (erosi dan dilasi). Pengujian ini juga bertujuan untuk menghitung nilai akurasi dari sistem klasifikasi kendaraan menggunakan metode KMEANS-GMM berbasis *hu moments* pada video lalu lintas. Tabel 5.1 menunjukkan hasil pengujian terhadap variasi video.

Tabel 5.1 Hasil pengujian variasi video

Data video	Perhitungan sistem		Pengamatan manusia		Nilai akurasi(%)
	Jumlah motor	Jumlah mobil	Data aktual kendaraan	Kesalahan	
data_vid1_testing1	4	1	5	0	100
data_vid1_testing2	2	1	5	0	60
data_vid1_testing3	6	0	5	1	80
data_vid1_testing4	3	2	6	0	83
data_vid1_testing5	3	3	6	0	100
data_vid2_testing1	1	2	4	1	50
data_vid2_testing2	4	0	4	0	100
data_vid2_testing3	2	2	4	0	100
data_vid2_testing4	3	1	4	0	100
data_vid2_testing5	2	1	4	0	75
Rata-rata					**84.8**

Tabel 5.1 menunjukkan hasil pengujian terhadap variasi video menggunakan perbandingan perhitungan sistem dengan pengamatan manusia. T*hreshold* (T), *learning rate* (α), dan morfologi citra (erosi dan dilasi) yang digunakan dalam pengujian ini adalah 0.4, 0.2, dan 1 kali erosi plus 4 kali dilasi. Sistem dapat mengklasifikasi jenis kendaraan beserta totalnya kemudian penglihatan manusia akan menentukan tingkat kesalahan yang dilakukan sistem sehingga mendapatkan nilai akurasi. Rata-rata akurasi yang didapat dari pengujian terhadap variasi video adalah 84.8%.

5.1.2 Pengujian terhadap parameter *learning rate* (α)

Pengujian terhadap parameter *learning rate* (α) dilakukan untuk mengetahui nilai parameter *learning rate* (α) tertentu yang memiliki nilai akurasi terbaik. Nilai parameter *learning rate* (α) yang diuji mulai dari skala angka 0.10 sampai 0.55. Data *testing* yang digunakan untuk menguji parameter *learning rate* (α) adalah *data_vid1_testing5*, karena data *testing data_vid1_testing5* merupakan salah satu data *testing* yang memiliki nilai akurasi 100%. Tabel 5.2 menunjukkan hasil pengujian terhadap parameter *learning rate* (α).

Tabel 5.2 Hasil pengujian parameter *learning rate* (α)

Parameter *learning rate* (α)	Perhitungan sistem		Pengamatan manusia		Nilai akurasi(%)
	Jumlah motor	Jumlah mobil	Data aktual kendaraan	Kesalahan	
0.10	0	0	6	0	0
0.15	4	2	6	1	83.3
0.20	3	3	6	0	100
0.25	0	0	6	0	0
0.30	0	0	6	0	0
0.35	0	0	6	0	0
0.40	0	0	6	0	0
0.45	0	0	6	0	0
0.50	0	0	6	0	0
0.55	0	0	6	0	0
Rata-rata					**18.3**

Tabel 5.2 menunjukkan hasil pengujian terhadap parameter *learning rate* (α) menggunakan perbandingan perhitungan sistem dengan pengamatan manusia. Sistem dapat mengklasifikasi jenis kendaraan beserta totalnya kemudian penglihatan manusia akan menentukan tingkat kesalahan yang dilakukan sistem sehingga mendapatkan nilai akurasi. Nilai parameter *learning rate* (α) 0.2 merupakan

parameter yang memiliki nilai akurasi tertinggi yaitu 100%. Rata-rata akurasi yang didapat dari pengujian terhadap parameter *learning rate* (α) adalah 18.3%.

5.1.3 Pengujian terhadap parameter *threshold* (T)

Pengujian terhadap parameter *threshold* (T) dilakukan untuk mengetahui nilai parameter *threshold* (T) tertentu yang memiliki nilai akurasi terbaik. Nilai parameter *threshold* (T) yang diuji mulai dari skala angka 0.1 sampai 1.0. Data *testing* yang digunakan untuk menguji parameter *threshold* (T) adalah *data_vid1_testing5*, karena data *testing data_vid1_testing5* merupakan salah satu data *testing* yang memiliki nilai akurasi 100%. Tabel 5.3 menunjukkan hasil pengujian terhadap parameter *threshold* (T).

Tabel 5.3 Hasil pengujian parameter *threshold* (T)

Parameter *threshold* (T)	Perhitungan sistem		Pengamatan manusia		Nilai akurasi(%)
	Jumlah motor	Jumlah mobil	Data aktual kendaraan	Kesalahan	
0.1	0	0	6	0	0
0.2	0	0	6	0	0
0.3	0	0	6	0	0
0.4	3	3	6	0	100
0.5	3	3	6	0	100
0.6	0	0	6	0	0
0.7	0	0	6	0	0
0.8	0	0	6	0	0
0.9	0	0	6	0	0
1.0	0	0	6	0	0
Rata-rata					**20**

Tabel 5.3 menunjukkan hasil pengujian terhadap parameter *threshold* (T) menggunakan perbandingan perhitungan sistem dengan pengamatan manusia. Sistem dapat mengklasifikasi jenis kendaraan beserta totalnya kemudian penglihatan manusia akan menentukan tingkat kesalahan yang dilakukan sistem sehingga mendapatkan nilai akurasi. Nilai parameter *threshold* (T) 0.4 dan 0.5 merupakan parameter yang memiliki nilai akurasi tertinggi yaitu 100%. Rata-rata akurasi yang didapat dari pengujian terhadap parameter *threshold* (T) adalah 20%.

5.1.4 Pengujian terhadap morfologi citra (erosi dan dilasi)

Pengujian terhadap morfologi citra (erosi dan dilasi) dilakukan untuk mengetahui kombinasi dari morfologi citra (erosi dan dilasi) yang memiliki nilai akurasi terbaik. Kombinasi morfologi citra (erosi dan dilasi) yang diuji ada 10 jenis kombinasi. Data

testing yang digunakan untuk menguji morfologi citra (erosi dan dilasi) adalah *data_vid1_testing5*, karena data *testing data_vid1_testing5* merupakan salah satu data *testing* yang memiliki nilai akurasi 100%. Tabel 5.4 menunjukkan hasil pengujian terhadap morfologi citra (erosi dan dilasi).

Tabel 5.4 Hasil pengujian morfologi citra (erosi dan dilasi)

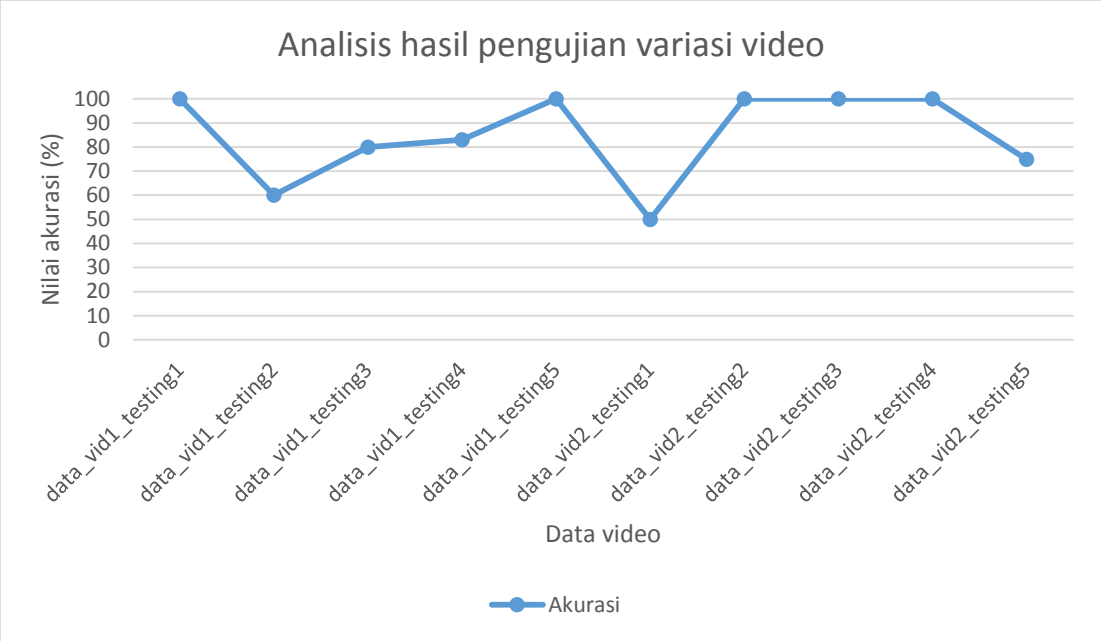
Morfologi citra		Perhitungan sistem		Pengamatan manusia		Nilai akurasi(%)
Total erosi	Total dilasi	Jumlah motor	Jumlah mobil	Data aktual kendaraan	Kesalahan	
0	**1**	9	3	6	6	0
0	**2**	9	0	6	6	0
0	**3**	7	1	6	5	16.6
0	**4**	9	0	6	6	0
0	**5**	7	0	6	5	16.6
1	**1**	7	1	6	5	16.6
1	**2**	4	2	6	3	50
1	**3**	2	3	6	1	66.6
1	**4**	5	1	6	3	50
1	**5**	3	3	6	0	100
Rata-rata						**31.6**

Tabel 5.4 menunjukkan hasil pengujian terhadap morfologi citra (erosi dan dilasi) menggunakan perbandingan perhitungan sistem dengan pengamatan manusia. Sistem dapat mengklasifikasi jenis kendaraan beserta totalnya kemudian penglihatan manusia akan menentukan tingkat kesalahan yang dilakukan sistem sehingga mendapatkan nilai akurasi. Nilai kombinasi morfologi citra (erosi dan dilasi) 1 dan 5 untuk masing-masing elemen erosi dan dilasi merupakan kombinasi yang memiliki nilai akurasi tertinggi yaitu 100%. Rata-rata akurasi yang didapat dari pengujian terhadap morfologi citra (erosi dan dilasi) adalah 31.6%.

5.2 Analisis

Sub bab analisis membahas tentang proses menganalisis dan mengambil kesimpulan berdasarkan hasil dari pengujian sistem klasifikasi kendaraan menggunakan metode KMEANS-GMM berbasis *hu moments* pada video lalu lintas. Setiap tahap pengujian yang dilakukan akan di analisis kemudian melakukan tindakan pengambilan keputusan berdasarkan hasil analisis yang dilakukan. Proses analisis yang dilakukan meliputi analisis hasil pengujian terhadap variasi video, analisis hasil pengujian terhadap parameter *learning rate* (α), analisis hasil pengujian terhadap parameter threshold (T), dan analisis hasil pengujian terhadap morfologi citra (erosi dan dilasi). Hasil dari analisis tiap pengujian akan ditampilkan dalam bentuk grafik.

5.2.1 Analisis hasil pengujian variasi video



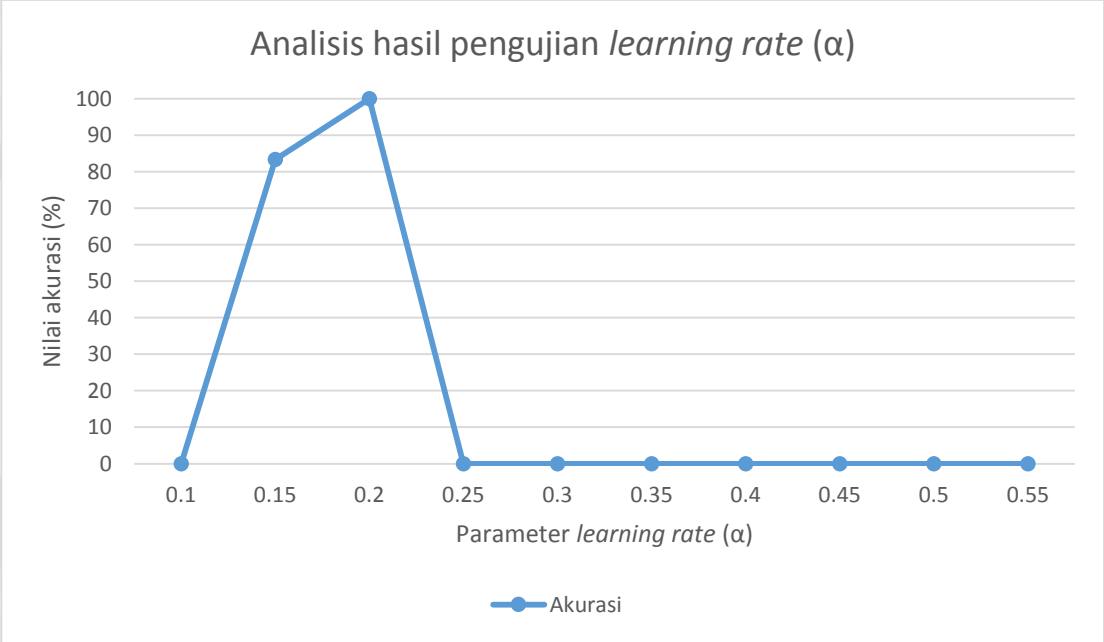
Gambar 5.2 Grafik analisis pengujian variasi video

Gambar 5.2 adalah grafik yang menjelaskan tentang hasil pengujian akurasi terhadap variasi video. Berdasarkan grafik nilai akurasi terbesar 100% berada pada *data_vid1_testing1*, *data_vid1_testing5*, *data_vid2_testing2*, *data_vid2_testing3*, dan *data_vid2_testing4,* sedangkan nilai akurasi terendah 50% berada pada *data_vid2_testing1*. Data video *testing* dapat mengklasifikasikan kendaraan hingga nilai akurasi 100% dikarenakan pada *data_vid1_testing1*, *data_vid1_testing5*, *data_vid2_testing2*, *data_vid2_testing3*, dan *data_vid2_testing4* kendaraan yang melintas tidak mengalami penumpukan artinya kendaraan melintas secara bergantian satu per satu. Sedangkan untuk data testing *data_vid2_testing1* mempunyai akurasi terendah yaitu 50%, karena pada data testing *data_vid2_testing1* kendaraan yang melintas mengalami penumpukkan atau melintas secara bersamaan yang menyebabkan nilai akurasi menjadi berkurang.

5.2.2 Analisis hasil pengujian parameter *learning rate* (α)

Gambar 5.3 adalah grafik yang menjelaskan tentang hasil pengujian akurasi terhadap parameter *learning rate* (α). Berdasarkan grafik nilai akurasi terbesar yaitu 100% berada pada angka 0.2 parameter *learning rate* (α), sedangkan nilai akurasi terkecil yaitu 0% berada pada angka 0.1, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, dan 0.55 parameter *learning rate* (α). *Learning rate* (α) merupakan parameter masukkan yang dapat mengubah nilai *prior weights* (ω), nilai *prior weights* (ω) merupakan nilai yang dapat digunakan untuk memisahkan objek *foreground* dengan objek *background*,

caranya yaitu dibandingkan dengan nilai *threshold* (T). Akurasi terbesar dari parameter *learning rate* (α) adalah 100%, hal ini dikarenakan nilai *learning rate* (α) yang dimasukkan menghasilkan nilai *prior weights* (ω) yang sesuai dengan *threshold* (T). Sedangkan untuk akurasi terendah parameter *learning rate* (α) adalah 0%, hal ini bisa terjadi karena nilai *learning rate* (α) yang dimasukkan menghasilkan nilai *prior weights* (ω) yang melebihi nilai *threshold* (T) sehingga proses pemisahan objek *foreground* dengan *background* mengalami kegagalan. Tujuan dari pengujian parameter *learning rate* (α) adalah untuk menemukan nilai parameter *learning rate* (α) yang optimal, maka berdasarkan grafik nilai parameter *learning rate* (α) yang paling optimal yang dapat digunakan untuk mengklasifikasi kendaraan adalah 0.2 dengan nilai akurasi 100%.

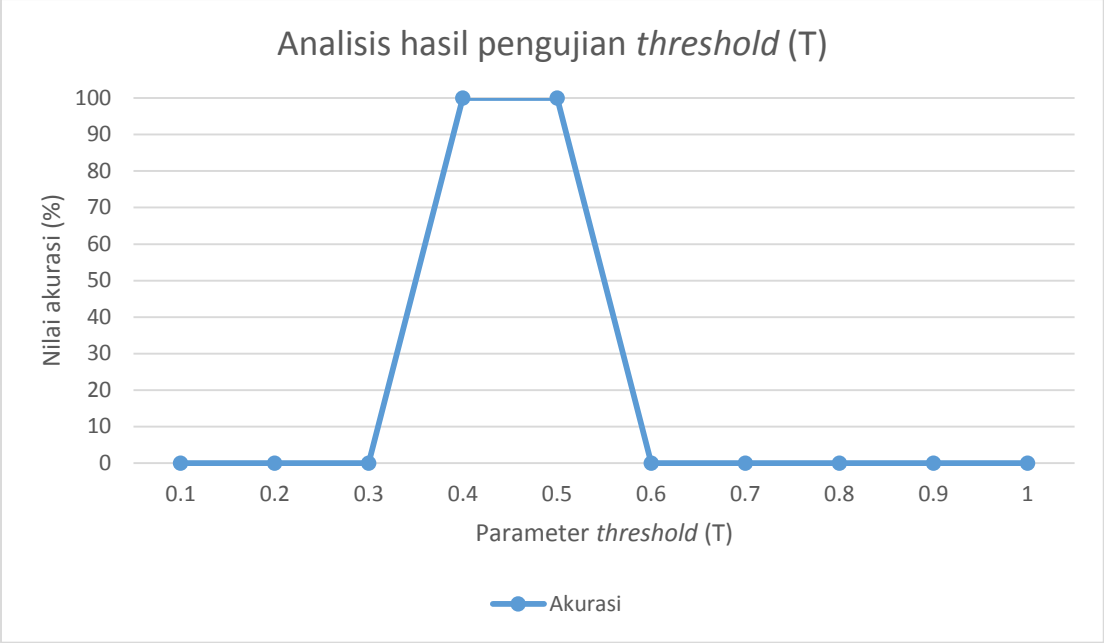


Gambar 5.3 Grafik analisis pengujian parameter learning rate (α)

5.2.3 Analisis hasil pengujian parameter *threshold* (T)

Gambar 5.4 adalah grafik yang menjelaskan tentang hasil pengujian akurasi terhadap parameter *threshold* (T). Berdasarkan grafik nilai akurasi terbesar yaitu 100% berada pada angka 0.4 dan 0.5 parameter *threshold* (T), sedangkan nilai akurasi terkecil yaitu 0% berada pada angka 0.1, 0.2, 0.3, 0.6, 0.7, 0.8, 0.9, dan 1.0 parameter *threshold* (T). *Threshold* (T) merupakan parameter masukkan yang digunakan sebagai batas pembanding nilai *prior weights* (ω) pada proses pemisahan objek *foreground* dengan *background*. Akurasi terbesar dari pengujian parameter *threshold* (T) adalah 100%, hal ini dapat terjadi karena nilai *threshold* (T) yang dimasukkan sesuai dengan nilai *prior weights* (ω) yang ada. Sedangkan untuk akurasi terendah adalah 0%, hal ini dikarenakan nilai *threshold* (T) yang dimasukkan lebih kecil ataupun lebih besar dari

nilai *prior weights* (ω) sehingga proses pemisahan objek *foreground* dengan *background* mengalami kegagalan. Tujuan dari pengujian parameter *threshold* (T) adalah untuk menemukan nilai parameter *threshold* (T) yang optimal, maka berdasarkan grafik nilai parameter *threshold* (T) yang paling optimal yang dapat digunakan untuk mengklasifikasi kendaraan adalah 0.4 dan 0.5 dengan nilai akurasi 100%.

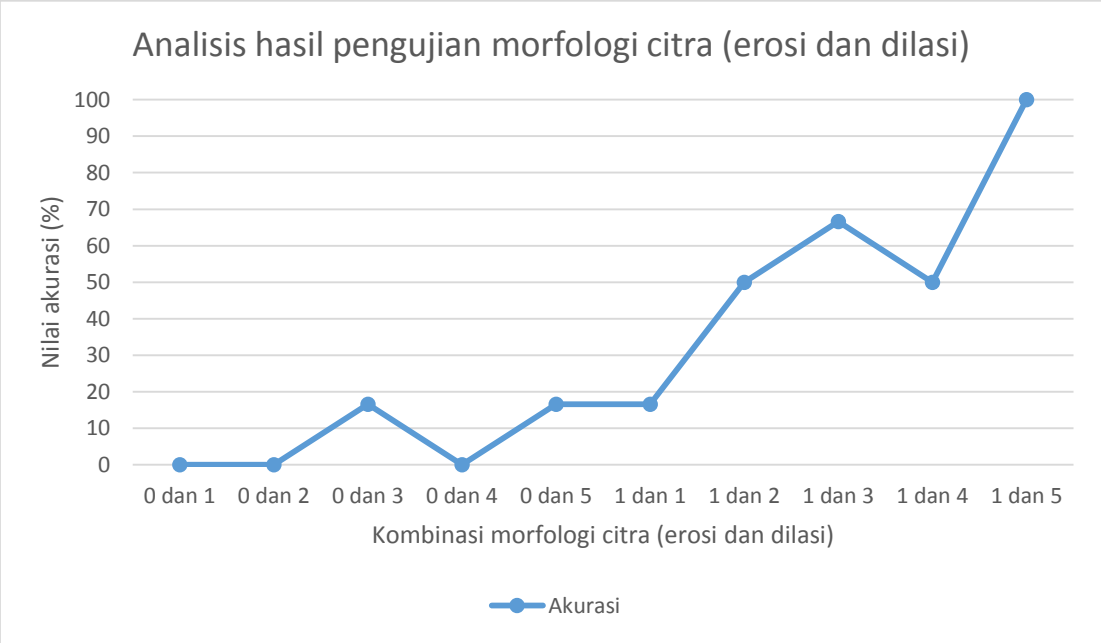


Gambar 5.4 Grafik analisis pengujian parameter *threshold* (T)

5.2.4 Analisis hasil pengujian morfologi citra (erosi dan dilasi)

Gambar 5.5 adalah grafik yang menjelaskan tentang hasil pengujian akurasi terhadap morfologi citra (erosi dan dilasi). Berdasarkan grafik nilai akurasi terbesar yaitu 100% berada pada kombinasi (1 dan 5) morfologi citra (erosi dan dilasi), sedangkan nilai akurasi terkecil yaitu 0% berada pada kombinasi (0 dan 1), (0 dan 2), (0 dan 4) morfologi citra (erosi dan dilasi). morfologi citra (erosi dan dilasi) merupakan proses untuk menghilangkan *noise* dan menggabungkan objek yang terpisah dari citra digital. . Akurasi terbesar dari pengujian morfologi citra (erosi dan dilasi) adalah 100%, hal ini dapat terjadi karena proses morfologi citra (erosi dan dilasi) yang dilakukan dapat menghilangkan *noise* dan menggabungkan objek pada citra digital secara optimal. Sedangkan untuk akurasi terendah adalah 0%, hal ini dikarenakan kombinasi morfologi citra (erosi dan dilasi) yang digunakan tidak dapat menghilangkan maupun menggabungkan objek yang terpisah sehingga proses klasifikasi kendaraan mengalami kegagalan. Tujuan dari pengujian morfologi citra (erosi dan dilasi) adalah untuk menemukan kombinasi morfologi citra (erosi dan dilasi) yang optimal, maka berdasarkan grafik kombinasi morfologi citra (erosi dan dilasi) yang paling optimal

yang dapat digunakan untuk mengklasifikasi kendaraan adalah kombinasi (1 dan 5) dengan nilai akurasi 100%.



Gambar 5.5 Grafik analisis pengujian morfologi citra (erosi dan dilasi)

BAB 6 PENUTUP

Pada bab penutup akan dibahas mengenai kesimpulan dan saran dari implementasi metode KMEANS-GMM berbasis *Hu Moments* untuk klasifikasi kendaraan pada video lalu lintas. Sub bab 6.1 dan 6.2 akan menjabarkan tentang kesimpulan dan saran.

6.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian yang telah dilakukan, maka dapat diambil kesimpulan sebagai berikut :

1. Aplikasi klasifikasi kendaraan dengan menggunakan metode KMEANS-GMM berbasis *Hu Moments* telah mampu melakukan klasifikasi pada video lalu lintas dengan baik dengan cara kerja melakukan *background subtraction* menggunakan metode KMEANS-GMM, kemudian melakukan ekstraksi fitur, dan terakhir melakukan klasifikasi dengan membandingkan kendaraan target dengan 5 data sampel kendaraan.
2. Aplikasi klasifikasi kendaraan pada video lalu lintas menggunakan metode KMEANS-GMM berbasis *Hu Moments* yang telah diimplementasikan menghasilkan nilai rata-rata akurasi sebesar 84.8% dari 10 jenis data *testing*, dengan nilai akurasi tertinggi 100% dan akurasi terendah 50%.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan pada penelitian selanjutnya, adalah sebagai berikut :

1. Penelitian selanjutnya diharapkan dapat menggunakan data video yang lebih bervariasi karena semakin banyak variasi data video maka akan semakin banyak pula variasi skema lalu lintas dan dapat melakukan pengujian yang lebih optimal dengan adanya penambahan variasi data video lalu lintas.
2. Melakukan pengembangan metode klasifikasi kendaraan dengan menggabungkan metode GMM dengan metode lain sehingga mendapatkan hasil *background subtraction* yang lebih optimal dan mampu mengatasi masalah kesalahan deteksi akibat adanya penumpukkan kendaraan.
3. Penelitian selanjutnya agar menggunakan dimensi data lebih dari 1 dimensi (nilai gray), karena pada metode GMM memungkinkan untuk dapat menggunakan dimensi data lebih dari 1 dimensi (nilai gray) seperti, 2 dimensi (warna yang sudah dinormalisasi atau *intensity-plus-range*), 3 dimensi(nilai warna *red*,*green*,*blue*), dan n dimensi(bergantung pada total n kolom vektor data yang digunakan).
4. Penelitian selanjutnya diharapkan dapat mendeteksi kendaraan yang melintas secara bersamaan, pada metode GMM tidak dijelaskan tentang cara yang dapat digunakan untuk mengatasi masalah kendaraan yang melintas secara bersamaan,

jadi agar dapat menyelesaikan masalah kendaraan yang melintas secara bersamaan maka diharapkan pada penilitian selanjutnya ditambahkan logika logika yang dibangun sendiri yang dapat mendeteksi kendaraan yang melintas secara bersamaan.

5. Penelitian selanjutnya diharapkan agar dapat membuat sebuah garis hitung kendaraan yang lebih fleksibel, yang artinya garis hitung kendaraan dapat diubah posisinya sesuai keinginan sehingga dapat menambah variasi pengujian yang dapat dilakukan.

DAFTAR PUSTAKA

Kartika, Unoviana. 2013. *Tiap Hari, Ada 300 Pengendara Ditilang di Jalur Transjakarta*. KOMPAS. Tersedia di : <http://sains.kompas.com/read/> [Diakses 11 Februari 2015]

Asaidi, H., Aarab, A., Bellouki, M., 2014. *Shadow Elimination and Vehicles Classification Approaches in Traffic Video Surveillance Context*. Elsevier. Jurnal of Visual Languages and Computing.

Meher, Saroj K., Murty M N., 2014. *Efficient Method of Moving Shadow Detection and Vehicle Classification*. Elsevier. International Journal of Electronics and Communications.

Xia, Yingjie., Shi, Xingmin., Song, Guanghua., Geng, Qiaolei., Liu, Yuncai., 2014. *Towards Improving Quality of Video-Based Vehicle Counting Method for Traffic Flow Estimation*. Elsevier. Signal Processing.

Wen, Xuezhi., Shao, Ling., Xue, Yu., Fang, Wei., 2014. *A Rapid Learning Algorithm for Vehicle Classification*. Elsevier. Information Sciences.

Chen, Thou-Ho., Lin, Yu-Feng., Chen, Tsong-Yi., 2007. *Intelligent Vehicle Counting Method Based on Blob Analysis in Traffic Surveillance*. IEEE.

Wang, Wei., Shang, Yulong., Guo, Jinzhi., Qian, Zhiwei., 2011. *Real-Time Vehicle Classification Based on Eigenface*. IEEE.

Badan Pusat Statistik Indonesia (BPS)., 2010. Tersedia di : <http://sp2010.bps.go.id/index.php> [Diakses pada 1 Januari 2015]

Badan Pusat Statistik Indonesia (BPS)., 2013. *Statistik Transportasi 2013*. Katalog BPS:06140.1401.

The World Economic Forum (TWEF)., 2013. *The Global Competitiveness Report 2013-2014*. ISBN-13: 978-92-95044-73-9.

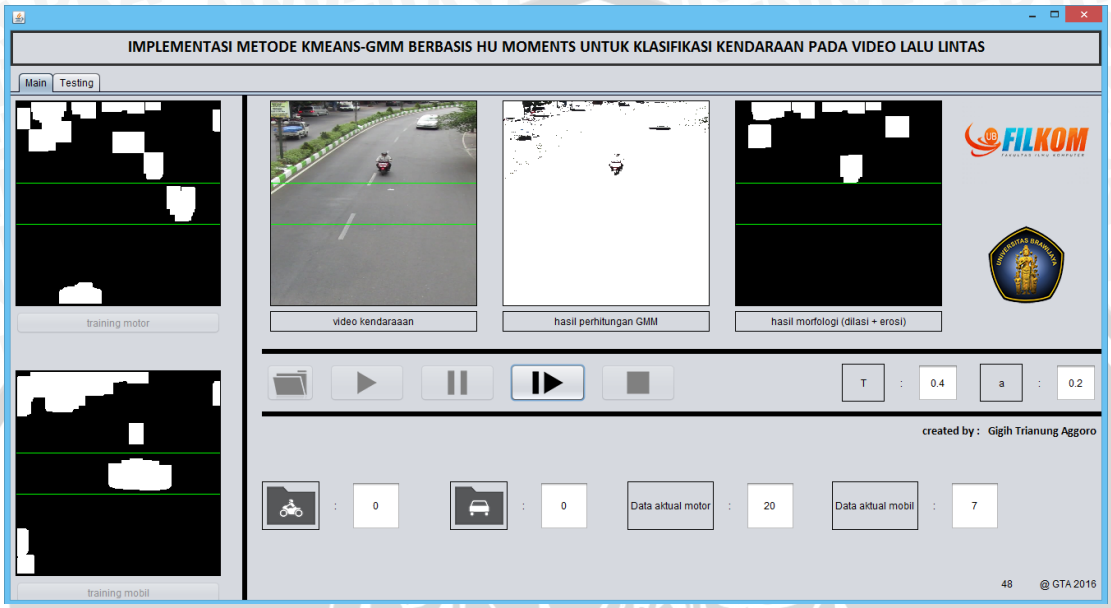
Agusta, Yudi., 2007. K-Means – Penerapan, Permasalahan dan Metode Terkait. *Jurnal Sistem dan Informatika*, Vol. 3, 47-60. Denpasar : STMIK STIKOM BALI.

Ong, Johan Oscar., 2013. Implementasi Algoritma K-MEANS Clustering untuk Menentukan Strategi Marketing President University. *Jurnal Ilmiah Teknik Industri*, Vol. 12, 1. Bekasi : President University.

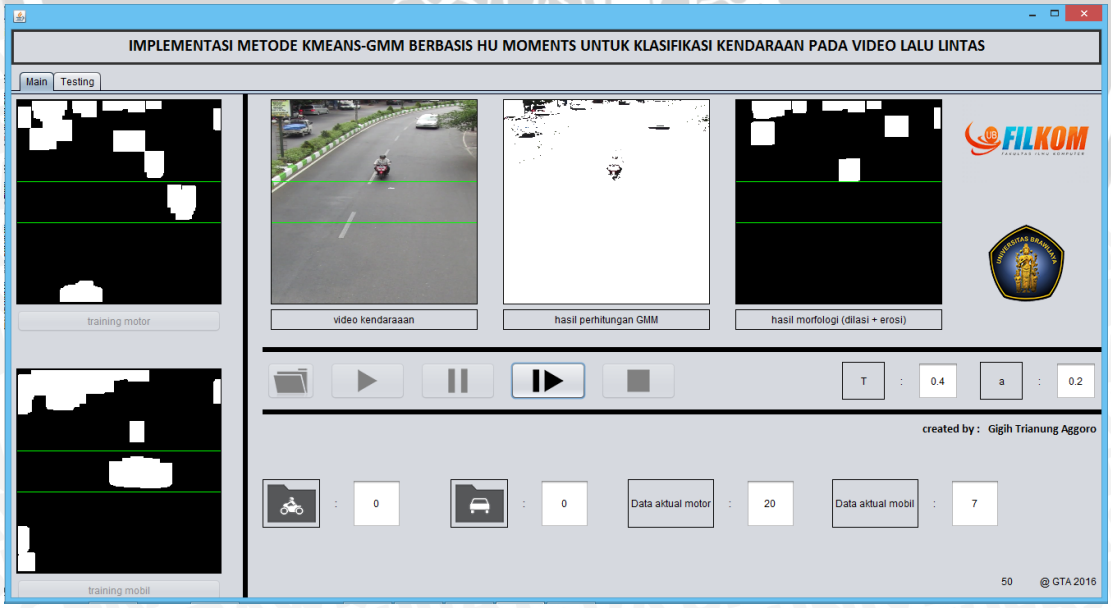
LAMPIRAN A *SCREEN CAPTURE* GUI

A.1 Gambar *step by step* deteksi sepeda motor (10 *frame*)

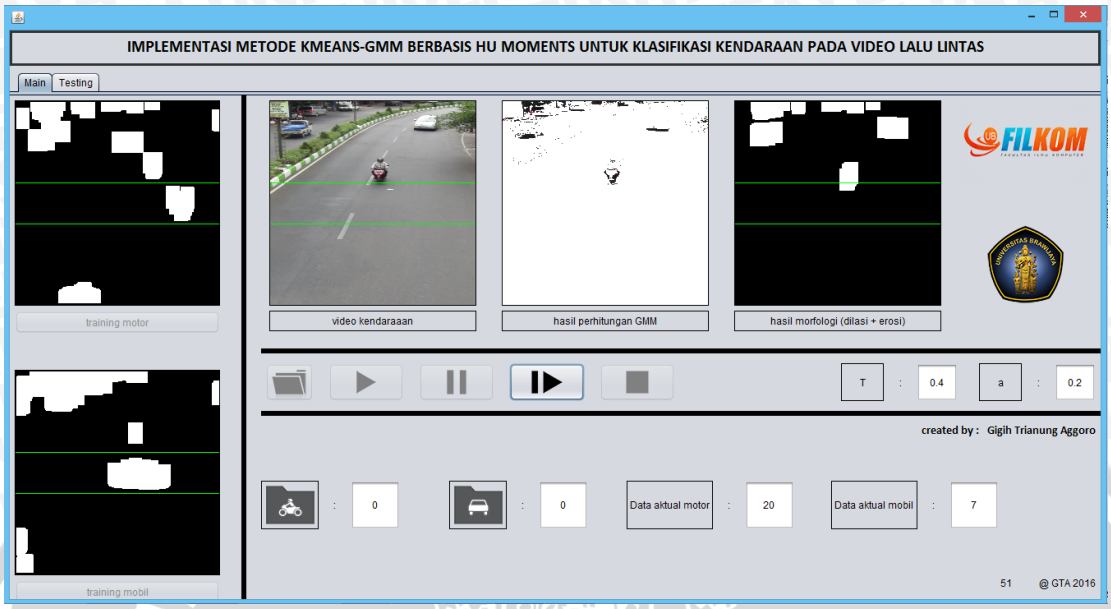
1. *Frame* 1



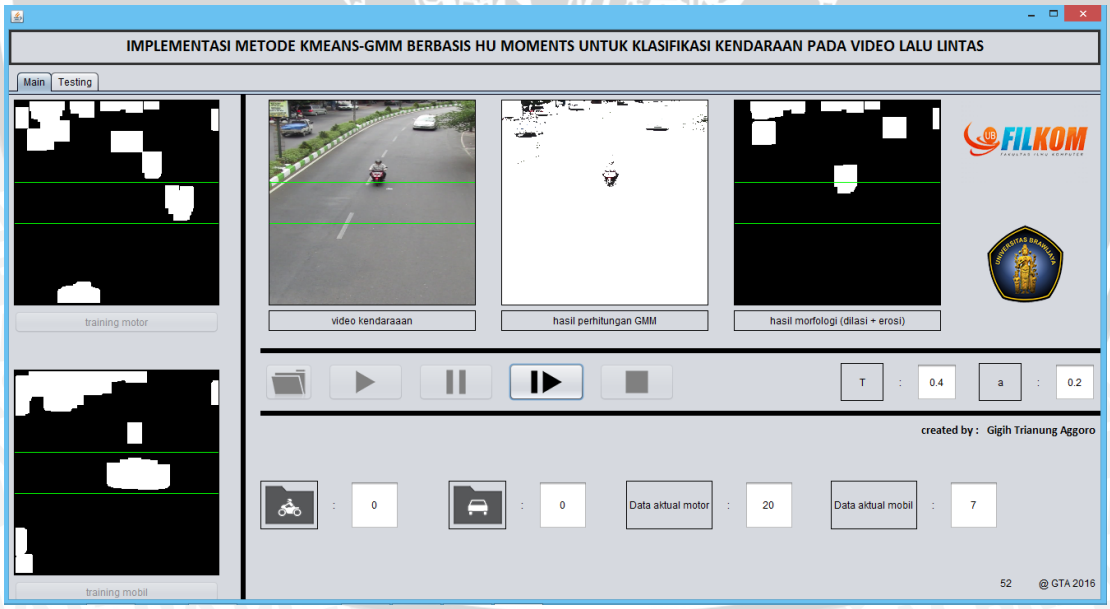
2. *Frame* 2



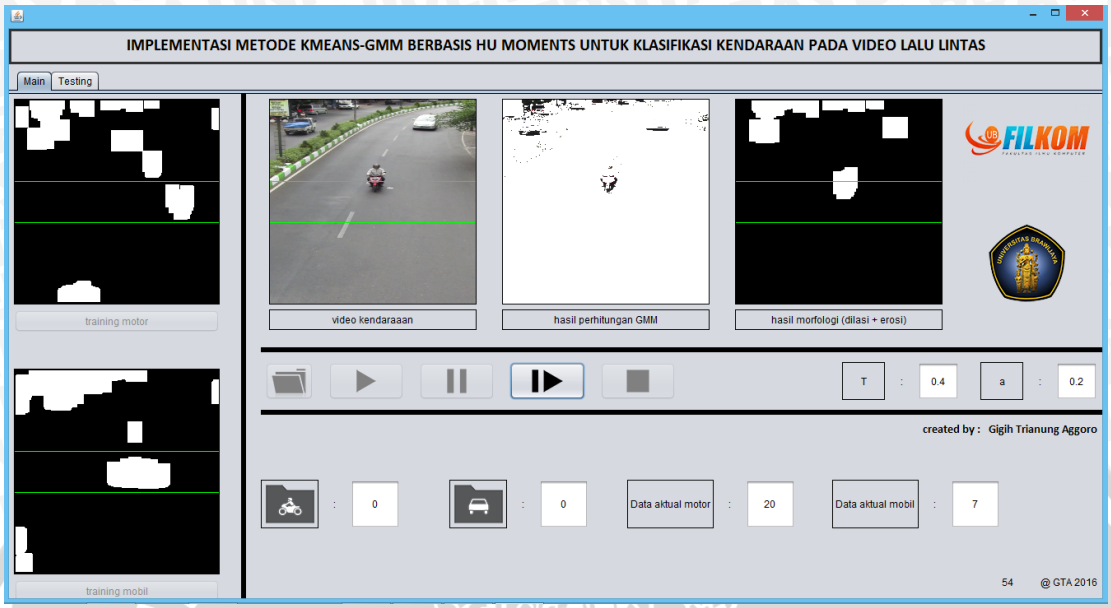
3. *Frame* 3



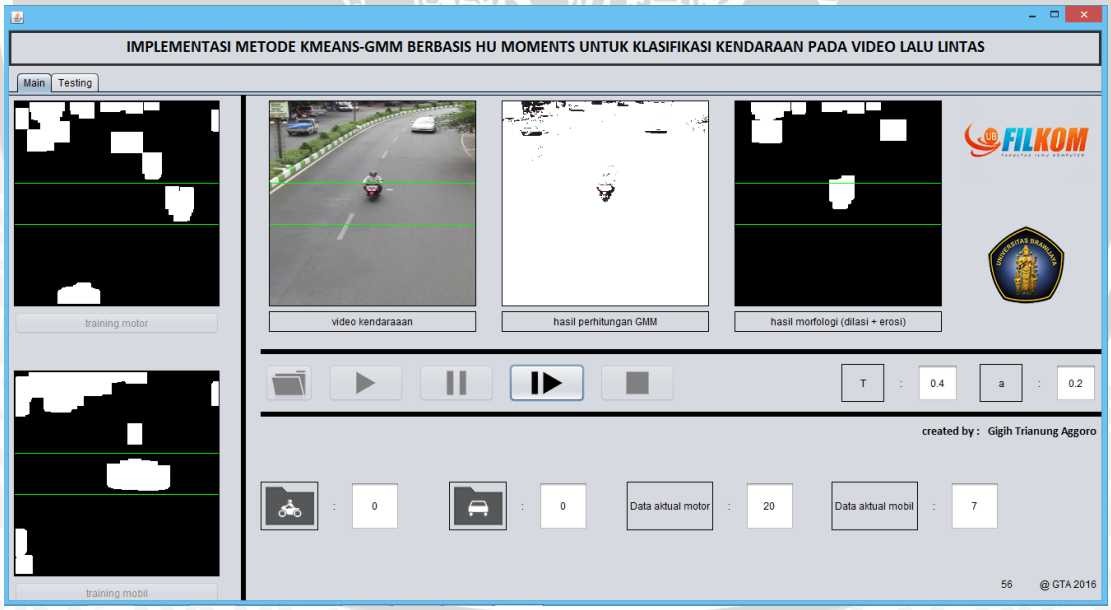
4. *Frame* 4



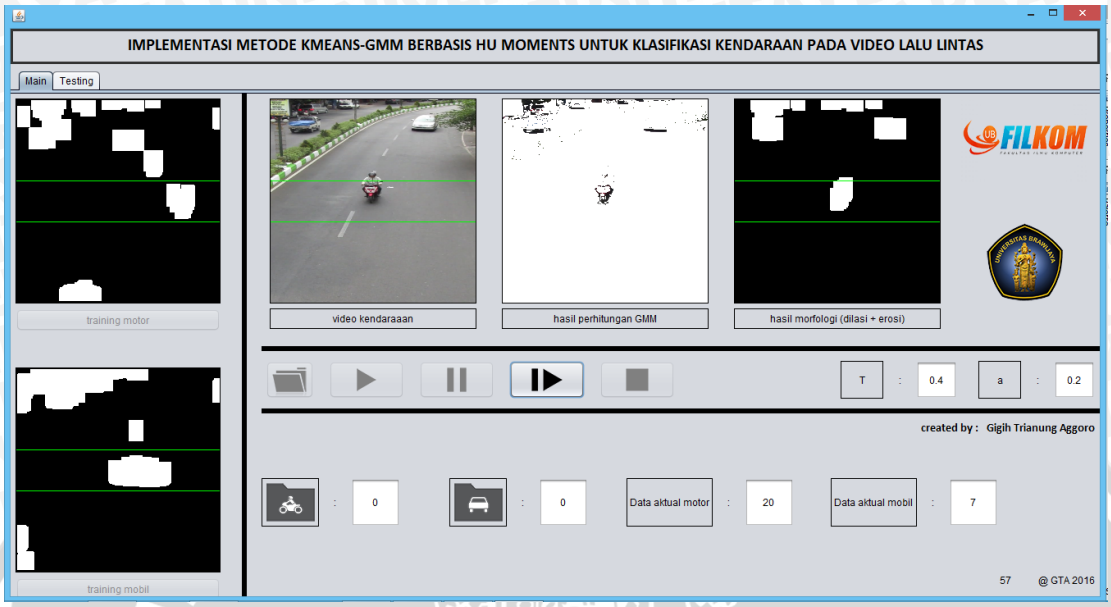
5. *Frame* 5



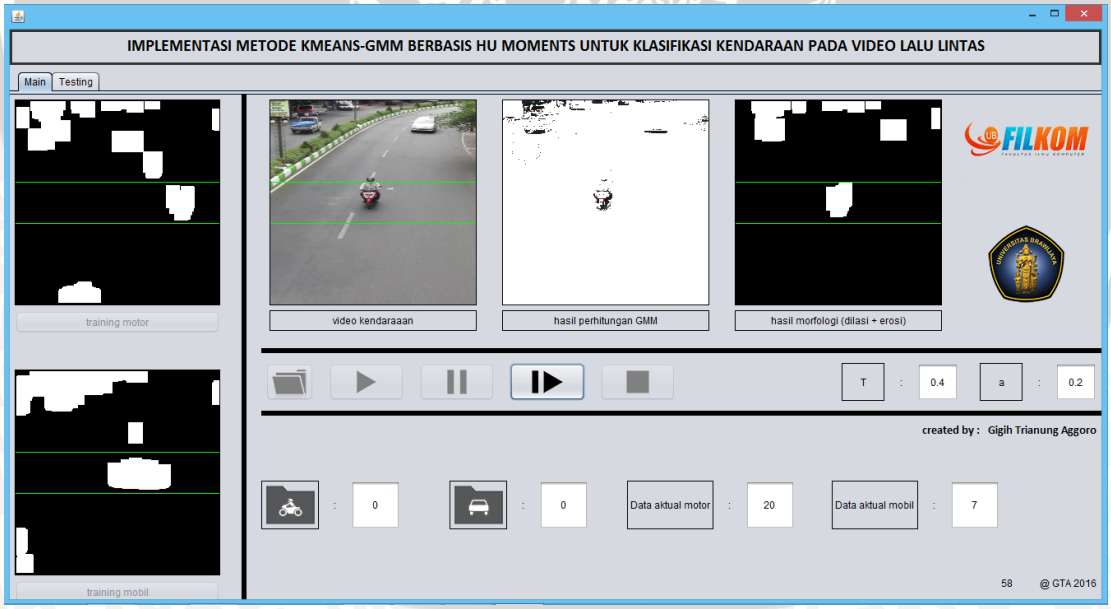
6. *Frame* 6



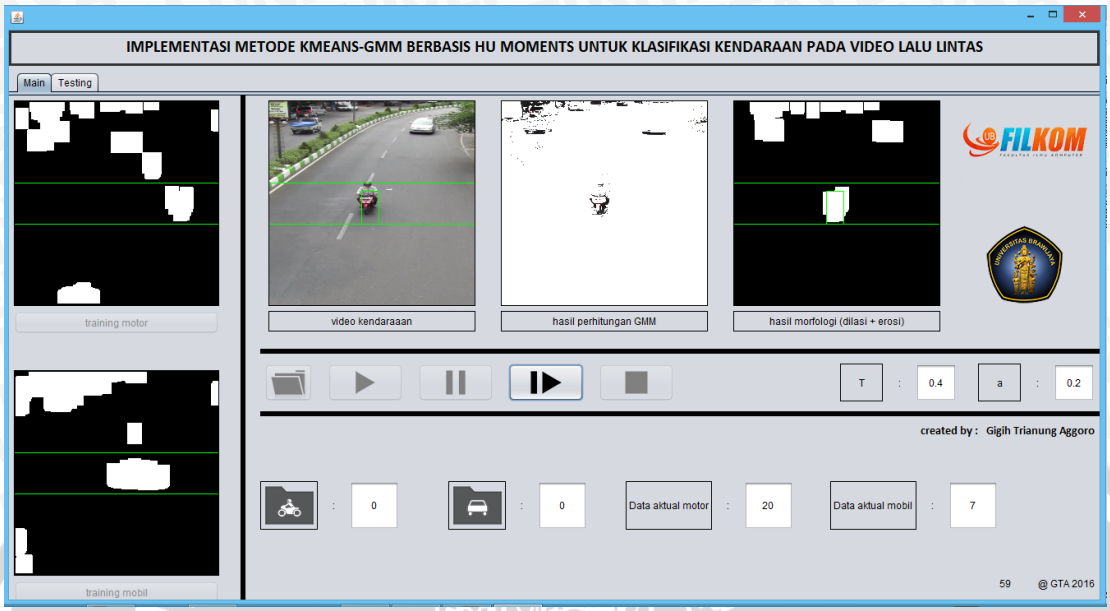
7. *Frame 7*



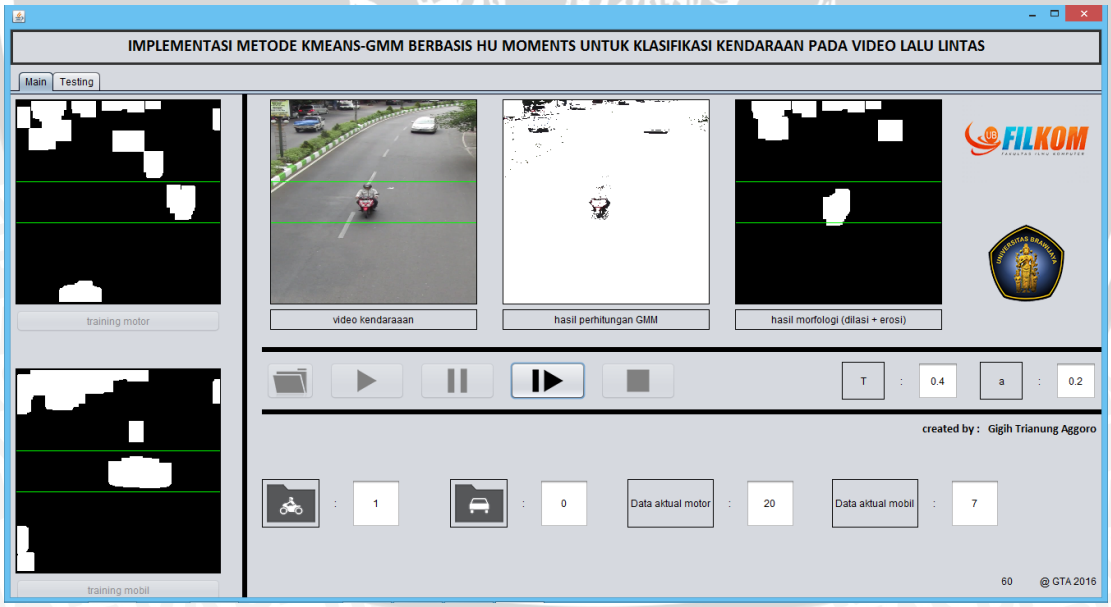
8. *Frame 8*



9. *Frame* 9

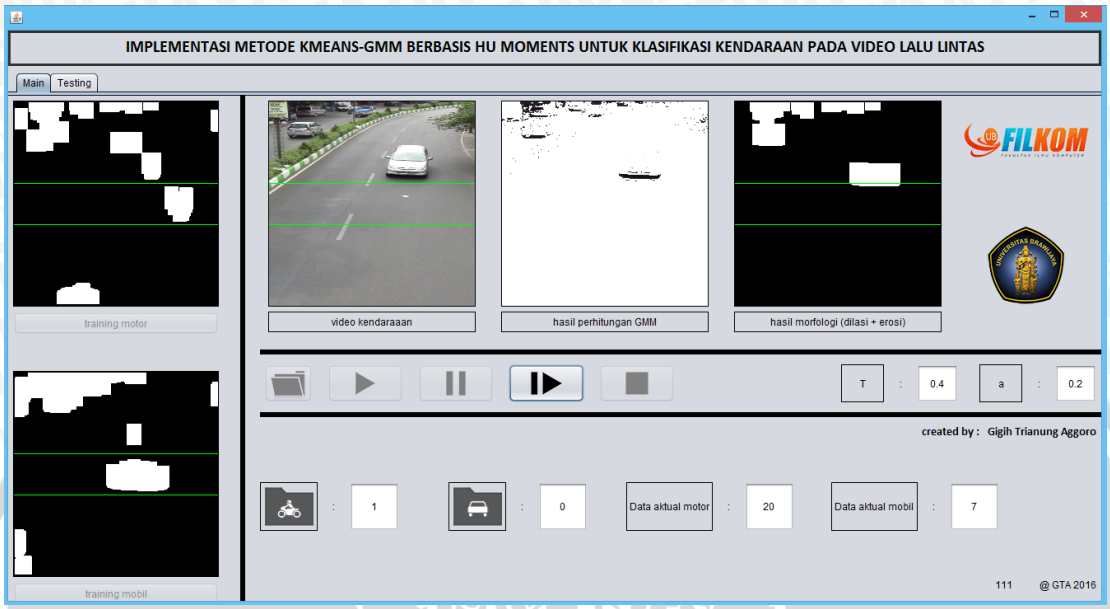


10. *Frame* 10

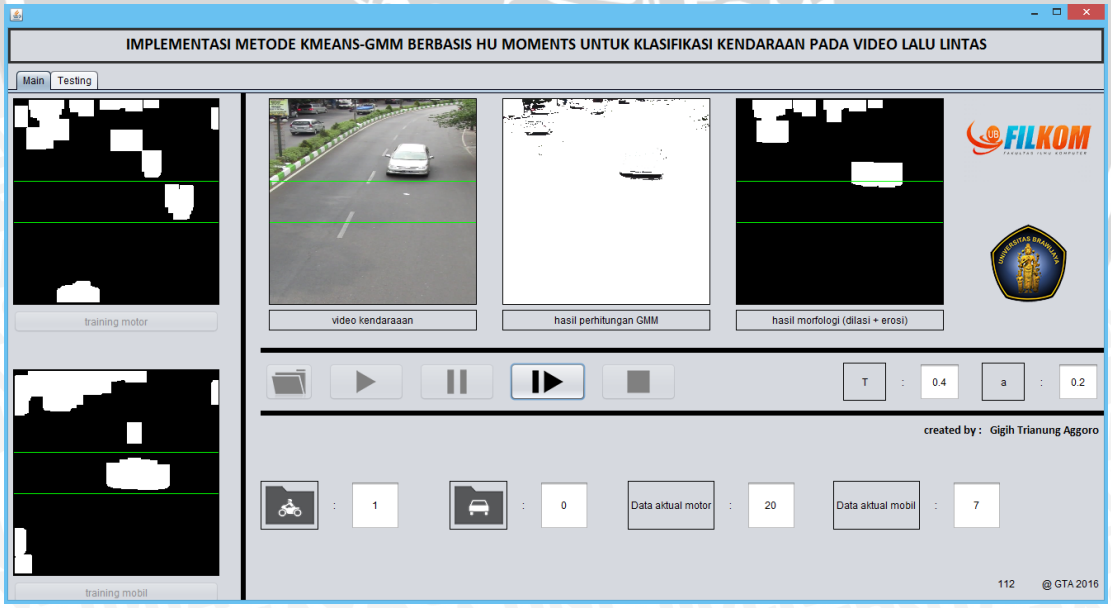


A.2 Gambar *step by step* deteksi mobil (10 *frame*)

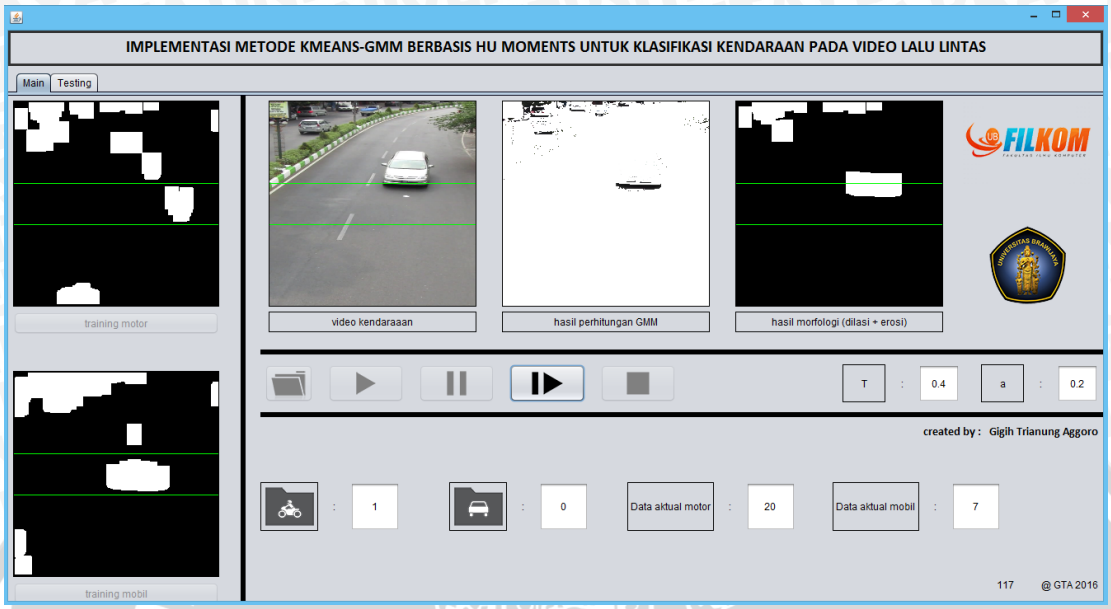
1. *Frame* 1



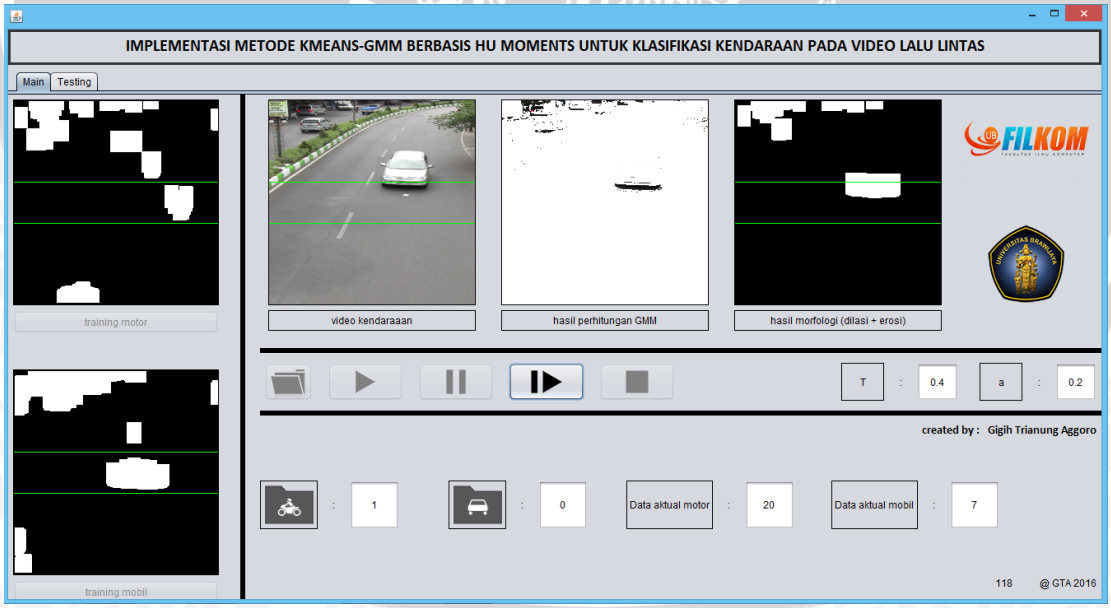
2. *Frame* 2



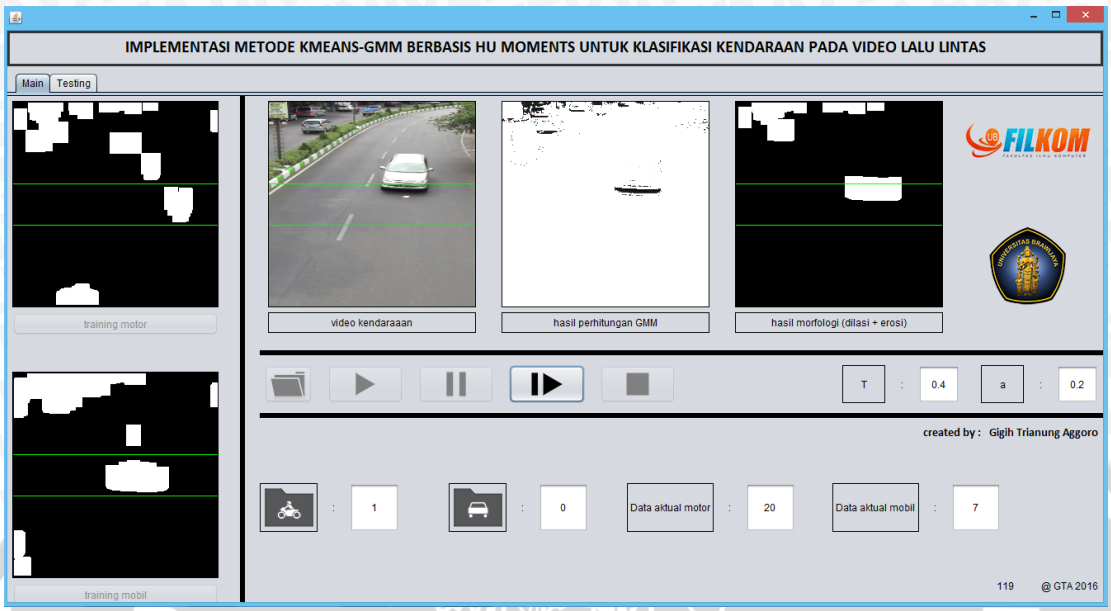
3. *Frame* 3



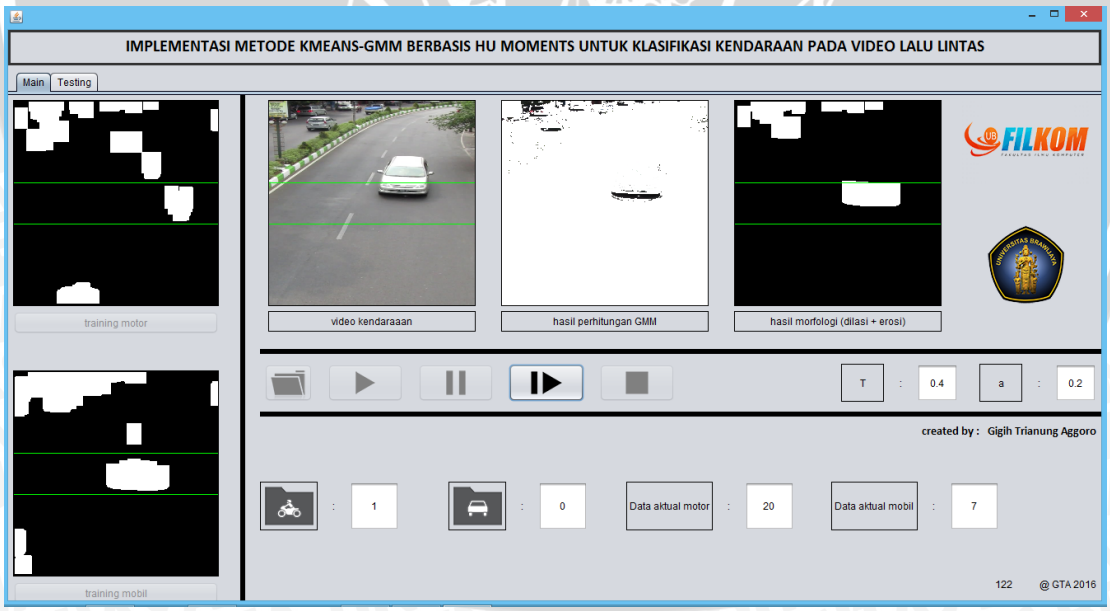
4. *Frame* 4



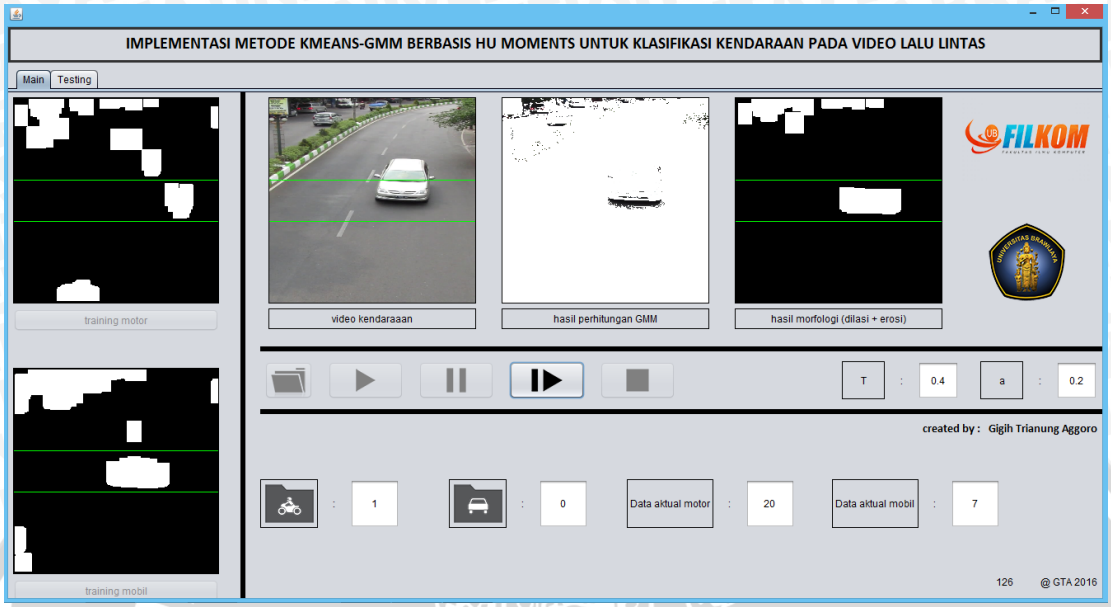
5. *Frame* 5



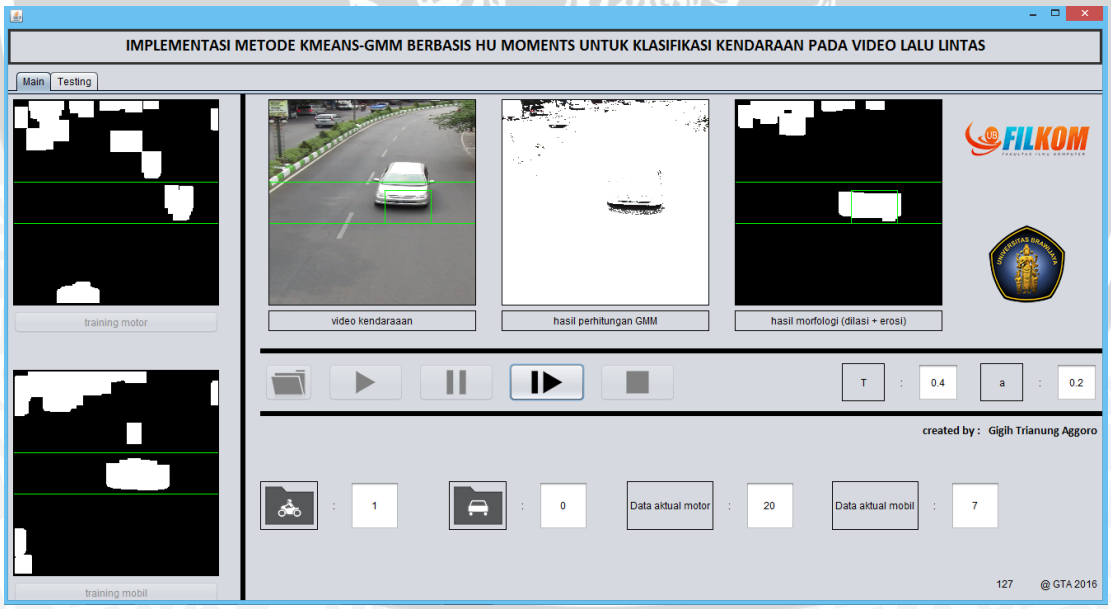
6. *Frame* 6



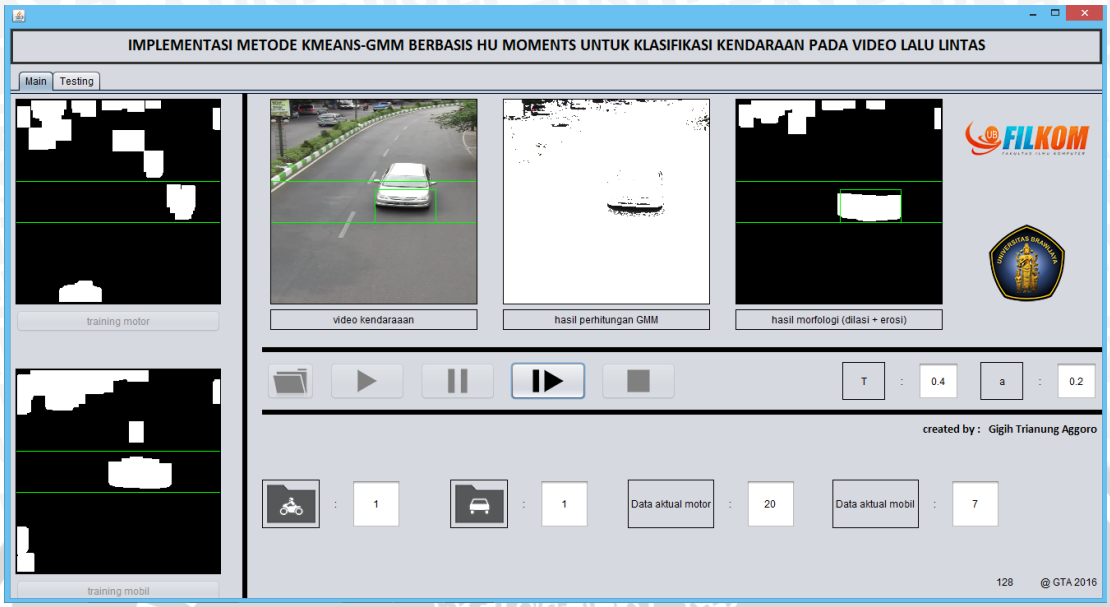
7. *Frame 7*



8. *Frame 8*



9. *Frame* 9



10. *Frame* 10