IMPLEMENTASI *BLOB ANALYSIS* UNTUK *VEHICLE COUNTING* PADA VIDEO LALU LINTAS

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Rina Christanti
NIM: 115060800111121

PRGRAM STUDI INFORMATIKA / ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

IMPLEMENTASI *BLOB ANALYSIS* UNTUK *VEHICLE COUNTING* PADA VIDEO LALU LINTAS

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Rina Christanti
NIM: 115060800111121

Skripsi ini telah diuji dan dinyatakan lulus pada
18 Januari 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Imam Cholissodin, S.Si., M.Kom	Ir. Sutrisno, M.T.
NIK: 850719 16 1 1 0422	NIP: 19570325 198701 1 001

Mengetahui
Ketua Program Studi Informatika

Drs. Marji., M.T.
NIP: 19670801 199203 1 001

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, 18 Januari 2016

Rina Christanti

NIM: 115060800111121

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah melimpahkan rahmat dan karunia-Nya kepada seluruh umat-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "**IMPLEMENTASI *BLOB ANALYSIS* UNTUK *VEHICLE COUNTING* PADA VIDEO LALU LINTAS"** dengan baik. Sholawat serta salam kami haturkan kepada junjungan kami Nabi Muhammad SAW, semoga kami selalu istiqomah dalam meneladaninya. Skripsi ini merupakan serangkaian mata kuliah dan syarat kelulusan yang harus ditempuh oleh mahasiswa sebagai pengaplikasian dari beberapa mata kuliah yang telah di pelajari di bangku kuliah.

Dalam pelaksanaan dan penulisan skripsi ini penulis mendapatkan banyak bantuan dari berbagai pihak baik secara moral maupun material. Dalam kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar – besarnya kepada :

1. Bapak Imam Cholissodin, S.Si,. M.Kom selaku dosen pembimbing I skripsi yang telah dengan sabar membimbing dan mengarahkan penulis, sehingga dapat menyelesaikan skripsi ini.
2. Bapak Ir. Sutrisno, M.T., selaku dosen pembimbing II skripsi yang telah dengan sabar membimbing dan mengarahkan penulis, sehingga dapat menyelesaikan skripsi ini.
3. Bapak Ir. Sutrisno, M.T., Bapak Ir. Heru Nurwasito, M.Kom., Bapak Himawat Aryadita, S.T., M.Sc., dan Bapak Eddy Santoso, S.Kom. selaku Ketua, Wakil Ketua 1, Wakil Ketua 2, dan Wakil Ketua 3 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
4. Bapak Drs. Marji, M.T dan Bapak Issa Arwani, S.Kom., M.Sc selaku Ketua dan Sekretaris Program Studi Teknik Informatika Universitas Brawijaya.
5. Bapak Eriq Muh. Adams Jonemaro, S.T, M.Kom selaku dosen penasehat akademik yang selalu memberikan nasehat kepada penulis selama menempuh masa studi.
6. Ibu Lailil Muflikhah, S.Kom,M.Sc selaku Ketua Laboratorium Komputasi dan Sistem Cerdas.
7. Orang tua dan seluruh keluarga atas segenap dukungan dan kasih sayang yang telah diberikan.
8. Seluruh Dosen Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya atas kesediaan membagi ilmu dan bantuan yang telah diberikan.
9. Seluruh Civitas Akademika Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.
10. Teman-teman angkatan 2011 dan Konsentrasi Cerdas dan Visualisasi dan seluruh pihak yang telah membantu kelancaran penulisan skripsi yang tidak dapat penulis sebutkan satu persatu.

11. Seluruh pihak yang telah membantu dalam penyelesaian penulisan skripsi ini yang tidak dapat penulis sebut satu persatu.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak kekurangan baik format penulisan maupun isinya. Oleh karena itu, saran dan kritik yang membangun dapat disampaikan melalui email penulis rnnyun@gmail.com. Penulis berharap skripsi ini dapat memberikan manfaat bagi semua pihak.

Malang, 7 Januari 2016

Penulis
rnnyun@gmail.com

ABSTRAK

Perkembangan teknologi multimedia saat ini telah berkembang sangat pesat, hal ini mendorong manusia untuk menciptakan teknologi yang dapat memudahkan dan memberikan dampak positif bagi kehidupan. Salah satu teknologi yang saat ini banyak diciptakan adalah menghitung jumlah kendaraan berbasis pengolahan citra, yang sebelumnya dilakukan secara manual dengan kamera CCTV atau alat *counter*. Metode yang akan digunakan adalah *blob analysis*. Proses perhitungan kendaraan dengan menggunakan *blob analysis* dapat diselesaikan dengan tiga tahap utama: segmentasi objek, *blob analysis*, dan *tracking*. Setiap objek yang berhasil terdeteksi akan dilakukan ektraksi fitur untuk mendapatkan ciri khusus dari objek tersebut. Cara untuk melacak objek yang bergerak dapat dicapai dengan membandingkan fitur yang sudah diekstrak dan mengukur jarak minimal setiap objek pada *frame* saat ini dan *frame* sebelumnya. Hasil pengujian akurasi dari 4 video yang berbeda dengan 10 data uji menunjukkan bahwa aplikasi ini memiliki nilai akurasi tertinggi yakni 86%. Sehingga dapat disimpulkan bahwa aplikasi ini mampu melakukan perhitungan kendaraan dengan baik.

Kata kunci: *blob analysis*, video lalu lintas, *background subtraction*

ABSTRACT

Nowadays, the advancement of multimedia technology has grown rapidly, it encourages people to create technology that can facilitate and provide a positive impact for living. For example, a technology that automatically count passing vehicle using image processing. Which was manually counted using CCTV Camera or Counting Tools. The method to be used in this paper is blob analysis. The process of calculating vehicles by using blob analysis is composed of three main steps: object segmentation, blob analysis, and tracking. Foreach successfully detected object, some feature will be extracted to get specific characteristic of the object. Tracking moving object can be achieved by comparing extracted feature in previous step and calculate the minimum distance of each object from different frames.The result of accuracy testing using 4 different videos with 10 data testing shows that the system proposed in this paper has best accuracy of 86%. So it can be concluded that this system capable of performing vehicle counting very well.

Keywords: *blob analysis*, *traffic surveillance*, *background subtraction*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR PERSAMAAN

DAFTAR *SOURCE CODE*

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Jumlah kendaraan di jalan raya meningkat tiap tahun. Meningkatnya jumlah kendaraan tidak didukung oleh pelebaran jalan sehingga menyebabkan terjadinya kemacetan di jalan raya. Dari data Badan Pusat Statistik Indonesia didapatkan bahwa jumlah kendaraan pada tahun 2001 adalah 20.922.235 sedangkan pada tahun 2010 adalah 76.907.127. Dari data tersebut terlihat bahwa terjadi peningkatan jumlah kendaraan dari tahun 2001 ke tahun 2010 (Taksiono, 2013). Selain dari jumlah kendaraan, penyebab kemacetan dapat dikarenakan banyaknya kendaraan dari jenis tertentu seperti kendaraan roda empat yang mempunyai lebar yang cukup besar dibandingkan dengan kendaraan roda dua.

Dampak dari kemacetan dapat dirasakan secara langsung maupun tidak langsung. Dampak yang dirasakan secara langsung adalah Bahan Bakar Minyak (BBM) yang terbuang sia – sia. Hal tersebut dikarenakan saat terjadi kemacetan, orang akan menunggu bahkan mencari celah agar kendaraan mereka dapat berjalan sehingga menjadikan mesin kendaraan tetap dalam kondisi menyala. Padahal saat ini masyarakat seharusnya menghemat BBM karena pasokan BBM sudah dikurangi dan harga yang tidak stabil. Sedangkan secara tidak langsung dampak dari kemacetan adalah menipisnya lapisan ozon. Seperti yang sudah dijelaskan sebelumnya saat terjadi kemacetan, kondisi mesin kendaraan akan dibiarkan menyala sehingga menyebabkan polusi udara yang berasal dari hasil pembuangan sisa - sisa pembakaran mesin tersebut. Polusi udara merupakan salah satu penyebab lapisan ozon semakin menipis dan hal tersebut akan membahayakan kelangsungan hidup manusia. Selain itu, kemacetan dapat menyebabkan menurunnya tingkat produktivitas seseorang. Kemacetan tanpa disadari akan membuat lelah karena menunggu dan harus berjalan pelan – pelan. Sehingga ketika sampai di tempat tujuan, kondisi badan yang sudah kelelahan akan menjadikan seseorang malas dan cenderung tidak menghasilkan sesuatu yang bermanfaat.

Pada umumnya deteksi pada objek kendaraan dilakukan secara manual dengan kamera CCTV (*Close Circuit Television*). CCTV memiliki kemampuan merekam dan mengamati objek dengan baik serta mampu memberikan informasi secara *real-time*. Namun penggunaan CCTV mempunyai beberapa kelemahan yaitu pengawasan pada monitor harus dilakukan tanpa henti agar mendapatkan informasi secara aktual dan tentunya terdapat faktor *human reliability* yang membatasi kemampuan manusia dalam melakukan pengawasan (Negara dan Rahman, 2011). Oleh karena itu, diperlukan sebuah sistem cerdas yang dapat melakukan pelacakan secara otomatis untuk mengenali objek yang bergerak.

Salah satu teknologi multimedia yang populer saat ini adalah *object tracking*. *Object tracking* adalah sebuah sistem yang digunakan untuk melakukan pelacakan terhadap suatu objek bergerak, sehingga pergerakan objek tersebut dapat dideteksi dengan tetap memperhatikan perubahan – perubahan yang terjadi di

sekitar objek tersebut (Agustina & Mukhlas, 2012). *Object tracking* secara umum mencakup dua tahap, yaitu deteksi objek dan pelacakan objek. Tahap deteksi objek adalah bagian dari *object tracking* untuk mendeteksi sebuah objek yang ada pada tiap *frame* video untuk dilacak pada tahap selanjutnya. Salah satu metode deteksi objek adalah *background subtraction*. Metode ini cukup sederhana dan mampu menghasilkan perubahan dari tiap *region* citra dengan baik, sehingga sangat sesuai untuk mendeteksi perubahan objek pada video (Rahman, Saha, & Khanum, 2009).

Sebuah penelitian yang dilakukan oleh Taksiono (2013) adalah penelitian yang menggunakan *background subtraction* untuk mendeteksi kendaraan bergerak. Hasil dari *background subtraction* berupa citra motor dan mobil yang bentuknya menyerupai kerangka badan dari motor dan mobil sehingga untuk mendapatkan bentuk kontur yang terbaik dilakukan beberapa proses seperti *thresholding* dan morfologi yaitu *dilation*, *erosion*, serta *smoothing*. Akan tetapi dari hasil pengujian sistem diambil kesimpulan bahwa kelancaran proses pendeteksian kendaraan dijalan raya dipengaruhi oleh pencahayaan, adannya bayangan dari kendaraan, sorot lampu dari kendaraan, dan penentuan sudut kemiringan dari peletakan kamera.

Pada penelitian Chen, Lin, & Chen (2007) penggunaan *blob analysis* adalah ketika objek bergerak yang sudah tersegmentasi dilakukan ekstraksi fitur dan kemudian diklasifikasikan. *Blob Analysis* pada penelitian ini merupakan bagian dari *Vehicle Analyzing Algorithm*. Cara kerja algoritma tersebut adalah serangkaian masukkan objek akan disegmentasi, objek tersebut akan mengalami *update background* secara terus – menerus sampai seluruh rangkaian masukkan objek tersegmentasi seluruhnya. Kemudian dilakukan *blob analysis* yaitu ekstraksi fitur seperti perhitungan *dispersedness*, *aspect ratio*, dan *area ratio* sehingga dapat dilakukan klasifikasi. Setelah itu dilakukan *tracking* dan *vehicle count*. Dengan menggunakan *blob analysis* didapatkan rata – rata hasil akurasi sebesar 91.7% dari tiga situasi yang berbeda, yaitu jalan dua arah, jalan satu arah dengan sudut pengambilan gambar dari arah depan dan arah belakang.

Berdasarkan latar belakang tersebut, pada tugas akhir ini penulis mengangkat judul **"Implementasi *Blob Analysis* Untuk *Vehicle Counting* pada Video Lalu Lintas"**. Diharapkan dengan mengimplementasikan metode tersebut sistem dapat melakukan deteksi kendaraan dan menghitung jumlah kendaraan, serta didapakan hasil yang optimal.

1.2 Rumusan masalah

Berdasarkan latar belakang yang ada maka dapat ditentukan rumusan masalah yang meliputi:

1. Bagaimana mengimplementasikan *Blob Analysis* untuk *vehicle counting* pada video lalu lintas.
2. Bagaimana tingkat akurasi dari hasil implementasi *Blob Analysis* saat digunakan untuk *vehicle counting* pada video lalu lintas.

1.3 Tujuan

Tujuan yang ingin dicapai pada penelitian ini adalah:

1. Mengimplementasikan *Blob Analysis* untuk *vehicle counting* pada video lalu lintas.
2. Mengetahui tingkat akurasi dari hasil implementasi *Blob Analysis* apabila digunakan untuk *vehicle counting* pada video lalu lintas.

1.4 Manfaat

Penelitian ini diharapkan mempunyai manfaat yang baik. Adapun manfaat yang diharapkan adalah:

1. Memudahkan dalam menghitung jumlah kendaraan roda empat dan roda dua dalam berbagai ruas jalan.
2. Membantu mengetahui tingkat kepadatan lalu lintas kendaraan roda empat dan roda dua dalam berbagai ruas jalan.

1.5 Batasan masalah

Adapun batasan masalah pada penelitian ini adalah:

1. Sistem yang dibuat akan menerima masukan berupa citra video yang terekam kamera CCTV pada beberapa ruas jalan di Kota Malang dengan *frame rate* 25fps.
2. Sistem akan melakukan pelacakan terhadap objek mobil dan sepeda motor.
3. Sistem yang dibuat akan menerima masukan berupa citra video yang terekam kamera CCTV pada siang hari.
4. Sistem yang dibuat akan menerima masukan berupa citra video dengan kualitas kamera minimal 12 *megapixel*.

1.6 Sistematika pembahasan

Penyusunan laporan proyek akhir ini menggunakan kerangka pembahasan yang tersusun sebagai berikut:

BAB I Pendahuluan

Menguraikan latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, sistematika penulisan dan jadwal penelitian.

BAB II Landasan Kepustakaan

Menguraikan tentang dasar teori dan referensi yang mendasari implementasi *Blob Analysis* untuk *vehicle counting* pada video lalu lintas.

BAB III Metodologi

Menguraikan tentang metode dalam mengimplementasikan *Blob Analysis* pada video lalu lintas. Langkah – langkah yang dijelaskan pada

bab ini meliputi studi literatur, analisis kebutuhan, kebutuhan data, implementasi perangkat lunak, dan pengujian perangkat lunak.

BAB IV Perancangan

Menguraikan tentang langkah kerja dalam mengimplementasikan *Blob Analysis* pada video lalu lintas. Langkah – langkah yang dijelaskan pada bab ini meliputi deskripsi sistem, perancangan perangkat lunak, perhitungan manual, perancangan *user interface*.

BAB V Implementasi

Membahas implementasi dari sistem deteksi yang sesuai dengan perancangan perangkat lunak yang telah dibuat.

BAB VI Pengujian dan Analisis

Memuat tentang hasil pengujian dan analisis terhadap sistem yang telah diimplementasikan.

Bab VII Penutup

Memuat kesimpulan serta saran dari hasil yang diperoleh dan diharapkan dapat bermanfaat dalam pengembangan selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi tinjauan pustaka yang meliputi kajian pustaka dan dasar teori yang mendukung dalam proses pembuatan skripsi. Kajian pustaka membahas penelitian yang telah ada sebelumnya dan akan diusulkan. Dasar teori membahas teori yang diperlukan untuk menyusun skripsi dengan topik yang diusulkan.

Kajian pustaka pada skripsi ini membahas penelitian sebelumnya yang berjudul '*Intellegent Vehicle Counting Method Based on Blob Analysis in Traffic Surveillance*'. Teori dasar yang akan dibahas pada bab ini yaitu Video *Digital*, Kamera CCTV, Algoritma *Vehicle Analysis*, serta Citra *Grayscale*.

2.1 Kajian Pustaka

Kajian pustaka pada skripsi ini membahas beberapa penelitian sebelumnya yang terkait dengan deteksi citra yang ditunjukkan melalui tabel 2.1. Diantaranya yaitu penelitian yang dilakukan oleh Chen, Lin, & Chen (2007) dalam penggunaan *blob analysis*.

Tabel 2.1 Kajian Pustaka

No	Judul	Objek	Metode	Hasil
1	Rancang Bangun Sistem Penghitung Laju dan Klasifikasi Kendaraan Berbasis Pengolahan Citra (Taksiono, 2013)	**Objek**: Melakukan pendeteksian untuk menghitung laju sepeda motor dan mobil dengan membandingkan tingkat *error* yang didapatkan dari beberapa metode yang berbeda **Input**: Video lalu lintas kendaraan	Langkah dan Metode: 1. Mengambil gambar. 2. Jika tidak ada gambar maka kembali mengambil gambar, jika ada gambar maka mengubah ukuran gambar 3. Melakukan pengurangan *background* 4. Memotong gambar 5. Mendeteksi dan melakukan klasifikasi 6. Menghitung laju kendaraan 7. Menampilkan hasil pada *display*	Jumlah *error* yang didapatkan dikarenakan pendeteksian kendaraan di jalan raya dipengaruhi oleh pencahayaan, adanya bayangan dari kendaraan, sorot lampu dari kendaraan, dan penentuan sudut kemiringan dari peletakkan kamera

Tabel 2.1 Kajian Pustaka (lanjutan)

No	Judul	Objek	Metode	Hasil
2	*Background Subtraction Using Gaussian Mixture Model Enhance by Hole Filling Algorithm* (GMMHF) (Nurhadiyatna, et al., 2013)	**Objek**: Membandingkan metode GMMHF dengan sepuluh metode lainnya. **Input**: Video yang diambil di Jl. Lenteng Agung Universitas Indonesia dan video dari VSSN 2006	Langkah dan Metode: 1. I : gambar biner $I(i,j) \in \{0,255\}$ $I(i,j)$: ukuran piksel dari gambar I 2. Membuat matriks baru I_{new}: x x y $I_{new}(x\text{-}k, j\text{-}k)$: $I(i.j)$ x : (h + (2 x k)) y : (w + (2 x k)) k : besar jari-jari dari HF h : tinggi dari I w : lebar dari I 3. $n(I_{new}(x\text{-}k, y\text{-}k) = 255)$: dianggap piksel *foreground* 4. $n(I_{new}(x\text{-}k, y\text{-}k) = 0)$: dianggap piksel *background*	Metode GMMHF yang merupakan kombinasi GMM yang diusulkan oleh Zivkovic dan Heijeden dengan algoritma *Hole Filling* dapat meningkatkan akurasi dan nilai Kappa dapat mengungguli metode lain. Nilai akurasi GMMHF mencapai 97,92% dan nilai Kappa mencapai 0.74
3	*Enhancement on Background Subtraction Techniques Using a Second Derivative in Gradient Direction Filter* (Rahman, et al., 2013)	**Objek**: Mengimplementasikan penggabungan filter SDGD dengan teknik dasar *background substraction* yaitu *Frame Differencing* (FD), *Approximate Median Filter* (AM), *Running Average* (RA), *Running Gaussian Average* (RGA).	Langkah dan Metode: 1. Membangkitkan referensi *frame* 2. Mengurangi *frame* saat ini dengan referensi *frame* untuk mendapatkan F_{diff} 3. Melakukan *thresholding* untuk mendapatkan F_x 4. Melakukan filter SDGD pada F_{diff} dan menghasilkan F_{sdgd} 5. Menggabungkan F_{sdgd} dengan F_x untuk menghasilkan F_g	Penggunaan filter SDGD menghasilkan ukuran piksel objek yang terekstraksi menjadi sedikit lebih luas dan lebih kompleks. Sehingga mampu memberikan gambar gumpalan yang lebih baik

Tabel 2.1 Kajian Pustaka (lanjutan)

No	Judul	Objek	Metode	Hasil
		Input: 6 video dengan situasi berbeda di dalam dan di luar ruangan	6. Melakukan operasi morfologi pada F_g dan menghasilkan F_{final} 7. Jika teknik *Frame Differencing* maka apakah *frame* terakhir. Jika iya berarti selesai. Jika tidak maka memulai lagi untuk menghasilkan F_{diff} 8. Jika bukan teknik *Frame Differencing* maka melakukan perbaruan referensi *frame*. Kemudian apakah *frame* terakhir. Jika iya berarti selesai. Jika tidak maka memulai lagi untuk menghasilkan F_{diff}	
4	*Intelligent Vehicle Counting Method Based on Blob Analysis in Traffic Surveillance* (Chen, Lin, & Chen, 2007)	**Objek**: Mengitung jumlah kendaraan dengan menggunakan *blob analysis* **Input**: Video lalu lintas dengan beberapa situasi yang berbeda	Langkah dan Metode: 1. Memasukkan serangkaian objek 2. Segmentasi objek 3. Meng-*update background* 4. Mendapatkan hasil *background* 5. Melakukan ekstraksi fitur dan klasifikasi terhadap objek bergerak 6. Melakukan *tracking* 7. Menghitung kendaraan	Menhasilkan rata – rata akurasi perhitungan kendaraan sebesar 91.7%. Hasil tersebut diperoleh dari rata – rata dari masing – masing situasi yang berbeda. Kesalahan mengukur kecepatan adalah kurang dari 5km/jam.

Sumber: Taksiono (2013), Nurhadiyatna, et al. (2013), Rahman, et al. (2013), Chen, Lin, & Chen (2007)

Pada penelitian Chen, Lin, & Chen (2007) melakukan proses pelacakan, klasifikasi, dan perhitungan dengan beberapa situasi video yang berbeda dimana objeknya adalah kendaraan (mobil dan sepeda motor). Proses yang dilakukan adalah dengan menggunakan tiga langkah utama yaitu segmentasi objek bergerak, *blob analysis*, dan pelacakan. Pada segmentasi objek bergerak penelitian tersebut menggunakan metode *background subtraction*.

Hasil yang diperoleh dari penelitian ini adalah rata – rata akurasi perhitungan kendaraan sebesar 91.7%. Hasil tersebut diperoleh dari rata – rata dari masing – masing situasi yang berbeda. Situasi pertama yaitu jalan dua arah, hasilnya adalah 83.3% untuk mobil yang terhintung sebanyak 6 dan total *error* 1, untuk motor sebesar 84.2% yang terhintung sebanyak 19 dan total *error* 3. Situasi kedua dan ketiga adalah jalan satu arah tetapi pada situasi kedua video diambil dari arah depan dan situasi ketiga diambil dari arah belakang. Pada situasi kedua, hasilnya adalah 92.8% untuk mobil yang terhintung sebanyak 14 dan total *error* 1, untuk motor sebesar 92.3% yang terhintung sebanyak 13 dan total *error* 1. Pada situasi ketiga, hasilnya adalah 94.4% untuk mobil yang terhintung sebanyak 18 dan total *error* 1, untuk motor sebesar 100% yang terhintung sebanyak 18 dan total *error* 0. Perbedaan yang dibuat penulis pada penyusunan skripsi ini adalah proses klasifikasi kendaraan hanya menggunakan rata – rata lebar dari kendaraan karena hanya membedakan dua jenis kendaraan yaitu sepeda motor dan mobil.

2.2 Video *Digital*

Video merupakan gabungan dari gambar - gambar yang sebetulnya tidak bergerak yang ditampilkan secara beruntun dalam suatu rentan waktu dengan kecepatan tertentu. Satu gambar yang digabung - gabungkan disebut dengan *frame* sedangkan kecepatan pembacaan *frame* disebut dengan *frame rate*, dengan satuan fps (*frame per second*). Jika *frame rate* video semakin tinggi maka pergerakan video tersebut semakin halus. Jadi halus atau tidaknya pergerakan video bergantung pada besar *frame rate* video tersebut (Agustina, 2012).

Karakteristik video digital ditentukan oleh resolusi (*resolution*) atau dimensi *frame* (*frame dimention*), kedalaman piksel (*pixel depth*) dan laju *france* (*frame rate*) (Agustina, 2012).

- Resolusi atau dimensi *frame* adalah ukuran tiap *frame* yang menyusun sebuah video digital. Semakin tinggi resolusi pada video maka semakin baik kualitas video tersebut.
- Kedalaman bit (*bit depth*) menentukan jumlah bit yang digunakan dalam merepresentasikan tiap piksel yang menyusun sebuah *frame*. Kedalaman bit dinyatakan dalam bit/piksel Semakin banyak jumlah bit yang digunakan untuk merepresentasikan sebuah piksel maka semakin tinggi kedalaman pikselnya maka hal ini berarti semakin tinggi pula kualitas videonya.
- Laju *frame* (*frame rate*) menunjukkan jumlah *frame* yang ditampilkan tiap detik dan dinyatakan dalam satuan fps atau *frame*/detik.

2.3 Kamera CCTV (*Closed Circuit Television*)

CCTV (*Closed Circuit Television*) adalah suatu alat yang yang dapat mengirimkan data berupa video melalui transmisi kabel coaxial, FO atau UTP bahkan tanpa kabel ke lokasi tertentu untuk dimonitor, direkam, atau untuk dianalisa. Trend saat ini penggunaan CCTV sudah mengarah ke IP *network* kamera (IP CCTV), walaupun dibeberapa tempat masih ada yang menggunakan analog karena disesuaikan dengan kebutuhan aplikasi pengguna (Autojaya, 2011).

Jadi CCTV berfungsi untuk memonitor suatu ruangan melalui layar televisi/monitor, dengan menampilkan gambar dari kamera yang dipasang di setiap ruangan (biasanya tersembunyi) yang diinginkan oleh bagian keamanan atau yang berkepentingan. Sistem kamera dan TV ini terbatas pada gedung tersebut (closed). Semua kegiatan di dalamnya dapat dimonitor di suatu ruangan atau secara remote. CCTV dapat bekerja selam 24 jam atau sesuai dengan kebutuhan, setiap gambar yang direkam dapat ditayangkan kembali pada waktu dan posisi yang diinginkan oleh operator (Autojaya, 2011).

Teknologi kamera CCTV dapat di kategorikan sebagai berikut (Autojaya, 2011):

1. Kamera Biasa, hanya menangkap gambar sesuai dengan yang di terima oleh CMOS (sensor kamera yang berfungsi menangkap gambar).
2. *Thermal* kamera, berfungsi untuk mendapatkan gambar dari suhu objek.
3. Infra *Red* Kamera, berfungsi untuk mendapatkan objek dari ruangan yang sangat gelap.

Dari sisi kategori bentuk CCTV dapat dibagi menjadi 2 macam, pertama CCTV yang berbentuk *fixed* (posisi kamera tidak berubah-ubah) seperti pada Gambar 2.1.

Gambar 2.1 *Fixed Camera*

Sumber: Autojaya (2011)

Bentuk CCTV yang kedua adalah berbentuk PTZ (*Pan*, *Tilt*, *Zoom*) yaitu kamera yang dapat digerakkan ke kiri dan ke kanan juga kebawah dan keatas serta memiliki kemampuan untuk *Zoom* (pembesaran) sasaran objek dengan kelipatan berkali-kali. Contoh kamera PTZ dapat dilihat bentuknya seperti pada Gambar 2.2.

Gambar 2.2 PTZ *Camera*

Sumber: Autojaya (2011)

Dengan adanya CCTV, kita dapat memantau kantor, pabrik, jalan, kantor atau daerah tertentu dari rumah dengan sangat mudah lewat monitor atau *handphone*.

2.3.1 Manfaat CCTV

Beberapa manfaat atau *benefit* yang dapat dirasakan jika menerapkan CCTV di lingkungan sesuai kebutuhan (Autojaya, 2011):

- Dapat memantau dan merekam segala bentuk aktifitas yang terjadi di lokasi dari jarak jauh dari mana saja, tanpa batasan jarak melalui koneksi internet.
- Dengan menggunakan CCTV juga dapat memantau dan merekam segala bentuk aktifitas yang terjadi dari luar lokasi dengan menggunakan PDA, laptop atau PC secara *realtime* dari mana saja dan kapan saja.
- Merekam dan menyimpan hasil monitoring secara otomatis kedalam *hard disk* selama full 24 jam atau hasil rekaman *motion detection* (merekam hanya jika ada gerakan yang terjadi pada daerah yang dipantau).
- Dapat berfungsi sebagai alarm, yang akan membunyikan suara (misalnya: sirene) atau mendial nomor telepon atau *handphone* secara otomatis pada saat ada kejadian yang tidak dikehendaki, jadi bukan hanya sebagai alat pantau untuk mengawasi saja, CCTV ini sekaligus berfungsi ganda memproteksi dan melindungi *asset* serta tempat yang ingin dilindungi dari ancaman kejahatan.
- Dengan mengkombinasi fungsi – fungsi *motion detect*, alarm *dial*, serta *inrecording*, maka tidak harus terus menerus mengawasi kamera CCTV.
- Mendapatkan bukti otentik jika terjadi peristiwa yang tidak dikehendaki.
- Mengawasi dan memproteksi asset berharga, karena CCTV dapat difungsikan sebagai alarm yang "cerdas".

2.3.2 Kamera IP CCTV

Kamera IP CCTV adalah kamera CCTV yang berbasis IP (Internet *Protocol*), cara kerjanya dengan mengkonversi gambar dan *audio* menjadi data kemudian mengirimkan data tersebut melalui jaringan atau koneksi internet (Xvision, 2013). Pada IP kamera memiliki *software internal* (Video *Analystics*) untuk menganalisa hasil kamera yang ditangkapnya, sedangkan pada kamera CCTV analog fungsi tersebut dikerjakan oleh DVR (*Digital Video Recording*) (Avtech, 2011). Salah satu contoh IP kamera adalah IP kamera merk TP-LINK seri TL-SC3000 3GPP yang bentuknya bisa dilihat pada Gambar 2.3.

Gambar 2.3 IP kamera TP-LINK TL-SC3000 3GPP

Sumber: Alnect (2015)

Kamera TL-SC3000 dari TP-LINK ini sangat ideal digunakan untuk lingkungan rumahan dan perkantoran. Perangkat ini merupakan kamera pengintai bersensor Panasonic MOS yang berbasis jaringan TCP/IP network dari jarak jauh melalui komputer yang terkoneksi ke jaringan. Memiliki fitur yang mendukung seperti 3 GPP, *dual codec,* dan *hybrid analog output* meningkatkan kelengkapan dan kompabilitas perangkat ini ke berbagai media. Spesifikasi lengkap mengenai IP kamera ini dapat dilihat pada Tabel 2.2.

Tabel 2.2 Spesifikasi IP Kamera TP-LINK TP-SC3000 3GPP

Spesifikasi	Keterangan
Color video	Available
Video streaming format	Dual Codec (MPEG 4/ MJPEG)
Quality Video Resolution	VGA, 640 x 480, 320 x 240 (default)
Frame Rate	NTSC (30 fps), PAL (25 fps)
Authentication	Tingkat akses dengan beberapa user menggunakan password
Supported Network Protocols	DDNS, PPPoE, DHCP, NTP, SNTP, TCP/IP, ICMP, SMTP, FTP, HTTP, RTP, RTSP
Notification Management	Alarm, time or motion detection-Image Buffer Triggers
Image Sensor	Yes, with 1/3.6'' Panasonic MOS 738(H) x 480(V) Pixels
Viewing Angle	Horizontal : 90°, Vertical : 55.6°
Lens Aperture	F2.0
Lens Focus Point	f3.6mm
Required Light Intensity	1 Lux / F2.0
Network Connection	Ethernet (10Base-T/100Base-TX)
Audio Codec	Not Available
Dimensions	152.5(L) x 115.2(W) x 40.2(H) mm

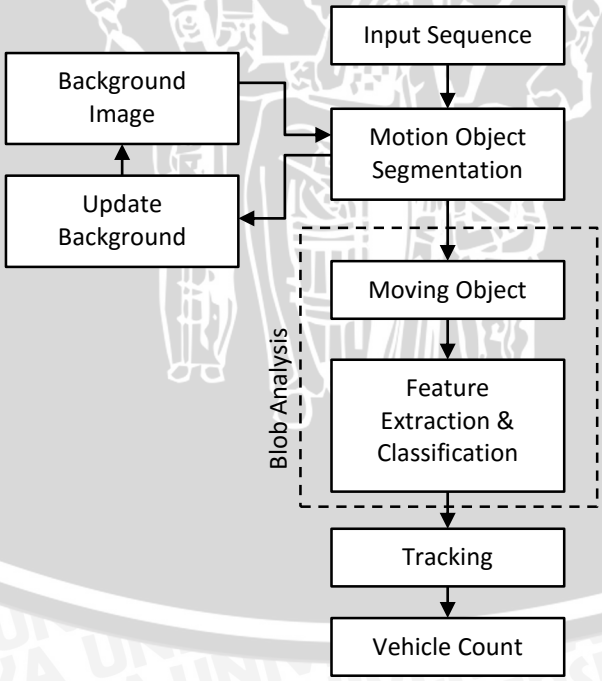
Sumber: Alnect (2015)

2.4 *Blob*

Blob merupakan sekumpulan area/objek yang diamati dan mempunyai nilai piksel yang sama dalam citra biner. Seluruh piksel yang merupakan bagian dari blob akan dianggap sebagai *foreground*, sedangkan piksel – piksel yang lain akan dijadikan *background*. Dalam citra biner, *background* mempunyai nilai 0 (hitam) sedangkan *foreground* mempunyai nilai 1 (putih) (Anonymous, 2012).

2.5 Algoritma *Vehicle Analysis*

Algoritma *vehicle analysis* yang diusulkan ditunjukkan pada Gambar 2.4. Pertama, benda bergerak tersegmentasi dari rangkaian gambar yang diambil dengan menggunakan deteksi perubahan dan memperbarui *background*. Deteksi perubahan digunakan untuk menganalisis informasi temporal antara *frame* yang berurutan dan lebih efisien daripada estimasi gerak. Kombinasi dari perbedaan *frame mask* dan *background subtraction mask* digunakan untuk memperoleh *mask* objek awal dan selanjutnya memecahkan masalah *background* yang masih menjadi masalah objek. Selain itu, penyempurnaan batas diperkenalkan untuk mengurangi pengaruh bayangan dan masalah ketinggalan *background*. Kemudian, setiap objek yang tersegmentasi menunjukkan kendaraan yang dibatasi menjadi persegi panjang. Tinggi, lebar, dan luas persegi panjang tersebut dianggap sebagai fitur kendaraan. Berdasarkan fitur tersebut, setiap kendaraan yang diklasifikasikan ke dalam mobil atau sepeda motor.



Gambar 2.4 Algoritma *Vehicle Analysis*

Sumber: Chen, Lin, & Chen (2007)

2.5.1 *Vehicle Segmentation*

Umumnya, *traffic-camera* biasanya ditempatkan pada tempat tertentu, karena itu *background*-nya tidak bergerak. Oleh karena itu, *background subtraction* cocok digunakan untuk mendeteksi kendaraan yang bergerak dalam proses deteksi perubahan. Awalnya, *background* statis menjadi kerangka acuan dan kemudian teknik *frame-difference* digunakan untuk deteksi perubahan. Fungsi deteksi ditunjukkan pada Persamaan 2.1.

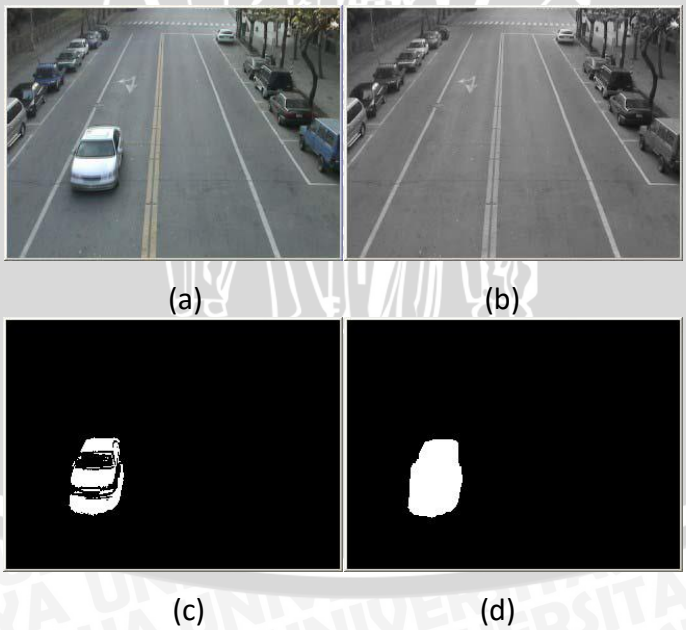
$$D_i(x,y) = C_i(x,y) - B_i(x,y) \tag{2.1}$$

Keterangan:
$D_i(x,y)$ = selisih gambar
$C_i(x,y)$ = gambar saat ini
$B_i(x,y)$ = *background* gambar
i = indeks *frame.*

Selisih gambar perlu ditransformasikan menjadi citra biner dengan menggunakan Persamaan 2.2.

$$R_i(x,y) = \begin{cases} 0, if \left|D_i(x,y)\right| < T \\ 255, otherwise \end{cases} \tag{2.2}$$

Keterangan:
$R_i(x,y)$ = citra biner
T = *threshold*.



(a) (b)

(c) (d)

Gambar 2.5 Hasil dari *Vehicle Segmentation*.
(a) *Input image*. (b) *Background Image*. (c) Perbedaan gambar dari (a) dan (b). (d) Hasil dari tampilan operasi morfologi

Sumber: Chen, Lin, & Chen (2007)

Dikarenakan ada piksel yang sama pada *foreground* dan *background*, maka akan menghasilkan beberapa rongga di citra biner dari objek yang bergerak. Untuk mengatasi masalah fenomena berongga, *erotion* dan *dilation* operasi morfologi digunakan. Hal ini tidak hanya dapat mengisi rongga, tetapi juga menghilangkan *noise* citra biner. Oleh karena itu, dapat memberikan penutup yang menyeluruh untuk objek yang bergerak agar mencapai ekstraksi yang lebih baik. Namun, urutan gambar yang diambil dapat berisi beberapa objek bergerak dan karena itu algoritma *multi-object segmentation* sederhana digunakan untuk mengekstrak setiap objek bergerak. Gambar 2.5 menunjukkan hasil dari metode segmentasi kendaraan yang diusulkan.

2.5.2 *Background Updating*

Ide dasar dari *background updating* adalah untuk membentuk *reliable background* yang dikurangi dari citra sehingga terlihat mirip dengan *background* dalam *frame* dari urutan gambar. *Background* diperbarui dengan mengambil rata-rata bobot dari *background* saat ini dan frame dari urutan gambar. Untuk mengekstrak piksel *background* dalam memperbarui *background* dari frame, persamaan yang digunakan dijelaskan dalam Persamaan 2.3.

$$B_{i+1}(x,y) = \begin{cases} B_i(x,y), & \text{if } R_i(x,y) \neq 0 \\ (1-\alpha)B_i(x,y) + \alpha C_i(x,y), & otherwise \end{cases} \quad (2.3)$$

Keterangan:
α = bobot dari setiap piksel pada *frame*.

2.5.3 *Feature Extraction and Vehicle Classification*

Banyak fitur yang ada di target bergerak seperti tekstur, warna, bentuk, dll. Fitur-fitur ini secara kasar dapat diklasifikasikan menjadi dua bagian: fitur spasial dan fitur temporal. Fitur spasial digunakan untuk membedakan objek yang berbeda pada saat yang sama, dan fitur temporal digunakan untuk mengenali objek yang sama pada waktu yang berbeda. Untuk mengenali objek yang berbeda, maka diperlukan mendapatkan fitur tertentu yang bermakna dan diskriminatif (Chen, Lin, & Chen, 2007).

Ketika kendaraan bergerak, fitur diekstrak, seperti perimeter dan area, mungkin berubah pada waktu ekstraksi yang berbeda. Untuk mengurangi masalah fitur yang berubah dari objek yang bergerak, fitur dari menganalisis *bounding-box* objek yang bergerak dimasukkan. Oleh karena itu, *dispersedness*, *aspect ratio*, dan *area ratio* diukur untuk menyediakan fitur yang stabil untuk objek yang bergerak, seperti yang dijelaskan dalam Persamaan 2.4, Persamaan 2.5, dan Persamaan 2.6.

$$Dispersedness = \frac{Perimeter^2}{Area} \quad (2.4)$$

$$Aspect\ Ratio = \frac{Height}{Width} \quad (2.5)$$

$$Area\ Ratio = \frac{Area}{ROI} \quad (2.6)$$

Dalam persamaan di atas, Perimeter berarti besarnya keliling dari batas *bounding-box* dan Area menunjukkan luas seluruh area dari citra, kemudian *Height*, *Width* dan ROI masing – masing merupakan tinggi, lebar dan luas (yaitu, Tinggi * Lebar) dari *bounding-box*. Untuk melakukan pelacakan dan perhitungan, titik pusat masing-masing objek harus dihitung sesuai dengan Persamaan 2.7.

$$x_0 = \frac{\sum\sum_{(x,y)\in R} x}{\sum\sum_{(x,y)\in R} 1}, \quad y_0 = \frac{\sum\sum_{(x,y)\in R} y}{\sum\sum_{(x,y)\in R} 1} \tag{2.7}$$

Keterangan:
x_0 = titik pusat dari objek
y_0 = titik pusat dari objek
R = letak piksel dari objek

Pada Gambar 2.6 menunjukkan hasil dari klasifikasi kendaraan dalam situasi yang berbeda. Bagian (a) dan (b) mendeskripsikan hasil klasifikasi dari jalan satu arah dalam situasi yang berbeda, yaitu pengambilan gambar dari arah depan dan belakang dan bagian (c) dan (d) menunjukkan hasil klasifikasi dari jalan dua arah dalam pandangan yang berbeda di jalan yang sama (Chen, Lin, & Chen, 2007).

(a) (b)

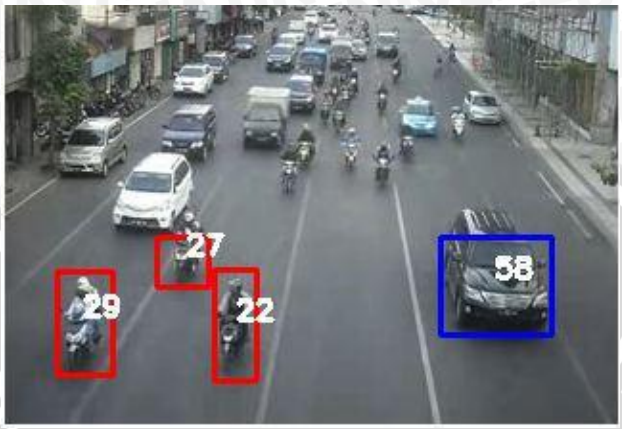
(c) (d)

Gambar 2.6 Hasil Klasifikasi Kendaraan dalam Berbagai Situasi yang Berbeda. (a) Satu arah situasi 1. (b) Satu arah situasi 2. (c) Dua arah situasi 1. (d) Dua arah situasi 2. (Simbol "C" dan "B" mengindikasikan c*ar* dan *bike* berturut - turut)

Sumber: Chen, Lin, & Chen (2007)

Untuk membedakan atau mengklasifikasikan dua jenis kendaraan sepeda motor dan mobil, dapat dilakukan dengan menggunakan salah satu fitur dari

kendaraan seperti panjang atau lebar dari dua jenis kendaraan tersebut. Dari percobaan didapatkan, jika lebar lebih dari 67 maka akan dianggap sebagai mobil sendangkan jika lebih kecil maka akan dianggap sepeda motor (Taksiono, 2013). Gambar 2.7 menunjukkan hasil klasifikasi kendaraan dengan menggunakan lebar untuk membedakan dua jenis kendaraan.



Gambar 2.7 Hasil Klasifikasi Kendaraan
(Kotak merah menandakan sepeda motor dan kotak biru menandakan mobil)

Sumber: Taksiono (2013)

2.5.4 *Vehicle Tracking*

Untuk mencapai penghitungan arus kendaraan, metode yang diusulkan akan melacak setiap kendaraan yang bergerak dalam *frame* gambar secara berturut-turut. Setelah segmentasi objek bergerak, objek tersebut dengan *bounding-box* dan *centroid* (titik pusat) diekstraksi dari setiap *frame*. Secara intuitif, dua objek yang secara spasial terdekat dalam *frame* yang berdekatan terhubung. Jarak Euclidean digunakan untuk mengukur jarak antar titik pusat objek tersebut. Selain itu, area kendaraan juga dipertimbangkan untuk meningkatkan pelacakan kendaraan. Untuk setiap objek dalam *frame*, objek dengan jarak minimum dan ukuran yang sama antara dua *frame* berturut-turut harus dicari dalam *frame* sebelumnya. Fungsi pencocokan dijelaskan dalam Persamaan 2.8 dan Persamaan 2.9.

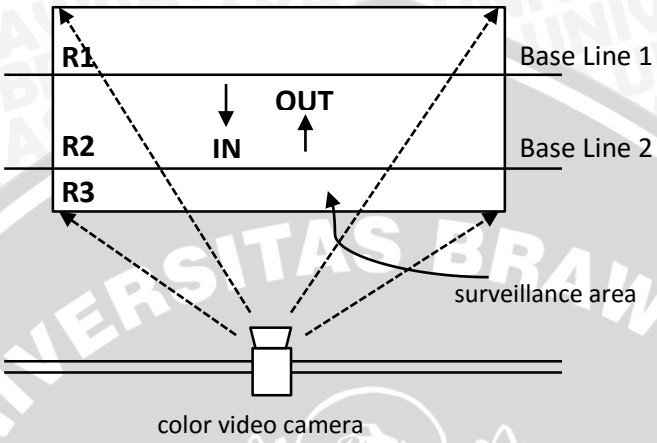
$$\rho_{obj}^{n} = \prod_{m} \left(\left| \rho_{obj_n}^{m} - \rho_{buff_n}^{m} \right| \right) \tag{2.8}$$

$$dist_{obj}^{n} = \left(P_{obj}, P_{buff} \right) = \sqrt{\left(x_{0_{obj}} - x_{0_{buff}} \right)^2 + \left(y_{0_{obj}} - y_{0_{buff}} \right)^2} < \delta \tag{2.9}$$

Keterangan:
ρ = parameter pelacakan
m = indeks fitur
n = jumlah objek
P_{obj} = objek dalam *frame* saat ini

P_{buff} = objek dalam *frame* sebelumnya
δ = *threshold*.

Jika ρ_{obj} dan $dist_{obj}$ *set* minimal dan memenuhi kondisi $dist_{obj}^{n} < \delta$, maka objek dalam frame dianggap objek yang sama dari frame sebelumnya (Chen, Lin, & Chen, 2007).



Gambar 2.8 Pengaturan Sistem *Vehicle Counting*

Sumber: Chen, Lin, & Chen (2007)

2.5.5 *Vehicle-flow and Counting*

Untuk melakukan penghitungan kendaraan yang melewati jalan dua arah secara otomatis, metode yang diusulkan adalah mengatur dua *base line*, seperti ditunjukkan pada Gambar 2.8. Sistem juga akan menghitung kecepatan kendaraan dalam proses penghitungan. Ketika kendaraan melewati daerah-R2, *frame* akan disimpan. Kendaraan yang bergerak dihitung ketika melewati *base line*. Dengan mengukur jarak di jalan yang sebenarnya dan kecepatan *frame rate* yang ditangkap, kecepatan dapat disimpulkan ketika kendaraan melewati daerah-R2.

2.6 Citra *Grayscale*

Merupakan proses yang dilakukan pertama kali pada suatu citra sebelum citra tersebut diproses dengan proses inti yang bertujuan untuk mempermudah proses pengolahan citra. Awalnya citra terdiri atas 3 *layer* warna yaitu R-*layer*, G-*layer* dan *B-layer*, untuk mengubah menjadi citra *grayscale* tiga layer warna tersebut akan diubah menjadi satu layer warna. Dalam citra *grayscale* yang ada hanyalah derajat keabuan, jadi tidak ada lagi istilah warna. Salah satu metode untuk menghasilkan citra *grayscale* yaitu dengan mengganti nilai RGB dari citra tersebut dengan satu nilai yang sama yaitu dengan menggunakan rata-rata dari nilai RGB piksel yang sama. Nilai *mean* dihitung dengan menggunakan Persamaan 2.10.

$$mean = \frac{Piksel_R + Piksel_G + Piksel_B}{3} \quad (2.10)$$

Dengan *mean* adalah nilai rata-rata, $Piksel_R$ adalah nilai piksel dari *Red*, $Piksel_G$ adalah nilai piksel dari *Green* dan $Piksel_B$ adalah nilai piksel *Blue* (Hapsani, 2014).

2.7 Operasi Morfologi

Operasi morfologi adalah teknik pengolahan citra yang didasarkan pada bentuk segmen atau region dalam citra. Operasi ini biasanya diterapkan pada citra biner karena difokuskan pada bentuk objek. Hasil operasi morfologi dimanfaatkan untuk pengambilan keputusan dengan analisis lebih lanjut. Operasi ini meliputi dilasi, erosi, *closing* (erosi kemudian dilasi), dan *opening* (dilasi kemudian erosi). Secara umum pemrosesan citra secara morfologi dilakukan dengan cara mempasing sebuah *Structure Element* (SE) terhadap sebuah citra dengan cara yang hampir sama dengan konklusi. SE dapat berukuran sembarang dan juga memiliki titik poros (Pratama, 2014).

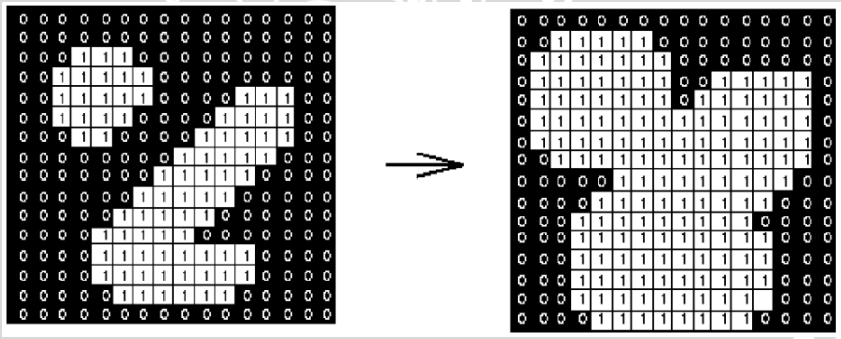
2.7.1 Dilasi

Operasi dilasi dilakukan untuk memperbesar ukuran segmen objek dengan menambah lapisan di sekeliling objek. Terdapat dua cara untuk melakukan operasi ini, yaitu dengan cara mengubah semua titik latar yang bertetangga dengan titik batas menjadi titik objek. Cara kedua yaitu dengan mengubah semua titik di sekeliling titik batas menjadi titik objek (Pratama, 2014). Rumus untuk dilasi menggunakan Persamaan 2.11 dan contoh untuk dilasi ditunjukkan pada Gambar 2.9.

$$g(x, y) = f(x, y) \oplus SE \tag{2.11}$$

SE

1	1	1
1	1	1
1	1	1



Gambar 2.9 Dilasi

Sumber: Pratama (2014)

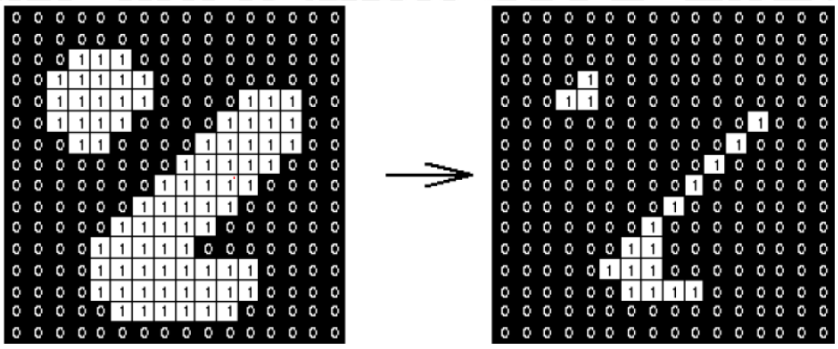
2.7.2 Erosi

Operasi erosi adalah kebalikan dari operasi dilasi. Pada operasi ini, ukuran objek diperkecil dengan mengikis sekeliling objek. Terdapat dua cara untuk melakukannya, cara pertama yaitu dengan mengubah semua titik batas menjadi titik latar dan cara kedua dengan menset semua titik di sekeliling latar menjadi titik objek (Pratama, 2014). Rumus untuk erosi menggunakan Persamaan 2.12 dan contoh untuk dilasi ditunjukkan pada Gambar 2.10.

$$g(x, y) = f(x, y) \Theta SE \qquad (2.12)$$

SE

1	1	1
1	1	1
1	1	1

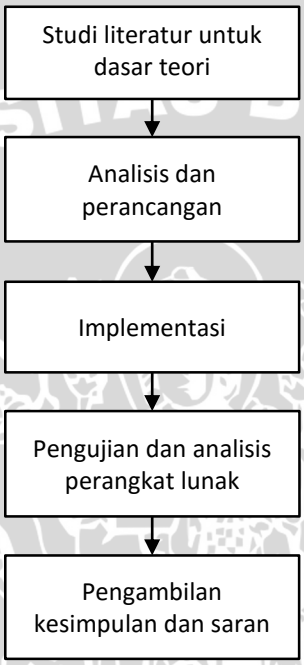


Gambar 2.10 Erosi

Sumber: Pratama (2014)

BAB 3 METODOLOGI

Pada bab ini akan dijelaskan mengenai metodologi penelitian yang akan digunakan dalam penelitian implementasi metode *blob analysis* untuk *vehicle counting* pada video lalu lintas. Metodologi berisi langkah – langkah yang digunakan dalam penelitian skripsi yang terdiri dari studi literatur untuk dasar teori, analisis kebutuhan dan perancangan, implementasi, pengujian, dan analisis perangkat lunak, serta pengambilan kesimpulan dan saran. Langkah – langkah yang akan dilakukan dalam penelitian ini dapat diilustrasikan pada Gambar 3.1.



Gambar 3.1 Diagram Alir Metodologi Penelitian

3.1 Studi Literatur untuk Dasar Teori

Studi literatur dilakukan dengan mencari referensi dari buku atau internet sebagai materi dasar teori untuk menunjang skripsi. Adapun teori – teori tersebut antara lain:

- Video Digital
- Kamera CCTV
- *Algoritma Vehicle Analysis*
- Citra *Grayscale*
- Operasi Morfologi

3.2 Analisis dan Perancangan

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan oleh sistem yang akan dibangun. Analisis kebutuhan dilakukan dengan mengidentifikasi kebutuhan sistem dan siapa saja yang terlibat di dalamnya.

Berikut analisis kebutuhan dalam aplikasi *vehicle counting* dengan menggunakan metode *blob analysis*.

3.2.1 Kebutuhan Antar Muka

Spesifikasi kebutuhan antarmuka untuk perangkat lunak ini antara lain:

- Tampilan antarmuka pengguna (*user interface*) harus mudah digunakan oleh pengguna.
- Perangkat lunak memiliki antarmuka untuk menerima masukan berupa data uji dalam *folder* kumpulan gambar dari hasil ekstraksi video (.avi) lalu lintas.
- Perangkat lunak memiliki antarmuka untuk menerima masukan berupa pengaturan yang dibutuhkan.
- Perangkat lunak memiliki antarmuka untuk menampilkan video yang tengah dilacak serta menandai daerah objek yang telah terdeteksi dengan balok/kotak merah.
- Perangkat lunak memiliki antarmuka untuk menampilkan perhitungan jumlah kendaraan yang terdeteksi.

3.2.2 Kebutuhan Data

Data yang akan diolah oleh perangkat lunak ini antara lain:

- Data berupa file video (.avi) lalu lintas dalam bentuk kumpulan gambar yang digunakan sebagai citra masukan. Data diunduh dari internet, kamera Nikon D3100, dan kamera Canon karena video CCTV mempunyai kualitas kurang baik. Video dari internet merupakan video dengan resolusi 1280x720 px dengan *frame rate* 25 fps, kemudian kamera merk Nikon dengan resolusi video 1920x1080 px dan *frame rate* 25 fps, sedangkan kamera Canon merupakan kamera dengan resolusi 720x1280 px dan *frame rate* 25 fps. Dimana kumpulan gambar diperkecil menjadi 320x180 px dan 180x320 px.
- Data berupa citra latar belakang tempat yang digunakan sebagai *background* untuk proses *background subtraction* pada proses deteksi objek.
- Data keluaran perangkat lunak ini akan langsung ditampilkan saat proses selesai.

3.2.3 Kebutuhan Fungsional

Fungsi – fungsi perangkat lunak ini antara lain:

- Perangkat lunak harus mampu menerima inputan data berupa *folder* kumpulan gambar hasil ekstraksi file video lalu lintas.
- Aplikasi harus mampu melakukan pendeteksian objek pada setiap *frame*.
- Perangkat lunak harus mampu melacak objek yang bergerak dengan menandai daerah objek yang terdeteksi menggunakan kotak merah pada gambar.

- Perangkat lunak harus mampu mengklasifikasikan objek berdasarkan lebar atau tinggi objek.
- Perangkat lunak harus mampu menghitung jumlah objek di dalam gambar.

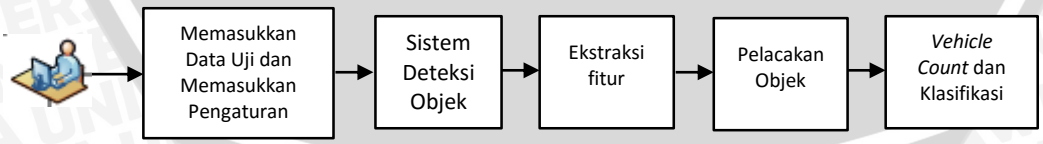
3.2.4 Arsitektur Program

Perancangan aplikasi *vehicle counting* pada video lalu lintas ini dapat dijelaskan sebagai berikut. Pertama, pengguna terlebih dahulu mengekstrak file video menjadi kumpulan gambar dengan menggunakan aplikasi tertentu, kemudian memilih citra latar belakang atau *background* pada salah satu gambar hasil ekstraksi video tanpa objek dan mengganti nama citra tersebut menjadi 0.png. Citra tersebut digunakan sebagai *background* saat dilakukan proses *background subtraction*.

Selanjutnya sistem akan memproses kumpulan gambar beserta citra latar belakang untuk deteksi, klasifikasi, dan *vehicle count*. Proses tersebut meliputi pengubahan kumpulan gambar dan citra latar belakang menjadi *frame – frame bitmap*, citra abu – abu (*grayscale*), *background subtraction*, operasi morfologi dengan menggunakan *library*, menandai objek dengan *bounding-box* dan memberikan *base line* untuk menandai daerah pelacakan. Hasil dari proses *preprocessing* ini adalah objek yang telah terdeteksi. Objek inilah yang akan menjadi target pelacakan dengan menggunakan metode *blob analysis*. Objek yang telah terdeteksi kemudian akan dilakukan ekstraksi fitur dengan menghitung *dispersedness*, *area ratio*, *aspect ratio* untuk *vehicle count* dan klasifikasi objek.

3.2.5 Diagram Blok Sistem

Secara umum, sistem ini akan dimulai dengan *user* memasukkan data uji berupa kumpulan gambar dan memilih salah satu gambar sebagai *background*. Setelah data uji telah siap, selanjutnya pengguna memasukkan data uji dan memasukkan pengaturan yang dibutuhkan untuk proses pendeteksian objek sebelum diolah dengan metode *blob analysis*. Metode *blob analysis* akan menghasilkan ekstraksi fitur yang kemudian diolah untuk pelacakan dan *vehicle counting* dan klasifikasi. Gambar 3.2 merupakan ilustrasi dari diagram blok aplikasi *vehicle counting* dengan metode *blob analysis*.



Gambar 3.2 Diagram Blok Sistem

Diagram blok sistem ini secara keseluruhan terdiri dari satu komponen utama yang dijalankan *user* yaitu memasukkan data uji dan memasukkan pengaturan. Selain itu terdapat empat komponen yang dilakukan oleh sistem yaitu deteksi objek dengan metode *background subtraction*, ekstraksi fitur, pelacakan objek, dan *vehicle count* dan klasifikasi.

Penjelasan lebih lanjut mengenai komponen utama dari sistem aplikasi ini dijelaskan sebagai berikut:

a) Memasukkan Data Uji dan Memasukkan Pengaturan

Bagi pengguna, proses memasukkan data uji ini bertujuan untuk menyediakan data berupa citra dua dimensi sebagai data yang akan diproses. Data uji didapatkan dari video yang diekstraksi menjadi kumpulan gambar dengan menggunakan aplikasi tertentu dan citra *background* ini didapatkan dari mengambil salah satu gambar dari data uji. Citra *background* ini merupakan pencitraan dari tempat pada keadaan dasar dimana tidak terdapat objek bergerak di dalamnya, dengan kata lain keadaannya statis dan terdiri dari objek-objek yang posisinya tidak akan berubah (*absolute*). Citra *background* ini digunakan pada proses *background substraction* pada tahap deteksi objek untuk metode *blob analysis*. Sedangkan memasukkan pengaturan bertujuan untuk mengatur beberapa kebutuhan sistem untuk proses *vehicle count*.

b) Sistem Deteksi Objek

Bagi pengguna, sistem deteksi objek ini merupakan proses untuk mendeteksi objek – objek yang ada di citra video yang akan dilacak. Sistem akan melakukan proses ini dengan mengubah citra data uji menjadi citra *grayscale,* melakukan *background substraction*, operasi morfologi, dan memberikan *bounding-box* untuk objek yang terdeteksi dan memberikan *base line* untuk menandai daerah pelacakan.

c) Ekstraksi Fitur

Bagi pengguna, ekstraksi fitur merupakan proses untuk mendapatkan fitur – fitur dari objek yang telah terdeteksi seperti *dispersedness*, *area ratio*, *aspect ratio* dan perhitungan *centroid* kemudian menggunakan fitur tersebut untuk *vehicle count* dan klasifikasi objek.

d) Pelacakan Objek

Bagi pengguna, pelacakan objek merupakan proses untuk melacak objek pada *frame* saat ini dengan *frame* sebelumnya merupakan objek yang sama atau bukan objek yang sama. Hasil dari pelacakan objek sangat penting untuk tahap *vehicle count*.

e) *Vehicle Count* dan Klasifikasi

Bagi pengguna, *vehicle count* merupakan proses untuk menghitung banyaknya objek dalam setiap data uji serta mengklasifikasikan objek menjadi *car* atau *motorcycle*.

3.3 Implementasi

Implementasi aplikasi pelacakan ini dilakukan dengan mengacu pada perancangan sistem. Implementasi perangkat lunak dilakukan menggunakan bahasa pemrograman *framework* C# .NET dan *library AForge* dengan

menggunakan *tools* Microsoft Visual Studio 2010. Implementasi aplikasi ini meliputi:

- Pembuatan antarmuka pengguna (*user interface*) berupa halaman yang menerima masukkan pengguna dan menampilkan proses dari *preprocessing* hingga *vehicle counting*.
- Melakukan proses deteksi objek terhadap data yang dimasukkan oleh pengguna yaitu data folder kumpulan gambar.
- Melakukan proses ekstraksi fitur dengan menggunakan metode *blob analysis*.
- Melakukan proses pelacakan dari dari frame-frame data uji.
- Memberikan hasil perhitungan jumlah kendaraan dan klasifikasi.

3.4 Pengujian Perangkat Lunak

Pengujian perangkat lunak pada penelitian ini dilakukan agar dapat menunjukkan bahwa perangkat lunak telah mampu bekerja sesuai dengan spesifikasi dari kebutuhan yang melandasinya. Pengujian yang dilakukan meliputi:

- Data berupa file video (.avi) lalu lintas dalam bentuk kumpulan gambar yang digunakan sebagai citra masukan. Data diunduh dari internet, kamera Nikon D3100, dan kamera Canon. Video dari internet merupakan video dengan resolusi 1280x720 px dengan *frame rate* 25 fps, kemudian kamera merk Nikon dengan resolusi video 1920x1080 px dan *frame rate* 25 fps, sedangkan kamera Canon merupakan kamera dengan resolusi 720x1280 px dan *frame rate* 25 fps. Dimana kumpulan gambar diperkecil menjadi 320x180 px dan 180x320 px. Data yang digunakan sebanyak 4 video dengan 10 variasi data uji. Satu video dapat mewakili 2 sampai 4 data uji karena dalam satu video diambil 30 detik.
- Skenario pengujian pertama yaitu pengujian akurasi terhadap variasi data video lalu lintas. Dimana yang akan diuji antara lain klasifikasi kendaraan dan jumlah kendaraan. Pengujian ini bertujuan untuk mengetahui jumlah kendaraan sesuai dengan klasifikasinya. Pengukuran tingkat akurasi program akan dihitung berdasarkan sesuai atau tidaknya jumlah objek yang terdeteksi oleh program dengan jumlah sebenarnya pada tiap video yang mampu dilihat mata manusia. Contoh tabel pengujian akurasi terhadap variasi data video lalu lintas dapat dilihat pada Tabel 3.1.

Tabel 3.1 Contoh Tabel Pengujian Akurasi

Video ke-	Data Uji ke-	Kendaraan	Hasil Sistem	Hasil Aktual	Akurasi tiap Data Uji (%)	Akurasi tiap Video (%)
1	1	Mobil				
		Sepeda Motor				
	2	Mobil				
		Sepeda Motor				
	3	Mobil				
		Sepeda Motor				

Tabel 3.1 Contoh Tabel Pengujian Akurasi (lanjutan)

Video ke-	Data Uji ke-	Kendaraan	Hasil Sistem	Hasil Aktual	Akurasi tiap Data Uji (%)	Akurasi tiap Video (%)
	4	Mobil				
		Sepeda Motor				
2	5	Mobil				
		Sepeda Motor				
	6	Mobil				
		Sepeda Motor				
3	7	Mobil				
		Sepeda Motor				
	8	Mobil				
		Sepeda Motor				
4	9	Mobil				
		Sepeda Motor				
	10	Mobil				
		Sepeda Motor				
Rata – rata						

- Skenario pengujian kedua yaitu pengujian terhadap *thresholding* untuk menentukan *Ri(x,y)*. Pengujian ini dilakukan dengan tujuan untuk mengetahui nilai thresholding yang terbaik sehingga dapat membentuk gambar biner dengan baik. Pengujian dilakukan pada setiap data uji dari video yang mempunyai akurasi tertinggi. Contoh tabel pengujian threshold untuk Ri(x,y) pada video 1 dapat dilihat pada Tabel 3.2 dan T untuk *threshold*, M untuk mobil, sedangkan S untuk sepeda motor.

Tabel 3.2 Contoh Tabel Pengujian *Threshold* untuk $R_i(x,y)$

T	Data Uji								Akurasi tiap *Threshold* (%)
	1		2		3		4		
	M	S	M	S	M	S	M	S	
15									
20									
25									
30									
35									
40									
45									
50									
55									
60									

- Skenario pengujian ketiga yaitu pengujian terhadap *alpha* (faktor pembobot). Nilai *alpha* berpengaruh terhadap perubahan *background* saat proses *update background*. Pengujian ini dilakukan dengan tujuan untuk mengetahui nilai *alpha* yang terbaik sehingga dapat membentuk *reliable background* yang dapat menyesuaikan dengan *background sequences image*. Pengujian dilakukan pada setiap data uji dari video yang mempunyai akurasi tertinggi. Contoh tabel pengujian *alpha* pada video 1 dapat dilihat pada Tabel 3.3 dan *α* untuk *alpha*, M untuk mobil, sedangkan S untuk sepeda motor.

Tabel 3.3 Contoh Tabel Pengujian *Alpha*

α	**Data Uji**								**Akurasi tiap *Alpha* (%)**
	1		2		3		4		
	M	S	M	S	M	S	M	S	
0									
0.1									
0.2									
0.3									
0.4									
0.5									
0.6									
0.7									
0.8									
0.9									

- Skenario pengujian keempat yaitu pengujian terhadap klasifikasi mobil dan sepeda motor menggunakan *threshold*. Pengujian ini dilakukan dengan tujuan untuk mengetahui nilai *threshold* yang terbaik sehingga dapat mengklasifikasikan mobil dan sepeda motor dengan baik. Pengujian dilakukan pada setiap data uji dari video yang mempunyai akurasi tertinggi. Contoh tabel pengujian klasifikasi menggunakan *threshold* pada video 1 dapat dilihat pada Tabel 3.4 dan T_{mobil} untuk threshold mobil, M untuk mobil, sedangkan S untuk sepeda motor.

Tabel 3.4 Contoh Tabel Pengujian Klasifikasi menggunakan *Threshold*

T_{mobil}	**Data Uji**								**Akurasi tiap T_{mobil} (%)**
	1		2		3		4		
	M	S	M	S	M	S	M	S	
30									
35									
40									
45									
50									
55									
60									
65									

Tabel 3.4 Contoh Tabel Pengujian Klasifikasi menggunakan *Threshold* (lanjutan)

T_{mobil}	Data Uji								Akurasi tiap T_{mobil} (%)
	1		2		3		4		
	M	S	M	S	M	S	M	S	
70									
75									

- Skenario pengujian kelima yaitu pengujian terhadap ukuran *structure element* untuk operasi morfologi. Pengujian ini dilakukan dengan tujuan untuk mengetahui ukuran *structure element* yang terbaik sehingga dapat membentuk *blob* dengan baik. Pengujian dilakukan pada setiap data uji dari video yang mempunyai akurasi tertinggi. Contoh tabel pengujian ukuran *structure element* untuk operasi morfologi pada video 1 dapat dilihat pada Tabel 3.5 dan *SE* untuk *structure element*, M untuk mobil, sedangkan S untuk sepeda motor.

Tabel 3.5 Contoh Tabel Pengujian Ukuran *Structure Element*

SE		Data Uji								Akurasi tiap *SE* (%)
		1		2		3		4		
Erosi	**Dilasi**	M	S	M	S	M	S	M	S	
3x3	**11x11**									
3x3	**13x13**									
3x3	**17x17**									
5x5	**13x13**									
5x5	**17x17**									

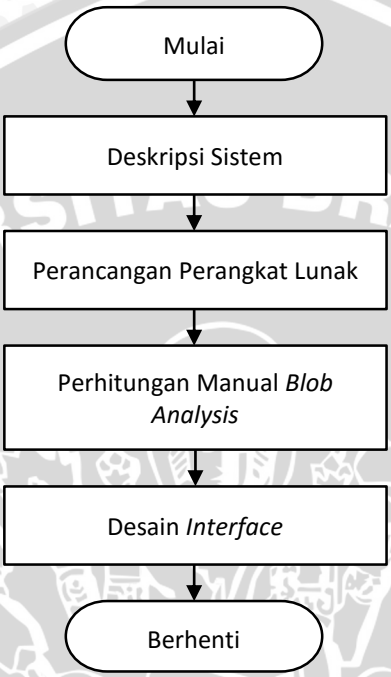
3.5 Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dari aplikasi yang telah dibuat dilakukan setelah semua tahapan perancangan dan pengujian sistem aplikasi telah selesai dilakukan. Pengambilan kesimpulan didasarkan pada kesesuaian antara teori dan praktek. Kesimpulan ini merupakan informasi akhir dari perancangan aplikasi yang berisi mengenai berhasil atau tidaknya aplikasi tersebut dijalankan.

Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi serta menyempurnakan penulisan.

BAB 4 PERANCANGAN

Pada bab ini akan dijelaskan mengenai perancangan yang akan digunakan dalam penelitian implementasi metode *blob analysis* untuk *vehicle counting* pada video lalu lintas. Perancangan yang dilakukan terdiri dari deskripsi umum sistem, perancangan program aplikasi, perhitungan manual, desain *interface*. Tahap – tahap perancangan yang akan dilakukan dapat diilustrasikan pada Gambar 4.1.



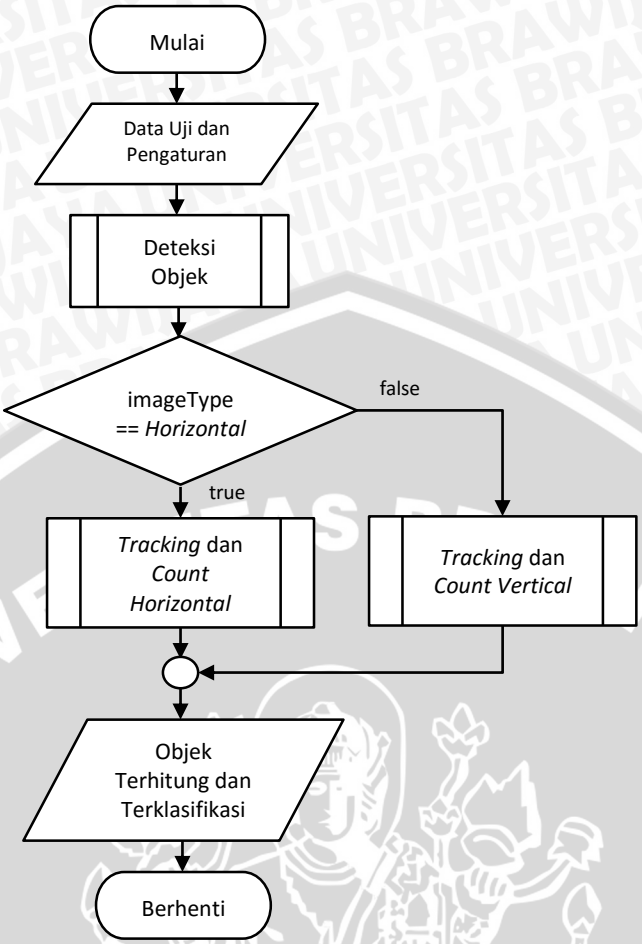
Gambar 4.1 Diagram Alir Perancangan Sistem

4.1 Deskripsi Sistem

Sistem ini dibuat untuk menerapkan metode *blob analysis* untuk *vehicle counting* ke dalam sebuah aplikasi. Sistem akan mengolah data inputan yang dimasukan user berupa kumpulan gambar dari hasil ekstraksi video. Setelah itu, sistem akan menghasilkan *output* berupa hasil *counting* kendaraan yang berhasil terdeteksi beserta informasi fitur – fitur dari kendaraan tersebut.

4.2 Perancangan Perangkat Lunak

Tahap perancangan perangkat lunak pada proses penerapan metode *blob analysis* untuk permasalahan *vehicle counting* yang dijalankan pada program aplikasi digambarkan dalam diagram alir sesuai dengan analisis yang dilakukan sebelumnya. Diagram alir sistem yaitu tahapan proses sistem dengan metode *blob analysis* yang ditunjukkan pada Gambar 4.2.



Gambar 4.2 Diagram Alir Proses Sistem

Proses yang dilakukan seperti yang terlihat pada diagram alir Gambar 4.2, pertama adalah memasukkan data uji, kemudian data tersebut di proses pada deteksi objek yang bertujuan mendapatkan segmentasi objek. Kemudian hasil dari deteksi objek akan diproses pada tahap selanjutnya, jika tipe gambar adalah *horizontal* maka proses yang dilakukan adalah *Tracking* dan *Count Horizontal* dan yang lainnya maka proses yang dilakukan adalah *Tracking* dan *Count Vertical*. Tujuan dari membedakan berdasarkan tipe gambar karena mempengaruhi pada proses *tracking*, klasifikasi, dan perhitungan kecepatan. Perbedaan tipe gambar dijelaskan pada Gambar 4.3.

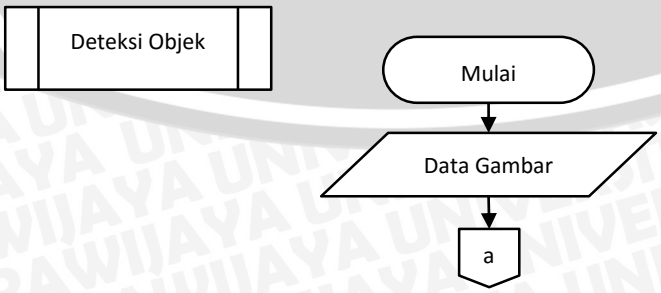
Pada proses *tracking* dan *count* akan dilakukan ekstraksi fitur pada objek yang berhasil terdeteksi, lalu membandingkan fitur tersebut pada setiap objek yang terdeteksi untuk mengetahui objek tersebut apakah objek yang sama atau objek yang berbeda. Kemudian jika memenuhi kriteria maka akan dihitung sebagai sebuah objek beserta klasifikasinya.

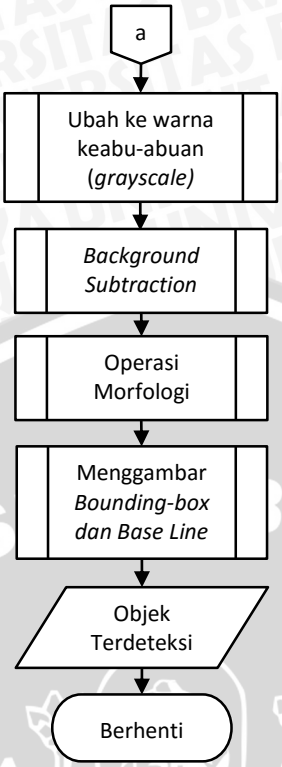
(a) (b)

Gambar 4.3 Tipe Gambar.
(a) *Horizontal* (Objek bergerak dari arah kiri ke kanan atau sebaliknya). (b) *Vertical* (Objek bergerak dari arah atas ke bawah atau sebaliknya).

4.2.1 Deteksi Objek

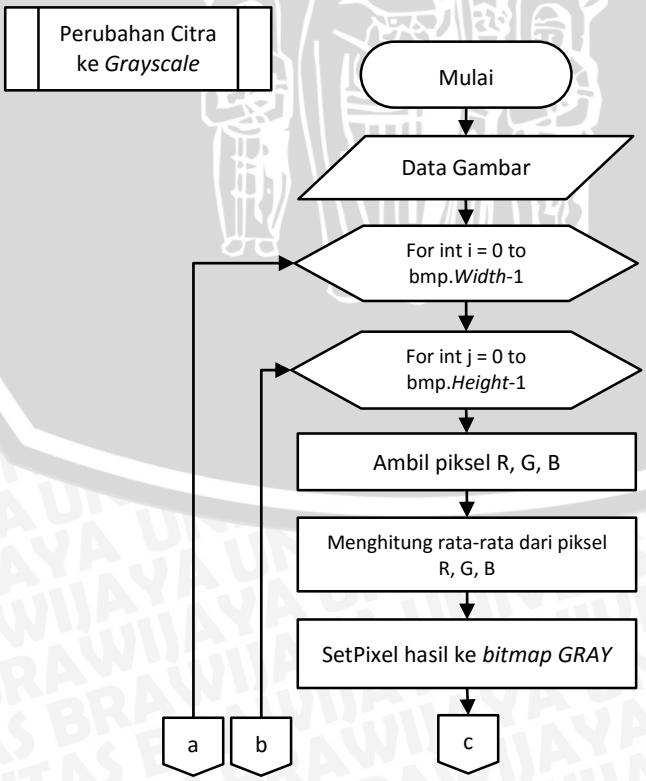
Deteksi objek bertujuan untuk mendeteksi objek yang akan dilacak dengan metode *blob analysis* nanti. Dengan adanya proses deteksi objek ini, pengambilan informasi atau pengolahan yang dilakukan terhadap citra akan lebih menghasilkan hasil yang maksimal. Pada proses ini pertama – tama sistem akan melakukan proses pengubahan data gambar ke citra *grayscale*. Proses ini bertujuan untuk mengubah 3 *channel* warna (*Red*, *Green*, *Blue*) pada citra menjadi 1 *channel* sehingga akan mempermudah pengolahan terutama pada proses *background subtraction* nanti. Selanjutnya citra hasil *grayscale* akan melalui proses *background subtraction* untuk mendeteksi adanya objek yang bisa dilacak dengan metode *blob analysis* atau tidak. Hasil yang didapatkan dari proses ini yaitu warna *foreground grayscale* objek yang berhasil dideteksi. Citra *background* yang lolos dari *background subtraction* akan dihilangkan dengan melakukan transformasi citra menjadi citra biner. Hasil dari citra biner diperkuat dengan operasi morfologi erosi dan dilasi untuk memperoleh hasil *blob* yang lebih baik. Setelah membentuk *blob* yang lebih baik, selanjutnya memberikan *bounding-box* pada *blob* tersebut dan *base line* untuk menandai daerah pelacakan. Diagram alir proses deteksi objek ditunjukkan pada Gambar 4.4.

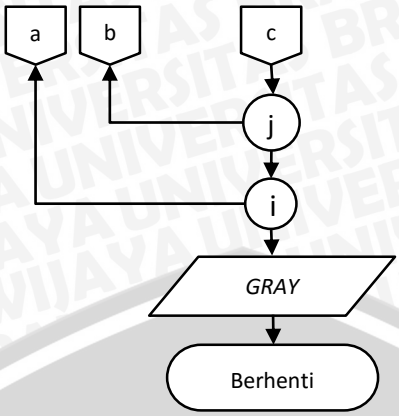




Gambar 4.4 Diagram Alir Proses Deteksi Objek

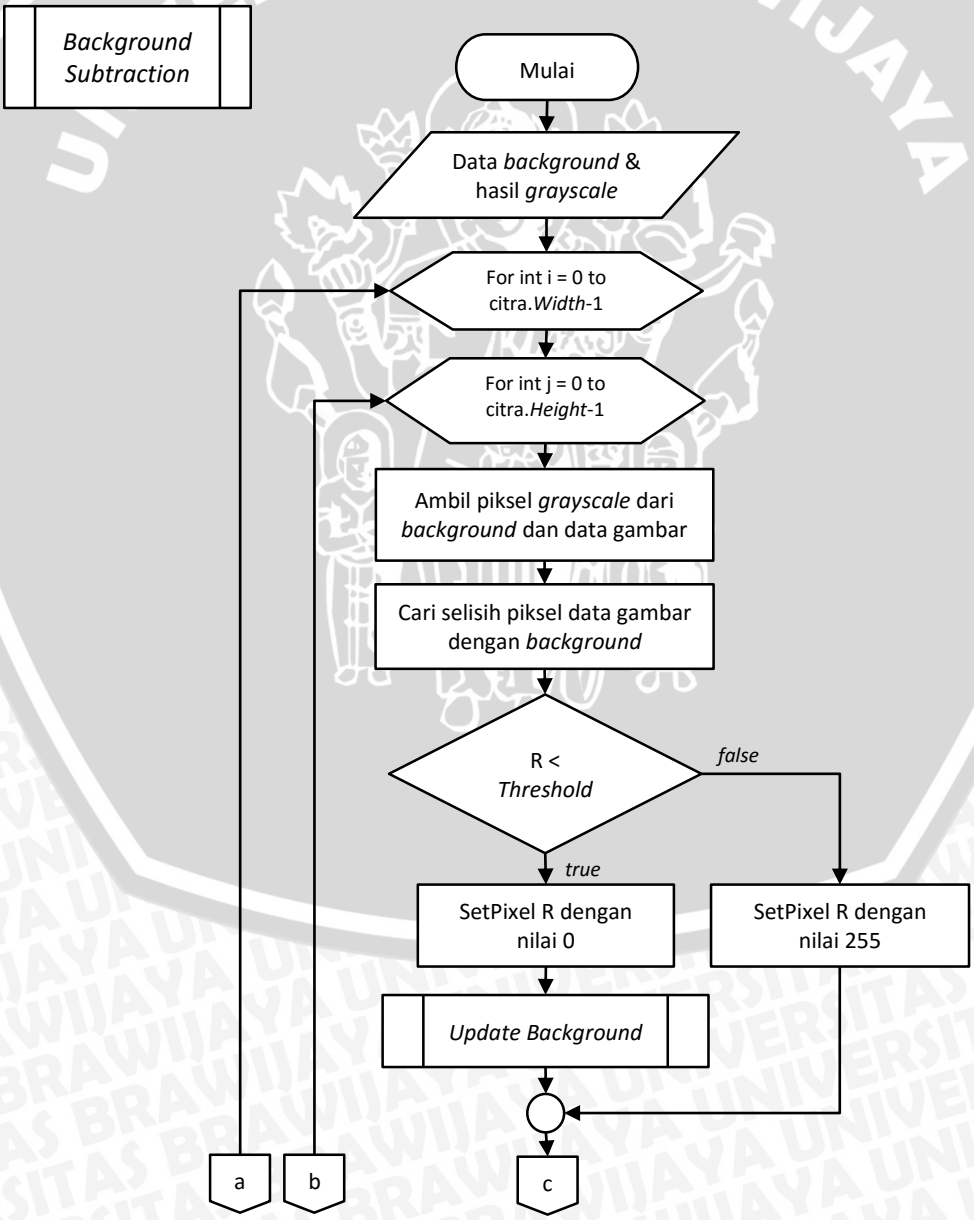
Pada proses deteksi objek, pertama – tama sistem akan mengubah tiga *channel* warna dari citra menjadi satu citra sehingga dihasilkan warna keabu – abuan (*grayscale*). Diagram alir sistem proses perubahan warna citra uji ke citra *grayscale* dapat dilihat pada Gambar 4.5.

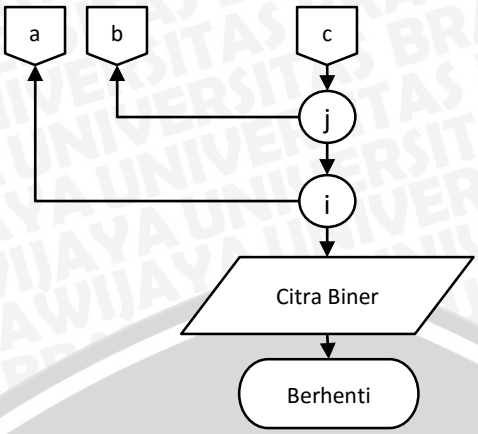




Gambar 4.5 Diagram Alir Proses Pengubahan Citra *Grayscale*

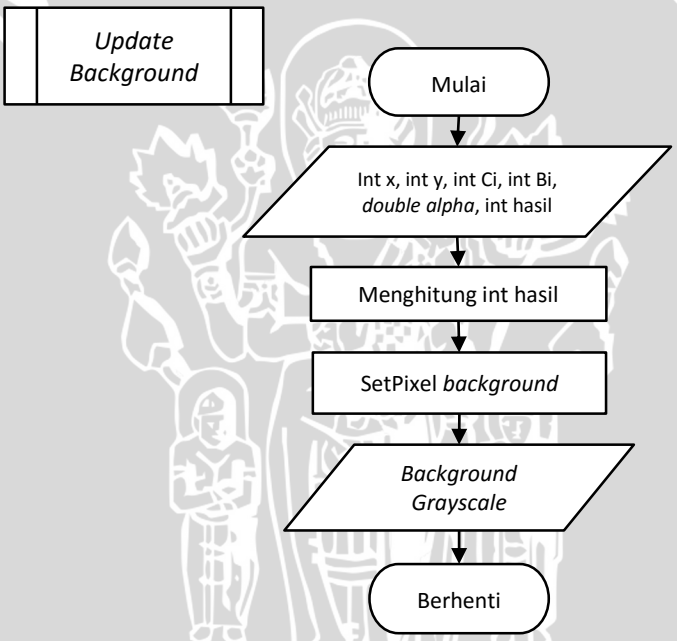
Selanjutnya dilakukan proses *background subtraction*. Diagram alir sistem untuk proses *background subtraction* ditunjukkan pada Gambar 4.6.





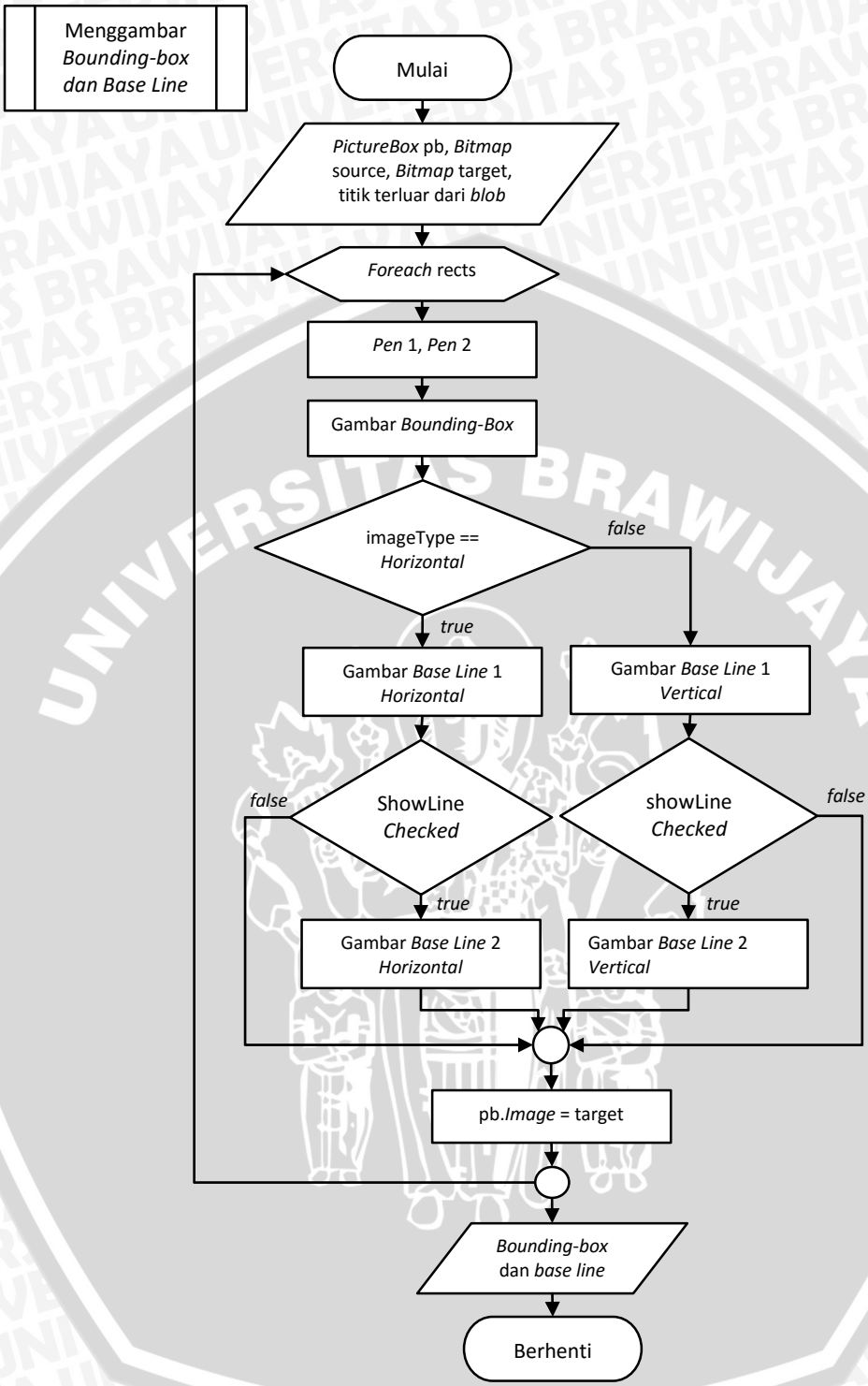
Gambar 4.6 Diagram Alir Proses *Background Substraction*

Pada proses *background subtraction* terdapat proses *update background* yang berfungsi untuk memperbarui *background*. Diagram alir *update background* diilustrasikan pada Gambar 4.7.



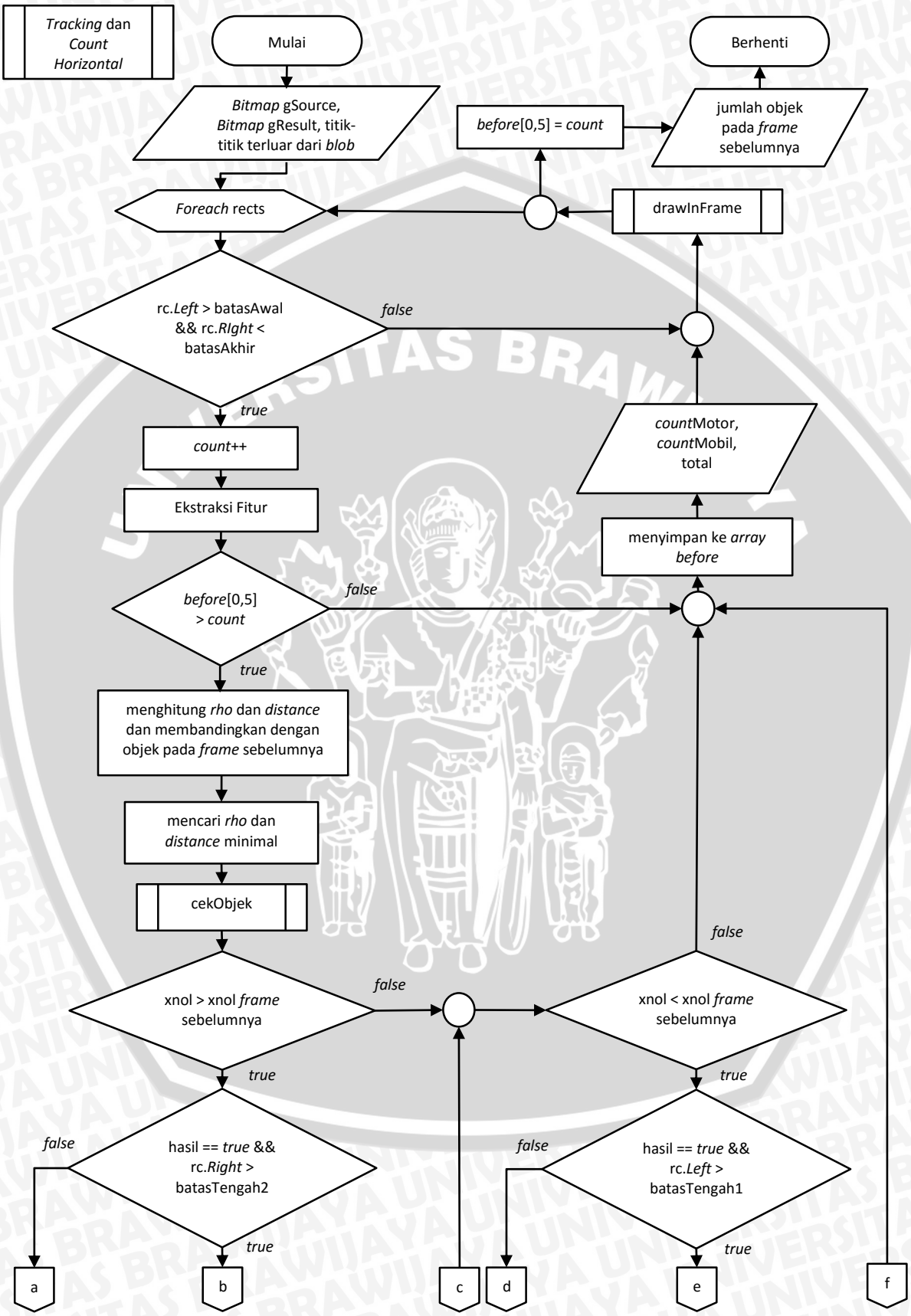
Gambar 4.7 Diagram Alir Proses *Update Background*

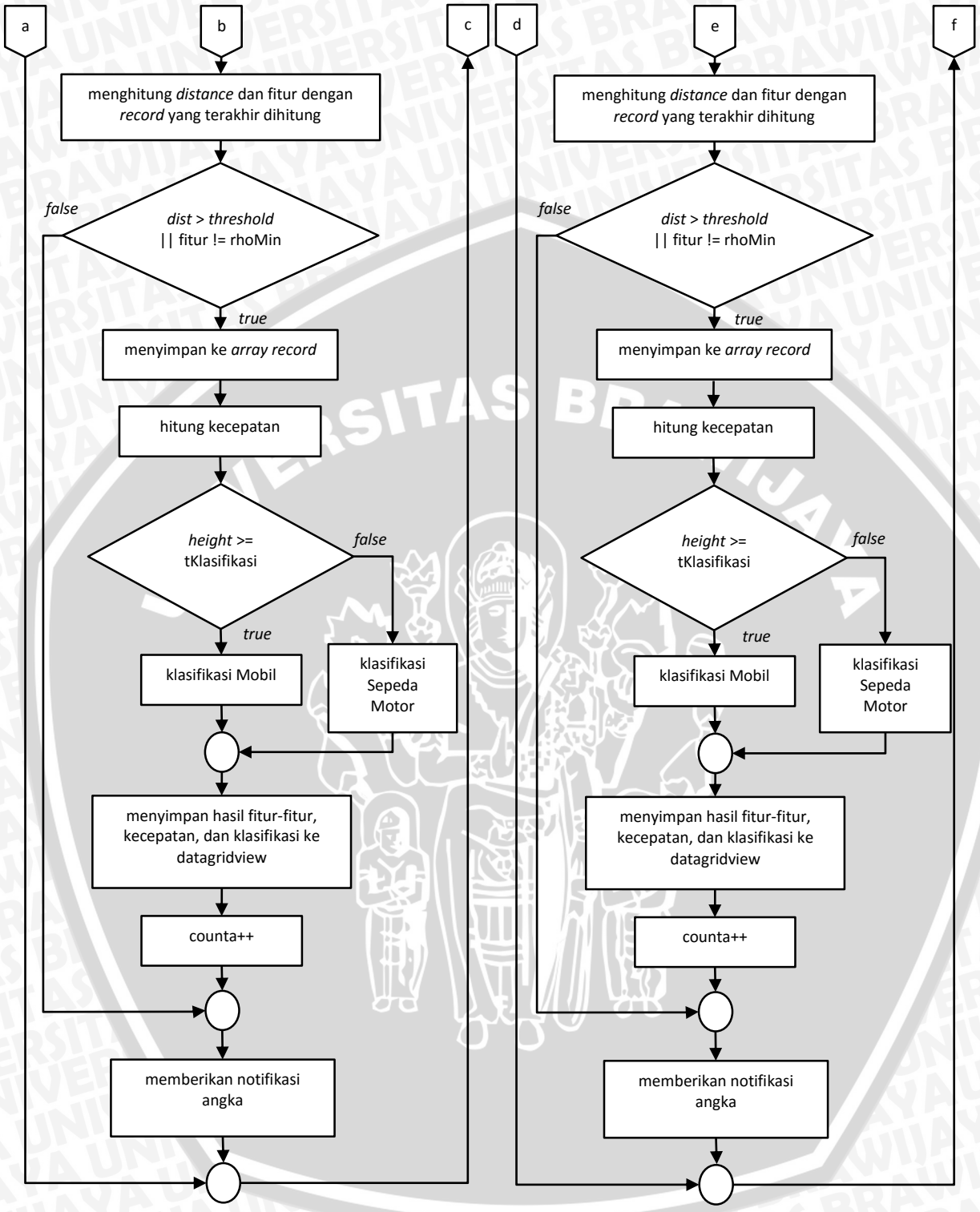
Setelah proses *background subtraction*, selanjutnya dilakukan oleh operasi morfologi untuk memperbaiki citra agar menjadi *blob* yang lebih baik. Proses ini diakukan oleh *library* AForge.NET meliputi *opening* (erosi dan dilasi) dengan *structure element* erosi 3x3 dan dilasi 17x17. Selanjutnya terdapat proses menggambar *bounding-box* dan *base line* yang berfungsi untuk menandai objek dan daerah pelacakan. Diagram alir proses menggambar *bounding-box* dan *base line* diilustrasikan pada Gambar 4.8.



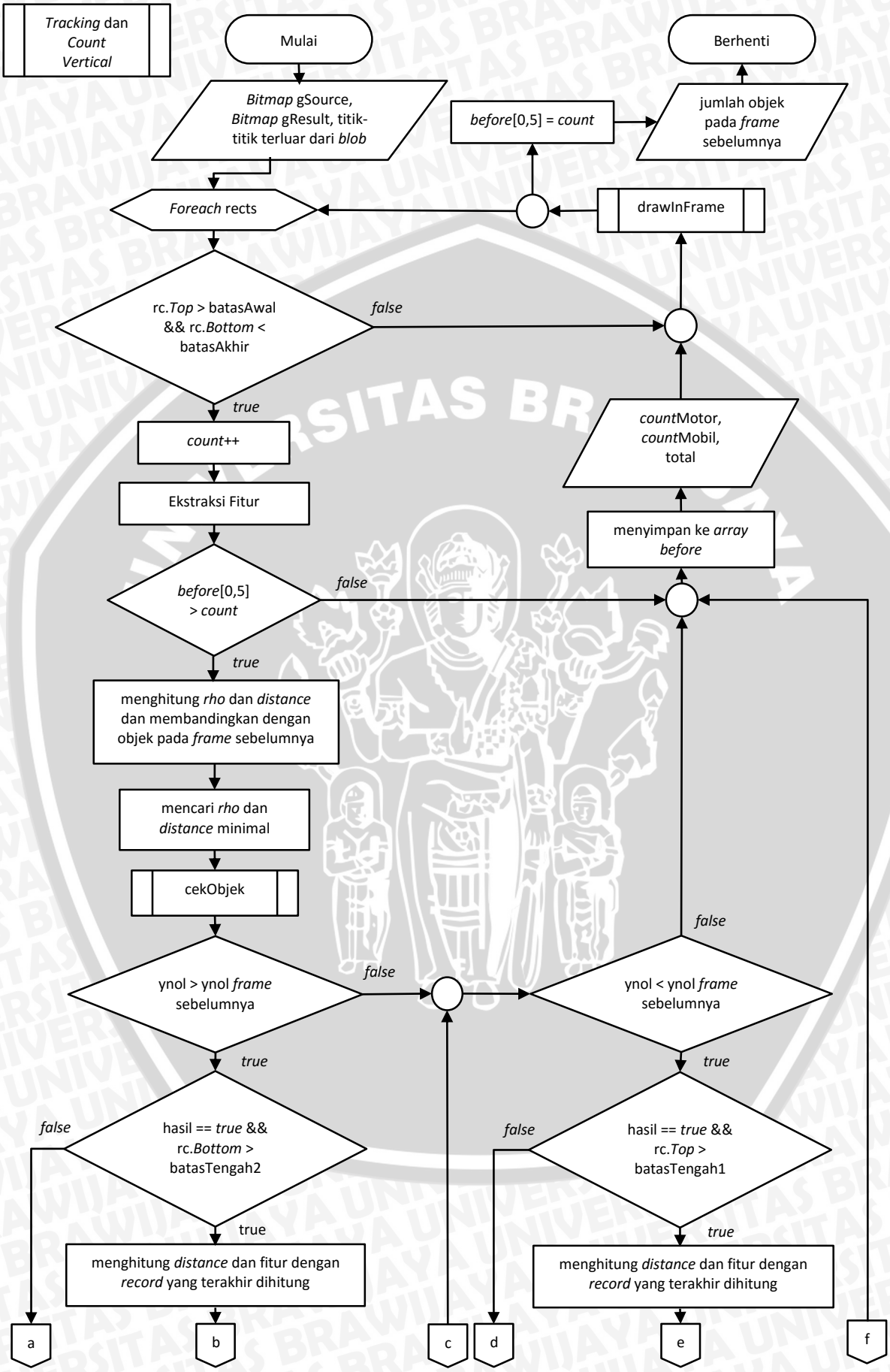
Gambar 4.8 Diagram Alir Proses Menggambar *Bounding-Box* dan *Base Line*

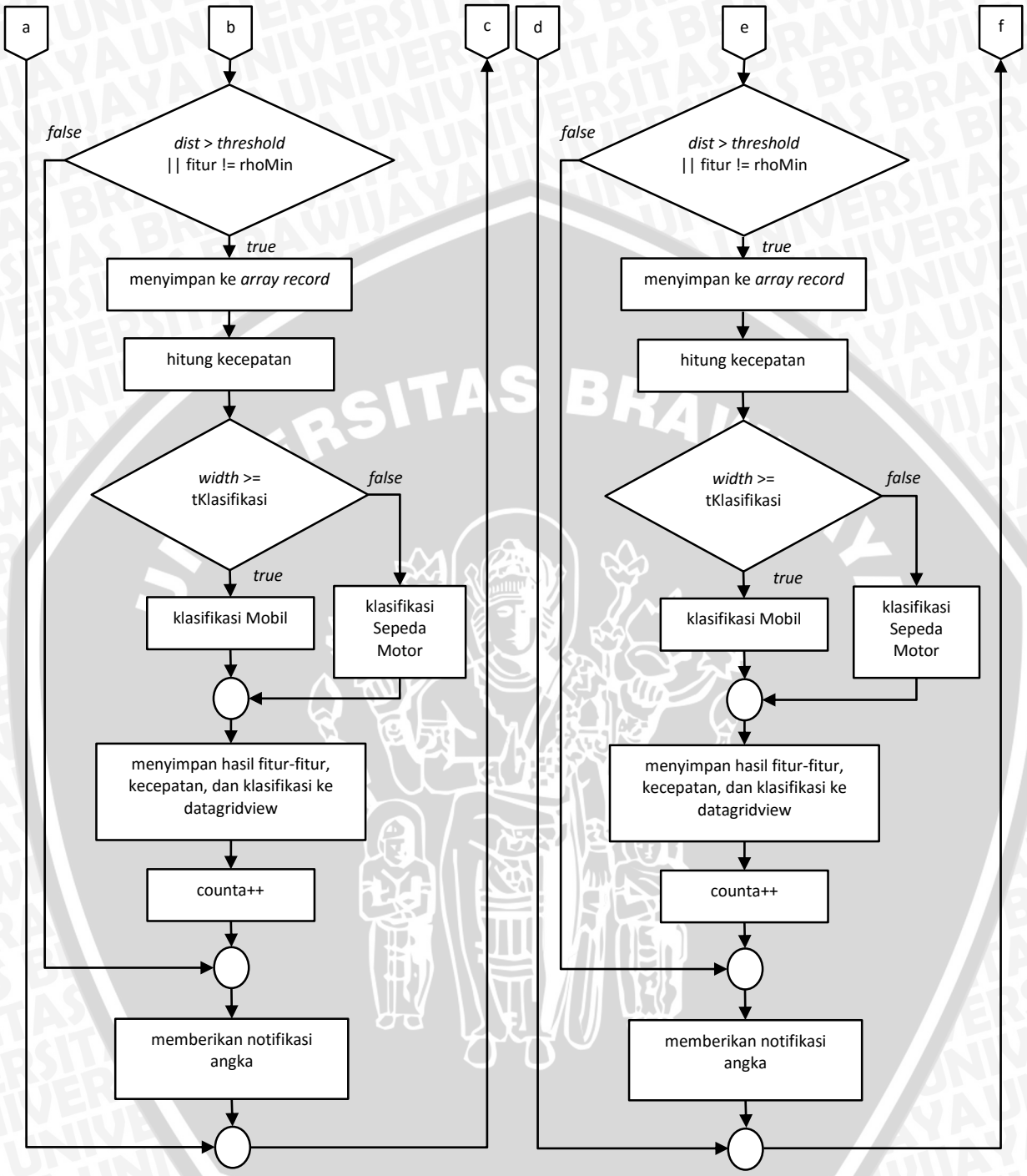
4.2.2 Tracking dan Count





Gambar 4.9 Diagram Alir Proses *Tracking* dan *Count Horizontal*


Tracking dan Count Vertical
Mulai
Bitmap gSource, Bitmap gResult, titik-titik terluar dari blob
Foreach rects
rc.Top > batasAwal && rc.Bottom < batasAkhir
false
true
count++
Ekstraksi Fitur
before[0,5] > count
false
true
menghitung rho dan distance dan membandingkan dengan objek pada frame sebelumnya
mencari rho dan distance minimal
cekObjek
ynol > ynol frame sebelumnya
false
true
hasil == true && rc.Bottom > batasTengah2
false
true
menghitung distance dan fitur dengan record yang terakhir dihitung
a
b
ynol < ynol frame sebelumnya
false
true
hasil == true && rc.Top > batasTengah1
false
true
menghitung distance dan fitur dengan record yang terakhir dihitung
c
d
e
f
menyimpan ke array before
countMotor, countMobil, total
drawInFrame
before[0,5] = count
jumlah objek pada frame sebelumnya
Berhenti

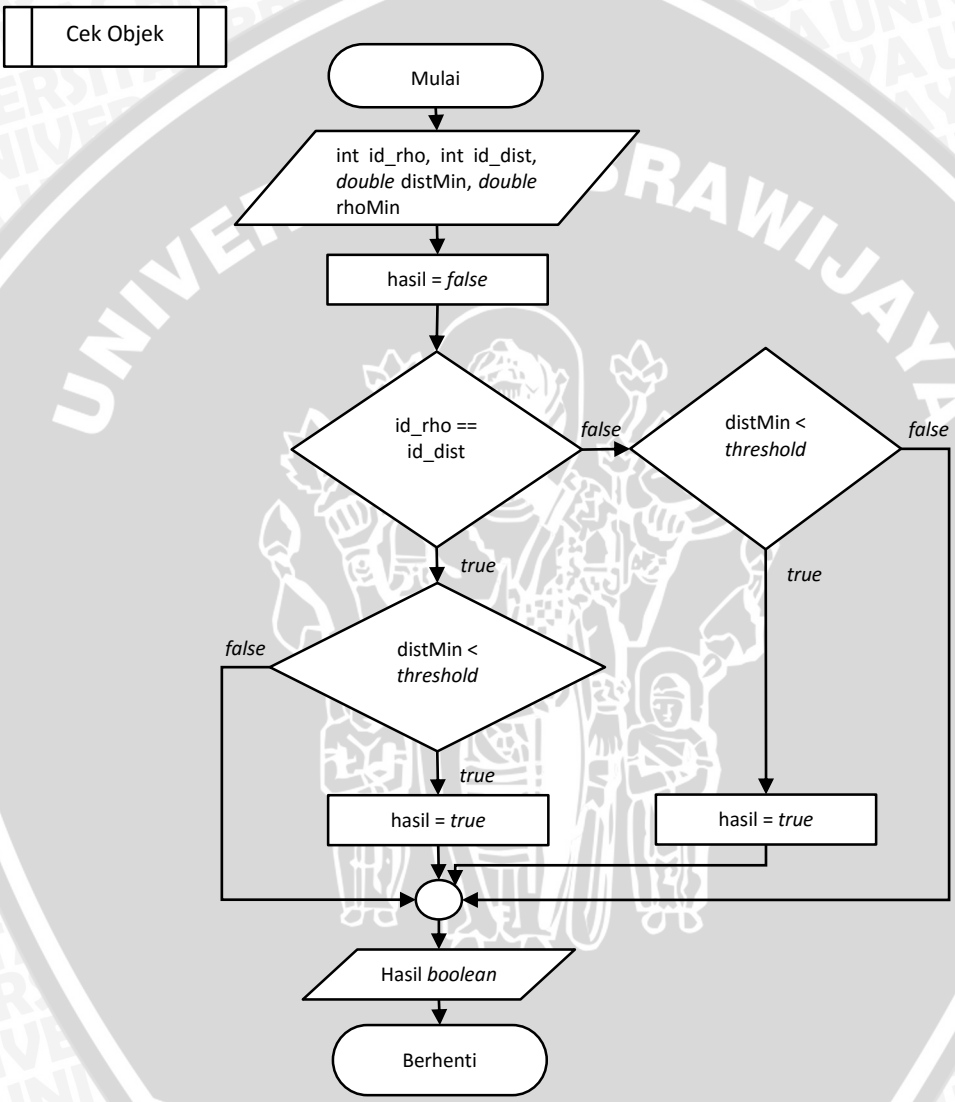


Gambar 4.10 Diagram Alir Proses *Tracking* dan *Count Vertical*

Tracking dan *Count* bertujuan untuk mendapatkan jumlah kendaraan yang berhasil terdeteksi. Proses ini dimulai dari mengekstraksi fitur – fitur dari *blob* hasil deteksi objek, kemudian dilakukan proses *tracking* untuk membandingkan antar *blob* adalah objek yang sama atau berbeda, dan setelah itu jika memenuhi kriteria tertentu objek akan dihitung, diklasifikasikan, dan dihitung kecepatannya. Proses *tracking* dan *count* dibedakan menjadi dua berdasarkan tipe gambar seperti pada

Gambar 4.3 yaitu *horizontal* dan *vertical*. Perbedaannya terletak pada saat menggambar *base line*, klasifikasi, dan menghitung kecepatan. Diagram alir proses *tracking* dan *count* dengan tipe gambar *horizontal* dan *vertical* diilustrasikan pada Gambar 4.9 dan Gambar 4.10.

Pada proses *tracking* dan *count* terdapat proses cek objek yang berfungsi untuk menentukan objek yang terdeteksi merupakan objek yang sama atau berbeda dengan *frame* sebelumnya. Diagram alir proses cek objek diilustrasikan pada Gambar 4.11.



Gambar 4.11 Diagram Alir Proses Cek Objek

4.3 Perhitungan Manual

Perhitungan manual berfungsi untuk memberikan gambaran umum perancangan sistem yang akan dibangun. Sistem *vehicle counting* pada penelitian ini menggunakan metode *blob analysis*. Pada manualisasi ini, proses yang dilakukan pada data video akan diwakilkan oleh contoh *frame bitmap* 8x8 untuk *background* dan *frame bitmap* 8x8 dari beberapa citra yang nantinya akan

diilustrasikan proses kalkulasi terhadapnya dan pada perhitungan manual nilai piksel *frame bitmap* sudah dalam keadaan *grayscale*. Proses *grayscale* yaitu mengubah tiga *channel* warna dari *piksel* menjadi hanya satu *channel* yakni warna abu-abu (*grayscale*). Proses pengubahan citra uji ke citra *grayscale* dilakukan dengan mencari nilai rata-rata dari tiga *channel* (*red*, *green*, *blue*) warna tiap *piksel*. Nilai rata-rata inilah yang akan menjadi nilai *channel* tunggal dari citra tersebut.

Contoh ilustrasi dari nilai piksel *frame bitmap* 8x8 pada beberapa citra tampak pada Gambar 4.12.

Frame 1:

213	40	200	203	176	3	230	245
242	164	260	18	1	35	60	170
40	34	60	243	221	223	75	65
81	37	112	225	49	104	10	21
106	108	91	34	224	59	10	175
248	128	93	107	18	246	144	202
82	209	61	221	18	32	250	33
117	246	195	95	201	68	216	39

Frame 2:

213	40	195	203	176	3	200	10
242	164	200	18	1	230	245	4
40	34	260	243	221	60	170	198
81	37	60	225	49	75	65	21
106	108	91	34	224	10	10	175
248	128	93	107	18	246	144	202
82	209	61	221	18	32	250	33
117	246	195	95	201	68	216	39

Frame 3:

213	40	195	203	176	3	200	10
242	164	235	18	1	35	21	4
40	200	37	243	230	245	24	198
81	260	112	225	60	170	117	21
106	60	91	34	75	65	10	175
248	128	93	107	10	246	144	202
82	209	61	221	18	32	250	33
117	246	195	95	201	68	216	39

Frame 4:

213	40	195	203	176	3	200	10
242	164	235	18	1	35	21	4
40	34	37	243	221	223	24	198
81	200	112	225	230	245	117	21
106	260	91	34	60	170	10	175
248	60	93	107	75	65	144	202
82	209	61	221	10	32	250	33
117	246	195	95	201	68	216	39

Frame 5:

213	40	195	203	176	3	200	10
242	164	235	18	1	35	21	4
40	34	37	243	221	223	24	198
81	37	112	225	49	104	117	21
106	200	91	34	230	245	10	175
248	260	93	107	60	170	144	202
82	60	61	221	75	65	250	33
117	246	195	95	10	68	216	39

Gambar 4.12 Ilustrasi Nilai Piksel *Frame Bitmap* 8x8 untuk *Frame* 5

Kemudian contoh nilai piksel *frame bitmap* 8x8 untuk data *background* yang akan digunakan pada manualisasi ini tampak pada Gambar 4.13.

213	40	195	203	176	3	200	10
242	164	235	18	1	35	21	4
40	34	37	243	221	223	24	198
81	37	112	225	49	104	117	21
106	108	91	34	224	59	10	175
248	128	93	107	18	246	144	202
82	209	61	221	18	32	250	33
117	246	195	95	201	68	216	39

Gambar 4.13 Ilustrasi Nilai Piksel *Frame Bitmap* 8x8 untuk *Background*

4.3.1 Tahap Deteksi Objek

Tahap pertama dari deteksi objek adalah merubah data uji menjadi citra *grayscale* seperti Gambar 4.12 dan Gambar 4.13. Kemudian melakukan *background subtraction* yaitu tahap untuk mencari nilai selisih atau *different image* dari *frame* dengan *background*. Untuk frame pertama hanya melakukan pengurangan *frame* dengan *background* original dan untuk *frame – frame* selanjutnya *background* merupakan hasil *background updating* yang merupakan kombinasi *background* sebelumnya dengan *frame* yang digunakan dan perubahan yang terjadi pada saat *background updating* bergantung pada hasil citra biner. Hasil perhitungan proses *background subtraction* untuk *frame* 1 dapat dilihat pada Gambar 4.14.

0	0	5	0	0	0	30	235
0	0	25	0	0	0	39	166
0	0	23	0	0	0	51	-133
0	0	0	0	0	0	-107	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

Gambar 4.14 Hasil Proses *Background Subtraction* untuk *Frame* 1

Setelah mendapatkan hasil *background subtraction frame* 1, kemudian menyeleksi hasil tersebut dengan *threshold* untuk menjadi citra biner (Persamaan 2.2). *Threshold* yang digunakan untuk contoh manualisasi adalah 5. Jadi jika nilai *absolute* piksel kurang dari 5 maka diganti menjadi 0, dan yang lainnya diganti 255. Citra biner untuk frame 1 dapat dilihat pada Gambar 4.15.

0	0	255	0	0	0	255	255
0	0	255	0	0	0	255	255
0	0	255	0	0	0	255	255
0	0	0	0	0	0	255	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

Gambar 4.15 Citra Biner untuk *Frame* 1

Citra biner menentukan piksel yang akan di-*update* untuk proses *background updating*. Pada Persamaan 2.3 piksel yang di *update* adalah piksel yang mempunyai nilai 0 dan untuk nilai faktor pembobot (α) untuk contoh perhitungan manual adalah 0.1. Berikut contoh perhitungan *background updating* pada piksel[0,0] dan piksel[0,2] dari hasil citra biner *frame* 1:

$$B_2(0,0), R = 0 => B_2(0,0) = ((1-0{,}1)213) + (0.1 \times 213) = 213$$

$$B_2(0,2), R = 255 => B_2(0,2) = 195$$

Hasil tersebut menggantikan nilai *background* sebelumnya. Hasil *background updating*, *background subtraction,* dan citra biner *frame* 2 dapat dilihat pada Gambar 4.16

Background Updating Frame 2

213	40	195	203	176	3	200	10
242	164	235	18	1	35	21	4
40	34	37	243	221	223	24	198
81	37	112	225	49	104	117	21
106	108	91	34	224	59	10	175
248	128	93	107	18	246	144	202
82	209	61	221	18	32	250	33
117	246	195	95	201	68	216	39

Background Subtraction Frame 2

0	0	0	0	0	0	0	0
0	0	-35	0	0	195	224	0
0	0	223	0	0	-163	146	0
0	0	-52	0	0	-29	-52	0
0	0	0	0	0	-49	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

Citra Biner *Frame* 2

0	0	0	0	0	0	0	0
0	0	255	0	0	255	255	0
0	0	255	0	0	255	255	0
0	0	255	0	0	255	255	0
0	0	0	0	0	255	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

Gambar 4.16 Hasil *Background Updating*, *Background Subtraction*, dan Citra Biner *Frame* 2

Setelah semua *frame* menjadi citra biner, tahap selanjutnya adalah operasi morfologi. Operasi morfologi yang digunakan untuk contoh perhitungan manual adalah *opening* (*erosi* dan *dilasi*) dengan *structure element* 3x1. Implementasi operasi morfologi tergantung pada citra yang dihasilkan, jika dengan *opening* saja citra masih mempunyai *noise* atau titik – titik putih yang tidak diperlukan maka bisa menambahkan *erosi* dan jika sebaliknya bisa menambakan *dilasi* atau mengubah ukuran *structure element* sampai menghasilkan *blob* yang lebih baik. Ilustrasi operasi morfologi pada citra biner *frame* 1 ditunjukkan pada Gambar 4.17.

Structure Element

255
255
255

Citra Biner *Frame* 1

0	0	255	0	0	0	255	255
0	0	255	0	0	0	255	255
0	0	255	0	0	0	255	255
0	0	0	0	0	0	255	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

Erosi

0	0	0	0	0	0	0	0
0	0	255	0	0	0	255	255
0	0	0	0	0	0	255	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

Dilasi

0	0	255	0	0	0	255	255
0	0	255	0	0	0	255	255
0	0	255	0	0	0	255	255
0	0	0	0	0	0	255	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

Gambar 4.17 Ilustrasi Operasi Morfologi *Frame* 1

Frame 1:

0	0	255	0	0	0	255	255
0	0	255	0	0	0	255	255
0	0	255	0	0	0	255	255
0	0	0	0	0	0	255	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

Frame 2:

0	0	0	0	0	0	0	0
0	0	255	0	0	255	255	0
0	0	255	0	0	255	255	0
0	0	255	0	0	255	255	0
0	0	0	0	0	255	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

Frame 3:

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	255	0	0	255	255	0	0
0	255	0	0	255	255	0	0
0	255	0	0	255	255	0	0
0	0	0	0	255	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Frame 4:

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	255	0	0	255	255	0	0
0	255	0	0	255	255	0	0
0	255	0	0	255	255	0	0
0	0	0	0	255	0	0	0
0	0	0	0	0	0	0	0

Frame 5:

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	255	0	0	255	255	0	0
0	255	0	0	255	255	0	0
0	255	0	0	255	255	0	0
0	0	0	0	255	0	0	0

→ Garis kuning

→ Garis merah

Gambar 4.18 Ilustrasi Operasi Morfologi dengan Pemberian *Bounding-box* dan *Base Line*

Gambar 4.18 merupakan ilustrasi hasil operasi morfologi dengan pemberian *bounding-box* dan *base line* untuk seluruh *frame*. Pemberian *bounding-box* berfungsi untuk menandai objek dengan kotak merah dengan batas piksel terluar objek dan pemberian *base line* untuk menandai daerah *tracking* objek dan menangkap nilai objek. Terdapat dua macam *base line*, *base line* terluar untuk daerah *tracking* dan *base line* di dalam untuk menangkap nilai objek. Untuk manualisasi ini dianggap sudah memasuki daerah *tracking* tetapi belum keluar dari daerah *tracking* dan pergerakan objek dari atas ke bawah hanya satu arah.

Garis berwarna merah adalah *base line* terluar yang menandakan daerah *tracking* dan merupakan dimulainya proses *tracking*. Jadi jika objek belum memasuki atau sudah melebihi daerah tracking maka proses *tracking* tidak akan dimulai atau sudah dihentikan. Sedangkan garis berwarna kuning merupakan batas untuk menangkap nilai objek.

4.3.2 Tahap Tracking dan Count

Tahap pertama yang dilakukan dalam *tracking* dan *count* adalah ekstraksi fitur dan perhitungan nilai *centroid* dari objek yang sudah diberi *bounding-box* pada proses deteksi objek. Fitur yang digunakan adalah *dispersedneess*, *aspect ratio*, dan *area ratio*. Fitur didapatkan dengan mencari *width*, *height*, *perimeter*, *area*, *dan* ROI terlebih dahulu. *Width* dan *height* adalah lebar dan tinggi dari *bounding-box*, *perimeter* adalah keliling dari *bounding-box*, *area* adalah luas dari *frame*, dan ROI adalah luas dari *bounding-box*. Hasil ekstraksi fitur dan perhitungan *centroid* dapat dilihat pada Tabel 4.1 dan Tabel 4.2. Berikut contoh perhitungan fitur dan *centroid* berdasarkan Persamaan 2.4, Persamaan 2.5, Persamaan 2.6, dan Persamaan 2.7 dari *frame* 1:

Frame 1

0	0	255	0	0	0	255	255
0	0	255	0	0	0	255	255
0	0	255	0	0	0	255	255
0	0	0	0	0	0	255	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

Objek 1

$$Dispersedness = \frac{8^2}{64} = \frac{64}{64} = 1$$

$$AspectRatio = \frac{3}{1} = 3$$

$$AreaRatio = \frac{64}{3} = 21.33$$

$Centroid$:

$$X_0 = \frac{(2)\times 3}{(1)\times 3} = 2$$

$$Y_0 = \frac{(0+1+2)\times 1}{(1+1+1)\times 1} = \frac{3}{3} = 1$$

Tabel 4.1 Hasil Ekstraksi Fitur

Frame		Diketahui		*Dispersedness*	*Aspect Ratio*	*Area Ratio*
1	Objek 1	Width Height Perimeter Area ROI	1 2 6 64 2	1	3	21.33
	Objek 2	Width Height Perimeter Area ROI	2 3 10 64 6	1.563	1.5	10.67
2	Objek 1	Width Height Perimeter Area ROI	1 3 8 64 3	1	3	21.33
	Objek 2	Width Height Perimeter Area ROI	2 4 12 64 8	2.25	2	8
3	Objek 1	Width Height Perimeter Area ROI	1 3 8 64 3	1	3	21.33
	Objek 2	Width Height Perimeter Area ROI	2 4 12 64 8	2.25	2	8
4	Objek 1	Width Height Perimeter Area ROI	1 3 8 64 3	1	3	21.33
	Objek 2	Width Height Perimeter Area ROI	2 4 12 64 8	2.25	2	8
5	Objek 1	Width Height Perimeter Area ROI	1 3 8 64 3	1	3	21.33

Tabel 4.2 Hasil Perhitungan *Centroid*

Frame		*Centroid*	
		X_0	Y_0
1	Objek 1	2	1
	Objek 2	6.5	2
2	Objek 1	2	2
	Objek 2	5.5	2.5
3	Objek 1	1	3
	Objek 2	4.5	3.5
4	Objek 1	1	4
	Objek 2	4.5	4.5
5	Objek 1	1	5

Setelah itu hasil ekstraksi fitur dan perhitungan *centroid* digunakan proses *tracking* yaitu membandingkan objek pada *frame* saat ini dengan objek pada *frame* sebelumnya untuk mengetahui apakah objek pada *frame* saat ini dengan objek pada *frame* sebelumnya adalah objek yang sama atau objek yang berbeda dengan menggunakan fungsi pencocokan (Persamaan 2.8 dan Persamaan 2.9) dan dicari nilai *minimal*-nya. Hasil perbandingan seluruh *frame* dapat dilihat pada Tabel 4.3. Contoh perhitungan berdasarkan Persamaan 2.8 dan persamaan 2.9 untuk *frame* saat ini (P_{obj}) adalah *frame* 2 dengan 2 objek dan *frame* sebelumnya (P_{buff}) adalah *frame* 1 dengan 2 objek (*background* tidak dihitung karena tidak mempunyai objek):

- Objek 1 *frame* 2 terhadap objek 1 *frame* 1

$$\rho = |1-1| \times |3-3| \times |21.33-21.33| = 0$$

$$dist = \sqrt{(2-2)^2 + (2-1)^2} = 1$$

- Objek 1 *frame* 2 terhadap objek 2 *frame* 1

$$\rho = |1-1.56| \times |3-1.5| \times |21.33-10.67| = 9$$

$$dist = \sqrt{(2-6.5)^2 + (2-2)^2} = 4.5$$

- Objek 2 *frame* 2 terhadap objek 1 *frame* 1

$$\rho = |2.25-1| \times |2-3| \times |8-21.33| = 16.67$$

$$dist = \sqrt{(5.5-2)^2 + (2.5-1)^2} = 3.808$$

- Objek 2 *frame* 2 terhadap objek 2 *frame* 1

$$\rho = |2.25-1.56| \times |2-1.5| \times |8-10.67| = 0.917$$

$$dist = \sqrt{(5.5-6.5)^2 + (2.5-2)^2} = 1.118$$

Tabel 4.3 Hasil Perbandingan Nilai Fitur (*ρ*) dan Jarak antar *Centroid* (*dist*)

Frame	Objek		*ρ* (*rho*)	*ρ* (*rho*) *minimal*	*dist*	*dist minimal*
	Current	*Previous*				
P_{obj} = 2 P_{buff} = 1	1	1	0	0	1	1
		2	9		4.5	
	2	1	16.67	0.917	3.808	1.118
		2	0.917		1.118	
P_{obj} = 3 P_{buff} = 2	1	1	0	0	1.414	1.414
		2	16.67		4.528	
	2	1	16.67	0	2.915	1.414
		2	0		1.414	
P_{obj} = 4 P_{buff} = 3	1	1	0	0	1	1
		2	16.67		3.536	
	2	1	16.67	0	3.808	1
		2	0		1	
P_{obj} = 5 P_{buff} = 4	1	1	0	0	1	1
		2	16.67		3.536	

Pada proses perbandingan, objek masih belum dihitung, tetapi pada saat objek melewati *base line* berwarna kuning (Gambar 4.18) dan memenuhi kriteria tertentu untuk dihitung sebagai satu objek. Kriterianya adalah:

1. Jika pada *frame* saat ini dan *frame* sebelumnya nilai *rho minimal* dan *dist minimal* terdapat pada objek yang sama, maka apakah nilai *dist minimal* berada dibawah threshold atau tidak. Jika benar, maka dianggap sebagai 1 objek.
2. Jika pada *frame* saat ini dan *frame* sebelumnya nilai *rho minimal* dan *dist minimal* terdapat pada objek yang berbeda, maka nilai *dist minimal* apakah berada dibawah *threshold* atau tidak. Jika benar, maka dianggap 1 objek.
3. Selain kedua kriteria maka dianggap objek yang berbeda.

Untuk perhitungan manual *threshold* yang ditentukan adalah 3. Pada Gambar 4.18, objek 2 pada *frame* 4 melewati *base line* berwarna kuning maka menandakan mulai proses pengambilan nilai objek dan *frame* saat ini adalah *frame* 4 dan sebelumnya adalah *frame* 3. Jika dilihat pada Tabel 4.3 menunjukkan P_{obj} = 4 dan P_{buff} = 3 pada objek 2 memenuhi kriteria pertama, yaitu nilai *rho minimal* dan *dist minimal* (ditandai blok kuning) kurang dari 3 maka objek 2 pada *frame* saat ini dan objek 2 pada *frame* sebelumnya dianggap 1 objek dan *count* bertambah 1. Dan pada *frame* 5 objek 1 melewati *base line* kuning juga memenuhi kriteria pertama, tetapi kemudian dihitung kembali apakah jarak dengan objek yang terakhir dihitung yaitu objek 2 mempunyai nilai lebih dari *threshold*. Jika nilai lebih dari *threshold*, maka dianggap sebagai 1 objek dan *count* bertambah 1. Contoh perhitungan jarak objek 2 pada *frame* 4 dengan objek 1 pada *frame* 5:

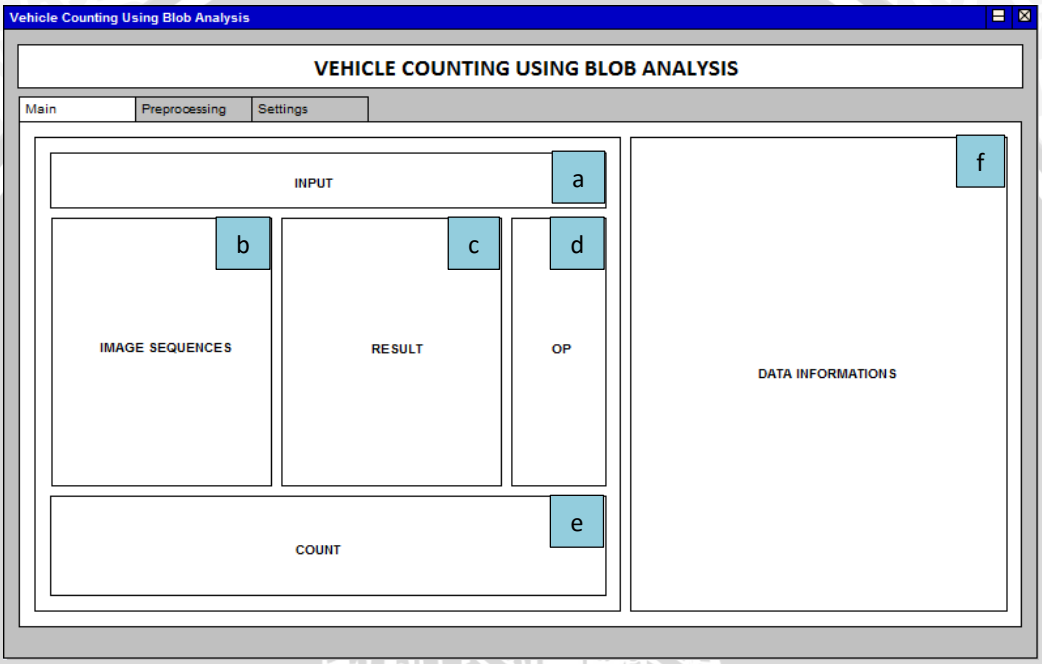
$$dist = \sqrt{(1-4.5)^2 + (5-4.5)^2} = 3.54$$

Jarak objek 2 pada *frame* 4 dengan objek 1 pada *frame* 5 adalah 3.54 yang berarti lebih dari *threshold* dan menunjukkan bahwa objek 1 berbeda dari objek 2. Jadi total objek yang berhasil dihitung adalah 2.

4.4 Perancangan *User Interface*

Perancangan antarmuka bertujuan untuk mewakili keadaan sebenarnya dari sistem yang akan dibangun. Sistem *vehicle counting* dengan metode *blob analysis* terdiri dari tiga halaman yaitu halaman *main*, halaman *preprocessing*, dan halaman *settings*.

1. Perancangan antarmuka pengguna halaman *main*.

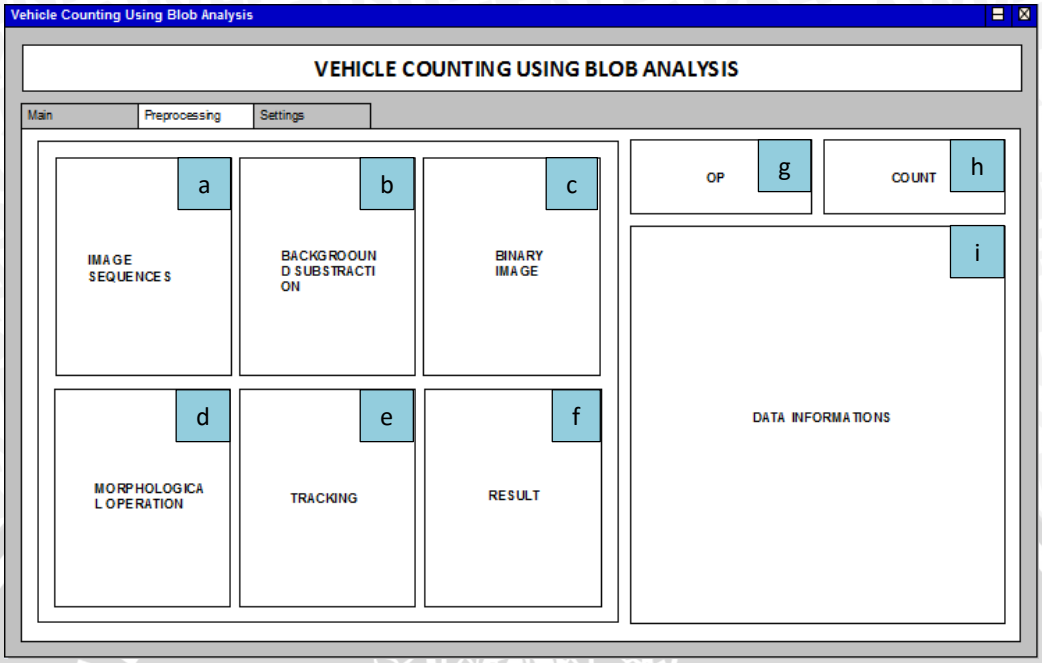


Gambar 4.19 Perancangan Antarmuka Halaman *Main*

Halaman *main* menampilkan hasil vehicle *counting* dan memasukkan data uji. Ilustrasi halaman *main* dapat dilihat pada Gambar 4.19. Halaman ini terdiri dari beberapa bagian antara lain:

a. *Input* untuk memasukkan data berupa folder yang berisi kumpulan gambar dari video yang sudah diekstraksi. Terdiri dari label dan openfolderdialog.
b. Picturebox *Image Sequences* untuk menampilkan gambar *frame by frame* dari hasil ekstraksi video original.
c. Picturebox *Result* untuk menampilkan gambar *frame by frame* beserta hasil *counting*.
d. *Operation* terdiri dari tombol *Play*, *Pause*, *Resume*, dan *Stop*. Tombol *Play* untuk menjalankan video. Tombol *Pause* untuk menhentikan sementara video. Tombol *Resume* untuk melanjutkan pemutaran video yang dihentikan sementara. Tombol *Stop* untuk mengentikan video.
e. *Count* untuk menampilkan hasil dari proses *counting* dan klasifikasi.
f. Data *Informations* menampilkan informasi dari setiap kendaraan yang terdeteksi.

2. Perancangan antarmuka pengguna halaman *preprocessing*.

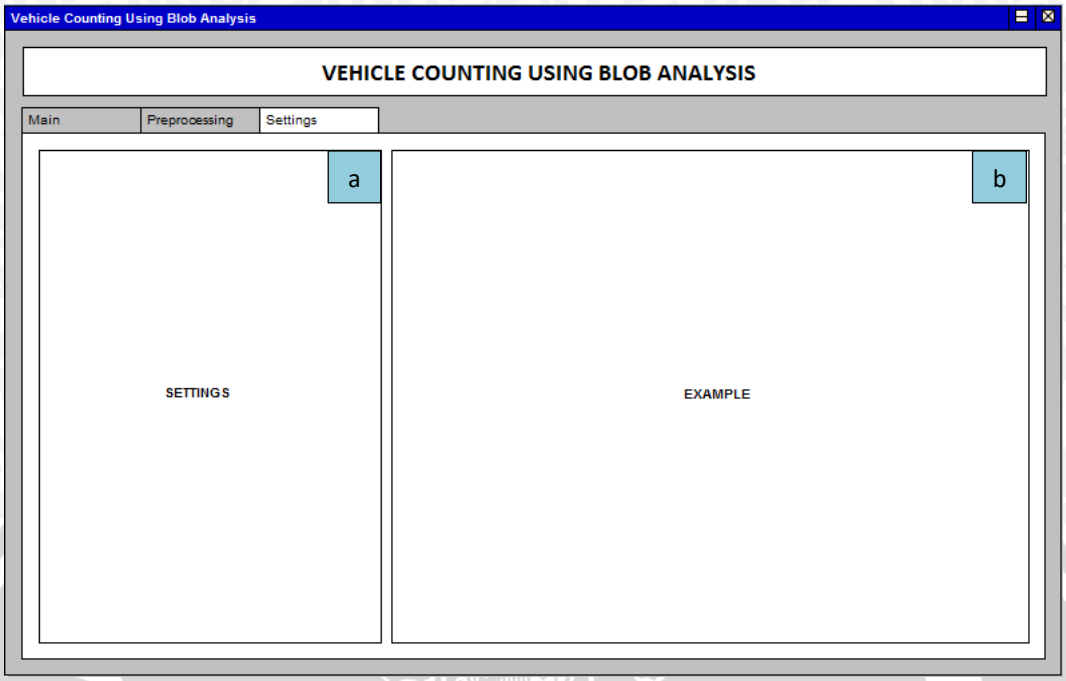


Gambar 4.20 Perancangan Antarmuka Halaman *Preprocessing*

Halaman *preprocessing* menampilkan proses yang terjadi mulai dari *image sequences* original sampai mendapatkan hasil *vehicle counting*. Ilustrasi halaman *preprocessing* dapat dilihat pada Gambar 4.20. Halaman ini terdiri dari beberapa bagian antara lain:

a. Picturebox *Image Sequences* untuk menampilkan gambar *frame by frame* dari hasil ekstraksi video original.
b. Picturebox *Background Subtraction* untuk menampilkan hasil dari pengurangan gambar saat ini dengan *background* gambar yang sudah ditentukan.
c. Picturebox *Binary Image* untuk menampilkan hasil dari *background subtraction* yang kemudian diseleksi dengan *threshold* dan diubah menjadi *binary image*.
d. Picturebox *Morphological Operation* untuk menampilkan hasil dari proses *morphological operation* yang terdiri dari *opening*.
e. Picturebox *Tracking* untuk menampilkan hasil dari deteksi kendaraan.
f. Picturebox *Result* untuk menampilkan gambar *frame by frame* beserta hasil *counting*.
g. Operation terdiri dari tombol *Pause* dan *Resume*. Tombol *Pause* untuk menghentikan gambar sementara. Tombol *Resume* untuk melanjutkan pemutaran gambar yang dihentikan sementara.
h. *Count* untuk menampilkan hasil dari proses *counting* dan klasifikasi.
i. Data *Informations* menampilkan proses kendaraan yang terdeteksi.

3. Perancangan antarmuka halaman *settings*.



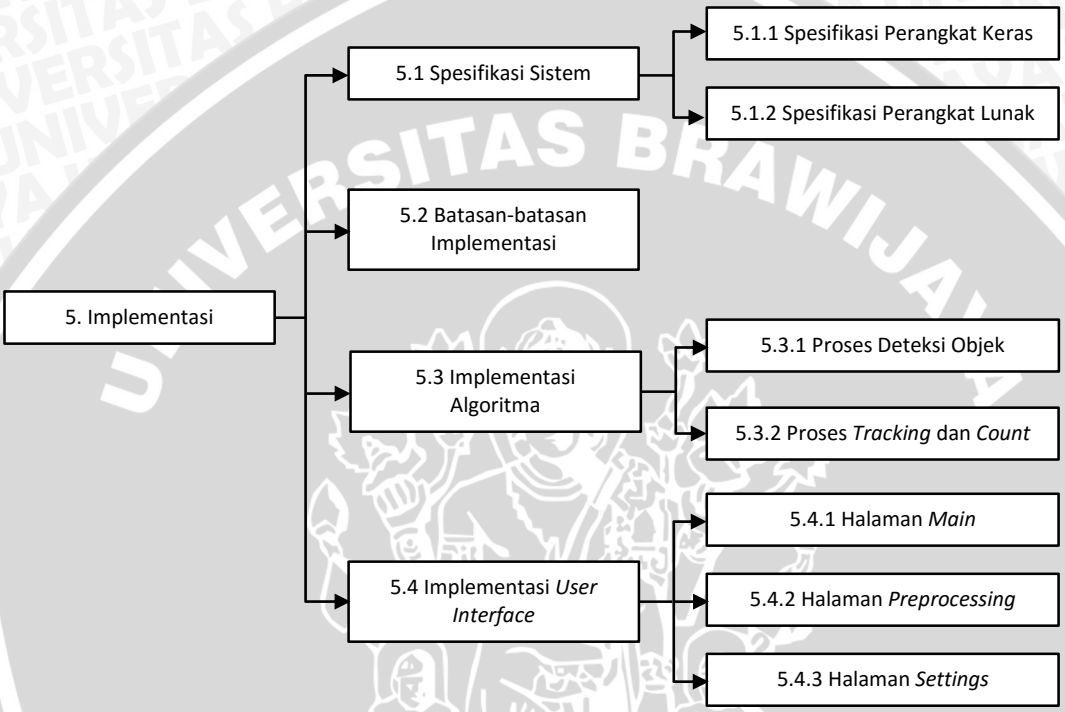
Gambar 4.21 Perancangan Antarmuka Halaman *Settings*

Halaman *settings* befungsi untuk memberikan masukan berupa pengaturan yang dibutuhkan sistem. Ilustrasi halaman *settings* dapat dilihat pada Gambar 4.21. Halaman ini terdiri dari beberapa bagian antara lain:

a. Grupbox *settings* beriisi textbox untuk memberikan masukan berupa pengaturan yang dibutuhkan. Terdiri dari pengaturan gambar, *base line*, dan *threshold*.
b. Grupbox *example* berfungsi memberikan contoh cara mengisi pengaturan.

BAB 5 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*. Tahap – tahap pembahasan implementasi yang dikerjakan digambatkan pada Gambar 5.1.



Gambar 5.1 Pohon Implementasi

5.1 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 yang dapat diimplementasikan pada spesifikasi perangkat keras dan perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras meliputi spesifikasi minimal dari prosesor, memori, dan hardisk yang akan digunakan pada sistem ini dapat dilihat pada Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat Keras Komputer

Nama Komponen	Spesifikasi
Prosesor	Intel® Core™ i3-2330M CPU @ 2.20GHz
Memori (RAM)	4 GB
Hardisk	WD kapasitas 500 GB

5.1.2 Spesifikasi Perangkat Lunak

Pengembangan subsistem menggunakan perangkat lunak dengan spesifikasi yang dijelaskan pada Tabel 5.2.

Tabel 5.2 Spesifikasi Perangkat Lunak Komputer

Nama Komponen	Spesifikasi
Sistem Operasi	Windows 10 Pro 64 bit
Tools pemrograman	Visual Studio 2010 Ultimate

5.2 Batasan – Batasan Implementasi

Batasan – batasan dalam mengimplementasikan sistem adalah sebagai berikut:

1. Sistem yang dibuat menerima masukan berupa video lalu lintas yang sudah diekstraksi menjadi kumpulan gambar dalam sebuah *folder*.
2. Sistem hanya mampu melakukan pelacakan terhadap dua objek berupa mobil dan sepeda motor dengan menggunakan tinggi atau lebar objek.
3. Sistem hanya mampu melakukan pelacakan di dalam daerah *tracking*.
4. Sistem akan memberikan informasi jumlah kendaraan yang terdeteksi beserta klasifikasi dan perkiraan kecepatan kendaraan.
5. Sistem akan memberikan konfirmasi berupa notifikasi atau tanda berupa angka saat objek terhitung sebagai respon jumlah objek yang telah terlacak.
6. Pembahasan ditekankan pada bagaimana mengimplementasikan metode *blob analysis* serta mengetahui tingkat kinerja metode tersebut apabila diterapkan untuk *vehicle counting*.

5.3 Implementasi Algoritma

Aplikasi Implementasi *Blob Analysis* untuk *Vehicle Counting* pada Video Lalu Lintas mempunyai berberapa proses utama yang terbagi dalam beberapa fungsi. Program aplikasi mempunyai 2 proses utama yaitu proses deteksi objek dan proses *tracking* dan *count*. Implementasi pembuatan program aplikasi dibuat dengan menggunakan bahasa pemrograman C#.

5.3.1 Proses Deteksi Objek

Proses deteksi objek dilakukan untuk mendeteksi objek tiap *frame*, data uji yang dimasukkan akan diubah menjadi citra *grayscale* dan dilakukan proses *background subtraction*. Setelah mendapatkan hasil *background subtraction*, citra diperbaiki dengan operasi morfologi menggunakan library dan kemudian diberikan *bounding-box* dan *base line*.

5.3.1.1 Proses Pengubahan Warna Citra ke *Grayscale*

Proses pengubahan warna citra ke *grayscale* dimaksudkan untuk mengubah tiga *channel* warna (*red*, *green*, dan *blue*) menjadi satu *channel* warna yakni *gray*. Dengan pengubahan ini maka akan mempermudah proses pengolahan citra.

```
1   private Bitmap im2gray(Bitmap bmp)
2   {
3       int numCol = bmp.Width;
4       int numRow = bmp.Height;
5       Bitmap GRAY = new Bitmap(numCol, numRow);
6
7       for (int i = 0; i < numCol; i++)
8       {
9           for (int j = 0; j < numRow; j++)
10          {
11              Color c = bmp.GetPixel(i, j); //mengambil warna
    piksel
12              int rd = c.R; int gr = c.G; int bl = c.B;
13              double d1 = ((double)rd + (double)gr + (double)bl) /
    3; //mencari nilai mean
14              int c1 = (int)Math.Round(d1); //pembulatan ke atas
                Color c2 = Color.FromArgb(c1, c1, c1);
15              GRAY.SetPixel(i, j, c2); //mengganti nilai color
16  menjadi gray
            }
17      }
18      return GRAY;
19  }
20
```

***Source Code* 5.1** *Listing Code* Proses Pengubahan Warna Citra ke *Grayscale*

Penjelasan *source code* 5.1 adalah:

1. Baris 11 – 12 berfungsi mengambil nilai warna dari bmp
2. Baris 13 – 14 berfungsi mencari nilai *mean* dari ketiga *channel* warna (*red*, *green*, dan *blue*) dan membulatkan nilai keatas
3. Baris 15 – 16 berfungsi mengganti warna ke *gray*

5.3.1.2 Proses *Background Subtraction*

Proses *background subtraction* bertujuan untuk melakukan proses *image subtraction* yakni proses pengurangan nilai piksel suatu citra oleh citra yang lain, dimana dalam hal ini antara citra pada *image sequences* dengan *citra background* sehingga dapat mengindikasikan jika terdapat objek.

```
1   //inisialisai
2   System.Drawing.Image image =
3   (System.Drawing.Image)pictureBox1.Image.Clone();
4   Bitmap img = new Bitmap(image);
5
6   Bitmap citra = new Bitmap(img);
7   Bitmap citraBiner = new Bitmap(img);
8   Bitmap gambar = new Bitmap(img);
9   Bitmap gambar_grayscale = im2gray(gambar);
10  int R;
11
12  //mulai melakukan background subtraction
13  for (int i = 0; i < citra.Width; i++)// x
14  {
15      for (int j = 0; j < citra.Height; j++)// y
```

```
16      {
17          int pixelgambar = gambar_grayscale.GetPixel(i, j).R;
18          int pixelbg = background_grayscale.GetPixel(i, j).R;
19          R = Math.Abs(pixelgambar - pixelbg); //pengurangan
    gambar dengan background
20          Color newcolor = Color.FromArgb(R, R, R);
21          citra.SetPixel(i, j, newcolor); //mengganti nilai gambar
    dengan hasil background subtraction
22
23          //untuk mengeleminasi piksel yg bernilai dibawah
    threshold
24          if (R < T)
25          {
26              citraBiner.SetPixel(i, j, Color.FromArgb(0, 0, 0));
    //jika nilai piksel = 0 maka akan melakukan update background
27              updateBackground(i, j, pixelgambar, pixelbg);
28          }
29          else
30          {
31              citraBiner.SetPixel(i, j, Color.FromArgb(255, 255,
    255));
32          }
33      }
34  }
```

***Source Code* 5.2** *Listing Code* Proses *Background Subtraction*

Penjelasan *source code* 5.2 adalah:

1. Baris 17 – 18 berfungsi mengganti citra dan background menjadi *grayscale*
2. Baris 19 berfungsi untuk pengurangan dengan *background substraction*
3. Baris 20 – 21 berfungsi mengganti nilai piksel menjadi hasi *background subtraction*
4. Baris 24 – 32 berfungsi untuk seleksi nilai selisih, jika bernilai lebih kecil dari *threshold* maka nilai piksel diset menjadi warna hitam (bernilai 0) dan mengalami *update background*. Sebaliknya nilai piksel di set menjadi warna putih (bernilai 255).

5.3.1.3 Proses *Update Background*

Proses *update background* berfungsi agar membentuk *reliable background* sehingga dapat menyesuaikan dan terlihat mirip dengan *background* dari *sequences image*.

```
1   private void updateBackground(int x, int y, int Ci, int Bi)
2   {
3       double alpha = 0.1;
4       int hasil = (int)(((1 - alpha) * Bi) + (alpha * Ci));
5       background_grayscale.SetPixel(x, y, Color.FromArgb(hasil,
    hasil, hasil));
6   }
```

***Source Code* 5.3** *Listing Code* Proses *Update Background*

Penjelasan *source code* 5.3 adalah:

1. Baris 4 berfungsi untuk mengganti nilai *background grayscale* dengan hasil *update background*

5.3.1.4 Proses Menggambar *Bounding-box* dan *Base Line*

Proses menggambar *bounding-box* dan *base line* berfungsi untuk menandai objek yang terdeteksi dengan kotak merah dan menandai daerah *tracking* dengan dua macam garis, yakni garis merah dan garis kuning.

```
1   private void drawInFrame(PictureBox pb, Bitmap source, Bitmap
    target)
2   {
3       //proses gambar menggunakan library
4       BlobCounter blobCounter = new BlobCounter(source);
5       rects = blobCounter.GetObjectsRectangles();
6       //perulangan untuk setiap frame
7       foreach (Rectangle rc in rects)
8       {
9           System.Drawing.Graphics gambar =
    System.Drawing.Graphics.FromImage(target);
10          //warna garis
11          System.Drawing.Pen pen1 = new Pen(Color.Red, 2);
12          System.Drawing.Pen pen2 = new Pen(Color.Yellow, 2);
13          //menggambar kotak
14          gambar.DrawRectangle(pen1, rc.Left, rc.Top, rc.Width,
    rc.Height);
15
16          if ((string)imageType.SelectedItem == "Horizontal")
17          {
18              //untuk gambar horisontal
19              //gambar garis batas luar
20              gambar.DrawLine(pen1, batasAwal, 0, batasAwal,
    source.Width);
21              gambar.DrawLine(pen1, batasAkhir, 0, batasAkhir,
    source.Width);
22              //gambar garis batas dalam *boleh ditampilkan/tidak
23              if (showLine.Checked)
24              {
25                  gambar.DrawLine(pen2, batasTengah1, 0,
    batasTengah1, source.Width);
26                  gambar.DrawLine(pen2, batasTengah2, 0,
    batasTengah2, source.Width);
27              }
28          }
29          else
30          {
31              //untuk gambar vertikal
32              //gambar garis batas luar
33              gambar.DrawLine(pen1, 0, batasAwal, source.Width,
    batasAwal);
34              gambar.DrawLine(pen1, 0, batasAkhir, source.Width,
    batasAkhir);
35              //gambar garis batas dalam *boleh ditampilkan/tidak
36              if (showLine.Checked)
37              {
38                  gambar.DrawLine(pen2, 0, batasTengah1,
    source.Width, batasTengah1);
39                  gambar.DrawLine(pen2, 0, batasTengah2,
    source.Width, batasTengah2);
40              }
41          }
42          pb.Image = target;
43      }
44  }
```

Source Code 5.4 *Listing Code* Menggambar *Bounding-box* dan *Base Line*

Penjelasan *source code* 5.4 adalah:

1. Baris 4 – 5 menjelaskan proses menggunakan *library* AForge.NET
2. Baris 11 – 12 berfungsi untuk membuat garis
3. Baris 14 berfungsi untuk menggambar *bounding-box* dengan informasi dari *library*
4. Baris 16 – 44 menjelaskan jika tipe gambar yang dipilih *horizontal* maka akan menggambar *base line* terluar sesuai dengan tipe gambar *horizontal* jika yang lainnya maka akan menggambar *base line* terluar untuk *vertical*. Dan jika *show line* dipilih maka akan menggambar *base line* di dalam.

5.3.2 Proses *Tracking* dan *Count*

Proses *tracking* dan *count* dilakukan untuk men-*tracking* objek kemudian menghitungnya. Proses ini dibagi menjadi dua berdasarkan tipe gambar, karena tipe gambar berpengaruh saat *tracking*, klasifikasi, dan perhitungan kecepatan. Proses yang terjadi adalah ekstraksi fitur, kemudian menggunakan fitur untuk proses *tracking* dengan membandingan tiap objek yang terdeteksi kemudian menghitung objek jika objek memenuhi kriteria.

5.3.2.1 Proses *Tracking* dan *Count Horizontal*

Proses *tracking* dan *count horizontal* berfungsi untuk mengekstraksi, men-*tracking*, dan menghitung objek dengan tipe gambar horizontal serta menghitung kecepatan dan klasifikasi objek.

```
1   private void trackingCountHorizontal()
2   {
3       //inisialisasi
4       System.Drawing.Image source = (System.Drawing.Image)
    pictureBox4.Image.Clone();
5       Bitmap gSource = new Bitmap(source);
6
7       System.Drawing.Image result =
    (System.Drawing.Image)pictureBox1.Image.Clone();
8       Bitmap gResult = new Bitmap(result);
9
10      int count = 0;
11      string lbl16 = "";
12
13      //proses gambar menggunakan library
14      BlobCounter blobCounter = new BlobCounter(gSource);
15      rects = blobCounter.GetObjectsRectangles();
16
17      //perulangan untuk setiap frame
18      foreach (Rectangle rc in rects)
19      {
20          if (rc.Left > batasAwal && rc.Right < batasAkhir)
    //gambar mulai diproses
21          {
22              count++;
23              lbl16 += "Objek " + count + "\n";
24
25              //ekstraksi fitur
26              double width = rc.Width;
27              double height = rc.Height;
28              double ki = rc.Left;
29              double ka = rc.Right;
```

```
30              double a = rc.Top;
31              double b = rc.Bottom;
32
33              double pCitra = gSource.Width;
34              double lCitra = gSource.Height;
35
36              double perimeter = 2 * width * height;
37              double area = pCitra * lCitra;
38              double roi = width * height;
39
40              double dispersedness = Math.Pow(perimeter, 2) /
    area;
41              double aspectratio = height / width;
42              double arearatio = area / roi;
43
44              //perhitungan centroid
45              double i;
46
47              double jumlahx = 0;
48              double counterx = 0;
49              for (i = ki; i <= ka; i++)
50              {
51                  jumlahx += i;
52                  counterx += 1;
53              }
54
55              double jumlahy = 0;
56              double countery = 0;
57              for (i = a; i <= b; i++)
58              {
59                  jumlahy += i;
60                  countery += 1;
61              }
62
63              double xnol = (jumlahx) * (countery) / ((1 *
    (counterx)) * (countery));
64              double ynol = (jumlahy) * (counterx) / ((1 *
    (countery)) * (counterx));
65
66
67              if (before[0, 5] > 0) //jika ada objek yang
    terdeteksi
68              {
69                  int countbefore = (int)before[0, 5];
70                  double[] rho = new double[countbefore];
71                  double[] dist = new double[countbefore];
72                  int j;
73
74                  //proses membandingkan objek dan mencari nilai
    minimal
75                  for (j = 1; j <= before[0, 5]; j++)
76                  {
77                      rho[j - 1] = Math.Abs(dispersedness -
    before[j, 0]) * Math.Abs(aspectratio - before[j, 1]) *
    Math.Abs(arearatio - before[j, 2]);
78                      lbl16 += "rho" + j.ToString() + " = " +
    rho[j - 1] + "\n";
79                      dist[j - 1] = Math.Sqrt(Math.Pow(xnol -
    before[j, 3], 2) + Math.Pow(ynol - before[j, 4], 2));
80                      lbl16 += "dist" + j.ToString() + " = " +
    dist[j - 1] + "\n";
81                  }
82
83                  double rhoMin = rho.Min();
84                  int id_rho = 0;
```

```
85                  for (j = 0; j < countbefore; j++)
86                  {
87                      if (rho[j] == rhoMin)
88                          id_rho = j;
89                  }
90                  double distMin = dist.Min();
91                  int id_dist = 0;
92                  for (j = 0; j < countbefore; j++)
93                  {
94                      if (dist[j] == distMin)
95                          id_dist = j;
96                  }
97                  lbl16 += "rho min = " + rhoMin + "\n";
98                  lbl16 += "dist min = " + distMin + "\n\n";
99
100                 Boolean hasil = cekObjek(id_rho, id_dist,
    distMin, rhoMin);
101
102                 if (xnol > before[id_dist + 1, 3]) //mengecek
    arah kendaraan (kiri ke kanan)
103                 {
104                     //melakukan proses perhitungan
105                     if (hasil == true && rc.Right >
    batasTengah2)
106                     {
107                         //jika sebelumnya belum ada objek yang
    terhitung
108                         if (counta == 0)
109                         {
110                             //menyimpan fitur2 objek yang
    terdeteksi ke array record
111                             record[counta, 0] = width;
112                             record[counta, 1] = height;
113                             record[counta, 2] = dispersedness;
114                             record[counta, 3] = aspectratio;
115                             record[counta, 4] = arearatio;
116                             record[counta, 5] = xnol;
117                             record[counta, 6] = ynol;
118
119                             //menghitung kecepatan
120                             double t = 0.12; //waktu perpindahan
    per piksel => (1/25)x3
121                             double s = (xnol - before[id_dist +
    1, 3]) * (0.059); //jarak perpindahan per piksel => 0.059 dari
    jarak 19m/320px
122
123                             kecepatan = (s / t) * 3.6; //3.6
    dari  3600/1000 dari m/s menjadi km/jam
124
125                             //klasifikasi
126                             if (height >= tKlasifikasi)
127                             {
128                                 countMobil++;
129                                 klasifikasi = "Car";
130                             }
131                             else
132                             {
133                                 countMotor++;
134                                 klasifikasi = "Motorcycle";
135                             }
136
137                             //menyimpan string ke array dan
    menampilkan ke dataGridView
138                             string[] row = new string[] {
    (counta + 1).ToString(), width.ToString(), height.ToString(),
```

```
        dispersedness.ToString(), aspectratio.ToString(),
        arearatio.ToString(), xnol.ToString(), ynol.ToString(),
        kecepatan.ToString(), string.Format(klasifikasi),
        string.Format("Right") };
139                             dataGridView1.Rows.Add(row);
140
141                             counta++;
142                         }
143                         else
144                         {
145                             //untuk mengetahui nilai dari objek
        yg terakhir dihitung
146                             double distance =
        Math.Sqrt(Math.Pow(xnol - record[counta - 1, 5], 2) +
        Math.Pow(ynol - record[counta - 1, 6], 2));
147                             double fitur =
        Math.Abs(dispersedness - record[counta - 1, 2]) *
        Math.Abs(aspectratio - record[counta - 1, 3]) *
        Math.Abs(arearatio - record[counta - 1, 4]);
148
149                             //mencegah perhitungan kendaraan yg
        sama lebih dari 1x
150                             if (distance > threshold || fitur !=
        rhoMin)
151                             {
152                                 //menyimpan fitur2 objek yang
        terdeteksi ke array record
153                                 record[counta, 0] = width;
154                                 record[counta, 1] = height;
155                                 record[counta, 2] =
        dispersedness;
156                                 record[counta, 3] = aspectratio;
157                                 record[counta, 4] = arearatio;
158                                 record[counta, 5] = xnol;
159                                 record[counta, 6] = ynol;
160
161                                 //menghitung kecepatan
162                                 double t = 0.12; //waktu
        perpindahan per piksel => (1/25)x3
163                                 double s = (xnol -
        before[id_dist + 1, 3]) * (0.059); //jarak perpindahan per
        piksel => 0.059 dari jarak 19m/320px
164
165                                 kecepatan = (s / t) * 3.6; //3.6
        dari  3600/1000 dari m/s menjadi km/jam
166
167                                 //klasifikasi
168                                 if (height >= tKlasifikasi)
169                                 {
170                                     countMobil++;
171                                     klasifikasi = "Car";
172                                 }
173                                 else
174                                 {
175                                     countMotor++;
176                                     klasifikasi = "Motorcycle";
177                                 }
178
179                                 //menyimpan string ke array dan
        menampilkan ke dataGridView
180                                 string[] row = new string[] {
        (counta + 1).ToString(), width.ToString(), height.ToString(),
        dispersedness.ToString(), aspectratio.ToString(),
        arearatio.ToString(), xnol.ToString(), ynol.ToString(),
```

```
        kecepatan.ToString(), string.Format(klasifikasi),
        string.Format("Right") };
181                                 dataGridView1.Rows.Add(row);
182
183                                 counta++;
184                             }
185                         }
186                         //menggambar notifikasi angka
187                         Graphics g =
        Graphics.FromImage(gResult);
188                         String drawString =
        Convert.ToString(counta);
189                         //mengatur font dan besar font
190                         Font drawFont = new Font("Arial Black",
        16);
191                         SolidBrush drawBrush = new
        SolidBrush(Color.Cyan);
192                         PointF drawPoint = new PointF(rc.X,
        rc.Y);
193
194                         //mengaplikasikan string ke layar
195                         g.DrawString(drawString, drawFont,
        drawBrush, drawPoint);
196                     }
197                 }
198                 else if (xnol < before[id_dist + 1, 3])
        //mengecek arah kendaraan (kanan ke kiri)
199                 {
200                     //melakukan proses perhitungan
201                     if (hasil == true && rc.Left < batasTengah1)
202                     {
203                         //jika sebelumnya belum ada objek yang
        terhitung
204                         if (counta == 0)
205                         {
206                             //menyimpan fitur2 objek yang
        terdeteksi ke array record
207                             record[counta, 0] = width;
208                             record[counta, 1] = height;
209                             record[counta, 2] = dispersedness;
210                             record[counta, 3] = aspectratio;
211                             record[counta, 4] = arearatio;
212                             record[counta, 5] = xnol;
213                             record[counta, 6] = ynol;
214
215                             //menghitung kecepatan
216                             double t = 0.12; //waktu perpindahan
        per piksel => (1/25)x3
217                             double s = (before[id_dist + 1, 3] -
        xnol) * (0.059); //jarak perpindahan per piksel => 0.059 dari
        jarak 19m/320px
218
219                             kecepatan = (s / t) * 3.6; //3.6
        dari  3600/1000 dari m/s menjadi km/jam
220
221                             //klasifikasi
222                             if (height >= tKlasifikasi)
223                             {
224                                 countMobil++;
225                                 klasifikasi = "Car";
226                             }
227                             else
228                             {
229                                 countMotor++;
230                                 klasifikasi = "Motorcycle";
```

```
231                             }
232
233                             //menyimpan string ke array dan
    menampilkan ke dataGridView
234                             string[] row = new string[] {
    (counta + 1).ToString(), width.ToString(), height.ToString(),
    dispersedness.ToString(), aspectratio.ToString(),
    arearatio.ToString(), xnol.ToString(), ynol.ToString(),
    kecepatan.ToString(), string.Format(klasifikasi),
    string.Format("Left") };
235                             dataGridView1.Rows.Add(row);
236
237                             counta++;
238                         }
239                         else
240                         {
241                             //untuk mengetahui nilai dari objek
    yg terakhir dihitung
242                             double distance =
    Math.Sqrt(Math.Pow(xnol - record[counta - 1, 5], 2) +
    Math.Pow(ynol - record[counta - 1, 6], 2));
243                             double fitur =
    Math.Abs(dispersedness - record[counta - 1, 2]) *
    Math.Abs(aspectratio - record[counta - 1, 3]) *
    Math.Abs(arearatio - record[counta - 1, 4]);
244
245                             //mencegah perhitungan kendaraan yg
    sama lebih dari 1x
246                             if (distance > threshold || fitur !=
    rhoMin)
247                             {
248                                 //menyimpan fitur2 objek yang
    terdeteksi ke array record
249                                 record[counta, 0] = width;
250                                 record[counta, 1] = height;
251                                 record[counta, 2] =
    dispersedness;
252                                 record[counta, 3] = aspectratio;
253                                 record[counta, 4] = arearatio;
254                                 record[counta, 5] = xnol;
255                                 record[counta, 6] = ynol;
256
257                                 //menghitung kecepatan
258                                 double t = 0.12; //waktu
    perpindahan per piksel => (1/25)x3
259                                 double s = (before[id_dist + 1,
    3] - xnol) * (0.059); //jarak perpindahan per piksel => 0.059
    dari jarak 19m/320px
260
261                                 kecepatan = (s / t) * 3.6; //3.6
    dari  3600/1000 dari m/s menjadi km/jam
262
263                                 //klasifikasi
264                                 if (height >= tKlasifikasi)
265                                 {
266                                     countMobil++;
267                                     klasifikasi = "Car";
268                                 }
269                                 else
270                                 {
271                                     countMotor++;
272                                     klasifikasi = "Motorcycle";
273                                 }
274
```

```
275                                 //menyimpan string ke array dan
    menampilkan ke dataGridView
276                                 string[] row = new string[] {
    (counta + 1).ToString(), width.ToString(), height.ToString(),
    dispersedness.ToString(), aspectratio.ToString(),
    arearatio.ToString(), xnol.ToString(), ynol.ToString(),
    kecepatan.ToString(), string.Format(klasifikasi),
    string.Format("Left") };
277                                 dataGridView1.Rows.Add(row);
278
279                                 counta++;
280                             }
281                         }
282                         //menggambar notifikasi angka
283                         Graphics g =
    Graphics.FromImage(gResult);
284                         String drawString =
    Convert.ToString(counta);
285                         //mengatur font dan besar font
286                         Font drawFont = new Font("Arial Black",
    16);
287                         SolidBrush drawBrush = new
    SolidBrush(Color.Cyan);
288                         PointF drawPoint = new PointF(rc.X,
    rc.Y);
289
290                         // Draw string to screen.
291                         g.DrawString(drawString, drawFont,
    drawBrush, drawPoint);
292                     }
293                 }
294             }
295             //menyimpan nilai objek yg terdeteksi tiap frame ke
    array before
296             before[count, 0] = dispersedness;
297             before[count, 1] = aspectratio;
298             before[count, 2] = arearatio;
299             before[count, 3] = xnol;
300             before[count, 4] = ynol;
301
302             //menampilkan label 16 ke layar
303             label16.Text = lbl16;
304
305             //menghitung total kendaraan
306             total = countMobil + countMotor;
307             //menampilkan ke layar
308             label13.Text = total.ToString();
309             label27.Text = total.ToString();
310             label21.Text = countMobil.ToString();
311             label22.Text = countMotor.ToString();
312         }
313         //menggambar bounding box dan base line pada gambar
    original
314         drawInFrame(pictureBox6, gSource, gResult);
315         drawInFrame(pictureBox8, gSource, gResult);
316     }
317     before[0, 5] = count; //untuk mengetahui sebelumnya ada
    berapa objek
318 }
```

***Source Code* 5.5** *Listing Code Tracking* dan *Count Horizontal*

Penjelasan *source code* 5.5 adalah:

1. Baris 14 – 15 menjelaskan proses menggunakan *library* AForge.NET
2. Baris 20 menyatakan jika objek berada diantara *base line* terluar (batasAwal dan batasAkhir) maka akan memulai memproses objek
3. Baris 26 – 64 menjelaskan tentang ekstraksi fitur objek.
4. Baris 67 menyatakan jika ada objek yang terdeteksi maka akan memulai proses membandingkan antar objek
5. Baris 75 – 96 menjelaskan tentang perhitungan fungsi pencocokan untuk membandingkan fitur dan jarak antar *centroid* objek dan kemudian dicari nilai minimal.
6. Baris 100 menyatakan untuk menjalankan fungsi cek objek
7. Baris 102 berfungsi untuk mengecek arah objek. Jika titik pusat *x* pada *current frame* lebih besar dari pada *frame* sebelumnya, berarti menyatakan bahwa objek bergerak dari kiri ke kanan
8. Baris 105 – 185 berfungsi untuk menangkap nilai objek saat melewati *base line* di dalam (batasTengan2) dan sebelum keluar dari daerah *tracking*. Jika objek dinilai merupakan objek yang sama dengan *frame* sebelumnya, objek dapat dihitung sebagai 1 objek. Serta dihitung perkiraan kecepatan, klasifikasi, dan menampilkan informasi objek pada dataGridView
9. Baris 187 – 195 berfungsi untuk menggambar angka yang merupakan notifikasi jumlah objek saat itu
10. Baris 198 berfungsi untuk mengecek arah objek. Jika titik pusat *x* pada *current frame* lebih kecil dari pada *frame* sebelumnya, berarti menyatakan bahwa objek bergerak dari kanan ke kiri
11. Baris 201 – 281 berfungsi untuk menangkap nilai objek saat melewati *base line* di dalam (batasTengah1) dan sebelum keluar dari daerah *tracking*. Jika objek dinilai merupakan objek yang sama dengan *frame* sebelumnya, objek dapat dihitung sebagai 1 objek. Serta dihitung perkiraan kecepatan, klasifikasi, dan menampilkan informasi objek pada dataGridView
12. Baris 283 – 291 berfungsi untuk menggambar angka yang merupakan notifikasi jumlah objek saat itu
13. Baris 296 – 300 berfungsi untuk menyimpan nilai objek yang terdeteksi tiap *frame* ke array *count*
14. Baris 303 – 311 menampilkan hasil ke layar
15. Baris 314 – 315 menampilkan *bounding-box* dan *base line* diatas gambar original
16. Baris 317 berfungsi untuk mengetahui jumlah objek pada *frame* sebelumnya

5.3.2.2 Proses *Tracking* dan *Count Vertical*

Proses *tracking* dan *count horizontal* berfungsi untuk mengekstraksi, men-*tracking*, dan menghitung objek dengan tipe gambar horizontal serta menghitung kecepatan dan klasifikasi objek.

```
1   private void trackingCountVertical()
2   {
3       //inisialisasi
```

```
4       System.Drawing.Image source =
    (System.Drawing.Image)pictureBox4.Image.Clone();
5       Bitmap gSource = new Bitmap(source);
6
7       System.Drawing.Image result =
    (System.Drawing.Image)pictureBox1.Image.Clone();
8       Bitmap gResult = new Bitmap(result);
9
10      int count = 0;
11      string lbl16 = "";
12
13      //proses gambar menggunakan library
14      BlobCounter blobCounter = new BlobCounter(gSource);
15      rects = blobCounter.GetObjectsRectangles();
16
17      //perulangan untuk setiap frame
18      foreach (Rectangle rc in rects)
19      {
20          if (rc.Top > batasAwal && rc.Bottom < batasAkhir)
    //gambar mulai diproses
21          {
22              count++;
23              lbl16 += "Objek " + count + "\n";
24
25              //ekstraksi fitur
26              double width = rc.Width;
27              double height = rc.Height;
28              double ki = rc.Left;
29              double ka = rc.Right;
30              double a = rc.Top;
31              double b = rc.Bottom;
32
33              double pCitra = gSource.Width;
34              double lCitra = gSource.Height;
35
36              double perimeter = 2 * width * height;
37              double area = pCitra * lCitra;
38              double roi = width * height;
39
40              double dispersedness = Math.Pow(perimeter, 2) /
    area;
41              double aspectratio = height / width;
42              double areararatio = area / roi;
43
44              //perhitungan centroid
45              double i;
46
47              double jumlahx = 0;
48              double counterx = 0;
49              for (i = ki; i <= ka; i++)
50              {
51                  jumlahx += i;
52                  counterx += 1;
53              }
54
55              double jumlahy = 0;
56              double countery = 0;
57              for (i = a; i <= b; i++)
58              {
59                  jumlahy += i;
60                  countery += 1;
61              }
62
63              double xnol = (jumlahx) * (countery) / ((1 *
    (counterx)) * (countery));
```

```
64              double ynol = (jumlahy) * (counterx) / ((1 *
    (countery)) * (counterx));
65
66
67              if (before[0, 5] > 0) //jika ada objek yang
    terdeteksi
68              {
69                  int countbefore = (int)before[0, 5];
70                  double[] rho = new double[countbefore];
71                  double[] dist = new double[countbefore];
72                  int j;
73
74                  //proses membandingkan dan mencari nilai minimal
75                  for (j = 1; j <= before[0, 5]; j++)
76                  {
77                      rho[j - 1] = Math.Abs(dispersedness -
    before[j, 0]) * Math.Abs(aspectratio - before[j, 1]) *
    Math.Abs(arearatio - before[j, 2]);
78                      lbl16 += "rho" + j.ToString() + " = " +
    rho[j - 1] + "\n";
79                      dist[j - 1] = Math.Sqrt(Math.Pow(xnol -
    before[j, 3], 2) + Math.Pow(ynol - before[j, 4], 2));
80                      lbl16 += "dist" + j.ToString() + " = " +
    dist[j - 1] + "\n";
81                  }
82
83                  double rhoMin = rho.Min();
84                  int id_rho = 0;
85                  for (j = 0; j < countbefore; j++)
86                  {
87                      if (rho[j] == rhoMin)
88                          id_rho = j;
89                  }
90                  double distMin = dist.Min();
91                  int id_dist = 0;
92                  for (j = 0; j < countbefore; j++)
93                  {
94                      if (dist[j] == distMin)
95                          id_dist = j;
96                  }
97                  lbl16 += "rho min = " + rhoMin + "\n";
98                  lbl16 += "dist min = " + distMin + "\n\n";
99
100                 Boolean hasil = cekObjek(id_rho, id_dist,
    distMin, rhoMin);
101
102                 if (ynol > before[id_dist + 1, 4]) //mengecek
    arah kendaraan (atas ke bawah)
103                 {
104                     //melakukan proses perhitungan
105                     if (hasil == true && rc.Bottom >
    batasTengah2)
106                     {
107                         //jika sebelumnya belum ada objek yang
    terhitung
108                         if (counta == 0)
109                         {
110                             //menyimpan fitur2 objek yang
    terdeteksi ke array record
111                             record[counta, 0] = width;
112                             record[counta, 1] = height;
113                             record[counta, 2] = dispersedness;
114                             record[counta, 3] = aspectratio;
115                             record[counta, 4] = arearatio;
116                             record[counta, 5] = xnol;
```

```
117                             record[counta, 6] = ynol;
118
119                             //menghitung kecepatan
120                             double t = 0.12; //waktu perpindahan
    per piksel => (1/25)x3
121                             double s = (ynol - before[id_dist +
    1, 4]) * (0.059); //jarak perpindahan per piksel => 0.059 dari
    jarak 19m/320px
122
123                             kecepatan = (s / t) * 3.6; //3.6
    dari  3600/1000 dari m/s menjadi km/jam
124
125                             //klasifikasi
126                             if (width >= tKlasifikasi)
127                             {
128                                 countMobil++;
129                                 klasifikasi = "Car";
130                             }
131                             else
132                             {
133                                 countMotor++;
134                                 klasifikasi = "Motorcycle";
135                             }
136
137                             //menyimpan string ke array dan
    menampilkan ke dataGridView
138                             string[] row = new string[] {
    (counta + 1).ToString(), width.ToString(), height.ToString(),
    dispersedness.ToString(), aspectratio.ToString(),
    arearatio.ToString(), xnol.ToString(), ynol.ToString(),
    kecepatan.ToString(), string.Format(klasifikasi),
    string.Format("Bottom") };
139                             dataGridView1.Rows.Add(row);
140
141                             counta++;
142                         }
143                         else
144                         {
145                             //untuk mengetahui nilai dari objek
    yg terakhir dihitung
146                             double distance =
    Math.Sqrt(Math.Pow(xnol - record[counta - 1, 5], 2) +
    Math.Pow(ynol - record[counta - 1, 6], 2));
147                             double fitur =
    Math.Abs(dispersedness - record[counta - 1, 2]) *
    Math.Abs(aspectratio - record[counta - 1, 3]) *
    Math.Abs(arearatio - record[counta - 1, 4]);
148
149                             //mencegah perhitungan kendaraan yg
    sama lebih dari 1x
150                             if (distance > threshold || fitur !=
    rhoMin)
151                             {
152                                 //menyimpan fitur2 objek yang
    terdeteksi ke array record
153                                 record[counta, 0] = width;
154                                 record[counta, 1] = height;
155                                 record[counta, 2] =
    dispersedness;
156                                 record[counta, 3] = aspectratio;
157                                 record[counta, 4] = arearatio;
158                                 record[counta, 5] = xnol;
159                                 record[counta, 6] = ynol;
160
161                                 //menghitung kecepatan
```

```
162                                 double t = 0.12; //waktu
    perpindahan per piksel => (1/25)x3
163                                 double s = (ynol -
    before[id_dist + 1, 4]) * (0.059); //jarak perpindahan per
    piksel => 0.059 dari jarak 19m/320px
164
165                                 kecepatan = (s / t) * 3.6; //3.6
    dari  3600/1000 dari m/s menjadi km/jam
166
167                                 //klasifikasi
168                                 if (width >= tKlasifikasi)
169                                 {
170                                     countMobil++;
171                                     klasifikasi = "Car";
172                                 }
173                                 else
174                                 {
175                                     countMotor++;
176                                     klasifikasi = "Motorcycle";
177                                 }
178
179                                 //menyimpan string ke array dan
    menampilkan ke dataGridView
180                                 string[] row = new string[] {
    (counta + 1).ToString(), width.ToString(), height.ToString(),
    dispersedness.ToString(), aspectratio.ToString(),
    arearatio.ToString(), xnol.ToString(), ynol.ToString(),
    kecepatan.ToString(), string.Format(klasifikasi),
    string.Format("Bottom") };
181                                 dataGridView1.Rows.Add(row);
182
183                                 counta++;
184                             }
185                         }
186                         //menggambar notifikasi angka
187                         Graphics g =
    Graphics.FromImage(gResult);
188                         String drawString =
    Convert.ToString(counta);
189                         //mengatur font dan besar font
190                         Font drawFont = new Font("Arial Black",
    16);
191                         SolidBrush drawBrush = new
    SolidBrush(Color.Cyan);
192                         PointF drawPoint = new PointF(rc.X,
    rc.Y);
193
194                         //mengaplikasikan string ke layar
195                         g.DrawString(drawString, drawFont,
    drawBrush, drawPoint);
196                     }
197                 }
198                 else if (ynol < before[id_dist + 1, 4])
    //mengecek arah kendaraan (bawah ke atas)
199                 {
200                     //melakukan proses perhitungan
201                     if (hasil == true && rc.Top < batasTengah1)
202                     {
203                         //jika sebelumnya belum ada objek yang
    terhitung
204                         if (counta == 0)
205                         {
206                             //menyimpan fitur2 objek yang
    terdeteksi ke array record
207                             record[counta, 0] = width;
```

```
208                             record[counta, 1] = height;
209                             record[counta, 2] = dispersedness;
210                             record[counta, 3] = aspectratio;
211                             record[counta, 4] = arearatio;
212                             record[counta, 5] = xnol;
213                             record[counta, 6] = ynol;
214
215                             //menghitung kecepatan
216                             double t = 0.12; //waktu perpindahan
    per piksel => (1/25)x3
217                             double s = (before[id_dist + 1, 4] -
    ynol) * (0.075); //jarak perpindahan per piksel => 0.059 dari
    jarak 24m/320px
218
219                             kecepatan = (s / t) * 3.6; //3.6
    dari  3600/1000 dari m/s menjadi km/jam
220
221                             //klasifikasi
222                             if (width >= tKlasifikasi)
223                             {
224                                 countMobil++;
225                                 klasifikasi = "Car";
226                             }
227                             else
228                             {
229                                 countMotor++;
230                                 klasifikasi = "Motorcycle";
231                             }
232
233                             //menyimpan string ke array dan
    menampilkan ke dataGridView
234                             string[] row = new string[] {
    (counta + 1).ToString(), width.ToString(), height.ToString(),
    dispersedness.ToString(), aspectratio.ToString(),
    arearatio.ToString(), xnol.ToString(), ynol.ToString(),
    kecepatan.ToString(), string.Format(klasifikasi),
    string.Format("Top") };
235                             dataGridView1.Rows.Add(row);
236
237                             counta++;
238                         }
239                         else
240                         {
241                             //untuk mengetahui nilai dari objek
    yg terakhir dihitung
242                             double distance =
    Math.Sqrt(Math.Pow(xnol - record[counta - 1, 5], 2) +
    Math.Pow(ynol - record[counta - 1, 6], 2));
243                             double fitur =
    Math.Abs(dispersedness - record[counta - 1, 2]) *
    Math.Abs(aspectratio - record[counta - 1, 3]) *
    Math.Abs(arearatio - record[counta - 1, 4]);
244
245                             //mencegah perhitungan kendaraan yg
    sama lebih dari 1x
246                             if (distance > threshold || fitur !=
    rhoMin)
247                             {
248                                 //menyimpan fitur2 objek yang
    terdeteksi ke array record
249                                 record[counta, 0] = width;
250                                 record[counta, 1] = height;
251                                 record[counta, 2] =
    dispersedness;
252                                 record[counta, 3] = aspectratio;
```

```
                                record[counta, 4] = arearatio;
                                record[counta, 5] = xnol;
                                record[counta, 6] = ynol;

                                //menghitung kecepatan
                                double t = 0.12; //waktu perpindahan per piksel => (1/25)x3
                                double s = (before[id_dist + 1, 4] - ynol) * (0.059); //jarak perpindahan per piksel => 0.059 dari jarak 19m/320px

                                kecepatan = (s / t) * 3.6; //3.6 dari  3600/1000 dari m/s menjadi km/jam

                                //klasifikasi
                                if (width >= tKlasifikasi)
                                {
                                    countMobil++;
                                    klasifikasi = "Car";
                                }
                                else
                                {
                                    countMotor++;
                                    klasifikasi = "Motorcycle";
                                }

                                //menyimpan string ke array dan menampilkan ke dataGridView
                                string[] row = new string[] { (counta + 1).ToString(), width.ToString(), height.ToString(), dispersedness.ToString(), aspectratio.ToString(), arearatio.ToString(), xnol.ToString(), ynol.ToString(), kecepatan.ToString(), string.Format(klasifikasi), string.Format("Top") };
                                dataGridView1.Rows.Add(row);

                                counta++;
                            }
                        }
                        //menggambar notifikasi angka
                        Graphics g = Graphics.FromImage(gResult);
                        String drawString = Convert.ToString(counta);
                        //mengatur font dan besar font
                        Font drawFont = new Font("Arial Black", 16);
                        SolidBrush drawBrush = new SolidBrush(Color.Cyan);
                        PointF drawPoint = new PointF(rc.X, rc.Y);

                        //mengaplikasikan string ke layar
                        g.DrawString(drawString, drawFont, drawBrush, drawPoint);
                    }
                }
            }
            //menyimpan nilai objek yg terdeteksi tiap frame ke array before
            before[count, 0] = dispersedness;
            before[count, 1] = aspectratio;
            before[count, 2] = arearatio;
            before[count, 3] = xnol;
            before[count, 4] = ynol;
```

```
301
302             //menampilkan label 16 ke layar
303             label16.Text = lbl16;
304
305             //menghitung total kendaraan
306             total = countMobil + countMotor;
307             //menampilkan ke layar
308             label13.Text = total.ToString();
309             label27.Text = total.ToString();
310             label21.Text = countMobil.ToString();
311             label22.Text = countMotor.ToString();
312         }
313         //menggambar bounding box dan base line pada gambar
    original
314         drawInFrame(pictureBox6, gSource, gResult);
315         drawInFrame(pictureBox8, gSource, gResult);
316     }
317     before[0, 5] = count; //untuk mengetahui sebelumnya ada
    berapa mobil
318 }
```

***Source Code* 5.6** *Listing Code Tracking* dan *Count Vertical*

Penjelasan *source code* 5.6 adalah:

1. Baris 14 – 15 menjelaskan proses menggunakan *library* AForge.NET
2. Baris 20 menyatakan jika objek berada diantara *base line* terluar (batasAwal dan batasAkhir) maka akan memulai memproses objek
3. Baris 26 – 64 menjelaskan tentang ekstraksi fitur objek.
4. Baris 67 menyatakan jika ada objek yang terdeteksi maka akan memulai proses membandingkan antar objek
5. Baris 75 – 96 menjelaskan tentang perhitungan fungsi pencocokan untuk membandingkan fitur dan jarak antar *centroid* objek dan kemudian dicari nilai minimal.
6. Baris 100 menyatakan untuk menjalankan fungsi cek objek
7. Baris 102 berfungsi untuk mengecek arah objek. Jika titik pusat *y* pada *current frame* lebih besar dari pada *frame* sebelumnya, berarti menyatakan bahwa objek bergerak dari atas ke bawah
8. Baris 105 – 185 berfungsi untuk menangkap nilai objek saat melewati *base line* di dalam (batasTengah2) dan sebelum keluar dari daerah *tracking*. Jika objek dinilai merupakan objek yang sama dengan *frame* sebelumnya, objek dapat dihitung sebagai 1 objek. Serta dihitung perkiraan kecepatan, klasifikasi, dan menampilkan informasi objek pada dataGridView
9. Baris 187 – 195 berfungsi untuk menggambar angka yang merupakan notifikasi jumlah objek saat itu
10. Baris 198 berfungsi untuk mengecek arah objek. Jika titik pusat *y* pada *current frame* lebih kecil dari pada *frame* sebelumnya, berarti menyatakan bahwa objek bergerak dari bawah ke atas
11. Baris 201 – 281 berfungsi untuk menangkap nilai objek saat melewati *base line* di dalam (batasTengah1) dan sebelum keluar dari daerah *tracking*. Jika objek dinilai merupakan objek yang sama dengan *frame* sebelumnya, objek dapat dihitung sebagai 1 objek. Serta dihitung perkiraan kecepatan, klasifikasi, dan menampilkan informasi objek pada dataGridView

12. Baris 283 – 291 berfungsi untuk menggambar angka yang merupakan notifikasi jumlah objek saat itu
13. Baris 296 – 300 berfungsi untuk menyimpan nilai objek yang terdeteksi tiap *frame* ke array *count*
14. Baris 303 – 311 menampilkan hasil ke layar
15. Baris 314 – 315 menampilkan *bounding-box* dan *base line* diatas gambar original
16. Baris 317 berfungsi untuk mengetahui jumlah objek pada *frame* sebelumnya

5.3.2.3 Proses Cek Objek

Proses cek objek berfungsi untuk menentukan objek yang terdeteksi merupakan objek yang sama atau berbeda dengan *frame* sebelumnya.

```
1   private bool cekObjek(int id_rho, int id_dist, double distMin,
    double rhoMin)
2   {
3       Boolean hasil = false;
4       if (id_rho == id_dist)
5       {
6           if (distMin < threshold)
7           {
8               hasil = true;
9           }
10      }
11      else
12      {
13          if (distMin < threshold)
14          {
15              hasil = true;
16          }
17      }
18     return hasil;
19  }
```

Source Code 5.7 *Listing Code* Cek Objek

Penjelasan *source code* 5.7 adalah:

1. Baris 3 menyatakan *default* hasil adalah *false*
2. Baris 4 – 10 menyatakan jika nilai minimal dari *rho* dan *dist* berada dalam satu objek yang sama maka mengecek apakah nilai *dist* minimal lebih kecil dari pada *threshold*. Jika lebih kecil dari *threshold* maka hasilnya adalah *true*
3. Baris 11 – 17 menyatakan jika nilai minimal dari *rho* dan *dist* tidak berada dalam satu objek yang sama maka mengecek apakah nilai *dist* minimal lebih kecil dari *threshold* atau tidak. Jika lebih kecil dari *threshold* maka hasilnya adalah *true*

5.4 Implementasi *User Interface* Aplikasi

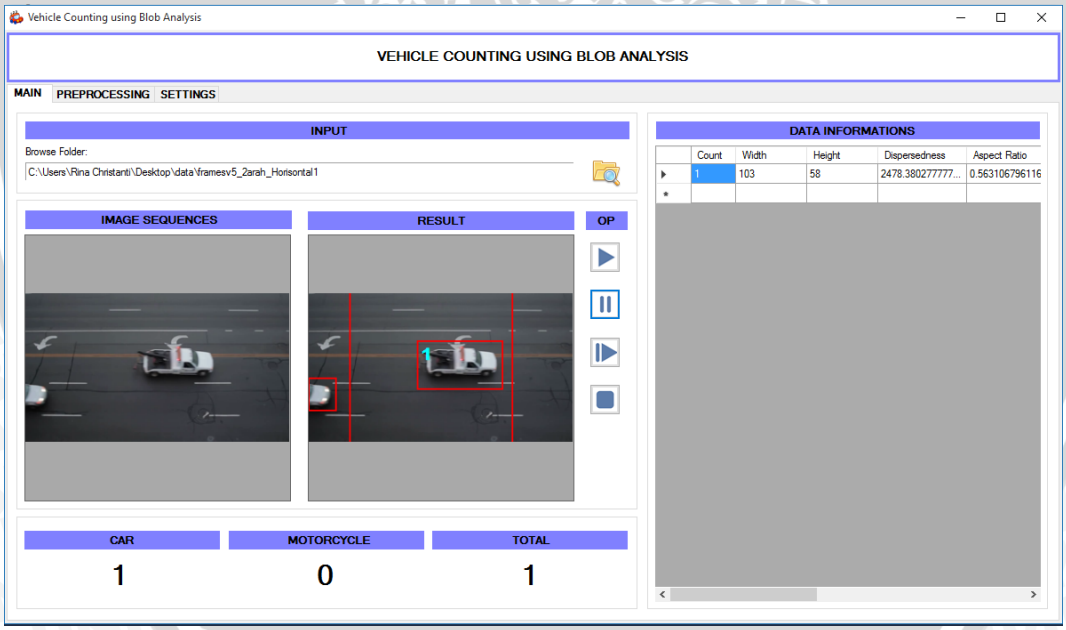
Interface Aplikasi *vehicle counting* pada video lalu lintas digunakan pengguna untuk berinteraksi dengan sistem perangkat lunak. *Interface* perangkat lunak ini dibagi menjadi tiga yaitu *interface* halaman *main*, *interface* halaman *preprocessing*, dan *interface* halaman *settings*.

5.4.1 Halaman *Main*

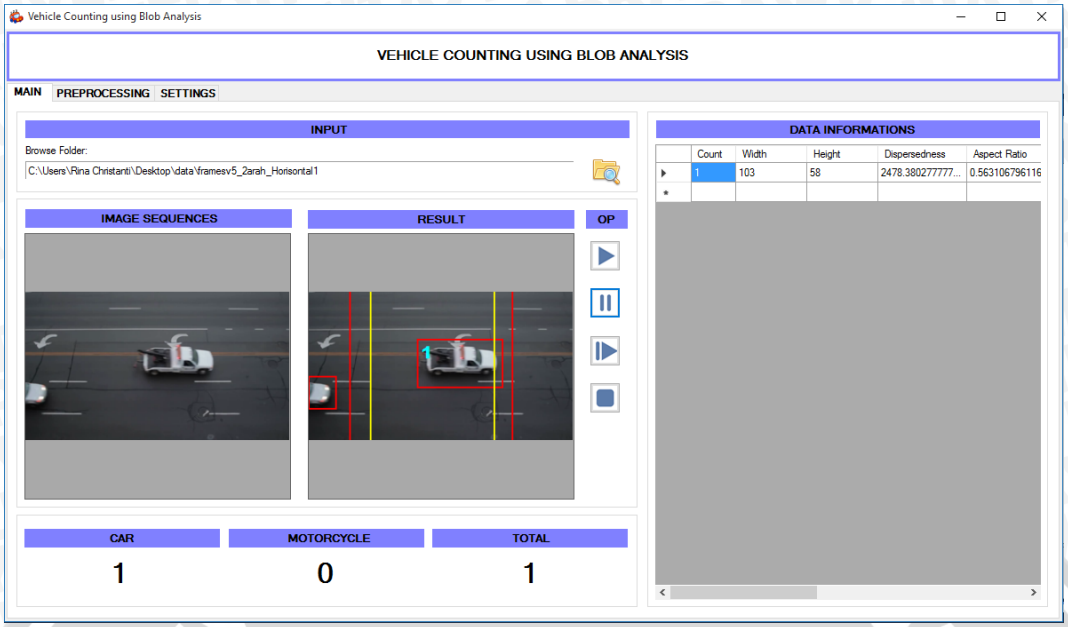
Halaman *main* berfungsi untuk memasukkan data uji yang akan dilacak dan dimana hasil *vehicle counting* akan ditampilkan. Gambar 5.2 dan 5.3 menunjukkan implementasi tampilan *interface* dari halaman *main* yang mengacu pada perancangan *interface* halaman *main*. Perbedaan pada Gambar 5.2 dan 5.3 adalah pada bagian *Result*, Gambar 5.2 tidak menampilkan *base line* di dalam dan Gambar 5.3 menampilkan *base line* di dalam (garis warna kuning) yang menyatakan batas dimana objek mulai dihitung. Sedangkan b*ase line* terluar (garis warna merah) menyatakan daerah *tracking* dimana objek tiap *frame* mulai diproses dari ekstraksi objek hingga perhitungan gambar. Jika objek melewati daerah *tracking*, maka objek diberhentikan prosesnya.

Adapun penjelasan Gambar 5.2 adalah sebagai berikut:

1. *Input* menampilkan alamat data uji setelah data uji diinputkan dengan OpenFolderDialog
2. *Image sequences* menampilkan urutan gambar original
3. *Car*, *motorcycle*, dan *total* menampilkan hasil perhitungan dari sistem berdasarkan klasifikasi kendaraan yakni mobil dan sepeda motor beserta total hasil keseluruhan kendaraan yang berhasil terdeteksi dan terhitung
4. Data *informations* menampilkan informasi dari objek yang sudah dihitung



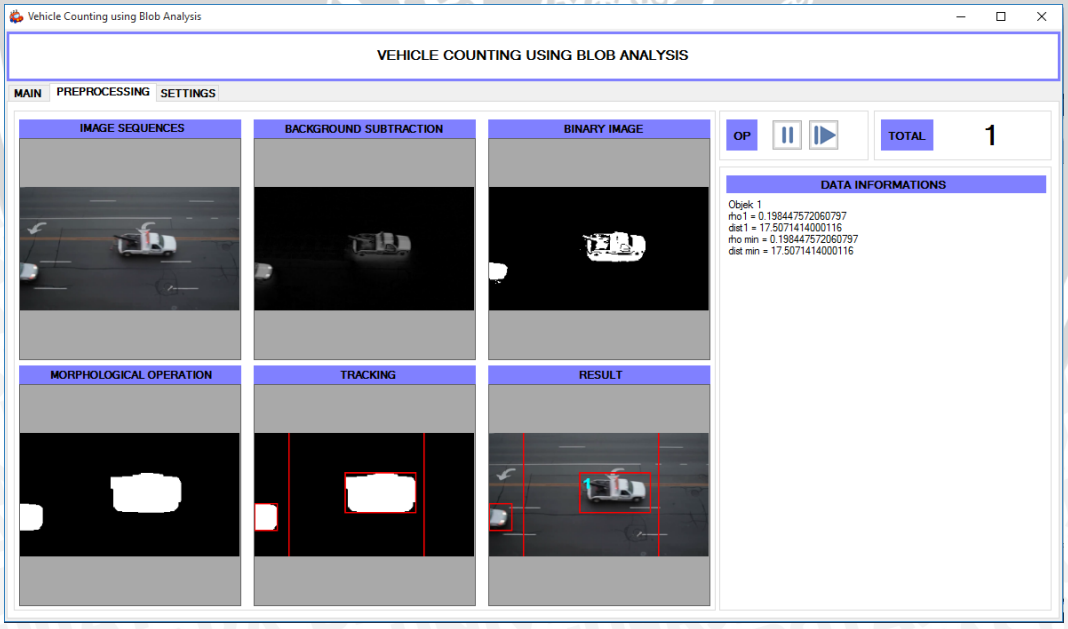
Gambar 5.2 Implementasi Halaman *Main* tanpa *Base Line* dalam



Gambar 5.3 Implementasi Halaman *Main* dengan *Base Line* dalam

5.4.2 Halaman *Preprocessing*

Halaman *preprocessing* berfungsi untuk menampilkan proses yang terjadi mulai dari *image sequences* original sampai mendapatkan hasil *vehicle counting*. Gambar 5.4 menunjukkan implementasi tampilan *interface* dari halaman *preprocessing* yang mengacu pada perancangan *interface* halaman *preprocessing*.



Gambar 5.4 Implementasi Halaman *Preprocessing*

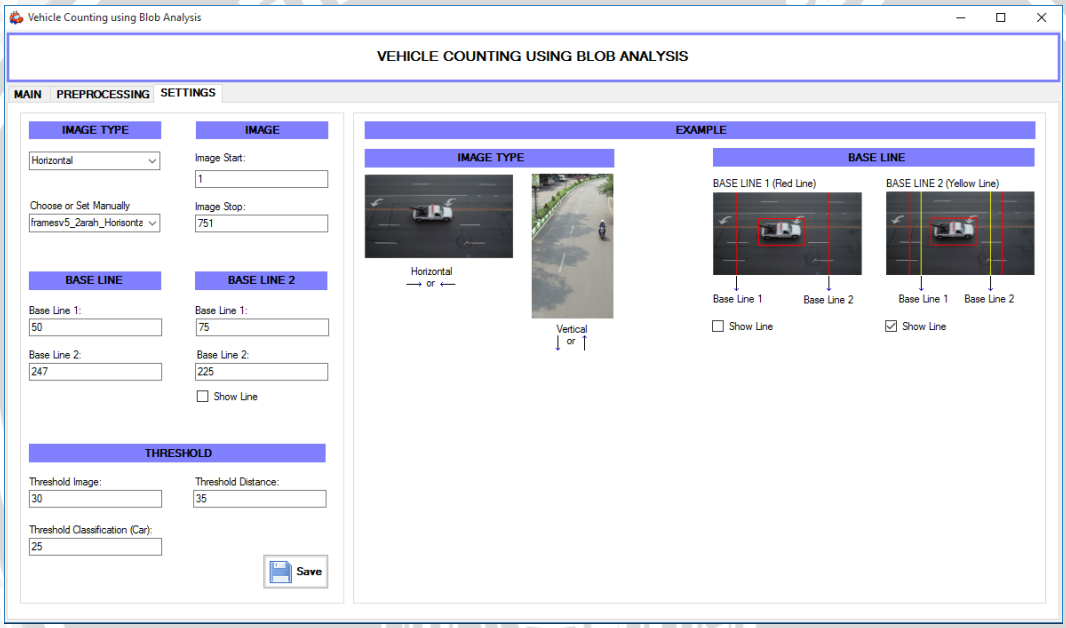
Adapun penjelasan Gambar 5.4 adalah sebagai berikut:

1. *Image sequences* menampilkan urutan gambar original
2. *Background subtraction* menampilkan hasil dari *background subtraction*
3. *Binary image* menampilkan hasil seleksi piksel dengan *threshold*

4. Morphological operation menampilkan hasil operasi morfologi yakni *opening* dengan menggunakan *library*
5. *Tracking* menampilkan hasil operasi morfologi dengan *bounding-box* dan *base line*
6. *Result* menampilkan gambar original dengan *bounding-box* dan *base line* serta hasil perhitungan objek dengan notifikasi angka
7. Total menampilkan jumlah kendaraan yang berhasil terdeteksi
8. Data *informations* untuk menampilkan proses *tracking*

5.4.3 Halaman *Settings*

Halaman *settings* berfungsi untuk memberikan masukan berupa pengaturan yang dibutuhkan sistem. Gambar 5.5 menunjukkan implementasi tampilan *interface* dari halaman *settings* yang mengacu pada perancangan *interface* halaman *settings*.



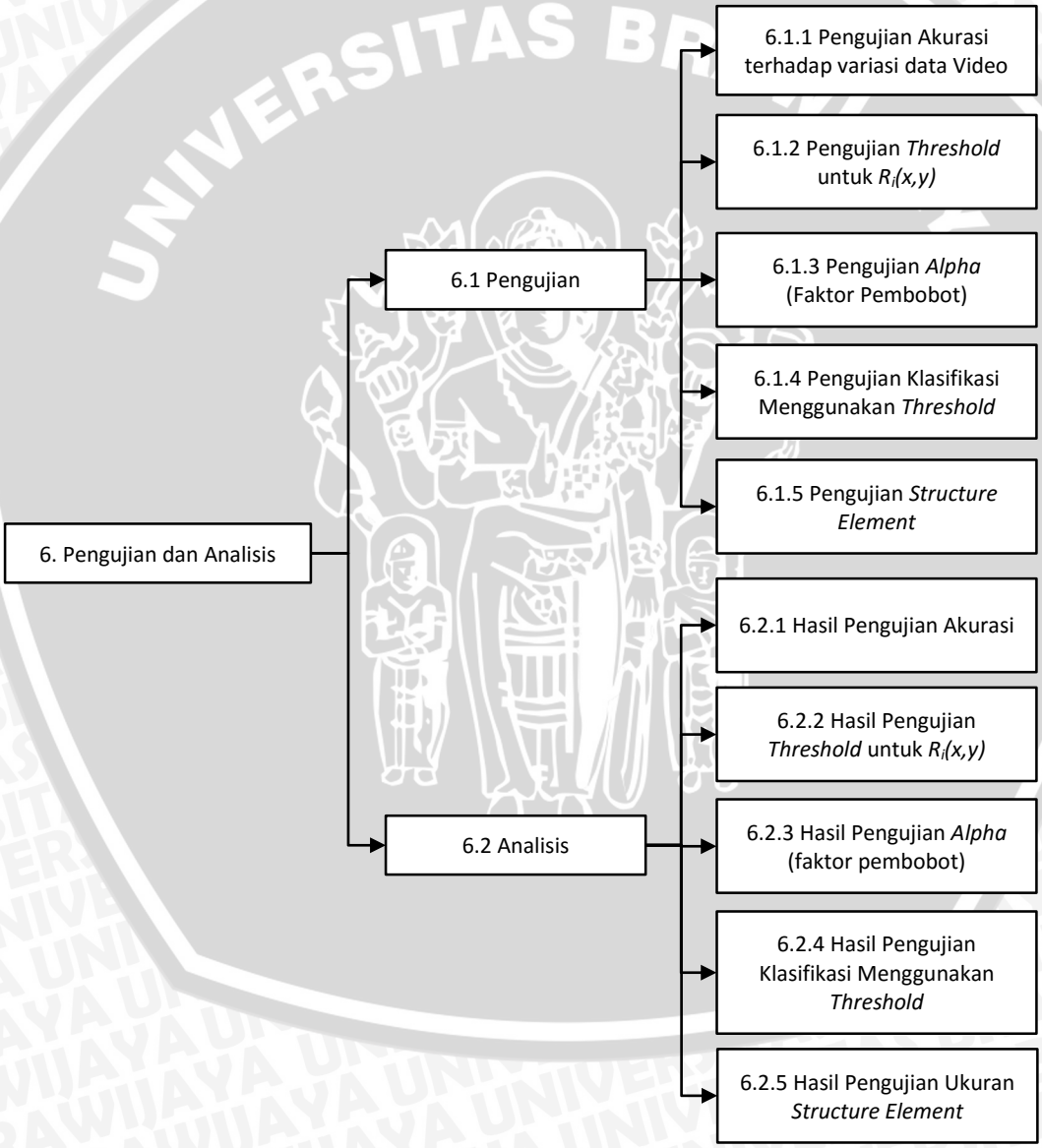
Gambar 5.5 Implementasi Halaman *Settings*

Adapun penjelasan Gambar 5.5 adalah sebagai berikut:

1. *Image type* untuk memilih tipe gambar yang akan diuji serta disediakan pilihan pengaturan yang dapat digunakan
2. *Image* untuk menentukan awal mulai dan berhentinya gambar diproses
3. *Base line* untuk menentukan dimana daerah *tracking* dan *base line* 2 adalah *base line* di dalam daerah *tracking*
4. *Threshold* untuk menentukan *threshold* yang akan digunakan. Ada tiga *threshold*, yaitu *threshold image* menentukan seleksi *image* untuk gambar biner, *threshold distance* untuk menentukan perhitungan objek, dan *threshold classification* untuk menentukan nilai mobil karena hanya membedakan dua jenis kendaraan. Jadi jika nilai diatas nilai *threshold classification* maka akan masuk kelas mobil dan dibawah *threshold classification* adalah sepeda motor.

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini membahas mengenai tahapan pengujian dan analisis sistem *vehicle counting* dengan menggunakan *blob analysis* pada video lalu lintas. Proses pengujian dilakukan dengan cara pengujian akurasi yang dihasilkan oleh sistem dengan beberapa kombinasi nilai konstanta tertentu. Pengujian akurasi dilakukan dengan cara menghitung akurasi pada setiap jumlah objek yang berhasil terdeteksi pada setiap video. Pencocokan akurasi dilakukan dengan membandingkan objek yang dideteksi oleh sistem dengan jumlah objek yang dideteksi berdasarkan pengamatan manusia. Gambar 6.1 menunjukkan ilustrasi dari pengujian aplikasi ini.

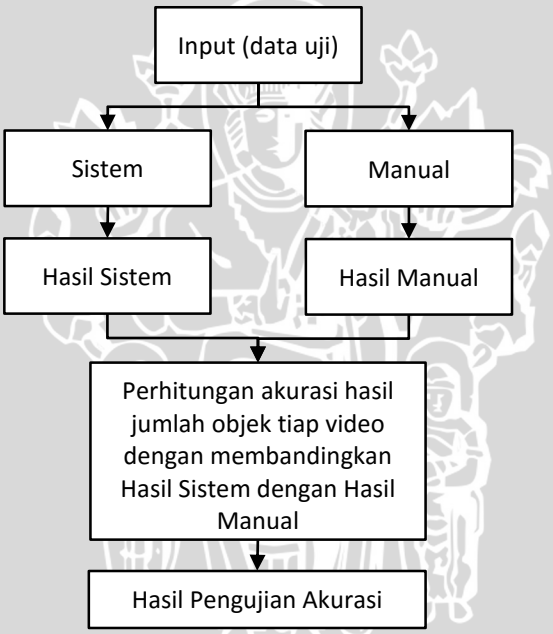


Gambar 6.1 Pohon Pengujian dan Analisis

6.1 Pengujian

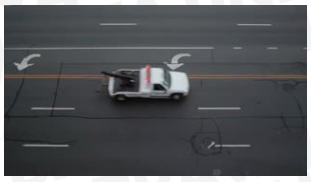
6.1.1 Pengujian Akurasi terhadap Variasi Data Video Lalu Lintas

Pengujian akurasi digunakan untuk mengetahui performa dari sistem *vehicle counting* menggunakan *blob analysis* dalam menghitung objek yang terdapat pada video lalu lintas. Pengujian ini dilakukan dengan cara membandingkan antara jumlah hasil *counting* yang dilakukan oleh sistem dengan jumlah hasil *counting* yang dilakukan mata manusia. Selain itu, perhitungan akurasi tersebut dilakukan dengan menggunakan nilai – nilai parameter terbaik pada setiap data uji seperti nilai *threshold* untuk $R_i(x,y)$, *threshold* untuk jarak antar *centroid*, *threshold* untuk klasifikasi, dan *alpha*. Sehingga akan didapatkan nilai akurasi tertinggi dari sistem. Data video yang digunakan sebanyak 4 video dengan *frame rate* 25 fps dan waktu 30 detik yang sudah diekstraksi menjadi kumpulan gambar menjadi 10 data uji serta resolusi sebesar 320x180 px dan 180x320 px. Diagram alir pengujian akurasi dapat dilihat pada Gambar 6.2 sedangkan data uji yang digunakan dapat dilihat pada Tabel 6.3.

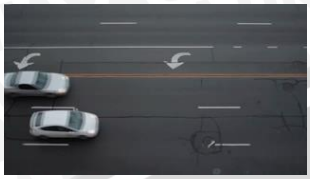
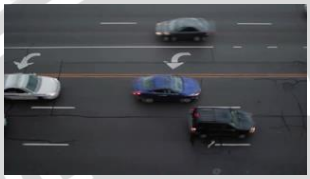
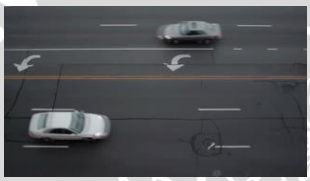
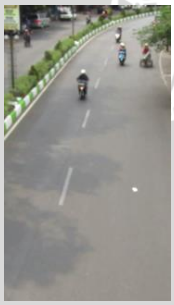


Gambar 6.2 Diagram Alir Proses Pengujian Akurasi

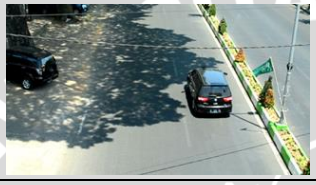
Tabel 6.1 Data Uji

Video	Data Uji (Contoh 1 *Frame*)		Keterangan	Hasil Aktual	
				Mobil	Sepeda Motor
1	1		Tipe data *horizontal*, 1 arah, kepadatan 26 kendaraan/30 detik	26	0

Tabel 6.1 Data Uji (lanjutan)

Video	Data Uji (Contoh 1 *Frame*)		Keterangan	Hasil Aktual	
				Mobil	Sepeda Motor
	2		Tipe data *horizontal*, 1 arah, kepadatan 20 kendaraan/30 detik	20	0
	3		Tipe data *horizontal*, 2 arah, kepadatan 38 kendaraan/30 detik	38	0
	4		Tipe data *horizontal*, 2 arah, kepadatan 28 kendaraan/30 detik	28	0
2	5		Tipe data *vertical*, 1 arah, kepadatan 18 kendaraan/30 detik	4	14
	6		Tipe data *vertical*, 1 arah, kepadatan 25 kendaraan/30 detik	6	19
3	7		Tipe data *vertical*, 1 arah, kepadatan 39 kendaraan/30 detik	7	32

Tabel 6.1 Data Uji (lanjutan)

Video	Data Uji (Contoh 1 *Frame*)		Keterangan	Hasil Aktual	
				Mobil	Sepeda Motor
	8		Tipe data *vertical*, 1 arah, kepadatan 25 kendaraan/30 detik	4	21
4	9		Tipe data *vertical*, 1 arah, kepadatan 34 kendaraan/30 detik	13	21
	10		Tipe data *vertical*, 1 arah, kepadatan 25 kendaraan/30 detik	5	20

Tabel 6.2 Nilai Kombinasi Parameter setiap Data Uji

Parameter / Data Uji ke-	*Threshold*			*Alpha*	*Structure Element*	
	$R_i(x,y)$	Jarak antar *Centroid*	Klasifikasi		Erosi	Dilasi
1	30	35	25	0.1	3x3	17x17
2	30	35	25	0.1	3x3	17x17
3	30	35	25	0.1	3x3	17x17
4	30	35	25	0.1	3x3	17x17
5	40	35	50	0.1	3x3	17x17
6	40	35	50	0.1	3x3	17x17
7	40	35	55	0.1	3x3	17x17
8	40	35	55	0.1	3x3	17x17
9	45	35	50	0.1	3x3	17x17
10	40	35	50	0.1	3x3	17x17

Tabel 6.2 menunjukkan nilai – nilai parameter yang digunakan pada setiap data uji. Sedangkan pada Tabel 6.3 menunjukkan hasil dari pengujian akurasi.

Tabel 6.3 Hasil Pengujian Akurasi

Video ke-	Data Uji ke-	Kendaraan	Hasil Sistem	Hasil Aktual	Akurasi tiap Data Uji (%)	Akurasi tiap Video (%)
1	**1**	Mobil	20	26	76.9	**78.3**
		Sepeda Motor	0	0		
	2	Mobil	16	20	80	
		Sepeda Motor	0	0		
	3	Mobil	31	38	81.6	
		Sepeda Motor	0	0		
	4	Mobil	24	28	85.7	
		Sepeda Motor	0	0		
2	**5**	Mobil	3	4	94.4	**86**
		Sepeda Motor	14	14		
	6	Mobil	6	6	84	
		Sepeda Motor	15	19		
3	**7**	Mobil	7	7	76.9	**76.6**
		Sepeda Motor	23	32		
	8	Mobil	5	4	80	
		Sepeda Motor	15	21		
4	**9**	Mobil	12	13	58.8	**66.1**
		Sepeda Motor	8	21		
	10	Mobil	5	5	76	
		Sepeda Motor	14	20		
Rata – rata						**76.75**

Berdasarkan hasil pengujian akurasi 4 video dengan 10 data uji didapatkan bahwa aplikasi *vehicle counting* menggunakan *blob analysis* memiliki nilai akurasi rata – rata yakni 76.75%. Dari hasil ini disimpulkan bahwa *blob analysis* mampu melakukan *vehicle counting* dengan baik.

6.1.2 Pengujian *Threshold* untuk $R_i(x,y)$

Pengujian *threshold* untuk *Ri(x,y)* dilakukan dengan tujuan untuk mengetahui nilai *threshold* terbaik sehingga dapat menyeleksi citra hasil *background subtraction* dan membentuk gambar biner dengan baik. Pengujian ini dilakukan dengan mengganti nilai *threshold* untuk *Ri(x,y)* pada kombinasi parameter Tabel 6.2 dan membandingkan hasil sistem dengan hasil aktual (Tabel 6.1). Pengujian dilakukan pada setiap data uji dari video yang mempunyai akurasi tertinggi pada pengujian akurasi terhadap variasi data video lalu lintas, yakni video 2. Tabel 6.4 menunjukkan hasil pengujian *threshold* untuk *Ri(x,y)* pada video 2.

Berdasarkan hasil pengujian *threshold* untuk *Ri(x,y)* pada video 2 dengan 2 data uji dan 10 nilai parameter *threshold* didapatkan bahwa *threshold* 40, 45 dan 50 memiliki nilai akurasi yang sama yakni 86%. Dari hasil ini, dapat disimpulkan bahwa *threshold* 40, 45, dan 50 merupakan *threshold* terbaik untuk video 2.

Tabel 6.4 Hasil Pengujian *Threshold* untuk *Ri(x,y)* Video 2

T	Data Uji				Akurasi tiap *Threshold* (%)
	5		6		
	M	S	M	S	
15	2	5	1	0	**18.6**
20	3	8	1	11	**54.5**
25	3	12	4	12	**72.1**
30	3	13	5	15	**83.7**
35	3	14	5	14	**83.7**
40	3	14	5	15	**86**
45	3	14	5	15	**86**
50	3	14	5	15	**86**
55	3	14	5	14	**83.7**
60	3	13	5	14	**81.4**

Keterangan:

T = *Threshold*
M = Mobil
S = Sepeda motor

6.1.3 Pengujian *Alpha* (Faktor Pembobot)

Pengujian *alpha* dilakukan dengan tujuan untuk mengetahui nilai *alpha* terbaik sehingga dapat membentuk *reliable background* yang dapat menyesuaikan dengan *background* gambar. Pengujian ini dilakukan dengan mengganti nilai *alpha* pada kombinasi parameter Tabel 6.2 dan membandingkan hasil sistem dengan hasil aktual (Tabel 6.1). Pengujian dilakukan pada setiap data uji dari video yang mempunyai akurasi tertinggi pada pengujian akurasi terhadap variasi data video lalu lintas, yakni video 2. Tabel 6.5 menunjukkan hasil pengujian *alpha.*

Tabel 6.5 Hasil Pengujian *Alpha* (α) Video 2

α	Data Uji				Akurasi tiap *Alpha* (%)
	5		6		
	M	S	M	S	
0	3	13	5	10	**72.1**
0.1	3	14	5	15	**86**
0.2	3	14	5	15	**86**
0.3	3	14	5	15	**86**
0.4	3	14	5	14	**83.7**
0.5	3	14	5	12	**79.1**
0.6	2	14	5	11	**74.4**
0.7	2	13	4	12	**72.1**
0.8	2	9	3	12	**60.5**
0.9	2	9	3	12	**60.5**

Keterangan:
α = *Alpha*
M = Mobil
S = Sepeda motor

Berdasarkan hasil pengujian *alpha* pada video 2 dengan 2 data uji dan 10 nilai parameter *alpha* didapatkan bahwa *alpha* 0.1, 0.2, dan 0.3 memiliki nilai akurasi yang sama yakni 86%. Dari hasil ini, dapat disimpulkan bahwa *alpha* 0.1, 0.2, dan 0.3 merupakan *alpha* terbaik untuk video 2.

6.1.4 Pengujian Klasifikasi Menggunakan *Threshold*

Pengujian *threshold* untuk klasifikasi dilakukan dengan tujuan untuk mengetahui nilai *threshold* terbaik sehingga dapat mengklasifikasikan jenis kendaraan dengan benar. Pengujian ini dilakukan dengan mengganti nilai *threshold* klasifikasi pada kombinasi parameter Tabel 6.2 dan membandingkan hasil sistem dengan hasil aktual (Tabel 6.1). Pengujian dilakukan pada setiap data uji dari video yang mempunyai akurasi tertinggi pada pengujian akurasi terhadap variasi data video lalu lintas, yakni video 2. Tabel 6.6 menunjukkan hasil pengujian *threshold* untuk klasifikasi.

Tabel 6.6 Hasil Pengujian *Threshold* Klasifikasi Video 2

T_{mobil}	Data Uji				Akurasi tiap T_{mobil} (%)
	5		6		
	M	S	M	S	
30	3	0	5	3	**25.6**
35	3	0	5	7	**34.9**
40	3	7	5	13	**65.1**
45	3	11	5	15	**79.1**
50	3	14	5	15	**86**
55	2	14	5	15	**83.7**
60	2	14	5	15	**83.7**
65	2	14	5	15	**83.7**
70	2	14	5	15	**83.7**
75	2	14	3	15	**79.1**

Keterangan:

T_{mobil} = *Threshold* klasifikasi
M = Mobil
S = Sepeda motor

Berdasarkan hasil pengujian klasifikasi menggunakan *threshold* pada video 2 dengan 2 data uji dan 10 nilai parameter *threshold* didapatkan bahwa *threshold* 50 memiliki nilai akurasi tertinggi dengan akurasi 86%. Dari hasil ini, dapat disimpulkan bahwa *threshold* 50 merupakan *threshold* untuk klasifikasi terbaik pada video 2.

6.1.5 Pengujian Ukuran *Structure Element*

Pengujian ukuran *structure element* untuk operasi morfologi dilakukan dengan tujuan untuk mengetahui ukuran *structure element* yang terbaik sehingga dapat membentuk *blob* dengan baik. Pengujian ini dilakukan dengan mengganti ukuran *structure element* pada kombinasi parameter Tabel 6.2 dan membandingkan hasil sistem dengan hasil aktual (Tabel 6.1). Pengujian dilakukan pada setiap data uji dari video yang mempunyai akurasi tertinggi pada pengujian akurasi terhadap variasi data video lalu lintas, yakni video 2. Tabel 6.7 menunjukkan hasil pengujian ukuran *structure element* untuk operasi morfologi.

Tabel 6.7 Hasil Pengujian Ukuran *Structure Element* Video 2

SE		**Data Uji**				**Akurasi tiap *SE* (%)**
		5		6		
Erosi	**Dilasi**	M	S	M	S	
3x3	**11x11**	3	13	6	14	**83.7**
3x3	**13x13**	3	13	5	17	**86**
3x3	**17x17**	3	14	5	15	**86**
5x5	**13x13**	3	13	6	14	**83.7**
5x5	**17x17**	2	11	5	14	**74.4**

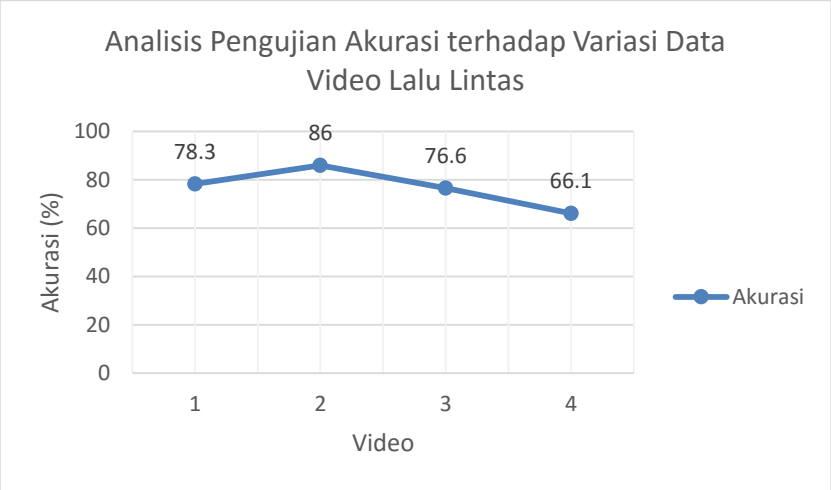
Keterangan:
SE = *Structure Element*
M = Mobil
S = Sepeda motor

Berdasarkan hasil pengujian ukuran *structure element* pada video 2 dengan 2 data uji dan 5 ukuran *structure element* didapatkan bahwa ukuran *structure element* dengan erosi 3x3 dilasi 13x13 dan erosi 3x3 dilasi 17x17 memiliki nilai akurasi tertinggi yakni 86%. Dari hasil ini, dapat disimpulkan bahwa ukuran *structure element* erosi 3x3 dilasi 13x13 dan erosi 3x3 dilasi 17x17 merupakan ukuran *structure element* untuk operasi morfologi terbaik pada video 2.

6.2 Analisis

Proses analisis bertujuan untuk mendapatkan kesimpulan dari hasil pengujian sistem vehicle counting yang telah dilakukan. Proses analisis mengacu pada hasil pengujian yang didapatkan. Analisis dilakukan terhadap hasil pengujian di setiap tahap pengujian. Proses analisis yang dilakukan meliputi analisis hasil pengujian akurasi, analisis hasil pengujian *threshold* untuk $R_i(x,y)$, analisis hasil pengujian *alpha*, analisis hasil pengujian klasifikasi menggunakan *threshold*, dan analisis pengujian *structure element*.

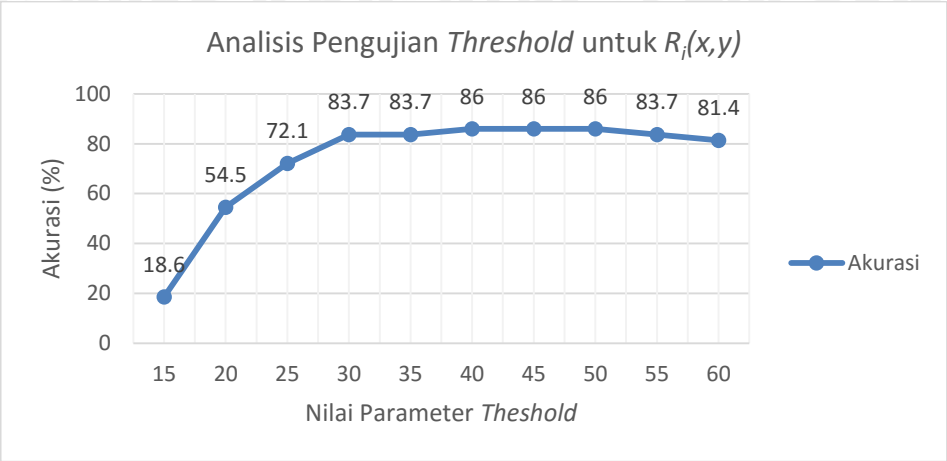
6.2.1 Hasil Pengujian Akurasi terhadap variasi Data Video Lalu Lintas



Gambar 6.3 Analisis Pengujian Akurasi terhadap Variasi Data Video Lalu Lintas

Gambar 6.3 adalah grafik yang menjelaskan tentang hasil pengujian akurasi terhadap variasi data video lalu lintas. Dari grafik dapat dilihat bahwa nilai akurasi terbesar yakni 86% ditunjukkan pada video 2 dan nilai akurasi terendah pada video 4 dengan akurasi 66.1%. Video 2 memiliki nilai akurasi tertinggi dikarenakan sudut pengambilan video dari depan kendaraan yang menyebabkan ketika kendaraan memasuki daerah *tracking*, kendaran semakin besar dan memisah dari kendaraan yang lainnya meskipun awalnya beberapa kendaraan menjadi satu kendaraan. Sebaliknya, video 4 memiliki nilai akurasi terendah dikarenakan sudut pengambilan video dari belakang kendaraan menyebabkan ketika kendaraan memasuki daerah *tracking,* kendaraan semakin kecil dan menyatu sebagai satu kendaraan. Video 3 dan video 4 memiliki sudut pengambilan gambar yang sama, akan tetapi perbedaannya adalah pada panjang jalan yang digunakan untuk daerah *tracking*. Daerah *tracking* adalah daerah yang dipilih pada saat rata – rata atau kebayakan objek dapat membentuk *blob* yang terbaik. Pada video 4, daerah *tracking* sangat terbatas karena untuk satu kendaraan mobil dalam membentuk *blob* yang terbaik sudah menghabiskan hampir setengah dari gambar. Jika daerah daerah *tracking* diperjauh lagi, maka kendaraan akan semakin mengecil dan menyebabkan banyak kendaraan yang menyatu menjadi satu kendaaraan. Dilihat dari segi kepadatan kendaraan dapat disimpulkan sistem mampu menghasilkan nilai akurasi yang lebih baik dengan rata – rata kendaraan pada setiap data uji yakni maksimal 25 kendaraan/30 detik. Data uji 9 dan 10 merupakan data uji dari video yang sama tetapi dengan kepadatan kendaraan yang berbeda dapat menyebabkan perbedaan akurasi yang cukup jauh. Data uji 9 dengan kepadatan kendaraan 34 kendaraan/30 detik memiliki akurasi 58,8%, sedangkan data uji ke 10 dengan kepadatan kendaraan 25 kendaraan/30 detik memiliki akurasi 76%. Data uji dengan kepadatan kendaraan lebih dari 25 kendaraan/30 detik dapat menyebabkan beberapa kendaraan menjadi satu kendaraan, sehingga menyebabkan akurasi berkurang.

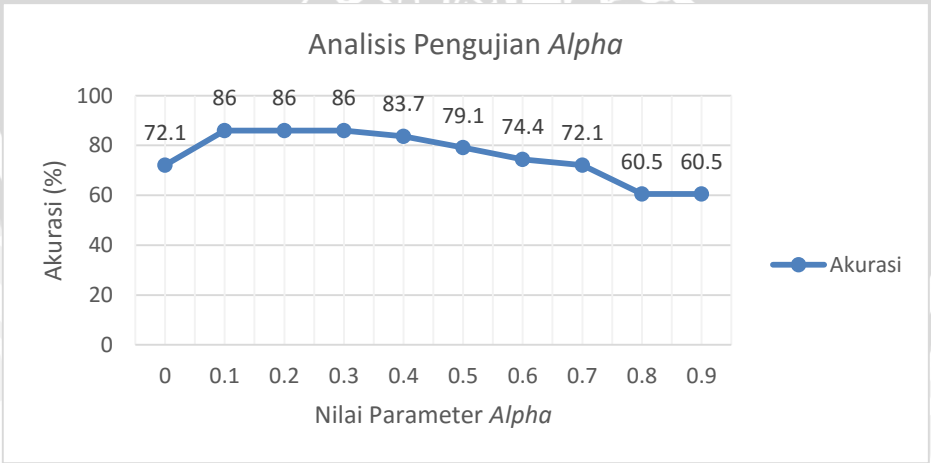
6.2.2 Hasil Pengujian *Threshold* untuk $R_i(x,y)$



Gambar 6.4 Analisis Pengujian *Threshold* untuk $R_i(x,y)$

Gambar 6.4 adalah grafik yang menjelaskan tentang hasil pengujian *threshold* untuk $R_i(x,y)$ pada video 2. Dari grafik dapat dilihat bahwa *threshold* 40, 45, dan 50 memiliki nilai akurasi tertinggi yakni 86%. Nilai akurasi dari *threshold* yang mendekati *threshold* 40, 45, dan 50 akurasinya semakin tinggi dan nilai akurasi yang menjauhi *threshold* 40, 45, dan 50 akurasinya semakin rendah. Hal ini disebabkan karena jika nilai *threshold* semakin kecil atau semakin besar maka sistem tidak mampu menyeleksi *background* dengan baik. Jika *threshold* semakin kecil, maka *background* dapat terdeteksi sebagai objek. Sedangkan jika *threshold* semakin besar, *blob – blob* objek akan menjadi lebih besar, sehingga dapat menyatu dengan *blob* yang lain.

6.2.3 Hasil Pengujian *Alpha* (Faktor Pembobot)

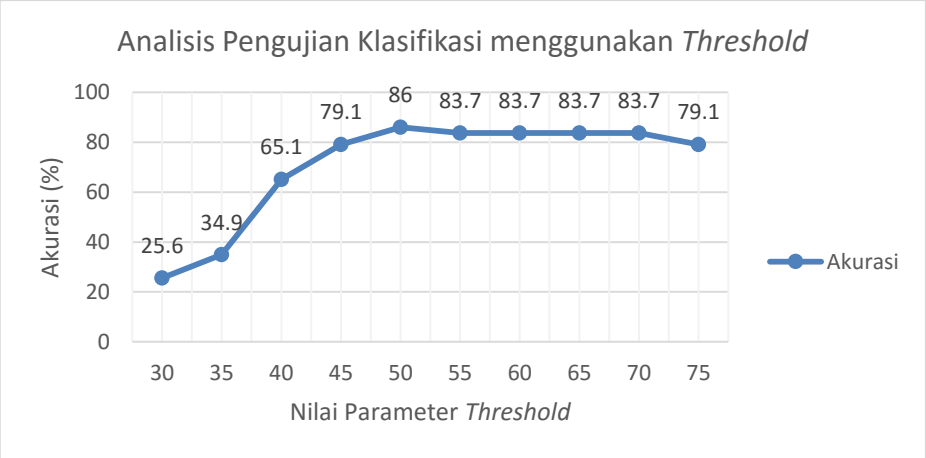


Gambar 6.5 Analisis Pengujian *Alpha* (Faktor Pembobot)

Gambar 6.5 adalah grafik yang menjelaskan tentang hasil pengujian nilai *alpha* pada video 2. Dari grafik dapat dilihat bahwa nilai akurasi tertinggi yakni 86% ditunjukkan pada nilai *alpha* 0.1, 0.2, dan 0.3. Semakin tinggi nilai *alpha*, nilai akurasi semakin kecil. Hal ini disebabkan *background* membentuk *blob – blob* baru

yang terhitung sebagai objek dan menganggu saat proses *tracking*, sehingga sistem tidak mampu melacak apakah objek tersebut merupakan objek yang sama atau objek yang berbeda dengan *frame* sebelumnya. Pada nilai *alpha* 0 yang memberikan hasil *background* sama dengan *background* yang tidak mengalami *update background* memiliki akurasi 72.1%, dimana akurasi lebih kecil daripada nilai *alpha* 0.1. Ini membuktikan bahwa *background updating* membantu dalam proses deteksi objek.

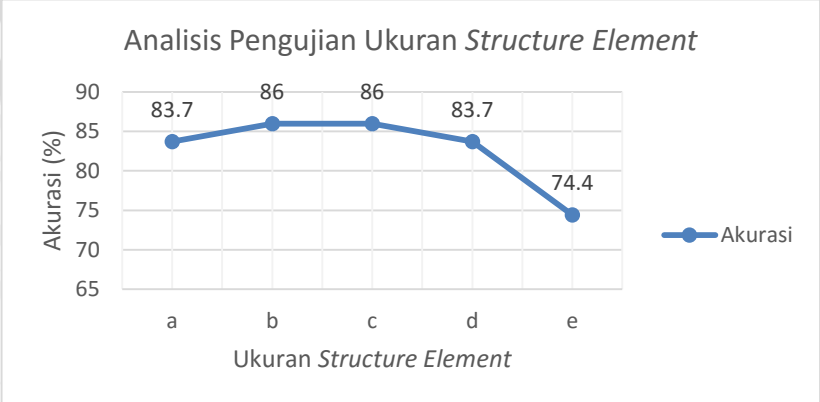
6.2.4 Hasil Pengujian Klasifikasi Menggunakan *Threshold*



Gambar 6.6 Analisis Pengujian Klasifikasi menggunakan *Threshold*

Gambar 6.6 adalah grafik yang menjelaskan tentang hasil pengujian klasifikasi menggunakan *threshold* pada video 2. Dari grafik dapat dilihat bahwa nilai akurasi tertinggi yakni 86% ditunjukkan pada nilai *threshold* 50. *Threshold* 50 menunjukkan bahwa pada video 2, lebar mobil rata – rata lebih dari 50 px dan lebar sepeda motor rata – rata kurang dari 50 px. Jika nilai *threshold* kurang dari 50 dan lebih dari 50 nilai akurasinya semakin rendah. Hal ini dapat disebabkan karena sepeda motor diklasifikasikan menjadi mobil atau mobil diklasifikasikan sebagai sepeda motor.

6.2.5 Hasil Pengujian Ukuran *Structure Element*



Gambar 6.7 Analisis Pengujian Ukuran *Structure Element*

Tabel 6.8 Ukuran *Structure Element*

SE	Erosi	Dilasi
a	3x3	11x11
b	3x3	13x13
c	3x3	17x17
d	3x3	13x13
e	3x3	17x17

Gambar 6.7 adalah grafik yang menjelaskan tentang hasil pengujian ukuran *structure element* (SE) pada video 2. Dari grafik dan Tabel 6.8 dapat dilihat bahwa nilai akurasi tertinggi yakni 86% ditunjukkan pada ukuran SE dengan erosi 3x3 dilasi 13x13 dan erosi 3x3 dilasi 17x17. Ukuran SE dengan erosi 3x3 dilasi 13x13 dan erosi 3x3 dilasi 17x17 menunjukkan bahwa pada video 2 sudah dapat melakukan segmentasi lebih baik dari pada ukuran SE yang lainnya. Ukuran SE yang tepat akan menghasilkan segmentasi *blob* yang baik. Jika ukuran erosi besar dan tidak diimbangi dengan dilasi yang tepat, maka *blob* akan terlihat lebih kecil dari ukuran sebenarnya atau dapat menyebabkan sebuah *blob* akan terpecah menjadi beberapa *blob*. Tetapi jika ukuran erosi terlalu kecil juga tidak mampu menghilangkan *noise* yang tidak diperlukan. Sebaliknya jika ukuran dilasi terlalu besar, maka akan menyebabkan ukuran *blob* lebih besar dari pada objek yang sebenarnya dan dapat menyebabkan beberapa *blob* bergabung menjadi sebuah *blob*.

BAB 7 PENUTUP

Pada bab penutup akan dibahas mengenai kesimpulan dan saran dari implementasi *blob analysis* untuk *vehicle counting* pada video lalu lintas. Berikut kesimpulan dan sarannya.

7.1 Kesimpulan

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

1. Aplikasi *vehicle counting* dengan menggunakan *blob analysis* telah mampu melakukan *counting* pada video lalu lintas secara baik dimana cara kerjanya dengan melakukan ekstraksi fitur dari objek yang terdeteksi kemudian membandingkan objek – objek yang terdeteksi merupakan objek yang sama atau objek yang berbeda.
2. Aplikasi *vehicle counting* dengan menggunakan *blob analysis* memiliki nilai akurasi tertinggi sebesar 86% dari 4 video berbeda dengan 10 data uji. Dari segi kepadatan kendaraan, sistem mampu menghasilkan nilai akurasi yang lebih baik dengan rata – rata kendaraan pada setiap data uji yakni maksimal 25 kendaraan/30 detik.
3. Sudut pengambilan video sangat berpengaruh pada aplikasi *vehicle counting* dengan menggunakan *blob analysis*. Sudut pengambilan video yang terbaik adalah ketika kendaraan semakin mendekati kamera, sedangkan sudut pengambilan video yang terburuk adalah ketika kendaraan semakin menjauhi kamera.

7.2 Saran

Saran yang diberikan untuk pengembangan penelitian selanjutnya, antara lain:

1. Diharapkan pada penelitian berikutnya dapat menggunakan data video yang lebih banyak dan lebih baik dalam pengambilan video, sehingga dapat mewakili sebagian besar dari variasi jalan dan kendaraan serta mengurangi kesalahan perhitungan dikarenakan pengambilan video yang kurang baik.
2. Melakukan pengembangan metode untuk deteksi kendaraan dengan menggabungkan metode *background subtraction* dengan metode lain agar mendapatkan hasil segmentasi kendaraan yang lebih baik. Hasil segmentasi yang baik tidak akan membuat dua kendaraan menjadi satu atau satu kendaraan menjadi dua kendaraan, sehingga akan minimalkan kesalahan perhitungan saat proses *counting*.
3. Diharapkan pada penelitian selanjutnya dapat membedakan tipe video secara otomatis sehingga tidak dilakukan secara manual dan agar dapat membentuk *base line* yang sesuai dengan tipe video.

4. Diharapkan pada penelitian selanjutnya dalam mengklasifikasikan tidak hanya menggunakan tinggi atau lebar objek tetapi menggunakan fitur – fitur yang dimiliki oleh objek.

DAFTAR PUSTAKA

AForge.NET Framework, 2013. AForge.NET Framework (2.2.5). [program komputer] AForge.NET. Tersedia di: <http://www.aforgenet.com/framework/downloads.html> [Diakses 30 April 2013]

Agustina, S.E. dan Mukhlash, I., 2012. Jurnal Sains Dan Seni. *Implementasi Metode Scale Invariant Feature Transform (SIFT) dan Metode Continuosly Adaptive Mean-Shift(Camshift) Pada Penjejakan Objek Bergerak*, Vol. 1, No. 1, 1-6.

Alnect Komputer, 2015. *3GPP IP Camera TP-Link TL-SC3000*. [online] Alnect Komputer. Tersedia di: <http://alnect.net/products.php?/13/88/265/622/Special-Tools/CCTV-IP-Camera/IP-Camera/3GPP-IP-Camera-TP-Link-TL-SC3000> [diakses pada 12 April 2015]

Anonymous, 2012. *BLOB Analysis (Introduction to Video and Image Processing) Part 1*. [online] what-when-how. Tersedia di: <http://what-when-how.com/introduction-to-video-and-image-processing/blob-analysis-introduction-to-video-and-image-processing-part-1/> [diakses pada: 21 Januari 2016]

Autojaya Idetech, 2011. *Multimedia Thru Wireless LAN*. [online] PT. Autojaya Idetech. Tersedia di: <http://autojaya.com/ind/bulletin/year_2011/vol_16.html> [diakses pada 11 April 2015]

Avtech Surabaya, 2011. *Mengenal Kelebihan IP Camera Avtech*. [online] Avtech Surabaya. Tersedia di: <http://avtechsurabaya.com/tutorial/cctv/kelebihan-ip-camera-avtech.html> [diakses pada 12 April 2015]

Chen, T.H., Lin, Y.F. dan Chen, T.Y, 2007. *Intellegent Vehicle Counting Method Based on Blob Analysis in Traffic Surveillance*. *Departement of Electronic Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan 807, R.O.C*.

Hapsani, A.G., 2014. *Implementasi Metode Scale Invariant Feature Transform (SIFT) untuk Multiple Object Tracking Pada Video CCTV*. S1. Universitas Brawijaya.

Negara, N.O. dan Rahman, A., 2011. *Perancangan Active Surveillance Camera Dalam Otomasi Pengawasasan Gedung*. Institut Teknologi Sepuluh November.

Nurhadiyatna, A., Jatmiko, W., Hardjono, B., Wibisono, A., Sina, I. dan Mursanto, P., 2013. *Background Subtraction Using Gaussian Mixture Model Enhanced by Hole Filling Algorithm (GMMHF)*. 2013 IEEE International Conference on Systems, Man, and Cybernetics.

Pratama, B.Y., 2014. *Operasi Morfologi Pada Citra Biner*. [pdf] Komunitas eLearning IlmuKomputer.Com. Tersedia di: <http://ilmukomputer.org/wp-content/uploads/2014/02/Batra-Operasi-Morfologi-Pada-Citra-Biner.pdf> [Diakses 5 Januari 2016]

Rahman, F.Y.A., Hussain, A., Zaki, W.M.D.W., Zaman, H.B. dan Tahir, N.M., 2013. *Enhancement of Background Subtraction Techniques Using a Second Derivative in Gradient Direction Filter*. Selangor: Jurnal of Electrical and Computer Engineering. Vol. 2013.

Rahman, M.S., Saha. A. dan Khanum, S., 2009. *Multi-Object Tracking in Video Sequences Based on Background Subtraction and SIFT Feature Matching*, Fourth International Conference on Computer Sciences and Convergence Information Technology.

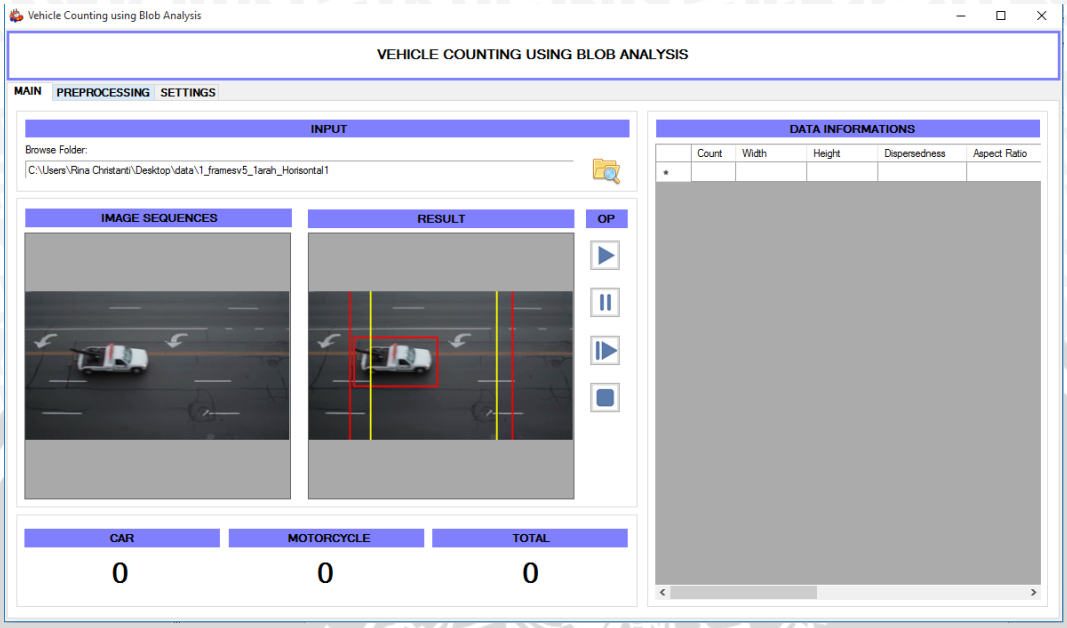
Taksiono, M.A., 2013. "Rancang Bangun Sistem Penghitung Laju dan Klasifikasi Kendaraan Berbasis Pengolahan Citra". Teknik Elektro, Fakultas Teknologi Industri. Institut Teknologi Sepuluh September.

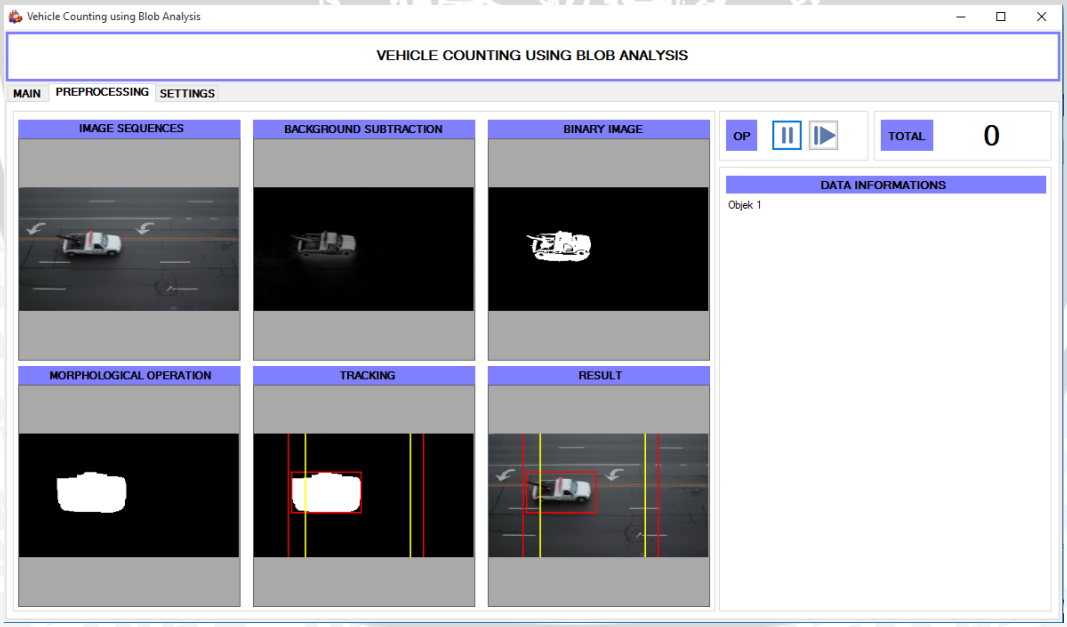
Xvision Group, 2013. *What Is IP CCTV?*. [online] Xvision Group. Tersedia di: <http:// http://www.ipcctv.com/article.php?xArt=10> [diakses pada 12 April 2015]

LAMPIRAN A *SCREEN CAPTURE* GUI

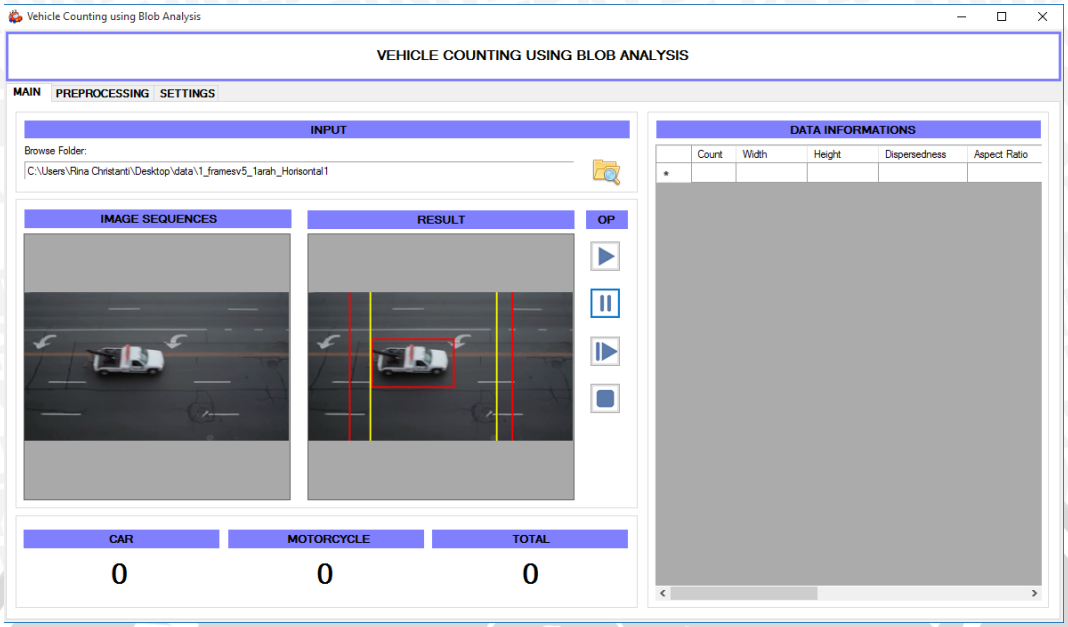
A.1 Gambar *Step By Step* Video 1 (5 *Frame*)

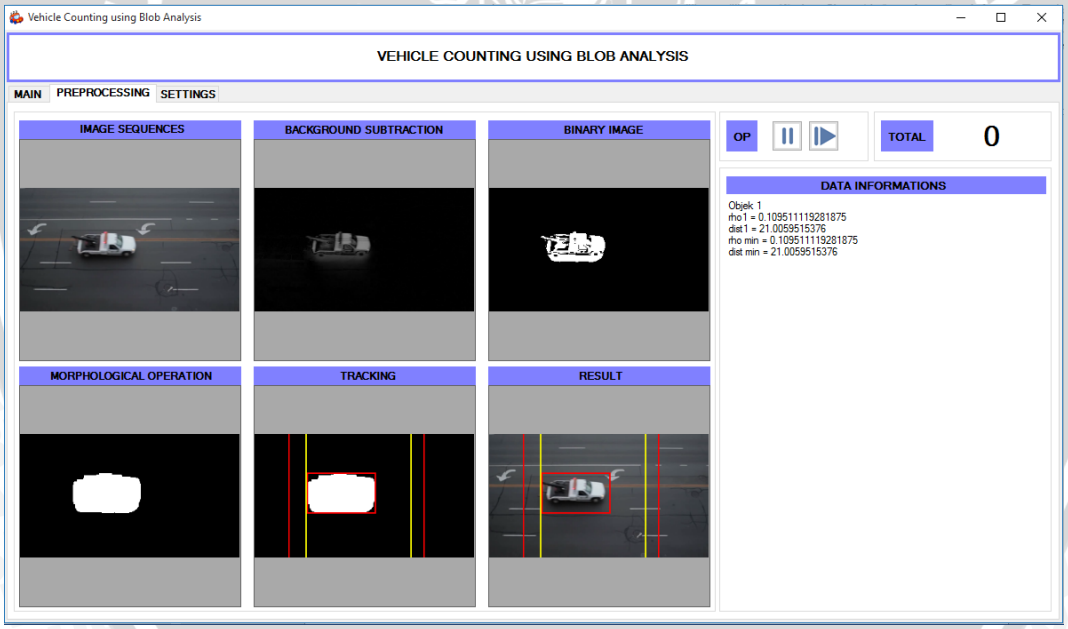
1. *Frame* 1



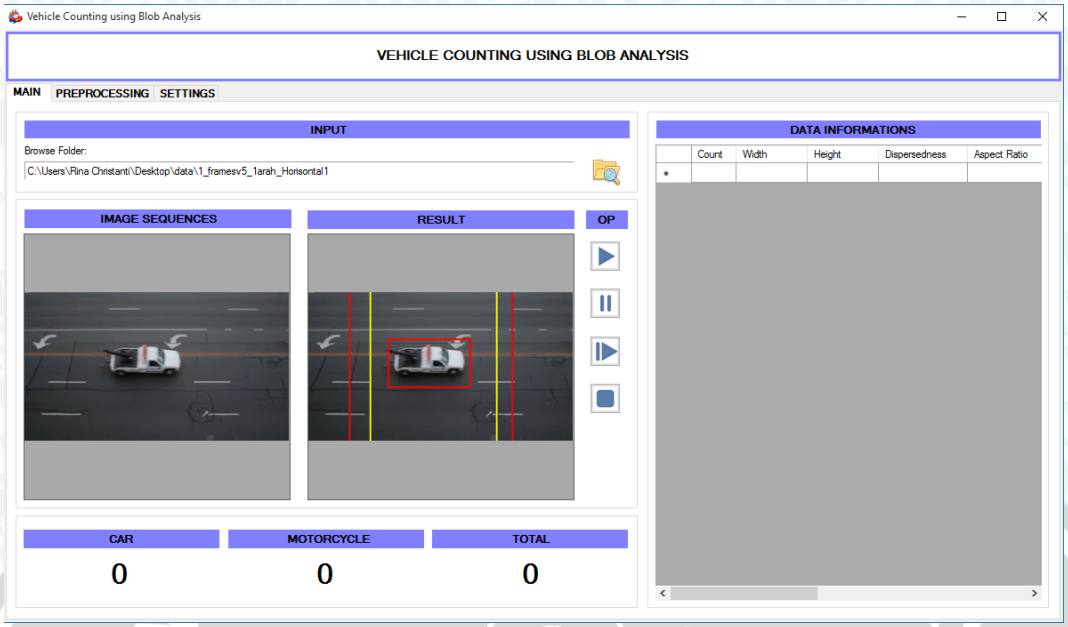


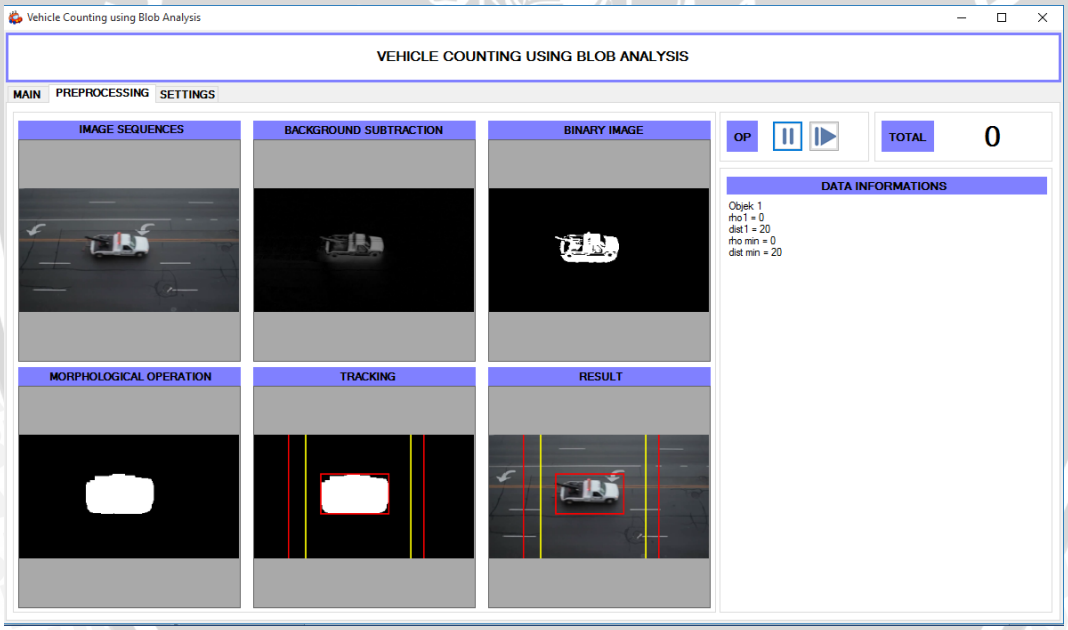
2. *Frame 2*


Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
INPUT
Browse Folder:
C:\Users\Rina Christanti\Desktop\data\1_framesv5_1arah_Horisontal1
DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
IMAGE SEQUENCES
RESULT
OP
CAR
MOTORCYCLE
TOTAL
0
0
0

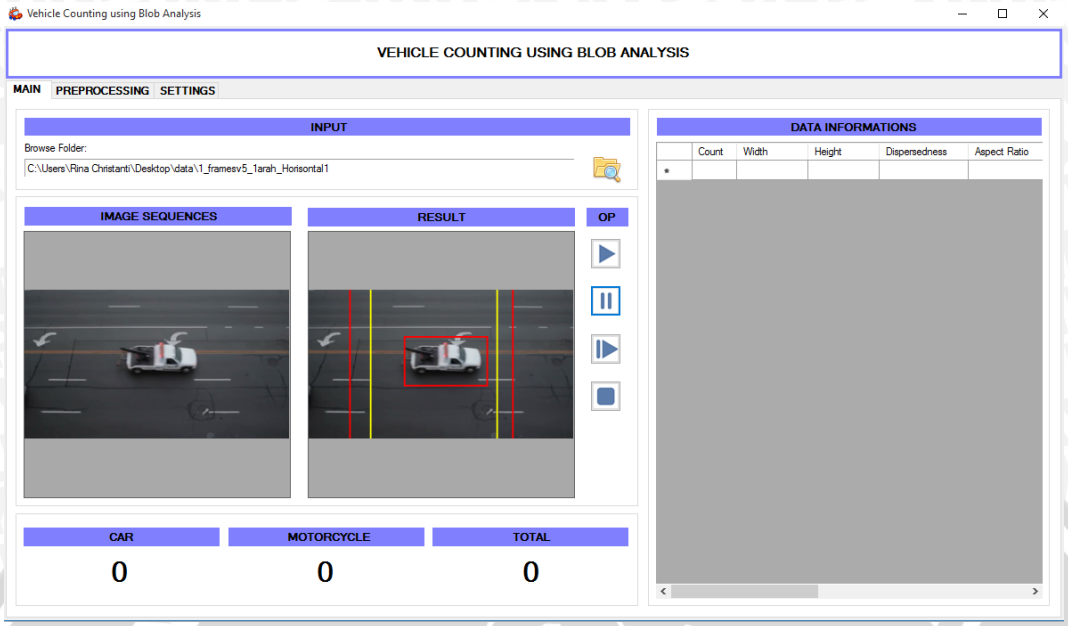

Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
IMAGE SEQUENCES
BACKGROUND SUBTRACTION
BINARY IMAGE
OP
TOTAL
0
DATA INFORMATIONS
Objek 1
rho1 = 0.109511119281875
dist1 = 21.0059515376
rho min = 0.109511119281875
dist min = 21.0059515376
MORPHOLOGICAL OPERATION
TRACKING
RESULT

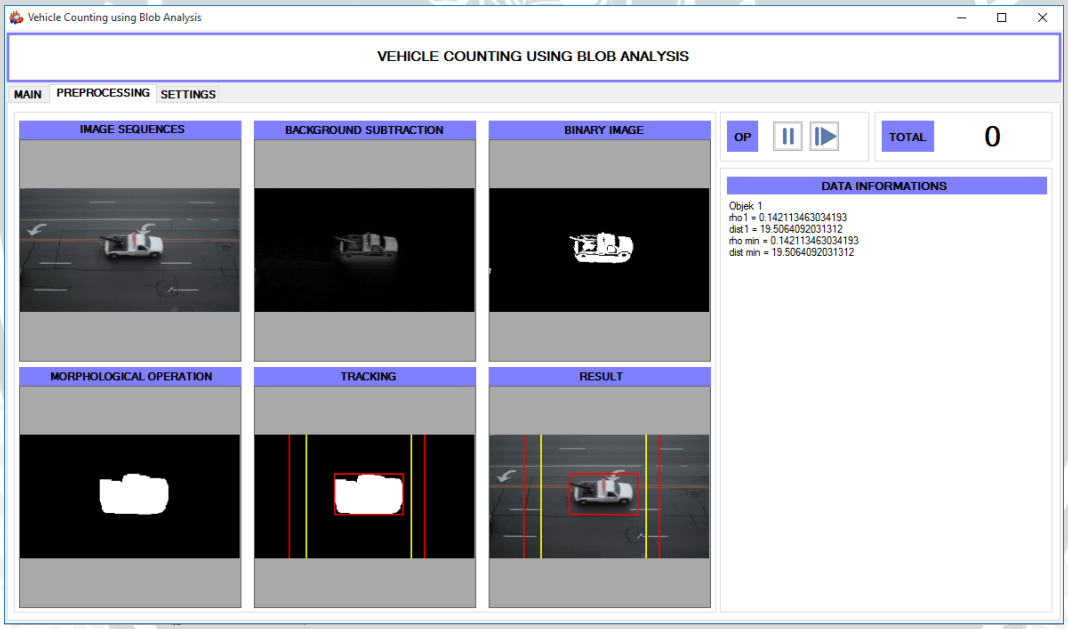
3. *Frame* 3


Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
INPUT
Browse Folder:
C:\Users\Rina Christanti\Desktop\data\1_framesv5_1arah_Horisontal1
DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
IMAGE SEQUENCES
RESULT
OP
CAR
MOTORCYCLE
TOTAL
0
0
0

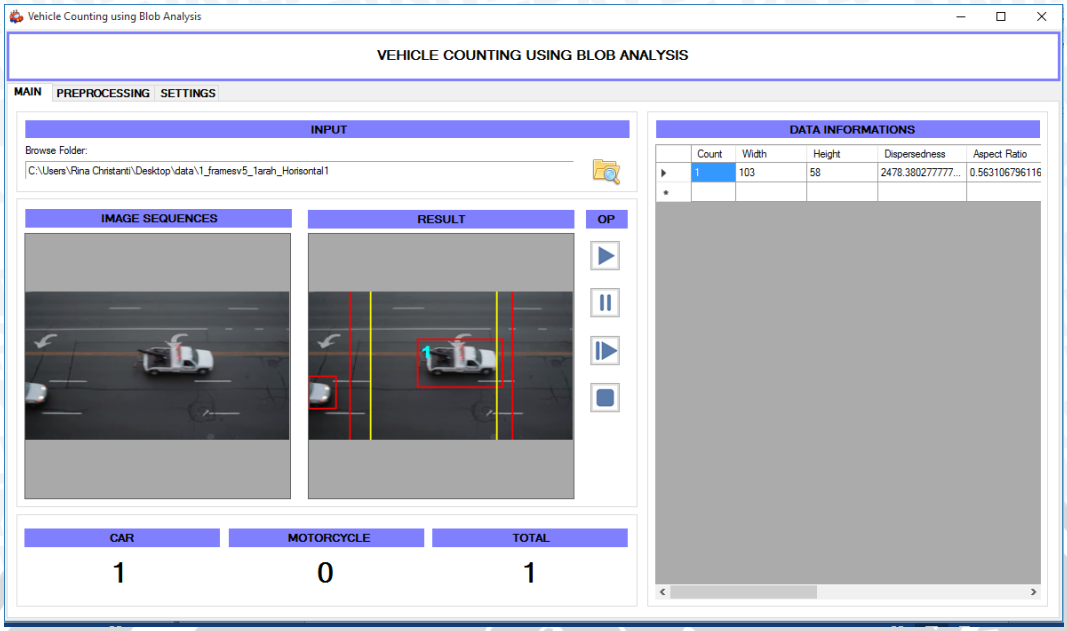

Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
IMAGE SEQUENCES
BACKGROUND SUBTRACTION
BINARY IMAGE
OP
TOTAL
0
DATA INFORMATIONS
Objek 1
rho1 = 0
dist1 = 20
rho min = 0
dist min = 20
MORPHOLOGICAL OPERATION
TRACKING
RESULT

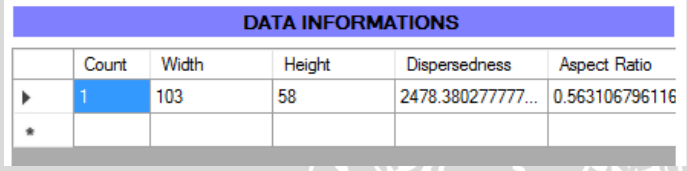
4. *Frame* 4

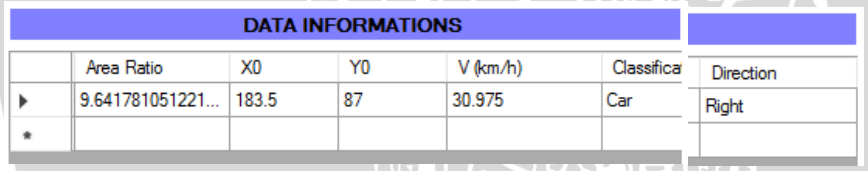

Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
INPUT
Browse Folder:
C:\Users\Rina Christanti\Desktop\data\1_framesv5_1arah_Horisontal1
DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
IMAGE SEQUENCES
RESULT
OP
CAR
MOTORCYCLE
TOTAL
0
0
0

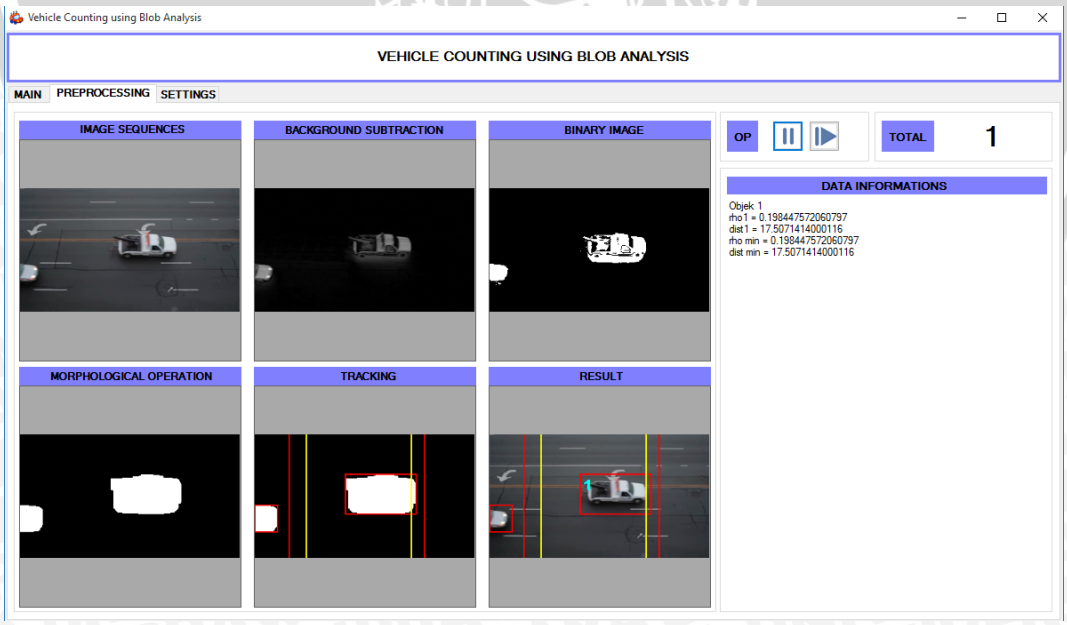

Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
IMAGE SEQUENCES
BACKGROUND SUBTRACTION
BINARY IMAGE
OP
TOTAL
0
DATA INFORMATIONS
Objek 1
rho1 = 0.142113463034193
dist1 = 19.5064092031312
rho min = 0.142113463034193
dist min = 19.5064092031312
MORPHOLOGICAL OPERATION
TRACKING
RESULT

5. *Frame* 5


Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
INPUT
Browse Folder:
C:\Users\Rina Christanti\Desktop\data\1_framesv5_1arah_Horisontal1
DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
1
103
58
2478.380277777...
0.563106796116
IMAGE SEQUENCES
RESULT
OP
CAR
MOTORCYCLE
TOTAL
1
0
1

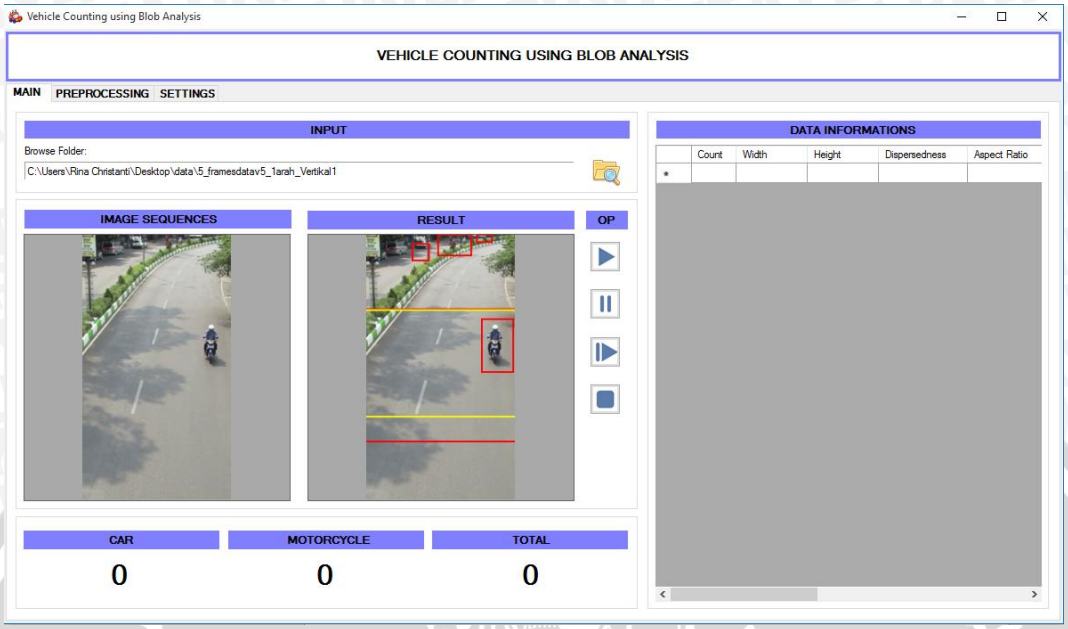

DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
1
103
58
2478.380277777...
0.563106796116

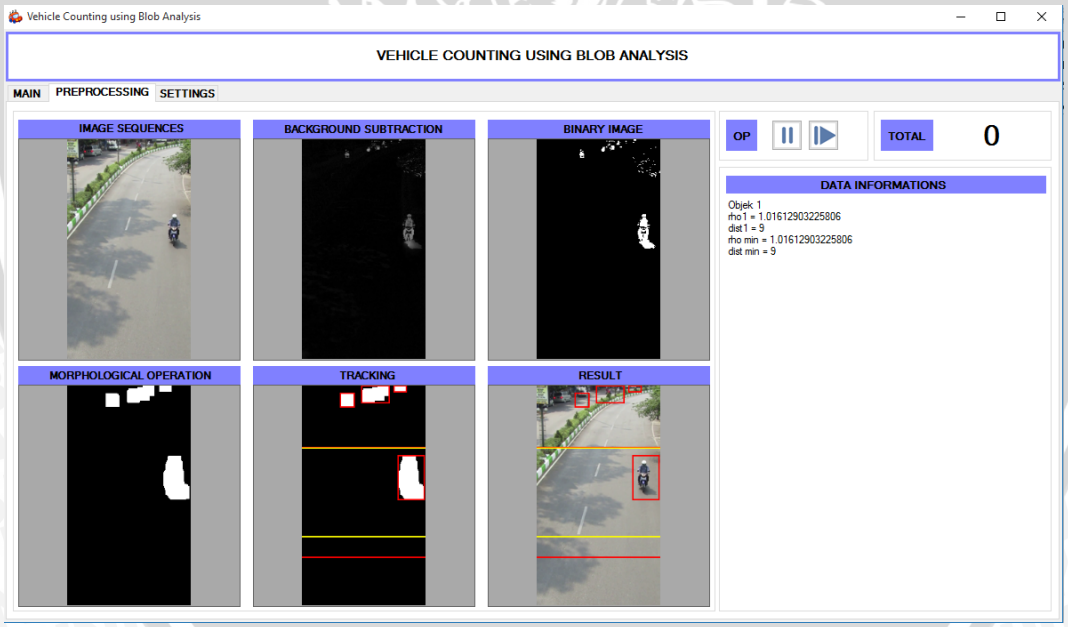

DATA INFORMATIONS
Area Ratio
X0
Y0
V (km/h)
Classifica
9.641781051221...
183.5
87
30.975
Car
Direction
Right


Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
IMAGE SEQUENCES
BACKGROUND SUBTRACTION
BINARY IMAGE
OP
TOTAL
1
DATA INFORMATIONS
Objek 1
rho1 = 0.19844572060797
dist 1 = 17.5071414000116
rho min = 0.19844572060797
dist min = 17.5071414000116
MORPHOLOGICAL OPERATION
TRACKING
RESULT

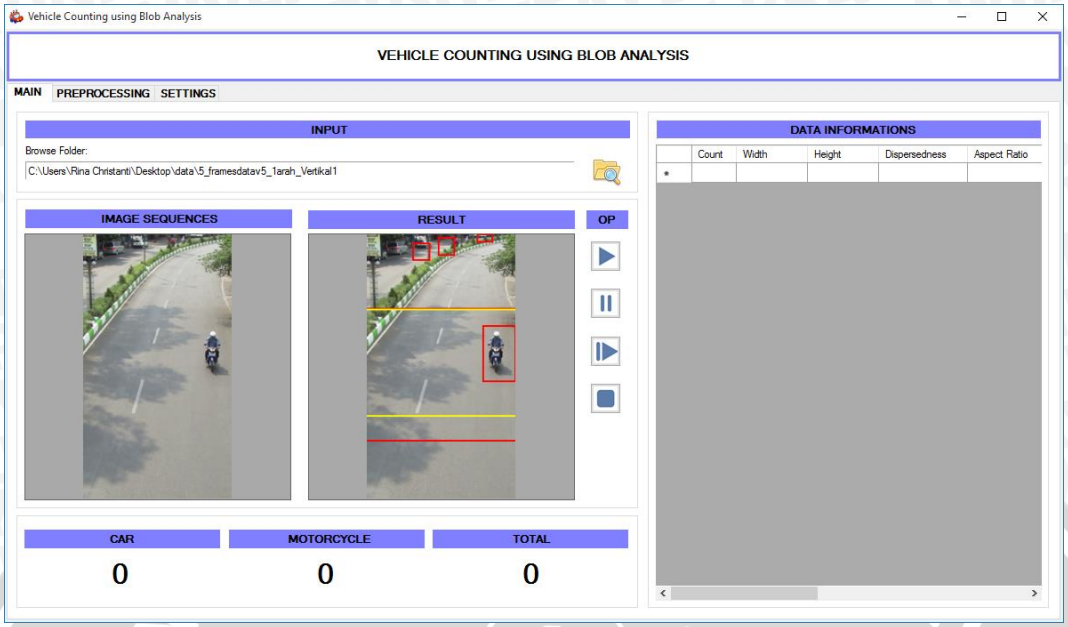
A.2 Gambar *Step By Step* Video 2 (5 *Frame*)

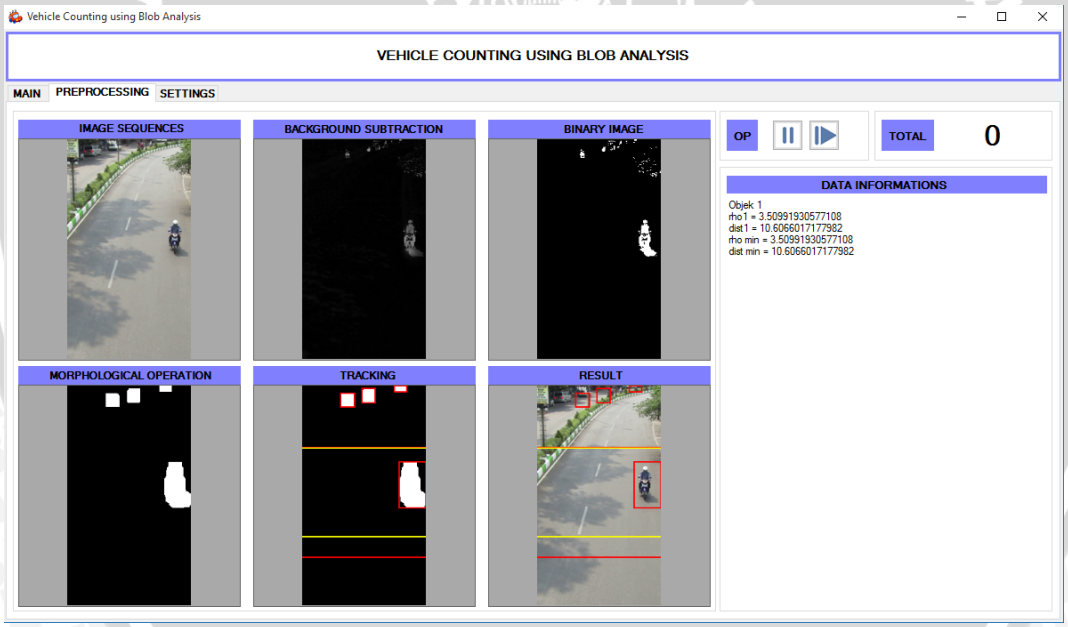
1. *Frame* 1



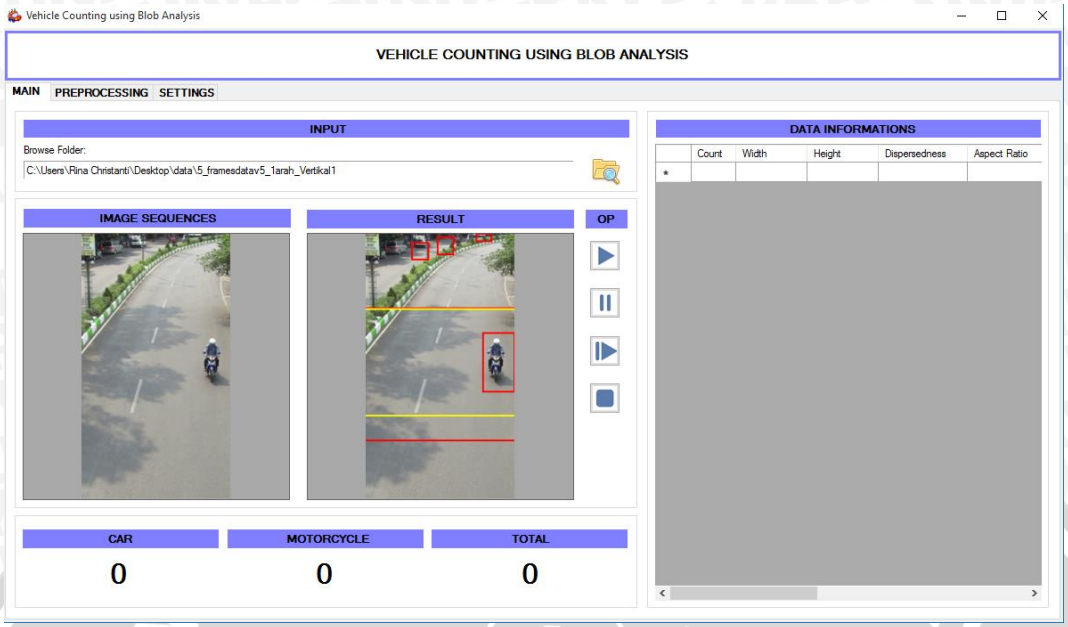


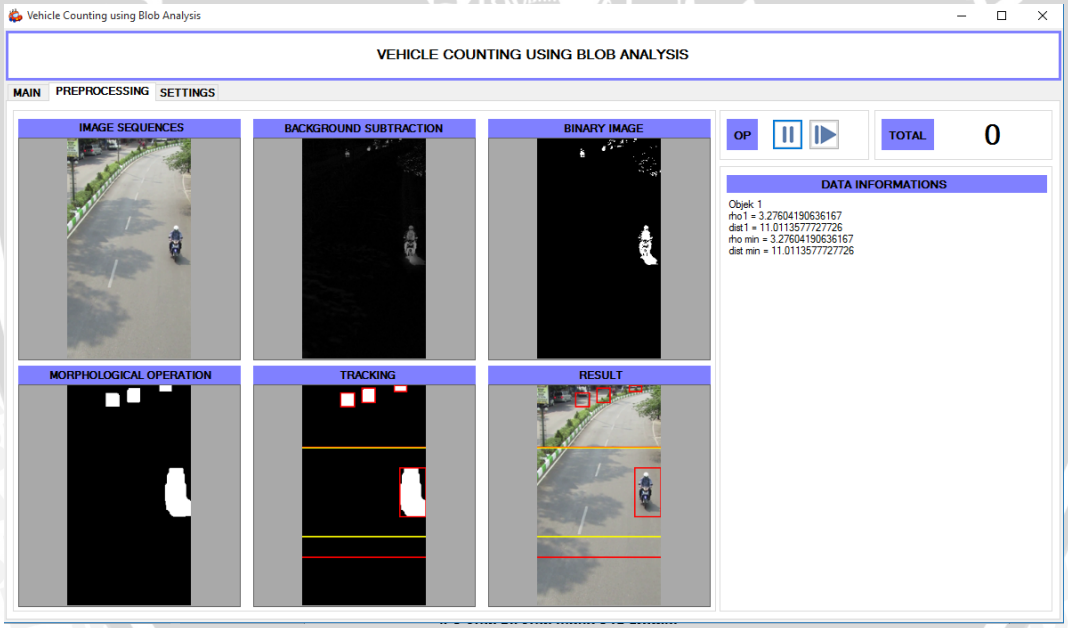
2. *Frame* 2


Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
INPUT
Browse Folder:
C:\Users\Rina Christanti\Desktop\data\5_framesdatav5_1arah_Vertikal1
DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
IMAGE SEQUENCES
RESULT
OP
CAR
MOTORCYCLE
TOTAL
0
0
0

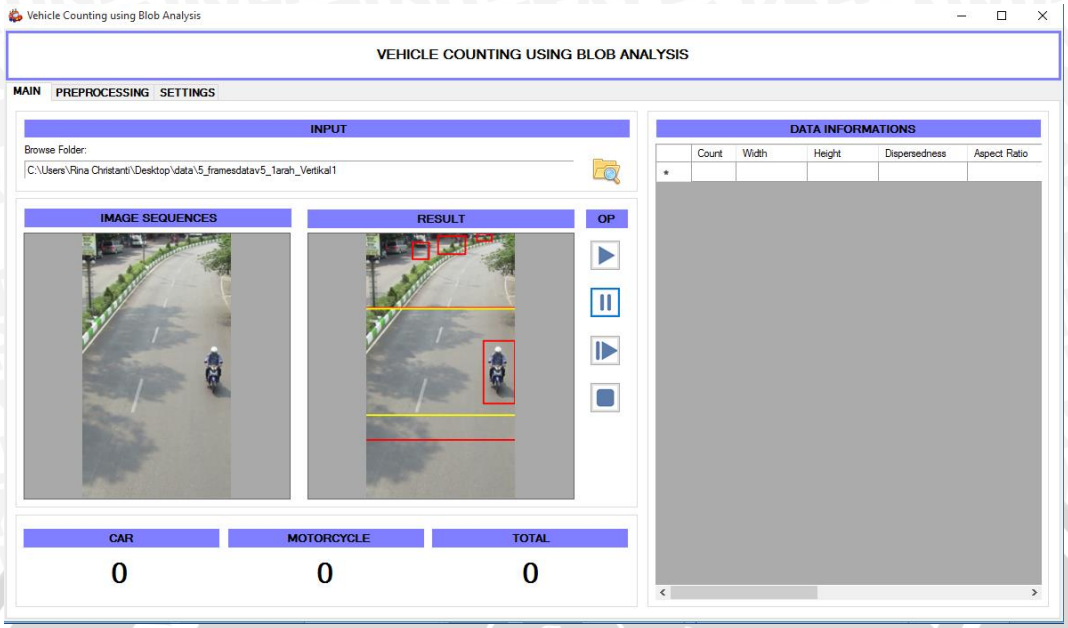

Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
IMAGE SEQUENCES
BACKGROUND SUBTRACTION
BINARY IMAGE
OP
TOTAL
0
DATA INFORMATIONS
Objek: 1
rho1 = 3.509919305771O8
dist 1 = 10.6066017177982
rho min = 3.509919305771O8
dist min = 10.6066017177982
MORPHOLOGICAL OPERATION
TRACKING
RESULT

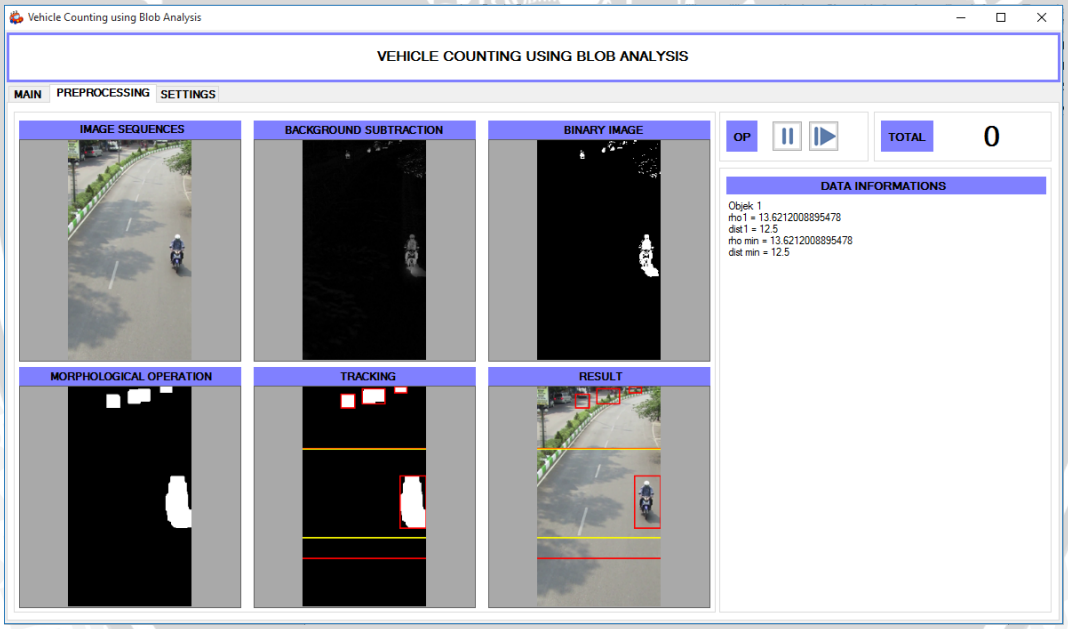
3. *Frame* 3


Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
INPUT
Browse Folder:
C:\Users\Rina Christanti\Desktop\data\5_framesdatav5_1arah_Vertikal1
DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
IMAGE SEQUENCES
RESULT
OP
CAR
MOTORCYCLE
TOTAL
0
0
0

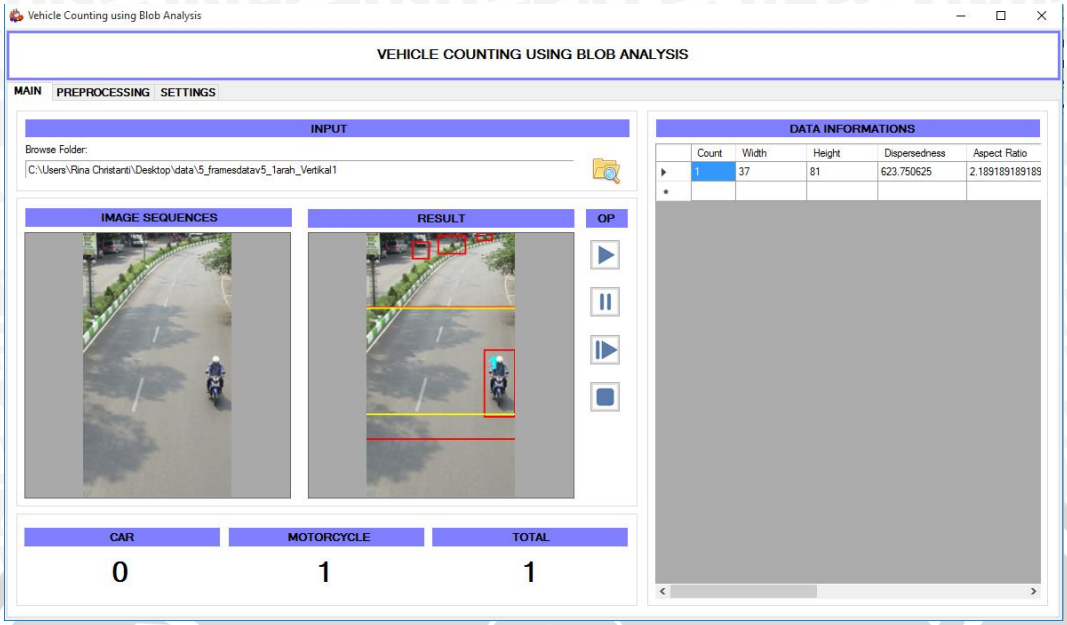

Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
IMAGE SEQUENCES
BACKGROUND SUBTRACTION
BINARY IMAGE
OP
TOTAL
0
DATA INFORMATIONS
Objek: 1
rho1 = 3.27604190636167
dist1 = 11.0113577727726
rho min = 3.27604190636167
dist min = 11.0113577727726
MORPHOLOGICAL OPERATION
TRACKING
RESULT

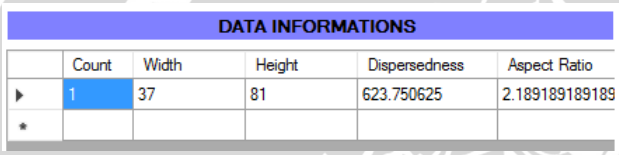
4. *Frame* 4

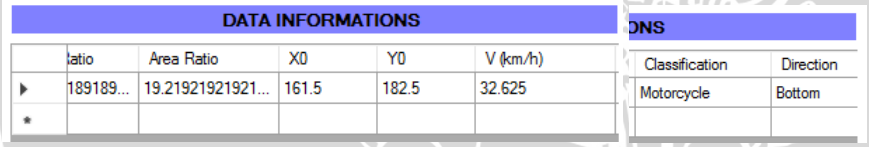

Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
INPUT
Browse Folder:
C:\Users\Rina Christanti\Desktop\data\5_framesdatav5_1arah_Vertikal1
DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
IMAGE SEQUENCES
RESULT
OP
CAR
MOTORCYCLE
TOTAL
0
0
0

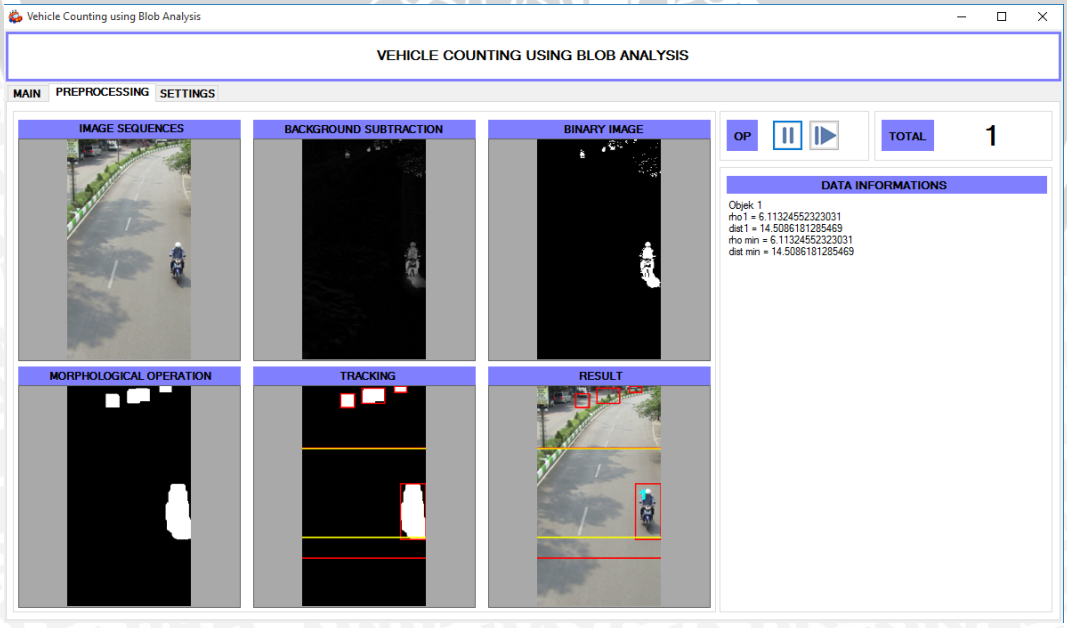

Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
IMAGE SEQUENCES
BACKGROUND SUBTRACTION
BINARY IMAGE
OP
TOTAL
0
DATA INFORMATIONS
Objek 1
rho1 = 13.6212008895478
dist1 = 12.5
rho min = 13.6212008895478
dist min = 12.5
MORPHOLOGICAL OPERATION
TRACKING
RESULT

5. *Frame* 5


Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
INPUT
Browse Folder:
C:\Users\Rina Christanti\Desktop\data\5_framesdatav5_1arah_Vertikal1
DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
1
37
81
623.750625
2.189189189189
IMAGE SEQUENCES
RESULT
OP
CAR
MOTORCYCLE
TOTAL
0
1
1

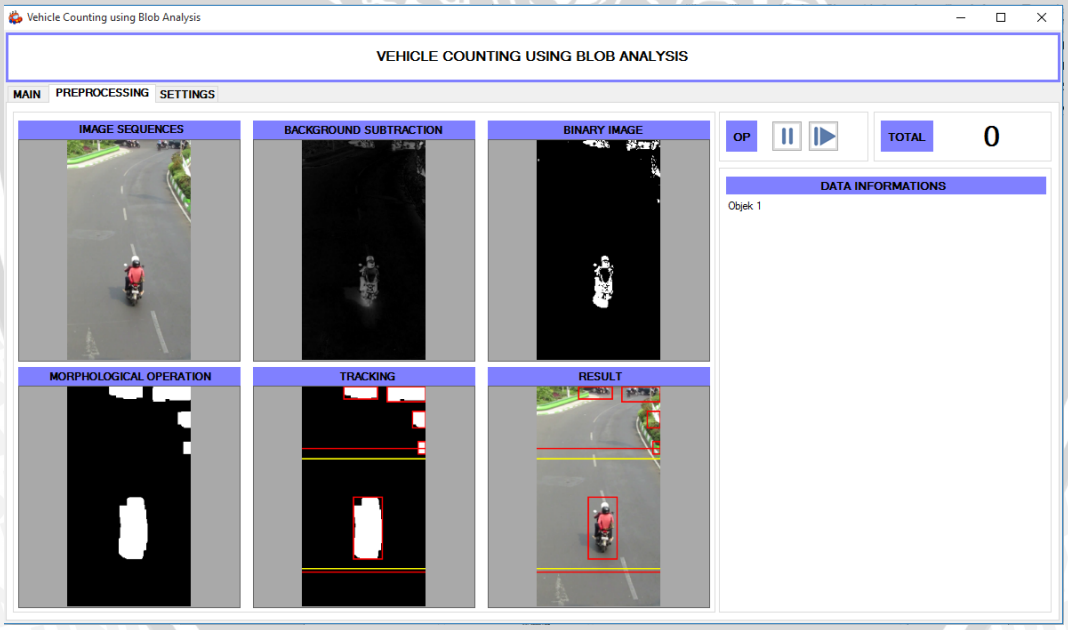

DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
1
37
81
623.750625
2.189189189189


DATA INFORMATIONS
Area Ratio
X0
Y0
V (km/h)
19.21921921921...
161.5
182.5
32.625
Classification
Direction
Motorcycle
Bottom

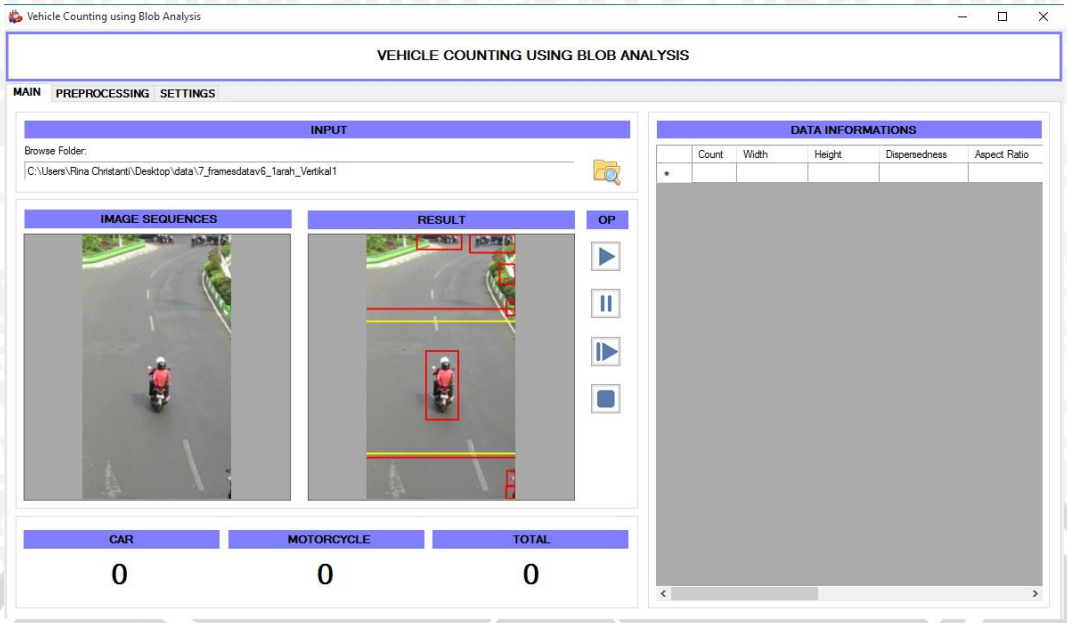

Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
IMAGE SEQUENCES
BACKGROUND SUBTRACTION
BINARY IMAGE
OP
TOTAL
1
DATA INFORMATIONS
Objek 1
rho1 = 6.11324552323031
dist1 = 14.5086181285469
rho min = 6.11324552323031
dist min = 14.5086181285469
MORPHOLOGICAL OPERATION
TRACKING
RESULT

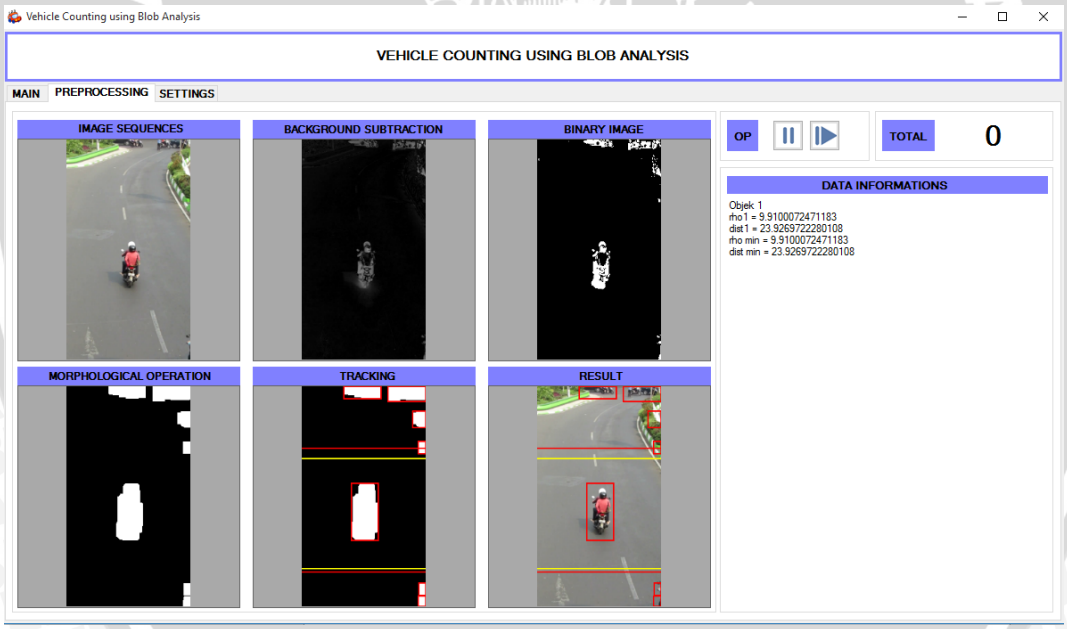
A.3 Gambar *Step By Step* Video 3 (5 *Frame*)

1. *Frame* 1

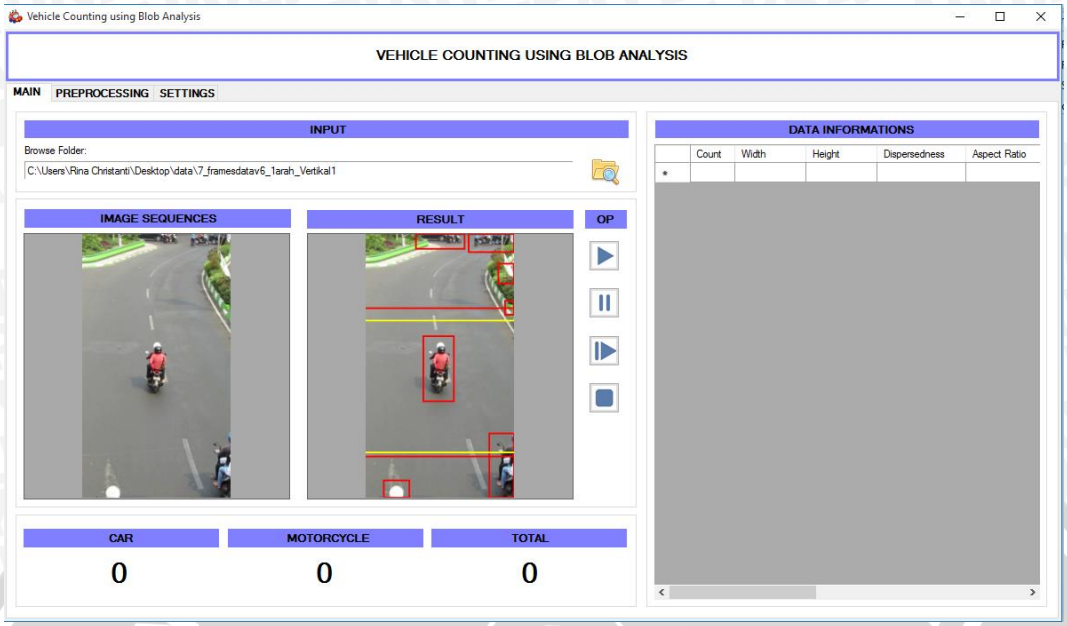


2. *Frame* 2


Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
INPUT
Browse Folder:
C:\Users\Rina Christanti\Desktop\data\7_framesdatav6_1arah_Vertikal1
DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
IMAGE SEQUENCES
RESULT
OP
CAR
MOTORCYCLE
TOTAL
0
0
0


Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
IMAGE SEQUENCES
BACKGROUND SUBTRACTION
BINARY IMAGE
OP
TOTAL
0
DATA INFORMATIONS
Objek 1
rho1 = 9.9100072471183
dist1 = 23.9269722280108
rho min = 9.9100072471183
dist min = 23.9269722280108
MORPHOLOGICAL OPERATION
TRACKING
RESULT

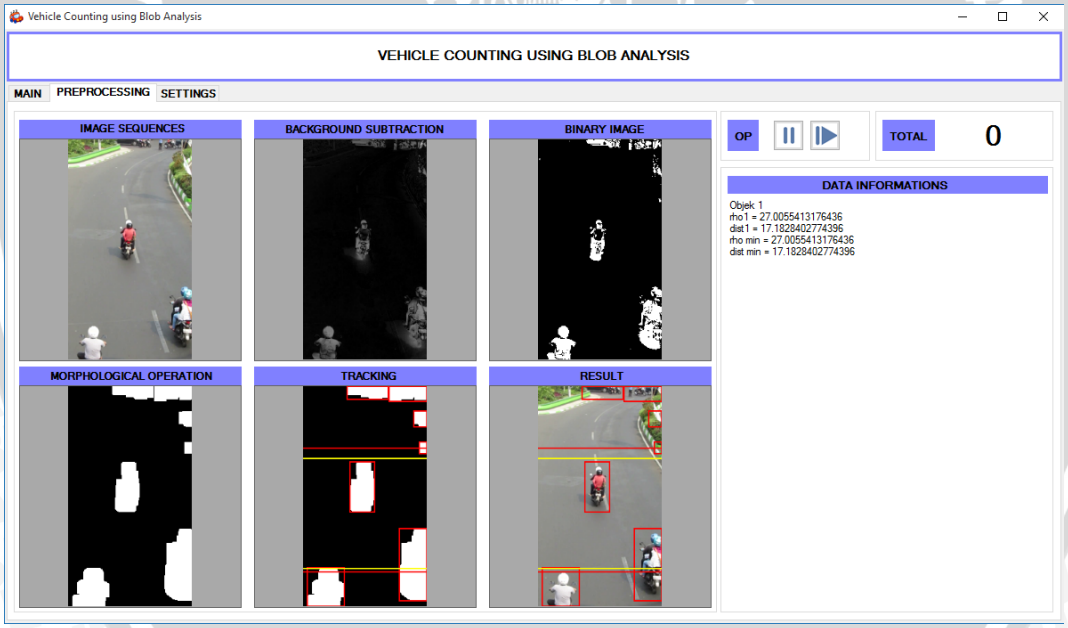
3. *Frame* 3


Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
INPUT
Browse Folder:
C:\Users\Rina Christanti\Desktop\data\7_framesdatav6_1arah_Vertikal1
DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
IMAGE SEQUENCES
RESULT
OP
CAR
MOTORCYCLE
TOTAL
0
0
0

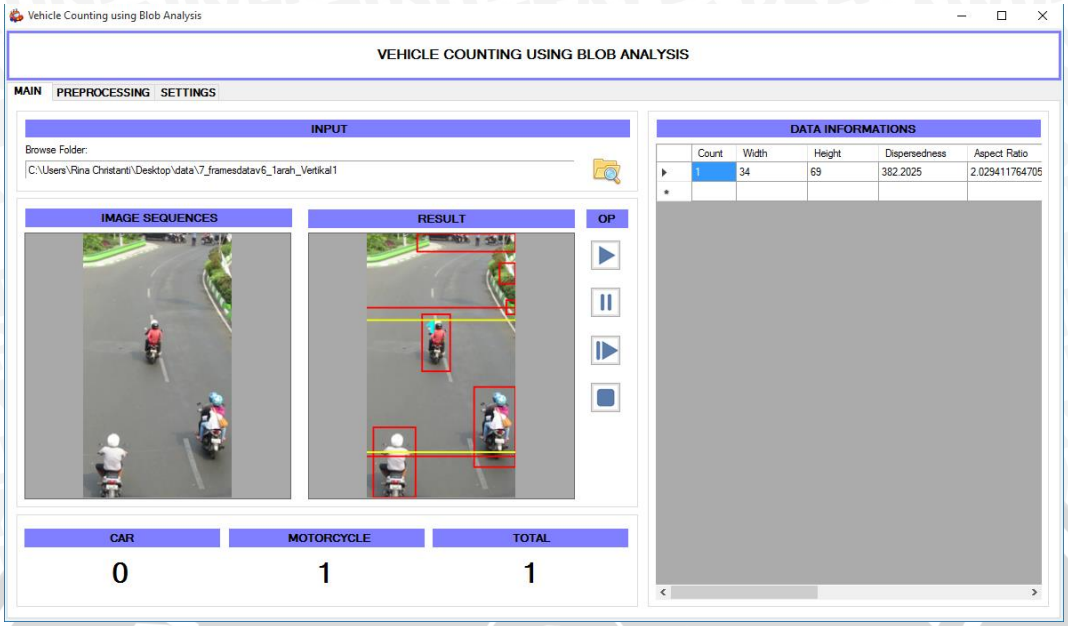

Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
IMAGE SEQUENCES
BACKGROUND SUBTRACTION
BINARY IMAGE
OP
TOTAL
0
DATA INFORMATIONS
Objek: 1
rho1 = 1.77935315936727
dist1 = 19.2353840616713
rho min = 1.77935315936727
dist min = 19.2353840616713
MORPHOLOGICAL OPERATION
TRACKING
RESULT

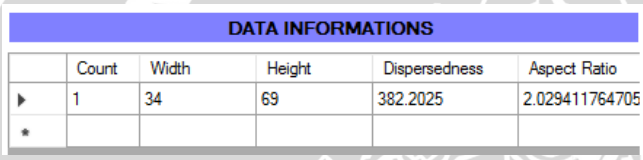
4. *Frame* 4

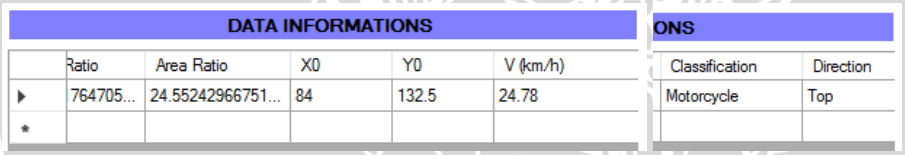

Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
INPUT
Browse Folder:
C:\Users\Rina Christanti\Desktop\data\7_framesdatav6_1arah_Vertikal1
DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
IMAGE SEQUENCES
RESULT
OP
CAR
MOTORCYCLE
TOTAL
0
0
0


Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
IMAGE SEQUENCES
BACKGROUND SUBTRACTION
BINARY IMAGE
OP
TOTAL
0
DATA INFORMATIONS
Objek: 1
rho1 = 27.0055413176436
dist1 = 17.1828402774396
rho min = 27.0055413176436
dist min = 17.1828402774396
MORPHOLOGICAL OPERATION
TRACKING
RESULT

5. *Frame* 5


Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
INPUT
Browse Folder:
C:\Users\Rina Christanti\Desktop\data\7_framesdatav6_1arah_Vertikal1
DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
1
34
69
382.2025
2.029411764705
IMAGE SEQUENCES
RESULT
OP
CAR
MOTORCYCLE
TOTAL
0
1
1

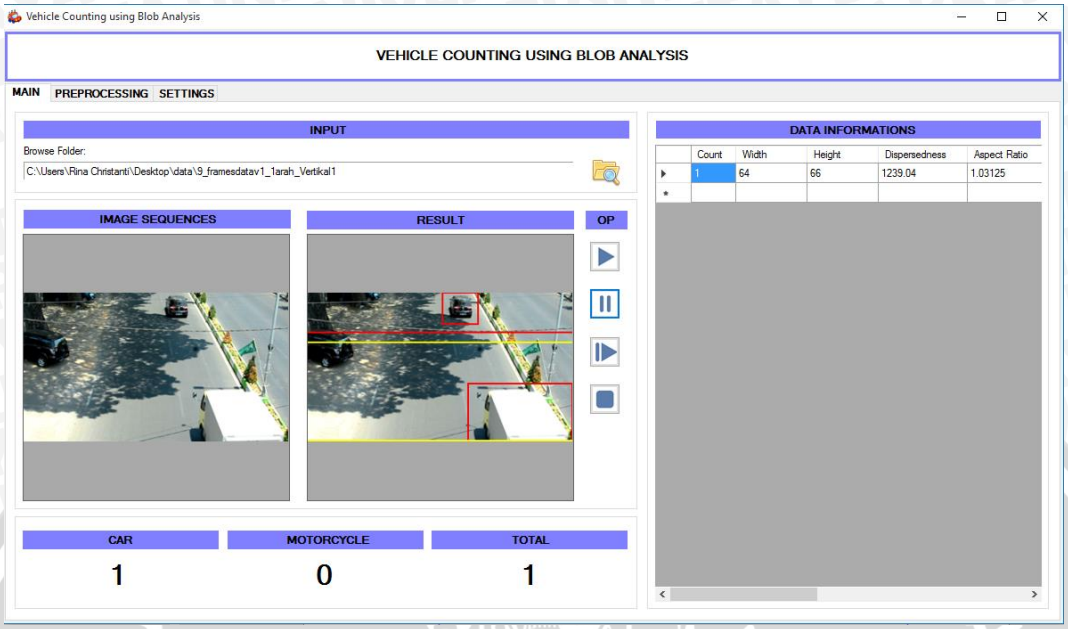

DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
1
34
69
382.2025
2.029411764705

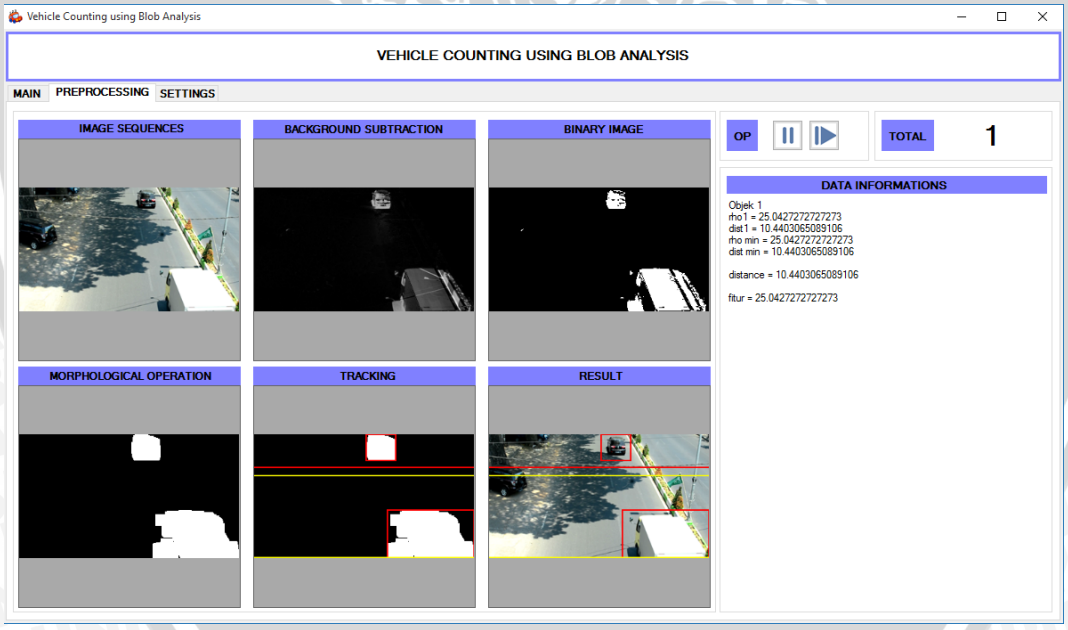

DATA INFORMATIONS
Ratio
Area Ratio
X0
Y0
V (km/h)
764705...
24.55242966751...
84
132.5
24.78
ONS
Classification
Direction
Motorcycle
Top


Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
IMAGE SEQUENCES
BACKGROUND SUBTRACTION
BINARY IMAGE
OP
TOTAL
1
DATA INFORMATIONS
Objek: 1
rho1 = 0.419333151518439
dist1 = 14.142135623731
rho min = 0.419333151518439
dist min = 14.142135623731
MORPHOLOGICAL OPERATION
TRACKING
RESULT

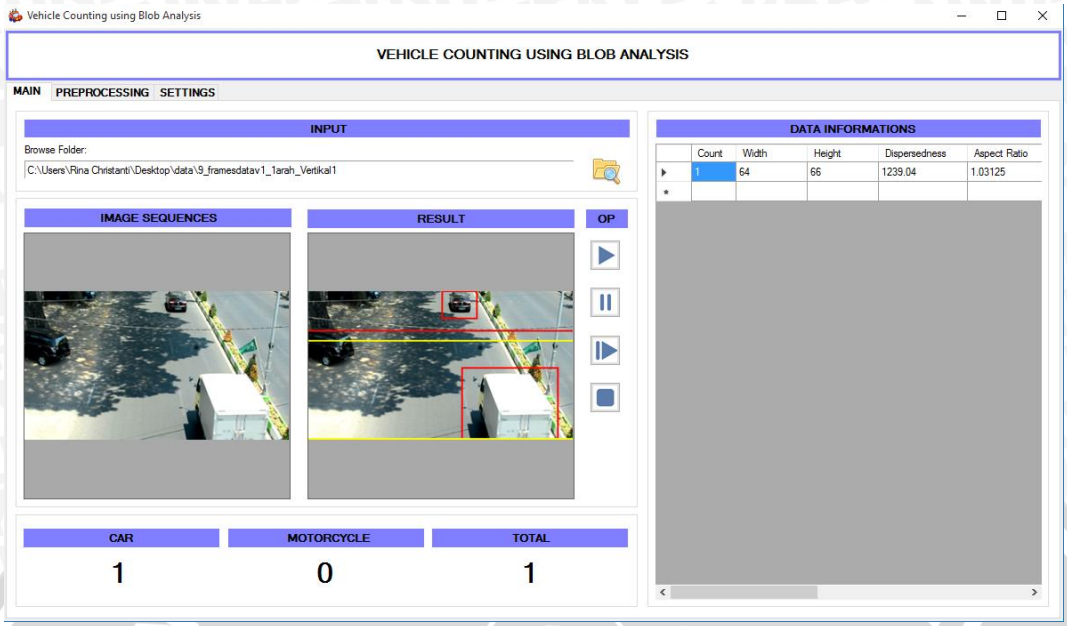
A.4 Gambar *Step By Step* Video 4 (5 *Frame*)

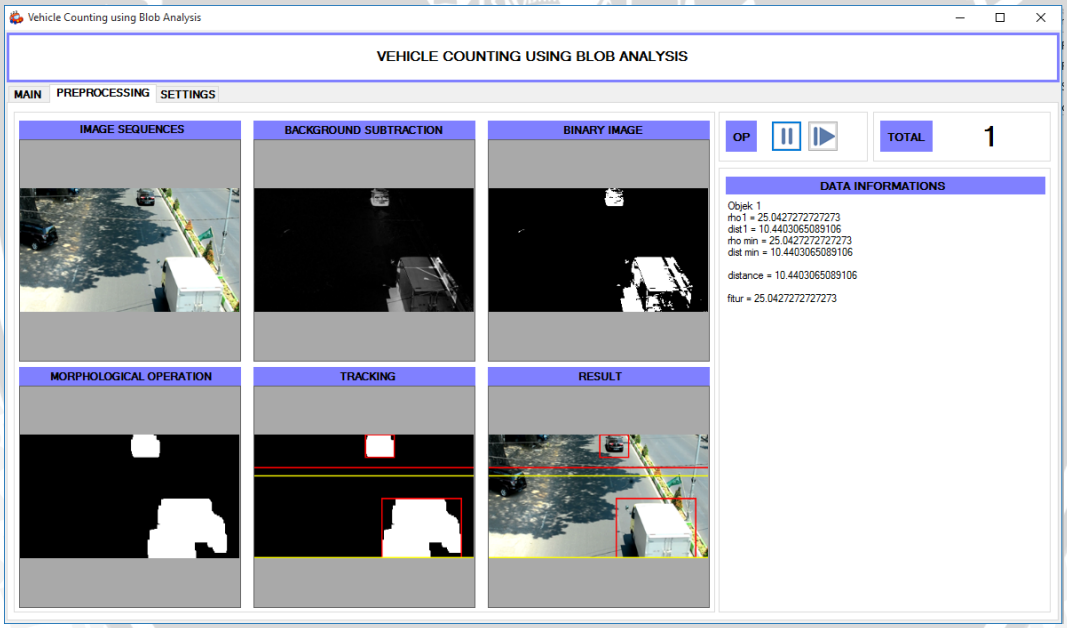
1. *Frame* 1



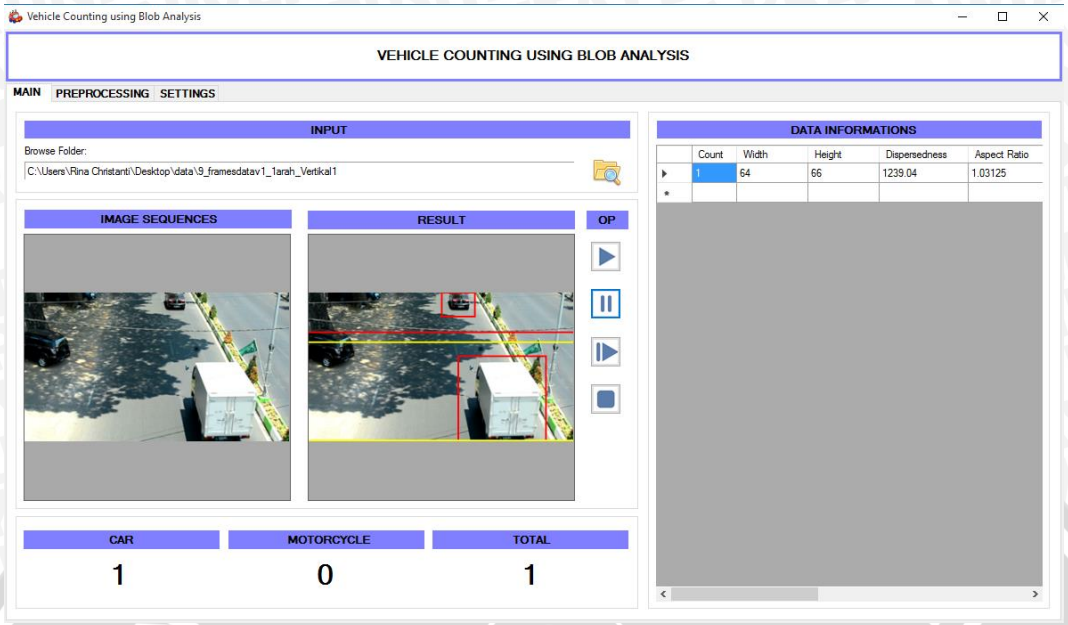


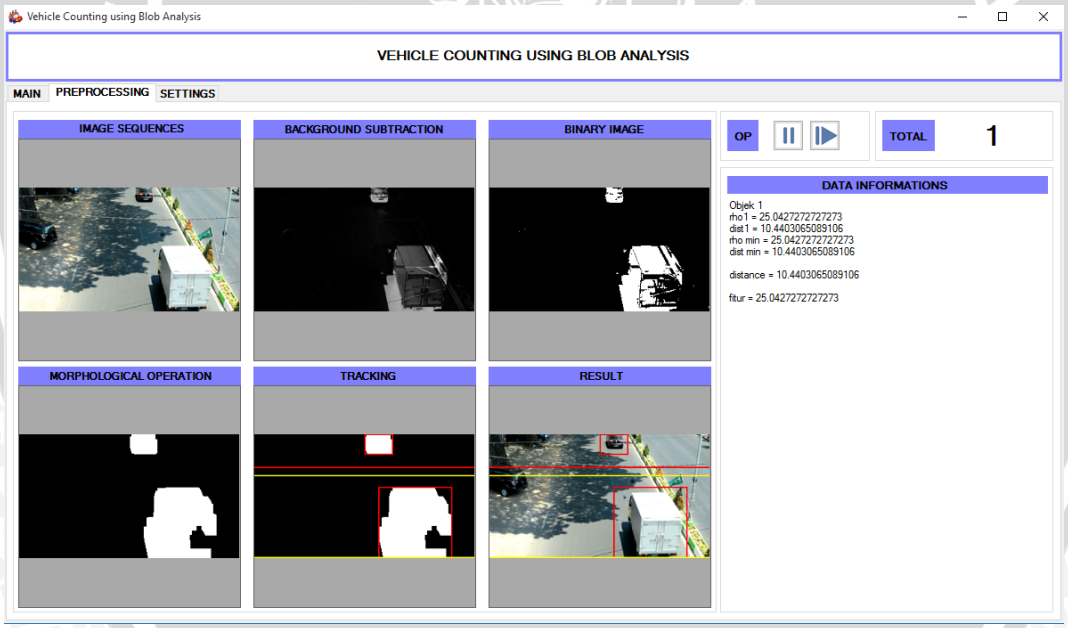
2. *Frame* 2


Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
INPUT
Browse Folder:
C:\Users\Rina Christanti\Desktop\data\9_framesdatav1_1arah_Vertikal1
DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
1
64
66
1239.04
1.03125
IMAGE SEQUENCES
RESULT
OP
CAR
MOTORCYCLE
TOTAL
1
0
1

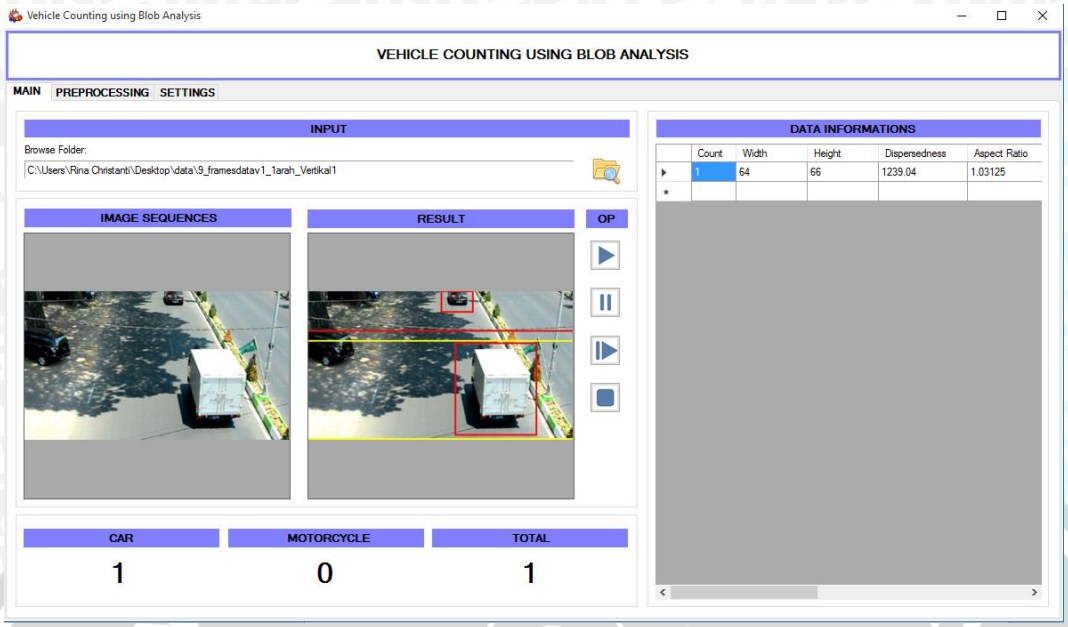

Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
IMAGE SEQUENCES
BACKGROUND SUBTRACTION
BINARY IMAGE
OP
TOTAL
1
DATA INFORMATIONS
Objek 1
rho1 = 25.0427272727273
dist1 = 10.4403065089106
rho min = 25.0427272727273
dist min = 10.4403065089106
distance = 10.4403065089106
fitur = 25.0427272727273
MORPHOLOGICAL OPERATION
TRACKING
RESULT

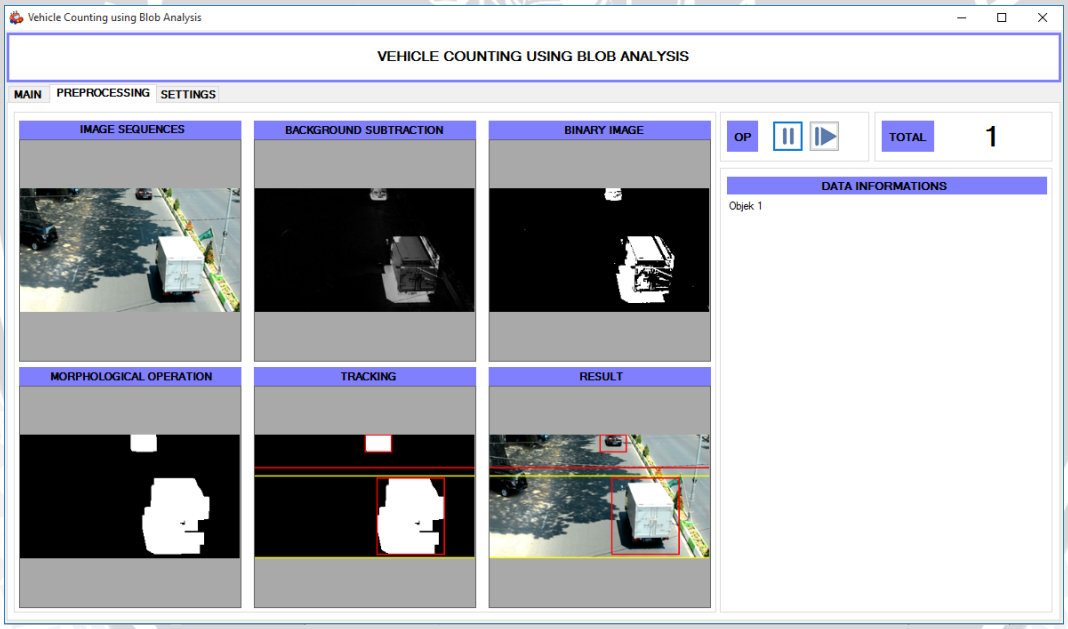
3. *Frame* 3


Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
INPUT
Browse Folder:
C:\Users\Rina Christanti\Desktop\data\9_framesdatav1_1arah_Vertikal1
DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
1
64
66
1239.04
1.03125
IMAGE SEQUENCES
RESULT
OP
CAR
MOTORCYCLE
TOTAL
1
0
1

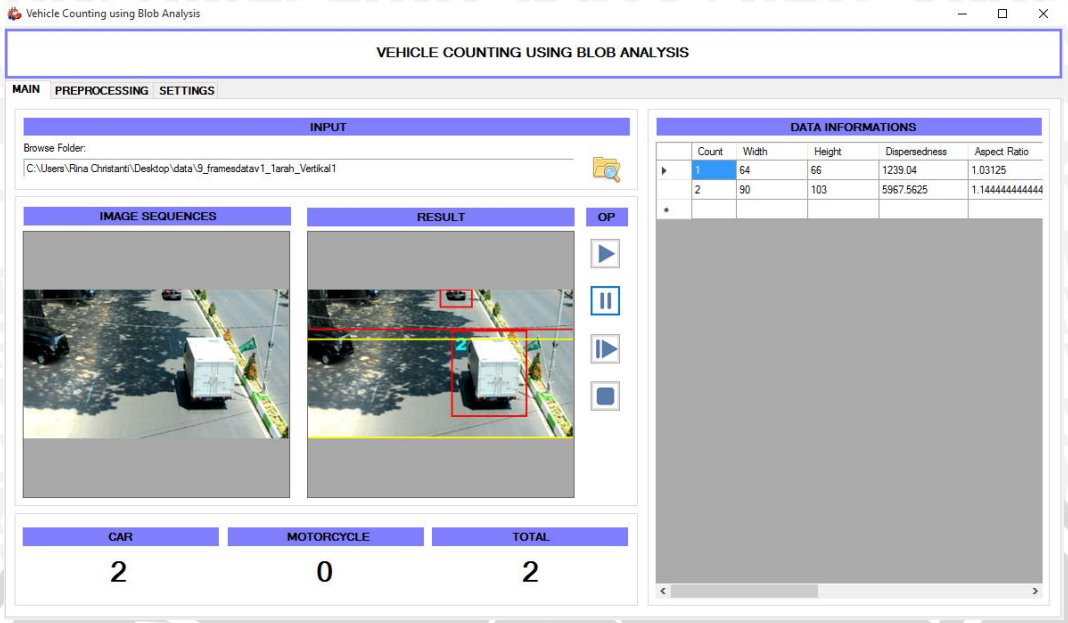

Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
IMAGE SEQUENCES
BACKGROUND SUBTRACTION
BINARY IMAGE
OP
TOTAL
1
DATA INFORMATIONS
Objek 1
rho1 = 25.0427272727273
dist1 = 10.4403065089106
rho min = 25.0427272727273
dist min = 10.4403065089106
distance = 10.4403065089106
fitur = 25.0427272727273
MORPHOLOGICAL OPERATION
TRACKING
RESULT

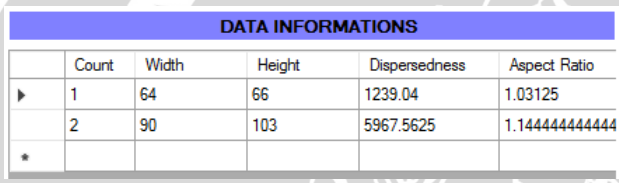
4. *Frame* 4

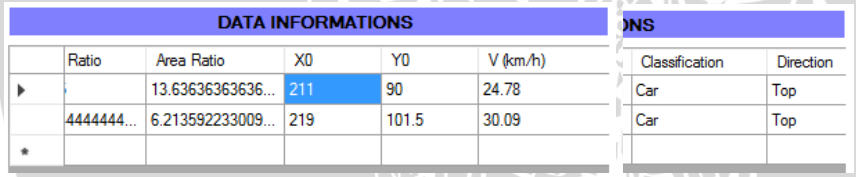

Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
INPUT
Browse Folder:
C:\Users\Rina Christanti\Desktop\data\9_framesdatav1_1arah_Vertikal1
DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
1
64
66
1239.04
1.03125
IMAGE SEQUENCES
RESULT
OP
CAR
MOTORCYCLE
TOTAL
1
0
1

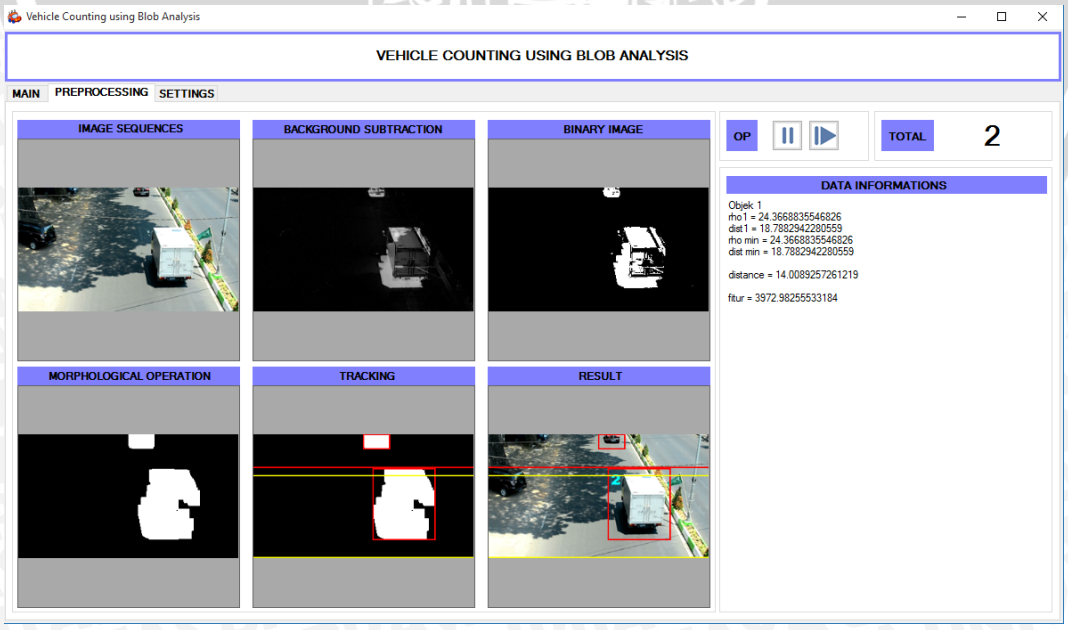

Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
IMAGE SEQUENCES
BACKGROUND SUBTRACTION
BINARY IMAGE
OP
TOTAL
1
DATA INFORMATIONS
Objek 1
MORPHOLOGICAL OPERATION
TRACKING
RESULT

5. *Frame* 5


Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
INPUT
Browse Folder:
C:\Users\Rina Christanti\Desktop\data\9_framesdatav1_1arah_Vertikal1
IMAGE SEQUENCES
RESULT
OP
DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
1
64
66
1239.04
1.03125
2
90
103
5967.5625
1.144444444444
CAR
MOTORCYCLE
TOTAL
2
0
2

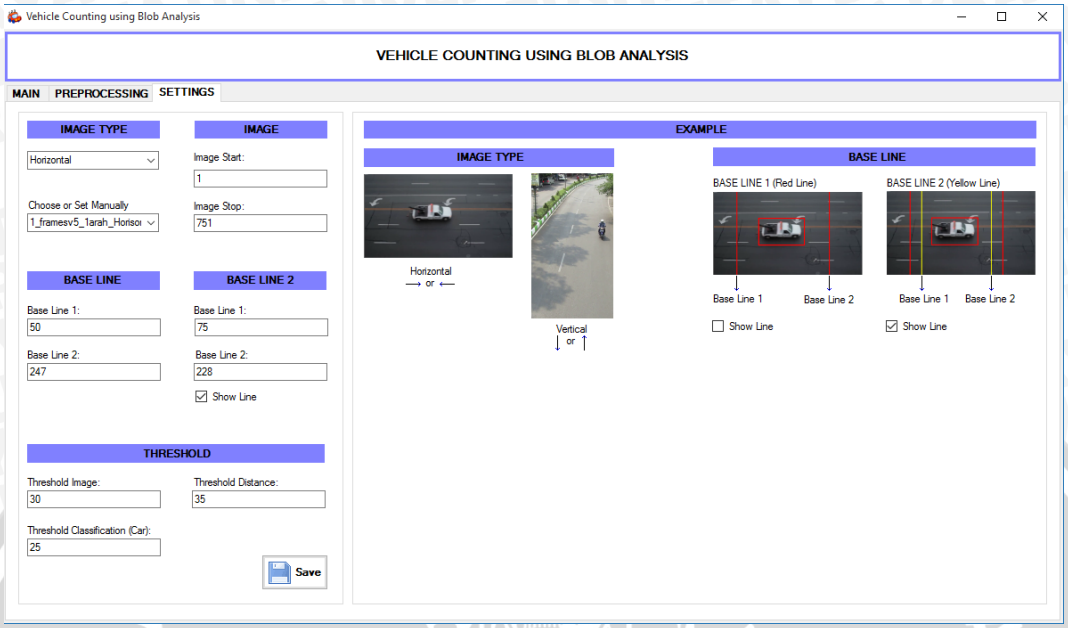

DATA INFORMATIONS
Count
Width
Height
Dispersedness
Aspect Ratio
1
64
66
1239.04
1.03125
2
90
103
5967.5625
1.144444444444


DATA INFORMATIONS
Ratio
Area Ratio
X0
Y0
V (km/h)
13.63636363636...
211
90
24.78
4444444...
6.213592233009...
219
101.5
30.09
ONS
Classification
Direction
Car
Top
Car
Top

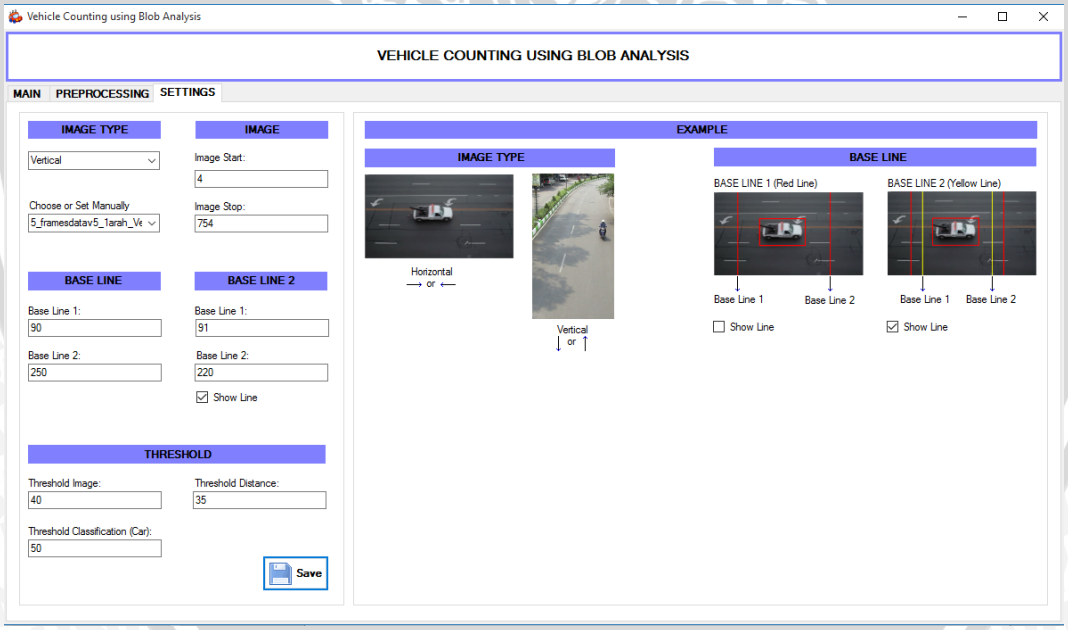

Vehicle Counting using Blob Analysis
VEHICLE COUNTING USING BLOB ANALYSIS
MAIN
PREPROCESSING
SETTINGS
IMAGE SEQUENCES
BACKGROUND SUBTRACTION
BINARY IMAGE
OP
TOTAL
2
DATA INFORMATIONS
Objek 1
rho1 = 24.3668835546826
dist 1 = 18.7882942280559
rho min = 24.3668835546826
dist min = 18.7882942280559
distance = 14.0089257261219
fitur = 3972.98255533184
MORPHOLOGICAL OPERATION
TRACKING
RESULT

A.5 Gambar Halaman *Settings*

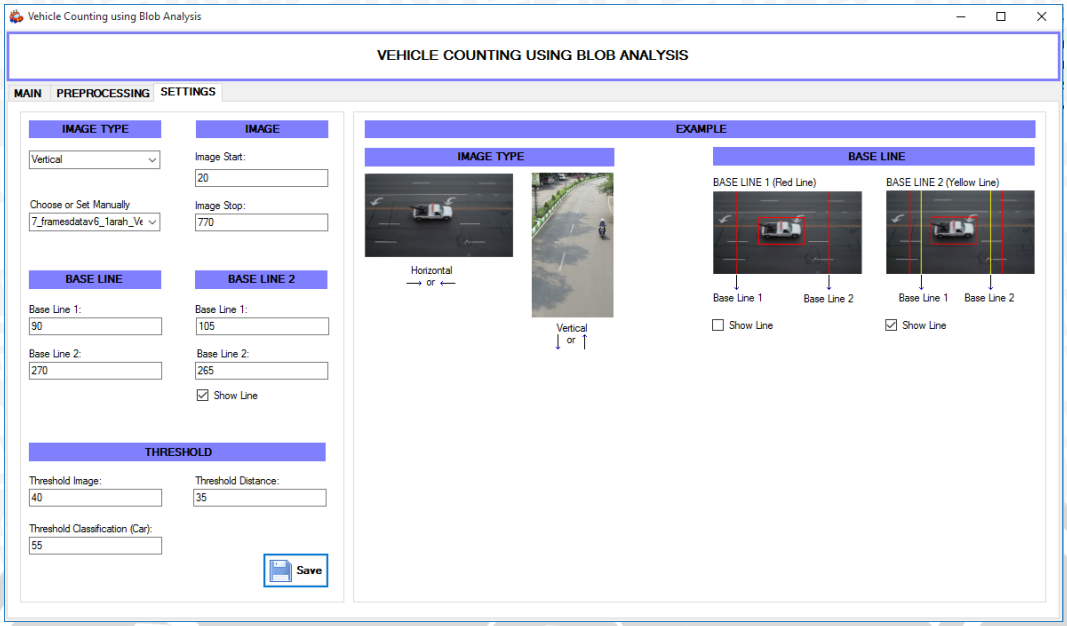
1. Video 1



2. Video 2



3. Video 3



4. Video 4