ABSTRAK

Nama Mahasiswa	:	Andi Muhammad Ali Mahdi Akbar
Judul Tugas Akhir	:	*Traffic IP Camera* untuk Menghitung Kendaraan Roda Empat Menggunakan Metode Luasan Piksel
Pembimbing	:	1. Arief Kurniawan, ST., MT. 2. Ahmad Zaini, ST., M.Sc.

Kemacetan yang terjadi pada persimpangan jalan raya sering diakibatkan oleh sistem pewaktuan yang tidak efisien. Sistem pewaktuan yang tidak efisien ini dapat diatasi dengan menerapkan *Smart Traffic Light* pada persimpangan jalan raya. *Smart Traffic Light* merupakan suatu sistem terintegrasi yang membutuhkan suatu kondisi kepadatan jalan sebagai sumber masukan. Kondisi kepadatan jalan bisa didapatkan menggunakan beberapa cara salah satunya dengan menggunakan sensor kamera. *IP Camera* dan sistem penghitungan kendaraan berbasis *Single Board Computer* diintegrasikan menjadi *Traffic IP Camera* sehingga dapat menghitung kendaraan yang melintas secara atau langsung. Metode yang digunakan untuk mendeteksi kendaraan adalah metode luasan piksel. Berdasarkan hasil pengujian secara langsung, menunjukkan bahwa sistem *Traffic IP Camera* dapat menghasilkan perhitungan kendaraan roda empat dengan tingkat akurasi hingga 82.18% pada waktu siang hari dan tingkat akurasi hingga 88.30% pada waktu malam hari.

Kata Kunci: *Smart Traffic Light*, *Traffic IP Camera*, *IP Camera*, *Single Board Computer*, Metode Luasan Piksel.

Halaman ini sengaja dikosongkan

ABSTRACT

Nama Mahasiswa	:	Andi Muhammad Ali Mahdi Akbar
Judul Tugas Akhir	:	*Traffic IP Camera for Counting Four-Wheeled Vehicle Using Area of Pixels Method*
Pembimbing	:	1. Arief Kurniawan, ST., MT. 2. Ahmad Zaini, ST., M.Sc.

Jam that occurs at the intersection of a highway often caused by inefficient timing system. This inefficient timing system can be overcame by applying Smart Traffic Light system at the intersection of a highway. Smart Traffic Light is an integrated system which requires a road density conditions as the input source. Road density conditions can be obtained by using several ways, one of them using camera sensor. IP Camera and vehicle counter using Single Board Computer are integrated into Traffic IP Camera therefore it can be used for counting vehicle directly. The method used for detecting vehicle is area of pixels. Based on the test results directly, show that the Traffic IP Camera system can produce four-wheeled vehicles detection with accuracy up to 82.18 % on daytime and accuracy up to 88.30 % at nighttime.

Keyword: Smart Traffic Light, Traffic IP Camera, IP Camera, Single Board Computer, Area of Pixels Method.

Halaman ini sengaja dikosongkan

BAB 2
TINJAUAN PUSTAKA

Pada Bab 2 dibahas mengenai konsep-konsep serta teori-teori dasar yang menunjang proses penelitian.

2.1 Kemacetan

Kemacetan adalah terhentinya proses lalu lintas yang sering kali diakibatkan oleh jumlah kendaraan yang melebihi batas kemampuan jalan. Kemacetan sering terjadi pada kota-kota besar yang transportasi umumnya kurang cukup baik atau memadai serta tidak seimbangnya kebutuhan jalan dengan jumlah penduduk pada kota tersebut. Terdapat 7 penyebab kemacetan, yaitu physical bottlenecks, kecelakaan lalu lintas, area pekerjaan, cuaca buruk, alat pengatur lalu lintas yang kurang memadai, acara khusus, dan fluktuasi pada arus normal.

1. Arus yang melewati bagan jalan melampaui kapasitas jalan yang seharusnya.
2. Adanya perbaikan jalan.
3. Terjadinya bencana alam seperti tanah longsor.
4. Kecelakaan lalu lintas.
5. Pelanggaran lalu lintas pada persimpangan.
6. Adanya kebakaran.
7. Terjadi banjir sehingga kecepatan kendaraan melamban.
8. dan lain sebagainya.

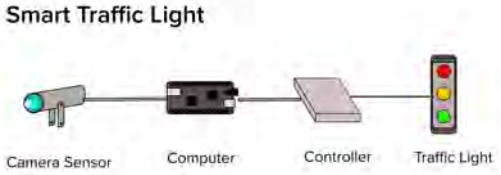
Kemacetan memiliki dampak buruk terutama pada pengguna jalan seperti kerugian waktu, pemborosan energi (bahan bakar), meningkatkan polusi udara, meningkatkan stress pengguna jalan, dan bahkan menghambat kendaraan-kendaraan penting seperti ambulans dan pemadam kebakaran.

Kemacetan sebenarnya dapat diminimalisir membiasakan menggunakan transportasi umum sehingga dapat mengurangi kemacetan yang diakibatkan kendaraan pribadi. Kemacetan juga dapat direduksi dengan cara memberikan sanksi keras pada pengguna jalan yang melanggar lalu lintas, pembuatan jalur alternatif pada kota-kota besar, pengoptimalan sistem lampu lalu lintas dan lain seba-

gainya.

2.2 *Smart Traffic Lights*

Smart Traffic Light dikenal juga dengan *Intelligent Traffic Lights* atau *Advanced Traffic Lights* adalah sistem lampu lalu lintas terintegrasi yang berfungsi untuk mengatur alur lalu lintas berdasarkan informasi yang diterima sehingga dapat beradaptasi terhadap kondisi jalan pada saat itu. *Smart Traffic Light* ditujukan untuk mengatasi kemacetan pada perempatan jalan karena waktu berhenti yang tidak efisien [5]. Sistem ini masih dalam tahap pengembangan namun sudah banyak diterapkan pada negara-negara maju ataupun berkembang. Sistem ini diharapkan dapat menghasilkan pewaktuan sinyal (*signal timing)* lampu lalu lintas yang lebih efisien sehingga tidak adanya antrian kendaraan yang panjang pada suatu persimpangan jalan yang dapat menimbulkan kemacetan.



Gambar 2.1: Konsep *smart traffic lights*

Salah satu contoh simpel permodelan *Smart Traffic Lights* seperti pada gambar 2.1. Sensor Kamera mendeteksi kendaraan yang melintas kemudian mengirimkan data tersebut menuju sebuah komputer server yang berfungsi sebagai pusat sistem untuk menentukan *signal timing* terbaik berdasarkan data yang diterima dari Sensor Kamera. Setelah itu komputer server mengubah *signal timing* pada *controller* agara *signal timing* sesuai dengan kepadatan jalan pada saat itu.

2.3 *IP Camera*

Internet Protocol Camera atau *IP Camera* adalah sebuah tipe video kamera digital yang seringkali digunakan untuk pengawas-

an, yang mana tidak seperti sebuah kamera CCTV (*Close Circuit Television*), *IP Camera* dapat mengirim dan menerima data melalui sebuah jaringan komputer dan *internet*. Walaupun kebanyakan kamera yang bisa melakukan hal ini adalah *webcams*, istilah "*IP Camera*" atau "*netcam*" biasanya diberikan pada kamera yang digunakan untuk pengawasan. Ada dua jenis *IP Camera*:

1. *Centralized IP Camera*, yang mana membutuhkan sebuah NVR (*network video recorder*) terpusat untuk menangani manajemen perekaman, video, dan alarm.

2. *Decentralized IP Camera*, yang mana tidak membutuhkan sebuah NVR terpusat, dikarenakan kamera mempunyai fungsi perekaman dan bisa merekam langsung menuju berbagai standar media penyimpanan, seperti *SD Cards*, NAS *Network-Attached Storage*, atau sebuah PC/Server.

2.3.1 *Traffic IP Camera*

Traffic IP Camera merupakan sebuah *IP Camera* yang dilengkapi kemampuan *video analytic*. Kamera ini biasanya digunakan untuk pengawasan pada jalan-jalan yang membutuhkan pengawasan yang khusus, seperti jalan bebas hambatan, persimpangan jalan raya, parkiran dan lain-lain. *Video analytic* pada *Traffic IP Camera* ini tujukan untuk melakukan proses analisa terhadap kondisi jalan. Beberapa contoh kemampuan *Traffic IP Camera* adalah seperti *Line Crossing* [2.2 (a)], *Path Tracking* [2.2 (b)], dan mendeteksi obyek asing [2.2 (c)].

(a) *Line Crossing* (b) *Unexpected Object* (c) *Path Tracking*

Gambar 2.2: Contoh *Video Analytic* pada *Traffic IP Camera* [1]

2.4 Segmentasi Citra Digital

Segmentasi citra digital merupakan bagian dari proses pengolahan citra digital. Proses segmentasi citra digital ini lebih banyak merupakan suatu proses pra-pengolahan pada sistem pengenalan objek dalam citra. Segmentasi citra (*image segmentation*) mempunyai arti membagi suatu citra menjadi wilayah-wilayah yang homogen berdasarkan kriteria keserupaan yang tertentu antara tingkat keabuan suatu piksel dengan tingkat keabuan piksel-piksel tetangganya, kemudian hasil dari proses segmentasi ini digunakan untuk proses tingkat tinggi lebih lanjut yang dapat dilakukan terhadap suatu citra, misalnya proses klasifikasi citra dan proses identifikasi objek.

2.4.1 *Thresholding* Citra digital

Pada pemprosessan citra digital, *Thresholding* adalah suatu teknik yang terkenal untuk segmentasi citra digital. Metode *Threshold* yang paling simpel adalah dengan mengganti setiap piksel dalam sebuah citra dengan piksel hital jika nilai intesitas citra kurang dari nilai konstan T, atau piksel putih jika intensitas citra lebih besar daripada nilai konstan T. Contoh citra hasil *thresholding* bisa dilihat pada Gambar 2.3.

Gambar 2.3: Contoh citra biner hasil thresholding

Otsu Threshold

Metode *otsu threshold* adalah membagi histogram citra *gray level* kedalam dua daerah yang berbeda secara otomatis tanpa membutuhkan bantuan *user* untuk memasukkan nilai ambang. Pende-

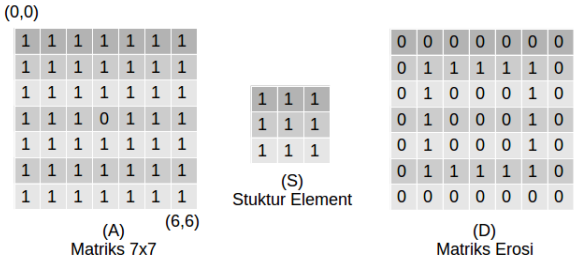
katan yang dilakukan oleh metode otsu adalah dengan melakukan analisis diskriminan yaitu menentukan suatu variabel yang dapat membedakan antara dua atau lebih kelompok yang muncul secara alami. Analisis diskriminan memaksimumkan *variable* tersebut agar dapat membagi objek latar depan (*foreground*) dan latar belakang (*background*).

2.4.2 *Erosion*

Erosion atau erosi merupakan metode *Morphological Operation* dasar pada citra digital yang berfungsi mengurangi nilai piksel dengan membandingkan nilai piksel terhadap nilai piksel tetangganya. Metode erosi sering digunakan untuk menghilangkan piksel noise pada sebuah citra digital. Erosi dimodelkan seperti pada gambar 2.4 dimana E merupakan matriks luaran dari proses erosi, A adalah matriks *input*, dan S adalah matriks *Kernel* atau *Structure Element*. Matriks *Kernel* atau *Structure Element* adalah matriks yang bertugas untuk menentukan apakah piksel poros berubah atau tidak. Ukuran dari *Structure Element* ini sendiri dapat ditentukan sesuai yang dibutuhkan dengan syarat ukuran matriks tersebut ganjil (contoh 1x1, 3x3 dll).

$$E(A,S) = A \Theta S$$

Gambar 2.4: Formula metode erosi.



Gambar 2.5: Contoh proses erosi pada matriks 7x7.

Contoh proses dan hasil erosi dapat dilihat pada gambar 2.5.

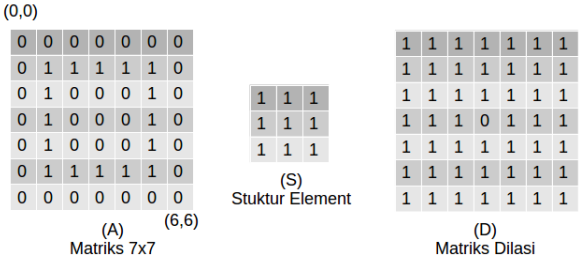
Ditentukan poros matriks S berada ditengah kemudian poros diletakkan pada posisi awal matriks A (0,0). Setelah itu dilakukan pengecekan apakah seluruh piksel matriks S yang *overlap* dengan matriks A terlingkupi oleh matriks A. Jika iya maka nilai matriks pada poros akan dipertahankan, selainnya akan dihapus. Proses ini dilakukan sebanyak ukuran matriks A.

2.4.3 *Dilation*

Dilation atau dilasi merupakan metode *Morphological Operation* dasar pada citra digital yang berfungsi menambahkan nilai piksel dengan membandingkan nilai piksel terhadap nilai piksel tetangganya. Dilasi sering digunakan untuk menutup piksel yang "bolong" pada sebuah citra digital. Dilasi dimodelkan seperti pada gambar 2.6 dimana D merupakan matriks luaran dari proses dilasi, A adalah matriks *input*, dan S adalah matriks *Kernel* atau *Structure Element*. Matriks *Kernel* atau *Structure Element* adalah matriks yang bertugas untuk menentukan apakah piksel yang disorot berubah atau tidak. Ukuran dari *Structure Element* ini sendiri dapat ditentukan sesuai yang dibutuhkan dengan syarat ukuran matriks tersebut ganjil (contoh 1x1, 3x3 dll).

$$D(A,S) = A \oplus S$$

Gambar 2.6: Formula metode dilasi.



Gambar 2.7: Contoh proses dilasi pada matriks 7x7.

Contoh proses dan hasil dilasi dapat dilihat pada gambar 2.7.

Ditentukan poros matriks S berada ditengah kemudian poros diletakkan pada posisi awal matriks A (0,0). Setelah itu dilakukan pengecekan apakah piksel matriks S yang *overlap* dengan matriks A terdapat piksel yang bernilai bukan 0. Jika iya maka nilai matriks pada poros akan ditambahkan, selainnya akan diabaikan. Proses ini dilakukan sebanyak ukuran matriks A.

2.5 *Background Reconstruction*

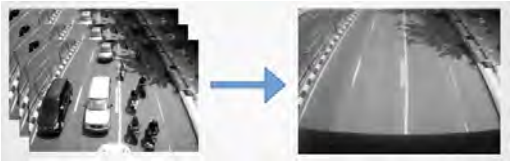
Background Reconstruction atau Rekonstruksi citra latar adalah metode pengolahan citra digital untuk merekonstruksi sebuah citra latar dari sebuah rangkaian frame atau video. Proses ini ditujukan untuk mendapatkan model citra latar yang biasanya digunakan dalam proses *Background Subtraction*. Ada banyak metode rekonstruksi citra latar dan hingga sekarang masih terus dikembangkan untuk mendapatkan model citra latar yang optimal.

Metode yang digunakan untuk melakukan proses rekonstruksi citra latar adalah dengan menggunakan bantuan fungsi dari pustaka *OpenCV*. Fungsi yang digunakan adalah fungsi *Accumulate Weighted* yang merupakan fungsi pustaka *OpenCV* untuk menghitung jumlah bobot piksel dari citra masukan dan kemudian dilakukan rata-rata berjalan (*Running average* terhadap citra tersebut. Formula *Accumulate Weighted* dapat dilihat pada gambar 3.4.

$$\texttt{dst}(x,y) \leftarrow (1-\texttt{alpha}) \cdot \texttt{dst}(x,y) + \texttt{alpha} \cdot \texttt{src}(x,y) \quad \text{if} \quad \texttt{mask}(x,y) \neq 0$$

Gambar 2.8: Rumus fungsi *accumulateWeighted*

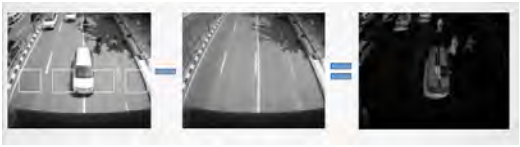
Salah satu contoh citra hasil rekonstruksi citra latar yang dihasilkan pada tugas akhir ini dapat dilihat pada gambar 2.9.



Gambar 2.9: Contoh proses rekonstruksi citra latar

2.6 *Background Subtraction*

Background Subtraction atau Subtraksi Citra Latar atau juga dikenal dengan *Foreground Detection* adalah sebuah teknik pada pengolahan citra dan visi komputer dimana untuk mengekstrak citra *foreground*. Tujuannya untuk mendapatkan objek yang berada pada citra *foreground*. Setelah tahapan pra-pemprosesan (*denoising*, *morphology* dan lain-lain) teknik ini digunakan untuk melokalisasi objek. *Background Subtraction* sering digunakan untuk pendekatan objek bergerak pada sebuah video dari kamera statis. Pendekatan yang digunakan untuk pendeteksian objek bergerak dengan membandingkan perbedaan frame sekarang dengan frame referensi, atau sering disebut citra latar atau model latar. *Background Subtraction* berperan penting pada visi komputer, seperti *Surveillance Tracking* atau estimasi pose manusia. Contoh proses *Background Subtraction* dapat dilihat pada gambar 2.10



Gambar 2.10: Contoh proses *Background Subtraction*

$$P[F(t)] = P[I(t)] - P[B]$$

Gambar 2.11: Formula simpel untuk melakukan *Background Subtraction*

Formula yang digunakan untuk menentukan *Foreground Image* dapat dilihat pada gambar 2.11. *P[F(t)]* merupakan citra hasil subtraksi antara citra masukan *P[I(t)]* terhadap citra latar *P[B]* pada waktu pada waktu *t*.

2.7 *Single Board Computer*

SBC atau *Single Board Computer* adalah sebuah komputer komplit yang dibangun pada sebuah papan circuit dengan microprosessor, memory, input/output, dan fitur lainnya yang dibutuhkan oleh sebuah komputer umumnya. SBC biasanya digunakan untuk

mendemonstrasikan sebuah sistem, untuk sistem pembelajaran atau bahkan digunakan untuk sistem kontroler komputer tertanam.

Tidak seperti komputer personal, SBC biasanya tidak mempunyai slot tambahan untuk slot *peripheral* atau slot ekspansi tujuan untuk menghemat harga sistem keseluruhan. Dengan menyimpan keseluruhan fungsi sistem pada sebuah papan, sebuah sistem yang lebih kecil bisa didapatkan sama halnya seperti pada komputer *notebook*. Konektor biasanya merupakan sumber masalah pada sebuah komputer, namun masalah ini dihilangkan pada sebuah SBC.

2.7.1 *BeagleBoard*

BeagleBoard merupakan salah satu SBC dengan daya minim dan juga termasuk perangkat yang *open-source*. Diciptakan oleh *Texas Instruments* dibantu oleh *Digi-Key* dan *Newark element14*. *BeagleBoard* juga dikembangkan dengan perangkat lunak yang bersifat *open source*.

BeagleBone Black merupakan salah satu SBC perkembangan dari *BeagleBone* dengan *Sitara ARM Cortex-A8* sebagai prosessornya, berjalan pada 1 GHz, dengan kapasitas RAM 512 MB, dan slot HDMI serta 2GB eMMC *flash memory*. Spesifikasi lebih lengkap dapat dilihat pada gambar 2.12.

Processor	AM3358BZCZ100, 1GHZ
Video Out	HDMI
DRAM	512MB DDR3L 800MHZ
Flash	4GB eMMC, uSD
OnBoard JTAG	Optional
Serial	Header
PWR Exp Header	No
Power	210-460 mA / 5V

Gambar 2.12: Spesifikasi Beaglebone Black

BeagleBone Black telah dilengkapi dengan *Linux kernel 3.8* membuat *BeagleBone Black* mampu melakukan *Direct Rendering Manager* (DRM). *BeagleBone Black Rev. C* datang dengan tambahan kapasitas *flash memory* sebesar 2GB (total 4GB) sehingga memungkinkan untuk menjalankan sistem operasi debian.

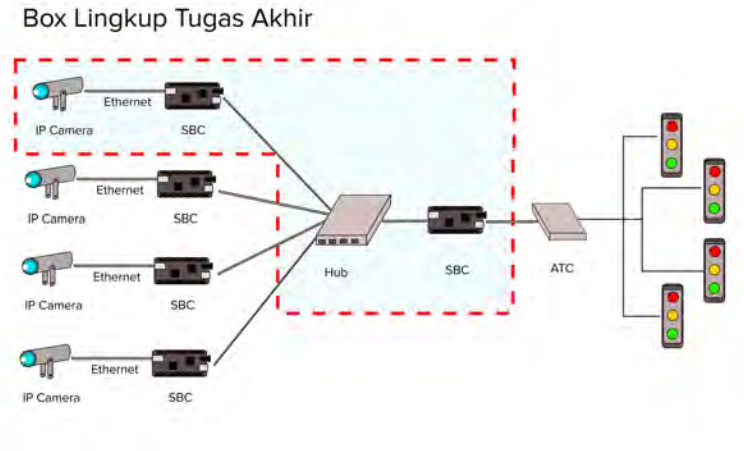
Halaman ini sengaja dikosongkan

BAB 3
DESAIN DAN IMPLEMENTASI SISTEM

Penelitian ini dilaksanakan sesuai dengan desain sistem berikut dengan implementasinya. Desain sistem merupakan konsep dari pembuatan dan perancangan infrastruktur dan kemudian diwujudkan dalam bentuk blok-blok alur yang harus dikerjakan. Pada bagian implementasi merupakan pelaksanaan teknis untuk setiap blok pada desain sistem.

3.1 Cakupan Tugas Akhir

Tugas akhir ini merupakan bagian dari sebuah sistem *Smart Traffic Light* yang merujuk pada *Intelligent Transportation System*. Desain keseluruhan sistem dapat dilihat pada Gambar 3.1, sistem yang dibahas dalam tugas akhir ini adalah bagian dalam box lingkup tugas akhir.

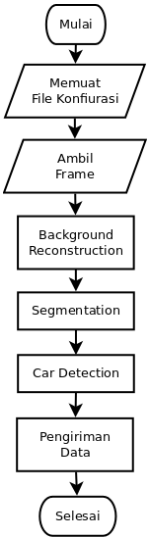


Gambar 3.1: Desain sistem keseluruhan

3.2 Desain Sistem

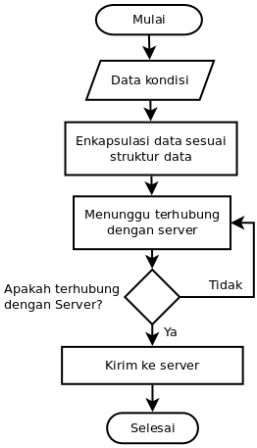
Pada tugas akhir ini, *Traffic IP Camera* dibangun pada sebuah SBC (*Single Board Computer*) dengan citra masukan dari *IP camera*. SBC bertugas sebagai media pengolahan citra untuk mendeteksi kendaraan sekaligus *client* yang mengirimkan kondisi terakhir jalan yang dipantau menuju SBC *server* yang melakukan perhitungan jumlah kendaraan.

Proses kerja sistem ini dijelaskan dalam beberapa diagram alir. Diagram alir pertama pada gambar 3.2, menjelaskan cara kerja sistem sebagai media pengolahan citra untuk mendeteksi kendaraan (*Car Counter*). Diagram alir kedua pada gambar 3.3 menjelaskan tentang cara komunikasi antar SBC (*Client to Server*).



Gambar 3.2: *Diagram alir Car Counter*

Proses pertama yang dilakukan SBC *Client* adalah memuat file konfigurasi yang diperlukan seperti *cam.conf* dan *roi.map*. *Cam.conf* merupakan file konfigurasi yang berisi parameter-parameter penting yang dibutuhkan untuk pendeteksian kendaraan. *Roi.map* merupakan file yang berisi tentang posisi serta besar ukuran ROI



Gambar 3.3: Diagram alir komunikasi antar SBC

(*Region of Interest* yang selanjutnya disebut daerah deteksi) yang digunakan. Setelah semua file konfigurasi dimuat, program selanjutnya meminta serta menerima citra digital masukan dari sensor kamera yang diolah untuk menghasilkan luaran berikutnya melalui sebuah protokol *streaming* RTSP (*Real-Time Streaming Protocol*). Citra yang didapatkan merupakan citra digital hasil tangkapan IP Kamera yang diletakkan pada ketinggian serta sudut kemiringan tertentu yang telah disesuaikan. Citra digital yang ditangkap diproses untuk mendeteksi kendaraan yang melintas dalam ROI yang telah ditentukan sebelumnya. Peletakan ROI umumnya diletakkan pada daerah lajur yang sering dilalui kendaraan.

Pada proses pendeteksian kendaraan (*Car Detection*), citra diubah terlebih dahulu menjadi citra *grayscale* kemudian citra tersebut digunakan dalam proses *Background Reconstruction* (Rekonstruksi citra latar) untuk membuat citra latar. Setelah citra latar didapatkan, dilakukan proses *Background Subtraction* (Subtraksi citra latar). Proses ini merupakan proses untuk menemukan *foreground image* dengan cara mendapatkan perbedaan absolut antara citra masukan dengan citra latar yang dibuat sebelumnya. Setelah mendapatkan *foreground image*, dilakukan proses segmentasi terha-

dap *foreground image* seperti *dilation* dan *erosion*. Setelah itu dilakukan proses pendeteksian kendaraan melalui perhitungan jumlah piksel bernilai 255 (piksel putih) pada daerah deteksi yang ditentukan sebelumnya. Sebelum itu, dilakukan penentuan nilai *Threshold* kendaraan berdasarkan persentase dari total luas daerah deteksi. Apabila jumlah piksel putih pada ROI lebih besar daripada nilai *threshold* kendaraan, dianggap ada kendaraan yang sedang melintas, dan ketika jumlah piksel frame selanjutnya lebih kecil daripada nilai *threshold* kendaraan, maka dianggap kendaraan telah melintas. Metode pendeteksian kendaraan dijelaskan lebih lanjut pada sub-bab berikutnya.

Selanjutnya ketika SBC mendeteksi adanya kendaraan yang melintas pada daerah deteksi, SBC langsung mengirimkan kondisi lajur tersebut menuju SBC *Server* menggunakan komunikasi *socket*. Data yang dikirimkan berupa *char*(karakter) yang berisi kondisi daerah deteksi pada tiap-tiap lajur pada ruas jalan yang dipantau pada saat itu. Apabila pengiriman sukses, SBC *server* melakukan pembacaan data dan melakukan perhitungan naik lajur sesuai data yang diterima. Penjelasan lebih lanjut dijelaskan pada sub-bab berikutnya.

Sistem juga melakukan Rekonstruksi citra latar berkala secara otomatis apabila waktu yang ditentukan untuk melakukan rekonstruksi telah dicapai. Sebagai contoh, apabila waktu untuk melakukan proses rekonstruksi citra latar ditentukan setiap 5 menit, maka sistem melakukan rekonstruksi citra latar setiap 5 menit sekali. Terdapat *down time* pada sistem ketika sistem melakukan rekonstruksi citra latar sehingga pemilihan waktu yang tepat nantinya sangat mempengaruhi kinerja sistem.

SBC *client* juga berfungsi sebagai *web server* yang menyediakan sebuah aplikasi web untuk melakukan konfigurasi terhadap program *Car Counter*. Pada aplikasi web tersebut terdapat berbagai fungsi untuk pengaturan seperti mengubah nilai-nilai pada file *cam.conf* yang nantinya digunakan pada program utama, pengaturan alamat jaringan pada SBC *Client*, dan juga digunakan untuk mengubah posisi serta ukuran daerah deteksi dan disimpan pada *Roi.map*. Penjelasan lebih lanjut dijelaskan pada sub bab berikutnya.

3.3 Alur kerja

Pembuatan tugas akhir ini dibagi menjadi beberapa bagian:

1. Perancangan program *Background Reconstruction*
2. Perancangan program *Car Detection*.
3. Desain struktur data.
4. Desain aplikasi web.

3.4 Perancangan program *Background Reconstruction*

Background Reconstruction atau Rekonstruksi citra latar merupakan suatu metode untuk menemukan sebuah citra latar dari suatu rekaman atau *streaming* video. Rekonstruksi citra latar ini sangat diperlukan dalam proses segmentasi terutama pada proses *Background Subtraction*. Syarat utama yang diperlukan dalam rekonstruksi citra latar adalah sebuah video yang *steady* (tidak bergerak) agar pembuatan citra latar dapat berjalan dengan baik. Formulasi fungsi *accumulateWeighted* dapat dilihat pada gambar 3.4.

$$\texttt{dst}(x,y) \leftarrow (1-\texttt{alpha}) \cdot \texttt{dst}(x,y) + \texttt{alpha} \cdot \texttt{src}(x,y) \quad \text{if} \quad \texttt{mask}(x,y) \neq 0$$

Gambar 3.4: Rumus fungsi *accumulateWeighted*

alpha pada rumus merupakan parameter yang mengatur seberapa cepat *accumulator* "melupakan" piksel sebelumnya. Semakin sering suatu objek muncul pada citra, maka objek tersebut dianggap sebagai citra latar. Hasil rekonstruksi citra latar dipengaruhi oleh jumlah citra yang digunakan. Semakin banyak citra yang dipakai, maka semakin baik hasil citra latar yang dihasilkan.

3.5 Perancangan program *Car Detection*

Program *Card Detection* dibuat dengan menggunakan bahasa C++ sebagai bahasa pemprograman. Pustaka *OpenCV* digunakan sebagai pustaka program untuk mempermudah pengaplikasian visi komputer pada SBC [6]. Pengerjaan perangkat lunak dilakukan pada *environment* Unix untuk menyesuaikan *environment* pada SBC.

Desain Perangkat lunak pada SBC *Client* dibagi menjadi beberapa tahapan:

1. Penentuan ROI dan *Threshold* Kendaraan.
2. Rekonstruksi citra latar.
3. Segmentasi kendaraan.
4. Pendeteksian kendaraan.

3.5.1 Penentuan ROI dan *Threshold* Kendaraan

Penentuan ROI atau daerah deteksi merupakan hal yang sangat penting untuk memfokuskan daerah yang ingin diamati. ROI digunakan sebagai daerah pengamatan untuk menentukan ada tidaknya kendaraan yang melintas. Daerah deteksi dibuat menggunakan fungsi pustaka *cv::rect* untuk menghasilkan sebuah daerah berbentuk *rectangle*. Parameter yang butuhkan adalah x, y, *width*, dan *height*. Proses penentuan posisi serta luas daerah deteksi selanjutnya ditangani oleh aplikasi web.

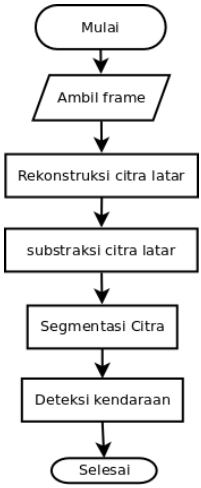
ROI digunakan sebagai acuan untuk menentukan nilai *Threshold* kendaraan berdasarkan luas daerah masing-masing ROI. Rumus untuk menentukan nilai *Threshold* kendaraan dapat dilihat pada 3.1.

$$ThresholdKendaraan = width * height * sensivitasDeteksi \quad (3.1)$$

Width & *Height* merupakan lebar & tinggi ROI, sedangkan Sensivitas deteksi merupakan paramater yang digunakan untuk menentukan seberapa sensitif nantinya program mendeteksi kendaraan. Contoh, luas ROI adalah 800 * 800 piksel, dan sensivitas deteksi yang digunakan adalah 80%, maka nilai *Threshold* yang dihasilkan adalah 800 * 800 * 80% = 512000 piksel.

3.5.2 Segmentasi kendaraan

Proses pendeteksian kendaraan dibagi menjadi beberapa tahapan proses. Proses kerja pendeteksian kendaraan dapat dilihat pada gambar 3.5.



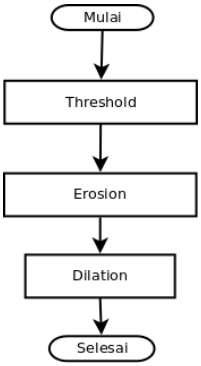
Gambar 3.5: Diagram alir proses pendeteksian kendaraan

Tahapan pertama, citra masukan diubah menjadi citra *grayscale* setelah itu dilakukan *Background Subtraction* dengan citra latar yang didapatkan dari proses *Background Reconstruction*. Kemudian dilakukan subtraksi untuk mendapatkan perbedaan absolut dari tiap piksel antar kedua citra. Hasil dari *Background Subtraction* dapat dilihat pada Gambar 3.6.

Gambar 3.6: Citra hasil *Background Subtraction*

Tahapan selanjutnya dilakukan beberapa proses *Image Segmentation* terhadap citra hasil proses *Background Subtraction* sebelum-

nya. Diagram alir *Image Segmentation* dapat dilihat pada gambar 3.7.



Gambar 3.7: Diagram alir *Image Segmentation*

Dilakukan proses *Thresholding* pada citra hasil proses *Background Subtraction.Thresholding* adalah proses mengubah nilai piksel suatu citra digital sesuai dengan nilai yang diinginkan apabila nilai piksel pada citra tersebut melebihi nilai ambang batas yang telah ditentukan. Pada proses ini digunakan fungsi *cv::threshold* dari pustaka OpenCV dan *Threshold* yang digunakan adalah *binary threshold* untuk mengubah citra menjadi citra biner. Digunakan metode *OTSU Threshold* sebagai nilai ambang batas untuk proses ini. Metode Otsu Threshold dapat menentukan secara otomatis nilai *Threshold* berdasarkan nilai intensitas histogram citra masukan. Hasil *thresholding* citra menggunakan *OTSU Threshold* dapat dilihat pada Gambar 3.8.

Proses selanjutnya adalah proses *Erosion* dan *Dilation* pada citra hasil *Thresholding*. Tujuan dari proses ini untuk menghilangkan piksel *noise* pada citra hasil *Thresholding* sehingga piksel *noise* tidak mengganggu pada proses berikutnya. Erosi atau *Erosion* merupakan teknik pengolahan citra untuk mengurangi nilai piksel tetangga atau sekitar pada suatu citra. Dilasi atau *Dilation* merupakan teknik pengolahan citra untuk menambahkan nilai piksel tetangga atau sekitar pada suatu citra. Hasil citra dari kedua teknik tersebut sangat dipengaruhi oleh ukuran *kernel* atau *Structure ele-*

Gambar 3.8: Citra hasil *thresholding*

ment yang digunakan. Contoh citra hasil erosi dapat dilihat pada Gambar 3.9 dan hasil dilasi dapat dilihat pada Gambar 3.10.

Gambar 3.9: Citra hasil *erosi*. Gambar kiri merupakan hasil erosi 1 kali. Gambar kanan adalah hasil erosi 2 kali

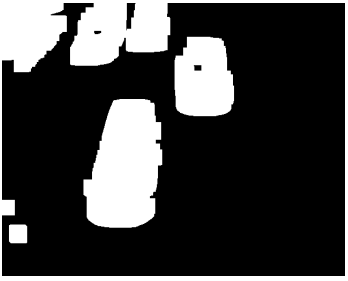
Gambar 3.10: Citra hasil *dilasi*. Gambar kiri merupakan hasil dilasi 1 kali. Gambar kanan adalah hasil dilasi 2 kali

Masing-masing proses dilakukan pada nilai *kernel* (5,5). Tam-

pak jelas pada gambar hasil erosi terjadi pengurangan nilai piksel sehingga *noise* pada citra tersebut berkurang, namun disisi lain fitur-fitur penting seperti piksel kendaraan juga ikut berkurang (Gambar 3.9). Hal ini adalah hal yang tidak diinginkan pada proses segmentasi citra. Oleh karena itu perlu dilakukan proses dilasi untuk menutup kembali "lubang" hasil proses erosi (Gambar 3.11).

Gambar 3.11: Citra hasil *erosi* serta *dilasi*

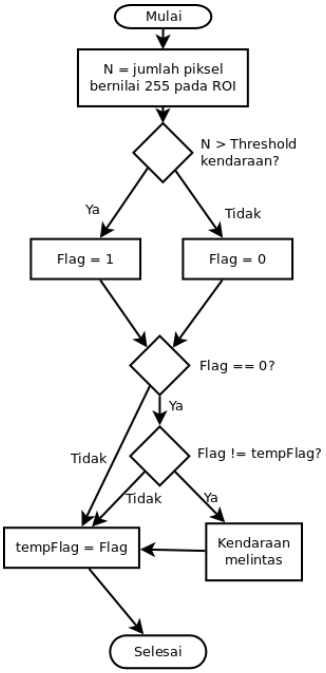
Untuk mendapatkan bentuk segmentasi yang optimal (pejal), dilakukan proses dilasi dan erosi berkali-kali hingga mendapatkan bentuk yang diinginkan. Contoh bentuk segmentasi yang diinginkan dapat dilihat pada gambar 3.12.



Gambar 3.12: Citra hasil *erosi* dan *dilasi*

3.5.3 Pendeteksian kendaraan

Pada sub bab ini menjelaskan penerapan metode luasan piksel pada proses pendeteksian kendaraan. Pada proses ini, ada tidaknya kendaraan yang melintas pada daerah deteksi ditentukan oleh jumlah luasan piksel pada daerah deteksi. Algoritma luasan piksel dapat dilihat pada gambar 3.13.



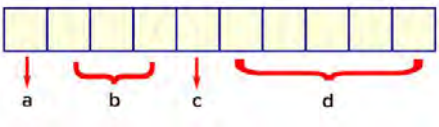
Gambar 3.13: Algoritma luasan piksel untuk mendeteksi kendaraan yang melintas pada daerah deteksi (ROI)

Citra hasil segmentasi kendaraan dari proses sebelumnya digunakan sebagai citra masukan pada proses ini. Proses pertama yang dilakukan adalah menghitung nilai N, yaitu jumlah piksel yang bernilai 255 (putih) pada daerah deteksi. Jumlah iterasi yang dilakukan tergantung pada luas daerah ROI yang digunakan, misalkan luas daerah ROI yang digunakan ada lah 800 x 800 piksel, maka dilakukan

proses iterasi sebanyak 800 x 800 iterasi. Proses selanjutnya adalah membandingkan nilai N dengan nilai *threshold* kendaraan. Apabila nilai N melebihi nilai *threshold* kendaraan, maka nilai *Flag* diberi nilai 1. *Flag* merupakan nilai untuk menandakan bahwa telah terdeteksi kendaraan pada daerah deteksi. *Temp* merupakan nilai kondisi *Flag* sebelumnya yang digunakan untuk membandingkan nilai *Flag* saat ini. Proses selanjutnya dilakukan pengecekan pada nilai *Flag* terhdap nilai *Flag* sebelumnya (*Temp*). Apabila nilai *Flag* bernilai 0, dan kondisi *Flag* tidak sama dengan nilai *Temp*, pada kondisi ini kendaraan dianggap telah melintas pada daerah deteksi.

3.6 Desain struktur data

Pengiriman data pada sistem ini menggunakan struktur data khusus. Hal ini berguna untuk mengatur sistematika data yang dikirim agar komunikasi antar *Client-Server* dapat saling mengerti satu dengan lainnya. Struktur data yang dibuat sepanjang 10 byte. Desain struktur data digambarkan pada gambar 3.14.



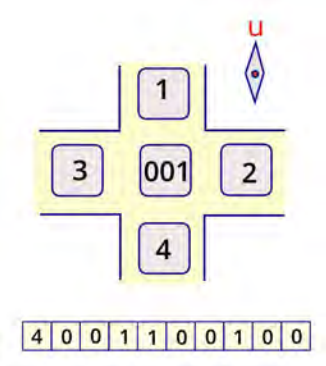
Gambar 3.14: Desain struktur data yang digunakan dalam komunikasi SBC

dimana:

a. Satu byte pertama merupakan data tipe simpangan. Tipe simpangan yang dimaksud adalah tipe persimpangan 3, 4, atau 5.
b. Tiga byte selanjutnya merupakan data mengenai ID simpangan. ID simpangan berguna untuk penyimpanan data kondisi lajur pada database SBC Server nantinya. Nilai ID simpangan dapat berkisar antara 0 hingga 999.
c. Satu byte berikutnya merupakan data mengenai ID simpangan. ID simpangan sangat penting untuk menentukan posisi kamera pada suatu persimpangan.
d. Lima byte terakhir merupakan data mengenai kondisi masing-

masing lajur pada jalur yang kamera pantau. Nilai pada data ini bersifat *bool*, yaitu antara 1 atau 0 yang menandakan ada tidaknya kendaraan.

Contoh struktur data yang dikirim dapat dilihat pada gambar 3.15. Pada gambar tersebut dapat diartikan bahwa SBC *Client* yang mengirimkan data tersebut merupakan SBC yang berada pada persimpangan bertipe "simpang empat". Lokasi SBC berada pada perempatan yang memiliki id '001' dan berada pada utara jalan raya (id kamera = '1'). Kondisi jalan pada saat itu terdeteksi ada kendaraan yang melintas pada lajur ketiga. Hal ini sangat penting untuk menentukan posisi serta orientasi SBC *Client*.



Gambar 3.15: Contoh data yang dikirimkan oleh SBC

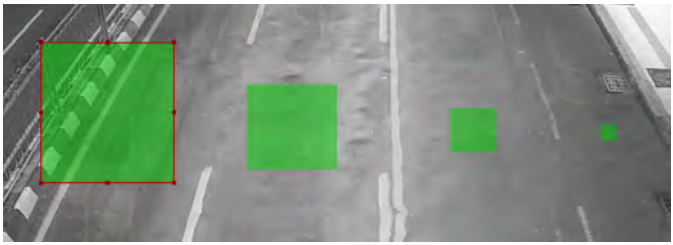
3.7 Desain aplikasi web

Desain aplikasi web dibuat dalam format *.php* dan dijalankan menggunakan *Apache Webserver*. Terdapat 3 halaman yang ditampilkan pada aplikasi web. Halaman pertama merupakan halaman untuk melakukan konfigurasi terhadap aplikasi *Car Counter*. Pada halaman ini dapat dilakukan pengaturan terhadap beberapa parameter penting yang ada pada perangkat lunak *Car Counter* antara lain seperti tipe simpangan, ID simpangan, ID kamera dan lain sebagainya. Parameter yang dapat diatur dapat dilihat pada Gambar 3.17.

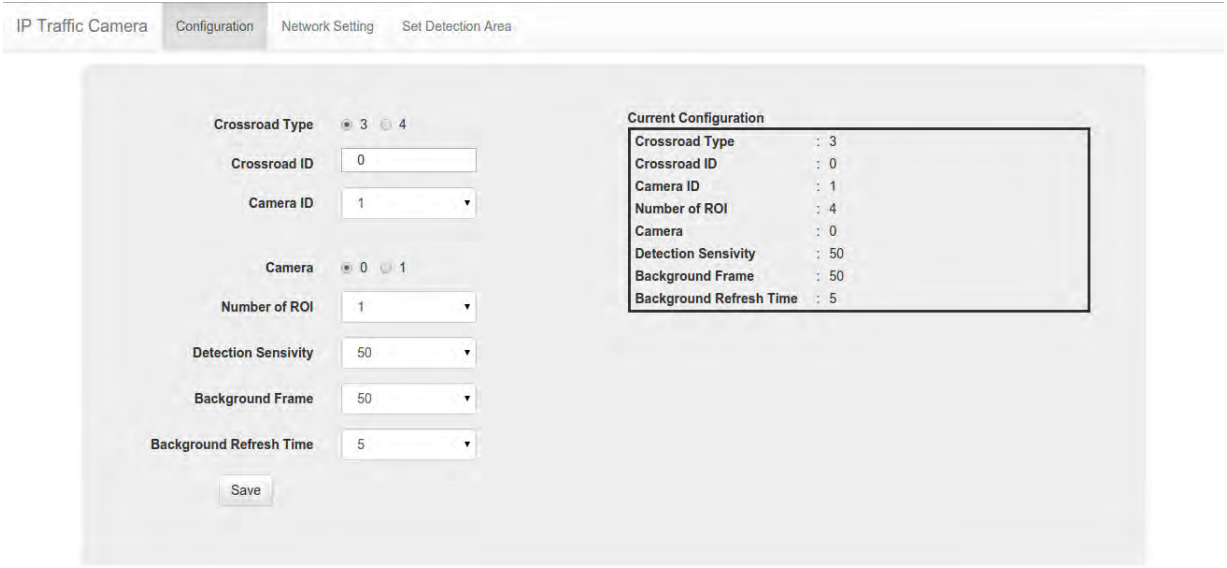
Crossroad ID, *Cross Type* dan, *Camera ID* merupakan parameter yang digunakan oleh SBC server untuk membedakan data pengirim. Dengan kata lain, ketiga parameter ini bertujuan sebagai identitas dari SBC client. *Number of ROI* adalah jumlah lajur yang ingin dideteksi. Parameter ini mempengaruhi jumlah daerah deteksi yang dibuat.*Camera* digunakan untuk memilih konfigurasi kamera yang digunakan. *Detection Sensivity* merupakan parameter yang digunakan sebagai nilai *Threshold* kendaraan. 50% berarti nilai *Threshold* kendaraan merupakan 50% dari luas daerah deteksi. *Background Frame* merupakan parameter untuk menentukan jumlah frame yang digunakan untuk proses rekonstruksi citra latar. 50 berarti jumlah citra yang digunakan untuk membuat citra latar adalah 50 frame. *Background Refresh Time* merupakan parameter untuk menentukan jeda waktu antar proses rekonstruksi citra latar. "5" artinya proses rekonstruksi citra latar dilakukan setiap 5 menit sekali. Semua konfigurasi pada halaman tersebut disimpan kedalam sebuah file *cam.conf*. File ini berisi semua konfigurasi yang tertera pada halaman pertama dan dimuat oleh perangkat lunak *Car Counter* pada saat pertama kali program dijalankan.

Halaman berikutnya adalah halaman untuk konfigurasi alamat jaringan SBC Client. Tujuannya untuk mengubah alamat jaringan SBC *Client* sesuai dengan kebutuhan pengguna. Ketika pengguna menekan tombol "Save", aplikasi web mengambil semua nilai pada masing-masing masukan kemudian mengirimkannya menuju *sudo.php*. *Sudo.php* berisi perintah-perintah *SystemCall* seperti *ifconfig* yang dijalankan melalui perintah *exec* pada php. Karena mengubah alamat jaringan membutuhkan *previlege sudo*, maka perlu ditambahkan pengguna (*user*) *www-data* pada *visudo* SBC.

Halaman yang ketiga adalah halaman untuk pengaturan daerah deteksi. Pada halaman ini terdapat citra digital yang merupakan citra latar hasil proses *Background Reconstruction* dan sejumlah *box* yang berfungsi sebagai daerah deteksi. Jumlah daerah deteksi tergantung oleh nilai parameter *Number of ROI* pada halaman konfigurasi. Posisi serta ukuran daerah deteksi dapat diatur menggunakan *pointer mouse* sehingga ukuran daerah deteksi dapat menyesuaikan kondisi jalan (gambar 3.16).

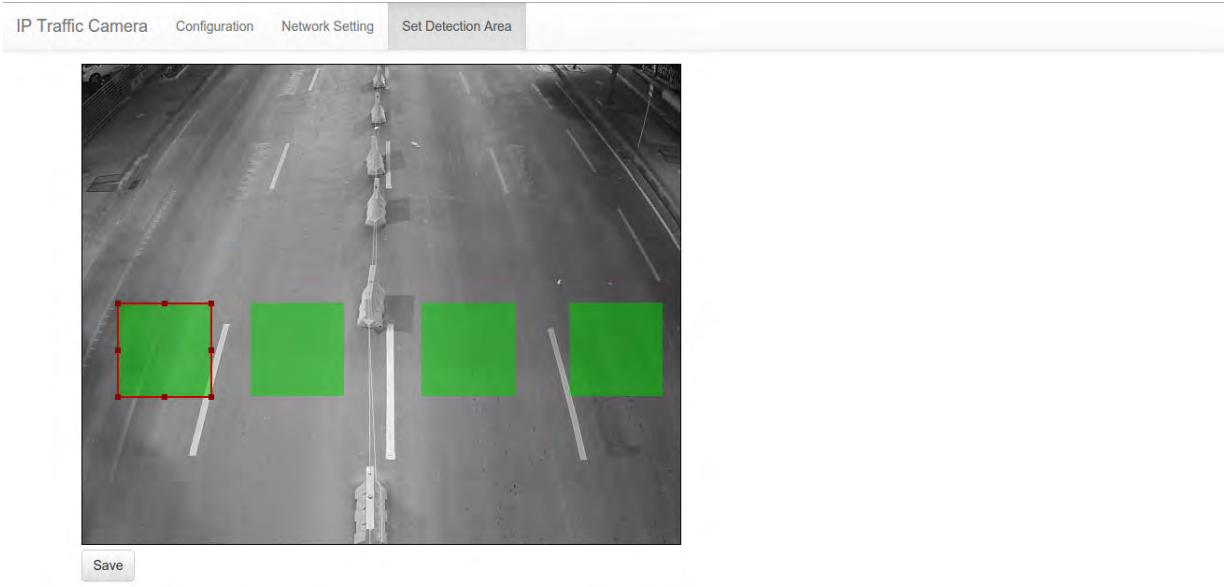


Gambar 3.16: Daerah deteksi yang dapat diubah ukuran dan posisinya



Gambar 3.17: Tampilan aplikasi web untuk konfigurasi

Gambar 3.18: Tampilan aplikasi web untuk pengaturan alamat jaringan SBC



Gambar 3.19: Tampilan aplikasi web untuk pengaturan ROI

BAB 4
PENGUJIAN DAN ANALISA

Pada bab ini dipaparkan hasil pengujian serta analisa dari desain sistem dan implementasi. Pengujian dibagi menjadi beberapa pengujian:

1. Pengujian *Background Reconstruction*.
2. Pengujian citra latar.
3. Pengujian *Delay Process*.
4. Pengujian ROI.
5. Pengujian tingkat akurasi program dengan video uji.
6. Pengujian tingkat akurasi program secara langsung.
7. Pengujian aplikasi web.

Sehingga dengan adanya pengujian tersebut, dapat ditarik beberapa kesimpulan dari pelaksanaan tugas akhir ini.

4.1 Pengujian *Background Reconstruction*

Pada bagian ini, pengujian dilakukan untuk membandingkan hasil dari proses *Background Reconstruction* terhadap jumlah *frame* yang digunakan. Pengujian dilakukan dengan mengubah jumlah *frame* yang digunakan pada proses *Background Reconstruction* kemudian menghitung waktu yang dibutuhkan untuk melakukan proses tersebut. Pengujian dilakukan secara *live* sehingga citra masukan merupakan citra yang didapat merupakan citra hasil tangkapan *IP Camera*.

Gambar 4.1: Citra hasil *Background Reconstruction* menggunakan 50 frame

Gambar 4.2: Citra hasil *Background Reconstruction* menggunakan 100 frame

Gambar 4.3: Citra hasil *Background Reconstruction* menggunakan 200 frame

Gambar 4.4: Citra hasil *Background Reconstruction* menggunakan 250 frame

Gambar 4.5: Citra hasil *Background Reconstruction* menggunakan 300 frame

Tabel 4.1: Tabel waktu yang dibutuhkan *Background Reconstruction* dalam satuan detik

	Frame				
Pengujian #	50	100	200	250	300
1	3	6	12	15	19
2	3	7	12	16	19
3	3	6	12	16	19

Dapat dilihat pada tabel 4.1, semakin banyak jumlah frame yang digunakan, maka semakin banyak waktu yang digunakan untuk melakukan proses *Background Rekonstruction*. Kenaikan jumlah waktu yang dibutuhkan dapat dikatakan konstan yaitu +-3 detik per 50 frame. Pada gambar 4.1 dapat dilihat citra latar yang dihasilkan menggunakan 50 frame sedikit gelap dibandingkan dengan citra latar yang dihasilkan menggunakan 100 frame. Ini di akibatkan karena proses pembuatan citra latar menggunakan citra *zero* sebagai *image accumulator*. Penggunaan citra *zero* sebagai *image accumlator* ini untuk mengindari terjadinya noise yang tidak diinginkan.

Dapat dilihat pada gambar 4.3, citra yang dihasilkan sudah dapat dikatakan baik, baik dari segi kecerahan maupun dari segi kebersihan citra latar dari noise yang tidak diinginkan. Contoh citra hasil *Background Reconstruction* yang tidak diinginkan dapat dilihat pada gambar 4.6. Terlihat terdapat bayangin kendaraan yang sedang berhenti dan dianggap menjadi sebuah latar pada citra

Gambar 4.6: Citra hasil *Background Reconstruction* pada saat kendaraan berhenti

latar. Ini merupakan galat yang dapat dihindari dengan cara mensinkronkan waktu pelaksanaan proses *Background Reconstruction* dengan waktu kendaraan sedang berhenti (lampu merah), sehingga proses pelaksanaan *Background Reconstruction* dapat dilakukan pada saat jalan tidak sedang berhenti (lampu merah). Penggunaan *frame* diatas 200 *frame* tidak menunujukkan perubahan yang signifikan dapat dilihat pada gambar 4.4 dan gambar 4.5.

4.2 Pengujian Citra Latar

Pada pengujian ini, dilakukan pengujian citra latar hasil proses *Background Reconstruction* pada proses *Segmentation*. Tujuan dari pengujian ini untuk mengetahui seberapa baik citra latar hasil *Background Reconstruction* pada pengujian sebelumnya untuk digunakan sebagai citra latar.

Dapat dilihat pada gambar 4.7 dan gambar 4.8, pada jendela hasil, segmentasi yang dihasilkan tidak sesuai dengan yang inginkan. Terdapat banyak sekali noise yang terjadi pada hasil segmentasi tersebut. Ini dapat diakibatkan karena citra latar yang digunakan masih belum baik atau optimal. Pada gambar 4.3, gambar 4.4, dan gambar 4.5, terlihat hasil segmentasi jauh lebih baik dibandingkan dengan hasil segmentasi menggunakan citra latar 50 *frame* dan 100 *frame*. Pada hasil segmentasi menggunakan citra latar 200 frame, hasil sudah dapat dikatakan baik dan perbedaan antar penggunaan 200 frame, 250 frame, dan 300 frame tidak terlalu signifikan.

Gambar 4.7: Citra hasil *Segmentation* menggunakan 50 frame

Gambar 4.8: Citra hasil *Segmentation* menggunakan 100 frame

Gambar 4.9: Citra hasil *Segmentation* menggunakan 200 frame

Gambar 4.10: Citra hasil *Segmentation* menggunakan 250 frame

Gambar 4.11: Citra hasil *Segmentation* menggunakan 300 frame

4.3 Pengujian *Delay process*

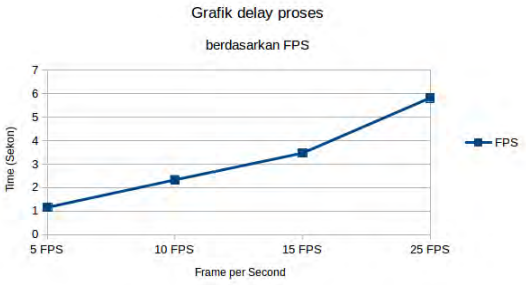
Pengujian ini ditujukan untuk menguji waktu proses (*Delay process)*yang dibutuhkan sistem untuk menjalankan proses pendeteksian dalam 1 siklus. Pengujian dilakukan dengan menghitung waktu yang dibutuhkan sistem dalam mengeksekusi semua proses utama (kecuali *Background Reconstruction*) seperti *Background Subtraction*, segmentasi, hingga pendeteksiannya, dengan mengubah jumlah ROI yang digunakan pada nilai FPS dan Resolusi yang berbeda-beda. Pengujian dilakukan masing-masing 3 kali kemudian dicari modusnya. Pencatatan waktu yang dilakukan sistem menggunakan fungsi *clock()* dari pustaka c. Dikarenakan ketebatasan kemampuan dari processor *Beaglebone*, sistem hanya mampu men-

catat waktu dengan resolusi ketelitian hingga dua digit dibelakang koma (milisekon). Hasil pengujian dapat dilihat pada tabel 4.2.

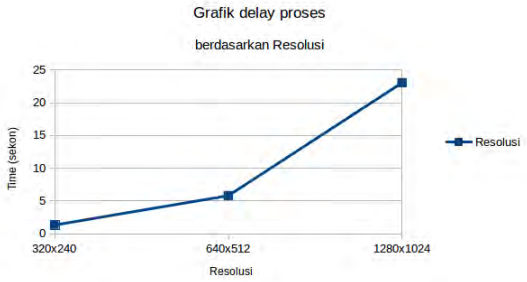
Tabel 4.2: Waktu yang dibutuhkan sistem dalam menjalankan proses. Nilai dalam satuan sekon.

Resolusi	Execution Time Per Frame	5 FPS	10 FPS	15 FPS	25 FPS
320 x 240	0,06	0,27	0,55	0,81	1,34
640 x 512	0,24	1,16	2,33	3.47	5,82
1280 x 1024	0,94	4,64	9,24	13,89	23,04

Berdasarkan hasil pengujian, perbedaan fps dan Resolusi sangat mempengaruhi *delay process* pada sistem. *Delay process* naik secara eksponensial terhadap jumlah *frame* yang digunakan. Perbedaan yang sangat tampak terlihat ketika dilakukan perubahan terhadap resolusi yang digunakan. Dengan menaikkan resolusi hingga 2 kali dari 640x512 menjadi 1280x1024, menyebabkan *Delay process* yang dihasilkan naik hingga nyaris 4 kali lipat terhadap nilai sebelumnya(0,24 sekon menjadi 0,94 sekon). Hal ini juga terjadi ketika menurunkan resolusi dari 640x512 menjadi 320x240, terjadi perubahan *delay process* menjadi hampir 1/4 kali dari nilai sebelumnya. Berdasarkan hasil pengujian, dapat dikatakan perubahan *Delay process* berubah eksponensial terhadap perubahan resolusi (grafik 4.13) dan berubah linear terhadap perubahan FPS (grafik 4.12).



Gambar 4.12: Grafik perubahan *Delay process* terhadap perubahan FPS.



Gambar 4.13: Grafik perubahan *Delay process* terhadap perubahan Resolusi.

4.4 Pengujian ROI

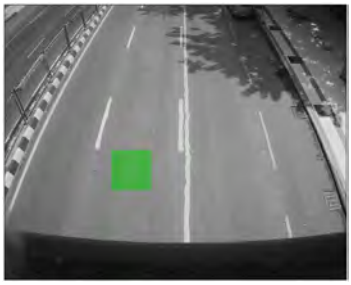
Pengujian ini untuk menguji pengaruh besar ROI terhadap hasil pendeteksian kendaraan. Pengujian dilakukan pada masukan dengan nilai *Frame Rate* 25 FPS dengan resolusi sebesar 640x512. Pengujian ini dilakukan dengan membandingkan hasil perhitungan manual dengan hasil perhitungan sistem dengan ukuran ROI yang berbeda-beda. Hasil Pengujian dapat dilihat pada tabel 4.3. Hasil perhitungan manual yang didapat adalah sebanyak 31 kendaraan roda 4.

Tabel 4.3: Tabel perbandingan hasil deteksi kendaraan dengan ukuran ROI yang berbeda

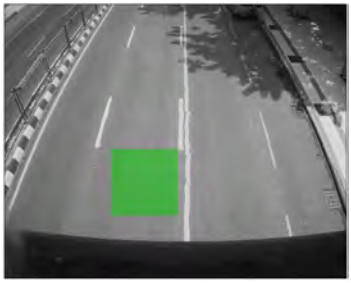
Luas ROI	Persen ROI	Terdeteksi	Error Absolute	Error %
75x75	1.72%	43	12	38.71%
100x100	3.05%	35	4	12.90%
125x125	4.77%	32	1	3.23%
150x150	6.87%	25	6	19.35%
100x125	3.81%	29	2	6.45%
100x150	4.58%	21	10	32.26%

Berdasarkan hasil pengujian, perbedaan ukuran ROI yang digunakan dapat dikatakan berpengaruh, walaupun tidak signifikan. Penggunaan ukuran ROI yang tepat dapat mempengaruhi hasil pendeteksian kendaraan seperti pada penggunaan ROI (75x75) piksel (gambar 4.14) dengan ROI (125x125) piksel (gambar 4.15). Dapat dilihat bahwa ROI 125x125 lebih mencangkup luasan roda 4 diban-

dingkan dengan ROI (75x75) piksel sehingga memungkinkan perhitungan yang lebih presisi dibandingkan dengan ROI (75x75) piksel. Pada ROI (75x75) piksel, memungkinkan terdeteksinya kendaraan roda 2 sehingga menimbulkan galat yang tidak diinginkan.



Gambar 4.14: ROI 75x75 piksel pada resolusi 640x512 piksel



Gambar 4.15: ROI 125x125 piksel pada resolusi 640x512 piksel

4.5 Pengujian tingkat akurasi program dengan video uji

Pengujian ini bertujuan untuk menguji tingkat akurasi pendeteksian kendaraan dengan menggunakan video uji. Video uji digunakan sebagai sumber masukan citra digital yang diproses sebagai uji coba apakah program dapat berjalan sesuai dengan yang diinginkan. Pengujian dilakukan dengan membandingkan hasil perhitungan manual terhadap hasil hitungan oleh program. Hasil perhitungan manual dapat dilihat pada tabel 4.4

Tabel 4.4: Hasil perhitungan secara manual

Lajur	FPS	
	25	
	Siang	Malam
1	72	84
2	91	89
3	43	27
4	1	9

4.5.1 Pengujian tingkat akurasi program pada *Frame Rate* yang berbeda

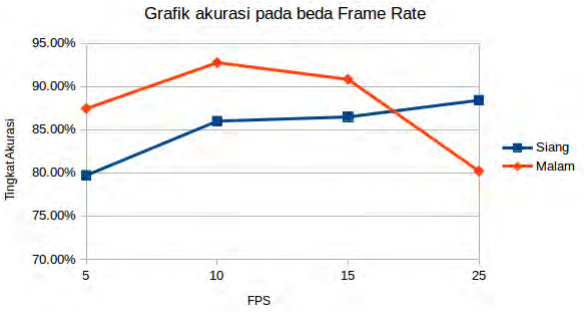
Pengujian pada *Frame Rate* dilakukan pada resolusi 640 x 512 dengan beberapa *Frame Rate* yang berbeda. Dipilih resolusi 640 x 512 dengan tujuan memperingan kerja sistem dalam memproses citra masukan serta lebih mudah dalam mendeteksi galat. Hasil perhitungan kendaraan menggunakan program dapat dilihat pada tabel 4.5.

Tabel 4.5: Hasil deteksi kendaraan menggunakan program pada *Frame Rate* yang berbeda

Lajur	FPS							
	5		10		15		25	
	Siang	Malam	Siang	Malam	Siang	Malam	Siang	Malam
1	57	74	67	91	68	91	71	97
2	65	75	74	91	78	93	81	104
3	42	28	50	32	54	32	56	35
4	1	10	1	10	1	12	1	14

Gambar 4.16 merupakan grafik persentase akurasi antara hasil perhitungan program dengan hasil hitungan manual pada siang dan malam hari. Dapat dilihat persentase akurasi menurun seiring penurunan nilai FPS dari 15 FPS menuju 5 FPS. Peningkatan nilai FPS dapat menyebabkan kesalahan deteksi, terutama pada malam hari.

Kesalahan deteksi pada beda FPS salah satunya disebabkan oleh perbedaan kecepatan tangkap kamera dengan kecepatan kendaraan yang melintas pada ROI. Kecepatan kendaraan yang lebih cepat dibandingkan dengan kecepatan tangkap kamera bisa mengakibatkan kesalahan deteksi pada kendaraan. Dianggap gambar 4.17 merupakan *frame* pada saat t sekon, sedangkan gambar 4.18 merupakan *frame* pada saat $t+1$ sekon. Pada saat pendeteksian pada

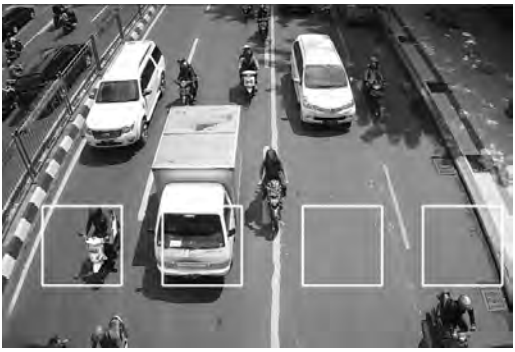


Gambar 4.16: Grafik persentase akurasi pada siang dan malam hari

frame (t+1), sistem masih menganggap kendaraan yang melintas merupakan kendaraan yang sama, padahal pada saat pergantian dari *frame t* menuju *frame t+1* terjadi pergantian kendaraan yang terdeteksi, sehingga menyebabkan terjadinya kesalahan pendeteksian.

Gambar 4.17: Contoh frame pada waktu (t)

Pada grafik 4.16, persentase akurasi pada malam hari lebih kecil dibanding siang hari. Berdasarkan analisa dari pengujian, kesalahan yang terjadi diakibatkan oleh pengaruh lampu sorot kendaraan roda 2 maupun roda 4 (gambar 4.19. Sorotan lampu yang mengenai permukaan jalan membuat intensitas pada citra masuk-



Gambar 4.18: Contoh frame pada waktu (t+1)

an meningkat, sehingga terjadi galat pada proses segmentasi citra (gambar 4.20).

Gambar 4.19: Lampu sorot kendaraan yang mengenai permukaan jalan

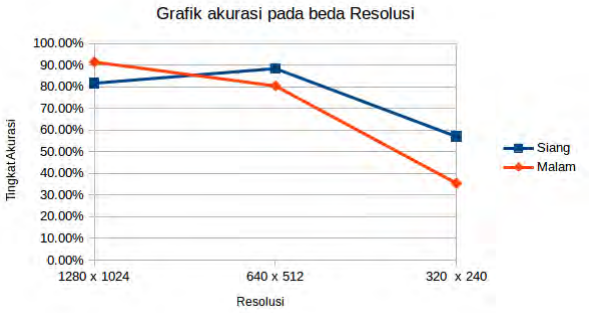
Gambar 4.20: Hasil segmentasi citra pada malam hari

4.5.2 Pengujian tingkat akurasi program pada Resolusi yang berbeda

Pada pengujian ini dilakukan pengujian pada resolusi yang berbeda untuk mengetahui pengaruh dari perbedaan resolusi terhadap hasil perhitungan program. Dipilih *Frame Rate* 25 FPS sebagai *Frame Rate* video uji agar lebih mudah dalam mendeteksi galat. Resolusi yang diuji adalah resolusi 1280x1024, 640x512, dan 320x240. Hasil deteksi kendaraan pada beda resolusi dapat dilihat pada tabel 4.6.

Tabel 4.6: Hasil deteksi kendaraan pada resolusi yang berbeda

Lajur	Resolusi					
	1280 x 1024		640 x 512		320 240	
	Siang	Malam	Siang	Malam	Siang	Malam
1	56	89	71	97	92	106
2	73	83	81	104	100	104
3	47	24	56	35	94	98
4	1	5	1	14	10	36



Gambar 4.21: Grafik persentase akurasi pada siang dan malam hari pada beda resolusi

Hasil pengujian menunjukkan perubahan resolusi berdampak pada akurasi pendeteksian kendaraan. Pada grafik 4.21, dapat dilihat terjadi penurunan akurasi pada resolusi 320x240. Perubahan resolusi mempengaruhi hasil segmentasi yang dihasilkan. Dapat dilihat pada gambar 4.22 dan gambar 4.23 bahwa citra dengan resolusi yang berbeda menghasilkan citra hasil segmentasi yang berbeda

pula. Citra pada resolusi 320x240 menghasilkan segmentasi kendaraan yang dapat dikatakan kurang baik dibandingkan pada resolusi 640x512, sehingga penggunaan resolusi yang semakin kecil memungkinkan terjadinya galat pada proses pendeteksian kendaraan.

Pada resolusi 1280x1024, hasil segmentasi yang dihasilkan terkadang masih belum sempurna, akibat jumlah piksel yang lebih banyak, sehingga pada proses segmentasi terkadang menghasilkan segmentasi yang kurang sempurna (gambar 4.24). Kesalahan segmentasi seperti ini sering terjadi pada kendaraan yang berwarna gelap, terutama pada siang hari ketika kendaraan terkena pantulan cahaya matahari.

Gambar 4.22: Citra pada resolusi 320 x 256

Gambar 4.23: Citra pada resolusi 640 x 512

Gambar 4.24: Hasil segmentasi pada resolusi 1280x1024

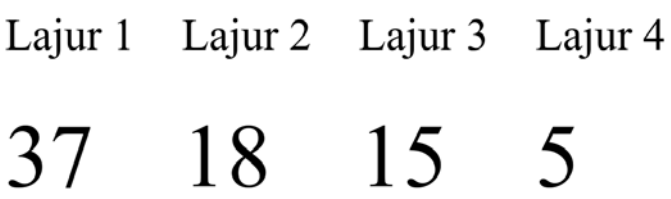
4.6 Pengujian tingkat akurasi program secara langsung

Pengujian secara langsung dilakukan di jalan raya. Pengujian ini dilakukan pada dua tempat yang berbeda, yang pertama di jalan Urip Sumoharjo, Surabaya dan di jalan Basuki Rahmat, Surabaya. Pengujian dilakukan diatas sebuah jembatan penyeberangan pejalan kaki dengan waktu yang berbeda-beda.

Gambar 4.25: Tata tempat untuk uji coba dijalan raya

Tata tempat yang dilakukan pada ujicoba dijalan dapat dilihat

pada gambar 4.25. *IP Camera* disambungkan menuju SBC *Client* menggunakan kabel *ethernet* dan SBC *Client* disambungkan menuju sebuah laptop yang nantinya berfungsi sebagai *Server Dummy*. *Server Dummy* berfungsi sebagai penerima data dari SBC *Client* dan juga mengkalkulasi jumlah kendaraan yang melintas. Jumlah kendaraan yang melintas ditampilkan pada sebuah web browser seperti gambar 4.26.



Gambar 4.26: Tampilan halaman web pada *Server dummy*

Ujicoba dilakukan dengan cara menjalankan program pendeteksi kendaraan pada SBC *Client* atau *Beaglebone* kemudian hasil deteksi dikirim menuju *Server Dummy* selanjutnya *Server Dummy* melakukan penjumlahan terhadap data yang diterima dari *Beaglebone* dan menampilkannya hasilnya pada halaman web. Proses tersebut direkam menggunakan kamera dan pada saat yang bersamaan kamera lainnya merekam kondisi jalan raya untuk nantinya digunakan pada perhitungan secara manual. Hasil perhitungan manual dibandingkan dengan hasil perhitungan program untuk mengetahui perbedaan perhitungan program dengan jumlah kendaraan yang melintas sesungguhnya. Proses ini dilakukan beberapa kali pada nilai *Detection Sensivity* yang berbeda serta waktu yang berbeda. *Frame Rate* yang digunakan adalah 25 FPS pada resolusi 640 x 512.

Galat yang didapatkan adalah selisih antar hasil perhitungan manual dan program. Dapat dilihat dari hasil pengujian, peningkatan nilai *Dectection Sensivity* dapat mengurangi galat yang disebabkan oleh kendaraan roda dua namun juga dapat meningkatkan galat pada pendeteksian kendaraan roda empat. Hal ini disebabk-

Tabel 4.7: Galat absolut di jalan Urip S. pada waktu pagi hari

Sensivitas deteksi	Lane 1	Lane 2	Lane 3	Lane 4
70%	4	1	6	0
80%	5	2	1	0
90%	1	12	3	0

	=	Over
	=	Less

Tabel 4.8: Galat absolut di jalan Urip S. pada waktu siang hari

Sensivitas deteksi	Lane 1	Lane 2	Lane 3	Lane 4
80%	12	7	2	2
90%	13	9	7	2

	=	Over
	=	Less

Tabel 4.9: Galat absolut di jalan Basuki R. pada waktu siang hari

Sensivitas deteksi	Lane 1	Lane 2	Lane 3	Lane 4
80%	9	4	3	2
90%	12	8	14	3

	=	Over
	=	Less

Tabel 4.10: Galat absolut di jalan Urip S. pada waktu malam hari

Sensivitas deteksi	Lane 1	Lane 2	Lane 3	Lane 4
70%	2	3	11	2
80%	0	6	9	0
90%	10	2	1	0

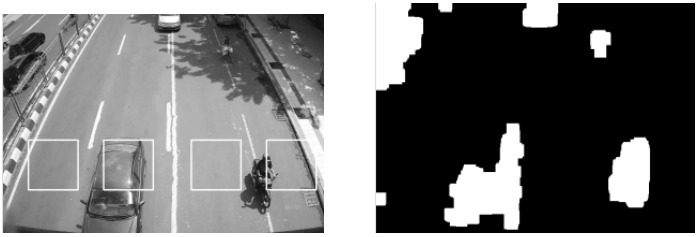
	=	Over
	=	Less

Tabel 4.11: Galat absolut di jalan Basuki R. pada waktu malam hari

Sensivitas deteksi	Lane 1	Lane 2	Lane 3	Lane 4
70%	2	7	0	2
80%	2	3	2	3
90%	2	5	9	9

	=	Over
	=	Less

an oleh jumlah piksel yang digunakan sebagai nilai *threshold* kendaraan semakin besar, sehingga hasil segmentasi kendaraan yang kurang sempurna dapat menyebabkan kesalahan deteksi. Seperti yang dijelaskan pada pengujian sebelumnya, kesalahan segmentasi citra dapat dikarenakan beberapa faktor seperti cahaya matahari yang jatuh tegak lurus terhadap kendaraan roda empat membuat warna pada kendaraan tampak lebih cerah sehingga mengubah nilai intensitas pada citra digital. Warna kendaraan juga sangat mempengaruhi hasil segmentasi kendaraan, seperti warna yang lebih gelap (menyerupai warna bagan jalan) menyebabkan kesalahan segmentasi seperti pada gambar4.27.



(a) citra masukan (b) hasil segmentasi kendaraan

Gambar 4.27: kesalahan pada proses segmentasi

Pada perhitungan malam hari (tabel 4.10), hasil perhitungan cendrung berlebih. Seperti yang dijelaskan pada pengujian sebelumnya, hal ini dapat diakibatkan pengaruh cahaya lampu sorot kendaraan yang mengenai bagan jalan. Dapat dilihat pada tabel 4.10, bahwa penggunaan sensivitas yang lebih besar dapat mengurangi kesalahan pada pendeteksian kendaraan roda 2 di malam hari namun juga memungkinkan tidak terdeteksinya kendaraan roda 4.

Tingkat akurasi perhitungan kendaraan dapat dilihat pada masing-masing tabel yang telah dilampirkan. Hasil pengujian membuktikan bahwa penggunaan sensivitas yang tepat dapat mengurangi kesalahan perhitungan. Pada pengujian kali ini, nilai sensivitas sebesar 80% cenderung menunjukkan hasil yang lebih baik dibandingkan nilai sensivitas 70% dan nilai sensivitas 90%.

Tabel 4.12: Tingkat akurasi di jalan Urip S. pada waktu pagi hari

Sensivitas deteksi	Akurasi
70%	71.05%
80%	80.95%
90%	61.90%

Tabel 4.13: Tingkat akurasi di jalan Urip S. pada waktu siang hari

Sensivitas deteksi	Akurasi
80%	68.92%
90%	59.74%

Tabel 4.14: Tingkat akurasi di jalan Basuki R. pada waktu siang hari

Sensivitas deteksi	Akurasi
80%	82.18%
90%	63.00%

Tabel 4.15: Tingkat akurasi di jalan Urip S. pada waktu malam hari

Sensivitas deteksi	Akurasi
70%	73.91%
80%	82.14%
90%	78.33%

Tabel 4.16: Tingkat akurasi di jalan Basuki R. pada waktu malam hari

Sensivitas deteksi	Akurasi
70%	88.30%
80%	87.50%
90%	68.35%

4.7 Pengujian aplikasi web

Pada pengujian ini dilakukan pengujian terhadap kemampuan aplikasi web yang dibuat apakah telah berjalan sesuai dengan yang diinginkan. Pengujian dilakukan pada masing-masing halaman aplikasi web.

Pengujian pertama dilakukan pada halaman *configuration* yaitu dengan menguji kemampuan aplikasi dalam menyimpan data masukan pada *file cam.conf*. Pengujian dilakukan dengan memban-

Tabel 4.17: Pengujian aplikasi web

Nama Pengujian	Hasil Pengujian
Save configuration	Berhasil
Load configuration	Berhasil
Save Network Setting	Berhasil
Save ROI position	Berhasil

dingkan data masukan terhadap data yang berhasil disimpan. Hasil pengujian didapatkan data masukan yang berhasil disimpan sama dengan data pada *file cam.conf*. Dapat dilihat pada gambar 4.28 bahwa yang disimpan sama dengan data yang dimasukkan.

Gambar 4.28: Pengujian *Save Configuration*

Pengujian kedua dilakukan masih pada halaman *configuration*. Pengujian dilakukan dengan membandingkan nilai yang dimasukkan dengan nilai yang ditampilkan pada box "*current configuration*". Dari hasil pengujian, nilai yang ditampilkan sudah sesuai dengan data masukkan yang disimpan (gambar 4.29).

Pengujian ketiga dilakukan pada halaman *Network Setting*. Pengujian ini dilakukan dengan membandingkan nilai yang dimasukkan terhadap alamat jaringan SBC. Pada pengujian ini dicoba untuk mengganti alamat jaringan SBC dari 10.122.1.105 menjadi 10.122.1.115. Hasil pengujian menunjukkan bahwa web aplikasi dapat berjalan dengan baik sesuai yang diharapkan (gambar 4.30 dan 4.30).

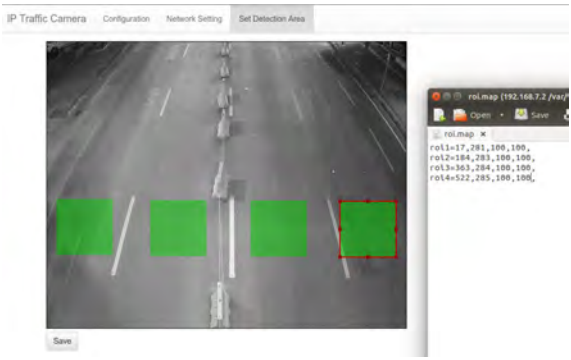
Pengujian yang terakhir adalah pegujian pada halaman *Set*

Gambar 4.29: Pengujian *Load Configuration*

Gambar 4.30: Pengujian *Save Network* sebelum diganti

Gambar 4.31: Pengujian *Save Network* setelah diganti

Detection Area. Pengujian dilakukan dengan menyimpan posisi serta luas ROI kemudian melihat nilai pada *file roi.map* apakah berhasil tersimpan dengan baik. Berdasarkan hasil pengujian, posisi serta luas ROI telah tersimpan dengan baik pada *file roi.map*. Nilai yang disimpan pada *roi.map* sesuai dengan jumlah ROI yang dimasukkan (gambar 4.32).



Gambar 4.32: Pengujian *Save ROI*

BAB 5
PENUTUP

5.1 Kesimpulan

Dari hasil implementasi dan pengujian sistem yang sudah dilakukan dapat ditarik beberapa kesimpulan sebagai berikut :

1. *Delay time* naik secara exponensial terhadap nilai resolusi yang digunakan. *Delay time* naik secara *linear* terhadap nilai FPS yang digunakan.
2. Penurunan nilai FPS memungkinkan peningkatan galat pada kondisi jalan yang kurang ramai, dikarenakan kecepatan kendaraan yang melintas melebihi kecepatan tangkap kamera. Bersadarkan hasil pengujian, penggunaan 15 FPS dinilai sudah cukup baik sebagai nilai FPS.
3. Penurunan Resolusi memungkinkan peningkatan kesalahan deteksi diakibatkan kesalahan pada proses segmentasi kendaraan. Berdasarkan hasil pengujian, resolusi 640x512 dinilai sudah cukup baik sebagai nilai resolusi pada sistem ini.
4. Peningkatan sensivitas deteksi dapat mengurangi galat pada pendeteksian kendaraan beroda dua, namun juga memungkinkan kendaraan beroda empat sulit terdeteksi. Penggunaan sensivitas 80% dinilai sudah cukup baik untuk digunakan sebagai nilai *Threshold* kendaraan.
5. Hasil pengujian secara *live* dijalan pada resolusi 640x512 dan 25 FPS, menunjukkan bahwa sistem yang diajukan dapat menghasilkan perhitungan kendaraan roda empat dengan tingkat akurasi hingga **82.18%** pada waktu siang hari dan tingkat akurasi hingga **88.30%** pada waktu malam hari.

5.2 Saran

Demi pengembangan lebih lanjut mengenai tugas akhir ini, disarankan beberapa langkah lanjutan sebagai berikut :

1. Meningkatkan *Hardware* yang digunakan, terutama pada media pengolahan citra atau SBC sehingga memungkinkan metode pengolahan citra yang lebih kompleks serta penurunan *delay process* yang dihasilkan.
2. Diperlukan pengukuran terhadap suhu kerja *Hardware* sehingga dapat diketaui pengaruh suhu terhadap kinerja sistem serta suhu kerja yang optimal untuk *Hardware* tersebut.
3. Sistem yang diajukan masih membutuhkan pengujian pada kondisi lingkungan yang berbeda-beda, seperti hujan, berawan, atau berkabut.