SISTEM DETEKSI KENDARAAN MENGGUNAKAN METODE *BACKGROUND SUBSTRACTION* PADA *EMBEDDED SYSTEM*

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
Memperoleh gelar Sarjana Komputer

Disusun Oleh :
Nama : Ibrahim Sholeh
NIM : 115060901111002

TEKNIK INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

SISTEM DETEKSI KENDARAAN MENGGUNAKAN METODE *BACKGROUND SUBSTRACTION* PADA *EMBEDDED SYSTEM*

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Ibrahim Sholeh
NIM: 115060901111002

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

Pembimbing I	Pembimbing II
Wijaya Kurniawan, S.T., M.T.	Fitri Utaminingrum, Dr.Eng., S.T., M.T.
NIP. 19820125 201504 1 002	NIP. 19820710 200812 2 001

Mengetahui
Ketua Program Studi Informatika

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

PERNYATAAN
ORISINALITAS SKRIPSI

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 dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

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

Malang, 14 Januari 2016

Ibrahim Sholeh
NIM. 115060901111002

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa, karena Rahmat-Nya lah penulis dapat menyelesaikan skripsi ini. Adapun maksud penyusunan skripsi ini adalah untuk memenuhi salah satu persyaratan dalam menempuh ujian Sarjana Program Teknologi Informasi dan Ilmu Komputer. Judul skripsi yang disusun adalah: **"Sistem Deteksi Kendaraan Menggunakan Metode Background Substraction pada Embedded System".**

Banyak kesulitan dan hambatan yang dialami penulis dalam menyusun skripsi ini terutama dalam mendapatkan data dan mengolahnya, tetapi semua dapat diatasi karena dukungan dari berbagai pihak. Ucapan terima kasih penulis sampaikan kepada:

1. Bapak dan ibu atas segenap do'a, serta dukungan dan kasih sayang yang telah diberikan sehingga skripsi ini terselesaikan.
2. Wijaya Kurniawan, S.T., M.T. selaku dosen pembimbing I yang telah memberikan pengarahan dalam penyusunan skripsi ini.
3. Fitri Utaminingrum, Dr.Eng., S.T., M.T. selaku dosen pembimbing II yang telah memberikan masukan dalam pengerjaan skripsi sehingga skripsi ini menjadi lebih baik.
4. Ir. Sutrisno, M.T. selaku Ketua Program Teknologi Informasi dan Ilmu Komputer dan selaku dosen penasihat akademik penulis.
5. Adharul Mutaqqin, S.T., M.T. selaku ketua program studi sistem komputer yang telah memberikan pengarahan dan motivasi dalam pembuatan skripsi ini.
6. Barlian Henryranu, S.T., M.T. selaku Kepala Laboratorium Sistem Komputer dan Robotika Program Teknologi Informasi dan Ilmu Komputer
7. Segenap dosen Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya atas segenap ilmu pengetahuan dan perhatian yang diberikan Segenap staff dan pegawai Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya atas segala bantuan yang bersifat administratif.
8. Teman-teman Program Studi Sistem Komputer angkatan 2011 tercinta yang selalu memberikan semangat, dorongan dan bantuan.

Penulis menyadari bahwa skripsi ini jauh dari sempurna, oleh karena itu untuk segala kritik dan saran yang membangun penulis ucapkan terima kasih.

Malang, 14 Januari 2016

Penulis
Ibrahim Sholeh

ABSTRAK

Ibrahim Sholeh 2015 : Sistem Deteksi Kendaraan Menggunakan Metode Image Processing pada Embedded System. Skripsi Program Studi Informatika/Ilmu Komputer, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.

Dosen Pembimbing : Wijaya Kurniawan, S.T., M.T. dan Fitri Utaminingrum, Dr.Eng., S.T., M.T.

Pertumbuhan penduduk yang terus meningkat menyebabkan bertambahnya jumlah kendaraan yang memicu terjadinya kemacetan lalu lintas, sehigga dibutuhkan kebijakan pemerintah untuk mencegah kemacetan yang lebih parah. Untuk itu, perlu menganalisa jumlah kendaraan pada setiap daerah yang dituju. Akan tetapi, hal tersebut masih dilakukan secara manual sehingga biaya operasionalnya tinggi. Oleh karena itu, diperlukan sistem yang secara otomatis melakukan hal tersebut sehingga lebih efisien.

Sistem diletakkan pada posisi yang tetap untuk mendeteksi kendaraan pada daerah tertentu. Sistem terdiri dari raspberry pi 2 sebagai perangkat keras, OpenCV sebagai perangkat lunak dan video sebagai data yang diproses. Video yang digunakan beresolusi 320x240 dengan *framerate* 29 fps (frametime 34,48 ms). Sistem menggunakan metode background substraction yaitu membandingkan gambar yang sekarang (foreground) ditangkap oleh sistem dengan yang sudah ditentukan sebelumnya (background). Jika terdapat perbedaan gambar, maka akan terdapat suatu objek. Lalu, citra disegmentasi dimana piksel putih menunjukkan objek, sedangkan piksel hitam adalah *backgroundnya*. Lalu piksel putih diukur luasnya, panjang dan lebarnya unutk menentukan objek tersebut adalah jenis kendaraan atau tidak. Hasil menunjukkan bahwa sistem ini dapat mendeteksi kendaraan berdasarkan jenisnya (mobil, truk dan bus) dengan akurasi sebesar 98% dan berdasarkan jumlahnya dengan akurasi sebesar 100%. Untuk performa, sistem membutuhkan waktu sekitar 138,3 ms untuk mengolah setiap *frame* video sedangkan pada komputer membutuhkan waktu sekitar 15,3 ms. Hal ini berarti *embedded system* belum mampu bekerja secara *realtime* karena waktu *embedded system* yang dibutuhkan untuk mengolah setiap *frame* video lebih besar dari *frametime* video. Oleh karena itu, butuh optimasi lebih lanjut untuk diimplementasikan secara langsung.

Kata kunci : *Raspberry pi 2, OpenCV, Background Substraction, Jumlah Kendaraan , Performa*

ABSTRACT

Ibrahim Sholeh 2015 : Sistem Deteksi Kendaraan Menggunakan Metode Image Processing pada Embedded System. Skripsi Program Studi Informatika/Ilmu Komputer, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.

Dosen Pembimbing : Wijaya Kurniawan, S.T., M.T. dan Fitri Utaminingrum, Dr.Eng., S.T., M.T.

The increasing number of population can cause higher number of vehicles that trigger the traffic congestion, so that it takes the government policy to prevent the worse traffic congestion. Hence, it needs traffic analysis on each intended area. But still, it done manually, so that it needs high net-operating cost. Therefore, it needs a system that perform traffic analysis automatically for the better efficiency cost.

The system is placed on a fixed position in detecting vehicles at a given place. It consist of raspberry pi 2 as hardware, OpenCV as software and the video as a input data. The video is set to 320x240 pixel in resolution with frametime of 34,48 ms (29 fps). It use background substraction method for detecting objects. That is compare the image that is now captured (as foreground) with a defined image before (as background). if there is difference between foreground and backgound, then there will be an object. Then, the image is segmented where white pixel is the object and black pixel is the background. then, the wide, width and length of the object is counted to determine whether vehicle or not. The result shows that system can detect vehicles with accuracy by 98% based on the type of vehicles, and can detect vehicle with accuracy by 100% based on the number of vehicles. In term of performance, the embedded system take 138,3 ms to process each frame of video, while desktop computer take 15,3 ms to process each frame of video. It means that embedded system still unable to real-time work because the time needed for embedded system to process each of frame of video is higher time than frametime of video. Therefore, it needs more optimation to deploy in real world.

Keyword : *Raspberry pi 2, OpenCV, Background Substraction, Number of Vehicle, Performance*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Jumlah penduduk di Indonesia terus menunjukkan peningkatan dari tahun ke tahun (DITRENDUK, 2010). Hal ini menyebabkan bertambahnya jumlah kendaraan dan menimbulkan kemacetan lalu lintas. Jika dibiarkan, maka kemacetan lalu lintas akan terus meningkat.

Kemacetan lalu lintas merupakan salah satu masalah terbesar di Indonesia karena dapat merugikan secara global : BBM terbuang sia-sia, kegiatan masyarakat terhambat, polusi udara dan lain-lain. Keterbatasan infrastruktur tidak sebanding dengan kebutuhan penduduk Indonesia yang terus meningkat dari tahun ke tahun juga merupakan salah satu penyebab utama kemacetan lalu lintas.

Oleh karena itu, dibutuhkan kebijakan dari pemerintah untuk mengatur sistem lalu lintas agar kemacetan dapat dikendalikan. Kebijakan pemerintah harus se-efektif mungkin agar hasil yang didapat maksimal. Untuk itu, perlu menganalisa tingkat kepadatan kendaraan terlebih dahulu di setiap daerah yang akan dijadikan objek penelitian.

Namun, masih terdapat kendala dalam menganalisa tingkat kepadatan jalan, salah satunya adalah penghitungan jumlah kendaraan secara manual dengan cara mengerahkan tenaga kerja untuk survey langsung di lapangan (Silvanus A., 2013). hal ini menyebabkan tingginya biaya operasional, membutuhkan konsentrasi tinggi sehingga rawan terjadi kesalah perhitungan. Selain itu, keterbatasan tenaga kerja juga membuat program pemerintah tidak dapat terlaksana dengan baik. Oleh karena itu, dibutuhkan suatu sistem yang dapat menangani masalah tersebut, sehingga proses analisa tersebut berjalan lebih efektif karena hasil hitung lalu lintas merupakan tolak ukur untuk tangani jalan (Litbang, 2013).

1.2 Rumusan Masalah

1. Bagaimana *Embedded System* dapat mendeteksi kendaraan yang melintas?
2. Berapa akurasi yang didapat dalam mendeteksi kendaraan yang melintas berdasarkan jenis kendaraan dengan kecepatan kendaraan yang sebenarnya, 2 kali dan 4 kali lebih cepat dari kecepatan yang sebenarnya?
3. Berapa akurasi yang didapat dalam mendeteksi kendaraan yang melintas berdasarkan jumlahnya dengan kecepatan kendaraan yang sebenarnya, 2 kali dan 4 kali lebih cepat dari kecepatan yang sebenarnya?
4. Bagaimana kualitas performa / kinerja *embedded* sistem dibandingkan dengan komputer dalam mendeteksi kendaraan?

1.3 Tujuan

1. Membuat "Sistem Deteksi Kendaraan menggunakan metode Background Substraction pada Embedded System".
2. Menggabungkan perangkat hardware dan software sedemikian rupa sehingga terciptalah Sistem Deteksi Kendaraan menggunakan metode Background Substraction pada Embedded System dalam menghitung jumlah kendaraan.

1.4 Manfaat

Membantu instansi pemerintah dalam menganalisa perkembangan jalan, situasi jalan, tingkat kepadatan jalan pada jam-jam tertentu yang nantinya akan digunakan sebagai tolak ukur atau parameter dalam menentukan kebijakan-kebijakan terkait dengan lalu lintas. Sehingga, kebijakan tersebut dapat berjalan efektif.

1.5 Batasan Masalah

1) Input dari sistem ini adalah gambar digital berupa video.
2) Sistem digunakan pada kondisi cahaya yang cukup (disiang hari).
3) Embedded system yang digunakan adalah raspberry pi 2.

1.6 Sistematika Pembahasan

Skripsi ini terbagi dalam 6 (enam) bab, uraian singkat mengenai ini masing-masing bab adalah sebagai berikut :

BAB I PENDAHULUAN

Memuat latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan.

BAB II KAJIAN PUSTAKA DAN DASAR TEORI

Membahas penelitian-penelitian sebelumnya yang berhubungan dan teori-teori yang mendukung dalam perencanaan sistem.

BAB III METODE PENELITIAN DAN PERANCANGAN

Berisi tentang langkah-langkah dalam merancang sistem.

BAB IV IMPLEMENTASI

Bagian ini berisi penjelasan tentang spesifikasi sistem, konfigurasi, dan prosedur implementasi .

Bab V PENGUJIAN DAN ANALISIS

Menguji dan menganalisa performansi sistem.

BAB VI PENUTUP

Memuat kesimpulan dan saran.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 KAJIAN PUSTAKA

Perhitungan kendaraan di jalan raya merupakan permasalahan yang sangat menarik untuk dipecahkan dengan *Image Processing*. Hal ini dibuktikan dengan banyaknya paper-paper terkait yang dipublikasikan pada jurnal-jurnal nasional maupun internasional.

Salah satunya adalah hasil penelitian Susmita Mesram S.A. et al. (2013), yang berjudul "*Traffic Surveillance by Counting and Classification of Vehicles from Video using Image Processing*". Penelitian ini mendeteksi kendaraan menggunakan metode deteksi tepi *Canny*. Masing-masing *background* dan *foreground* diproses menggunakan deteksi tepi *Canny.* Kemudian, masing-masing hasilnya dibandingkan satu sama lain. Jika terdapat perbedaan objek yang terdeteksi, maka dianggap ada kendaraan yang melintas, sedangkan jika tidak ada perbedaan objek yang terdeteksi, maka dianggap tidak ada kendaraan yang melintas .

Pendeteksian kendaraan dengan menggunakan sensor SRF02 diusulkan oleh Ahmad S. et al. (2008). Sensor SRF02 dapat mendeteksi objek dengan memancarkan gelombang ultrasonik. Sensor ini terdiri dari 2 bagian utama yaitu *transmitter* dan *receiver*. Bagian *transmitter* berfungsi untuk mengirimkan gelombang ultrasonik pada suatu objek dan dipantulkan menuju bagian *receiver*. Jika terdapat perubahan waktu antara pengiriman sinyal dari bagian *transmitter* dengan penerimaan sinyal ke bagian *receiver*, maka dianggap ada objek yang melintasinya.

Ada pula penelitian lain dari Permana M.I. (2015), yang merancang sistem deteksi / pengenalan kendaraan menggunakan metode *haar-Cascade*. Input yang digunakan berupa video. Sistem ini mengolah setiap *frame* dari video tersebut. Jika pada *frame* video tersebut terdapat objek yang mirip dengan data *training haar-Cascade,* maka dianggap ada kendaraan yang melintas. Akan tetapi, jika pada *frame* selanjutnya terdeteksi kendaraan yang sama, maka kendaraan tersebut juga dihitung. Sehingga, 1 kendaraan dihitung 2 kendaraan atau lebih.

Berdasarkan penelitian sebelumnya, maka diusulkan penghitung jumlah kendaraan menggunakan metode *Background Substraction* adaptif di mana jika ada kendaraan yang sama dalam *frame* yang berbeda tetap dihitung 1 kendaraan.

2.2 DASAR TEORI

Dasar teori berisi tentang teori-teori dasar yang berkaitan dengan sistem yang akan dibuat. Kemudian, teori-teori tersebut dikembangkan dan

dihubungkan menjadi satu yang digunakan untuk mendukung dalam pembuatan sistem.

2.2.1. Pengertian Citra

Citra adalah suatu fungsi yang berbentuk 2 dimensi yang menunjukkan suatu informasi seperti warna, tingkat kecerahan dari sebuah pandangan yang terlihat. Citra merupakan proyeksi objek berbidang 3 dimensi ke dalam bidang 2 dimensi (Jayaraman S. et al., 2011).

2.2.2. Citra Digital

Citra digital adalah sebuah gambar yang terdiri dari kumpulan elemen yang disebut dengan piksel. Piksel merupakan sampel terkecil yang berbentuk kotak dari suatu gambar. Setiap piksel mempunyai nilai yang menunjukkan warna dan intensitas cahaya pada bagian gambar tersebut (Jayaraman S. et al., 2011).

Gambar 2.1. Contoh Citra Digital

2.2.3. Pengolahan Citra Digital (PCD)

PCD merupakan suatu proses komputasi citra digital yang dilakukan oleh komputer. Secara umum PCD digunakan untuk memanipulasi citra digital dan mendapatkan informasi tertentu dari citra digital yang diolah (Jayaraman S. et al., 2011).

2.2.4. Representasi Citra Digital

Citra digital tersusun atas array 2 dimensi yaitu baris dan kolom yang mempunyai ukuran tertentu (Darma P., 2010). Misalnya, suatu citra digital mempunyai ukuran 1280 x 800. Artinya, citra tersebut mempunyai kolom sebanyak 1280 piksel dan baris sebanyak 800 piksel. Citra digital dapat dinotasikan dalam bentuk fungsi $f(x, y)$ dimana variabel x adalah posisi baris, variabel y adalah posisi kolom dan f adalah variabel suatu citra digital dengan

tipe data matriks. Sehingga, $f(x, y)$ merupakan nilai yang menunjukkan informasi warna dan tingkat kecerahan citra digital f pada koordinat (x, y). Misalnya citra digital M mempunyai ukuran kolom P dan baris L. maka citra digital M(L,P) dapat dituliskan sebagai berikut:

$$M = \begin{bmatrix} M(x_1,y_1) & M(x_2,y_1) & M(x_3,y_1) & \dots & M(x_p,y_1) \\ M(x_1,y_2) & M(x_2,y_2) & M(x_3,y_2) & \dots & M(x_p,y_2) \\ \dots & \dots & \dots & \dots & \dots \\ M(x_1,y_L) & M(x_1,y_L) & M(x_3,y_L) & \dots & M(x_p,y_L) \end{bmatrix} \quad \text{.....(2.1)}$$

Keterangan :
M = suatu variabel citra digital dengan tipe data matriks
P = kolom matriks
L = baris matriks

2.2.5. Citra Warna

Citra warna merupakan citra yang terdiri atas 3 komponen warna dasar yaitu merah, hijau, dan biru. Sehingga, setiap pikselnya terdiri dari 3 komponen warna tersebut (Darma P., 2010). Nilai dari masing-masing warna adalah bilangan adalah antara rentang 0-255. Matriks nilai-nilai citra warna berbentuk 3 dimensi dimana bagian *depth* (kedalaman) menunjukkan komponen-komponen warnanya. sedangkan urutannya terserah. Misalnya, citra warna berukuran 3 x 3 pikel sebagai berikut :

a)

$$\begin{bmatrix} (255,0,0) & (0,255,0) & (0,0,255) \\ (128,128,0) & (0,128,128) & (0,0,0) \\ (50,50,50) & (255,255,255) & (100,200,120) \end{bmatrix}$$

b)

Gambar 2.2. a) contoh citra warna, b) nilai setiap piksel-pikselnya

Gambar 2.2 bagian a merupakan contoh citra warna ukuran 3x3 piksel, sedangan gambar 2.2 bagian b adalah nilai setiap piksel-piksel dari citra warna tersebut. Setiap piksel terdiri dari 3 nilai array yaitu *red* (merah), *green* (hijau) dan *blue* (biru).

2.2.6. Citra Grayscale

Citra *Grayscale* merupakan suatu jenis citra keabu-abuan yang hanya menunjukkan tingkat intensitas / kecerahan cahaya. Setiap nilai pikselnya hanya terdiri dari 1 kanal 8 bit bilangan bulat positif dengan rentang nilai 0 – 255, dimana nilai 0 merupakan warna hitam sempurna dan nilai 255 merupakan warna putih sempurna (Darma P., 2010). Secara umum, citra *grayscale* digunakan untuk mengukur intensitas / kecerahan warna setiap piksel pada citra warna atau dapat digunakan sebagai langkah awal dalam suatu metode pengolahan citra digital. Nilai setiap piksel citra *grayscale* dapat diperoleh dari citra warna menggunakan rumus matematika 2.2 berikut :

*Gs = 0.2989 * R + 0.5870 * G + 0.1140 * B*.....(2.2)

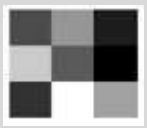
Keterangan :
Gs = nilai *grayscale* citra digital
R = nilai intensitas warna red (merah)
G = nilai intensitas warna green (hijau)
B = nilai intensitas warna blue (biru)

Misalnya, pada citra warna pada gambar 2.2 akan dikonversi menjadi citra *grayscale*, maka dengan menggunakan rumus 2.2 untuk mendapatkan nilai *grayscale* setiap pikselnya :

0.2989*255+0,587*0+ 0.114*0	0.2989*0+0,5870*255+0. 1140*0	0.2989*0+0,587*0+ 0.114*255
0.2989*128+ 0,587*128 + 0.114*0	0.2989*0+0,5870*128 +0.1140*128	0.2989*0+0,587*0+ 0.114*0
0.2989*50+0,587*50 + 0.114*50	0.2989*255+0,587*255+0 .114*255	0.2989*100+0,587*200+0 .114*120

Gambar 2.3. Contoh perhitungan nilai *grayscale*

Hasil citra *grayscale* dari gambar 2.2 dan hasil perhitungan dari gambar 2.3 diatas ditunjukkan pada gambar 2.4 dibawah ini :



a)

$$\begin{bmatrix} 76 & 150 & 29 \\ 203 & 90 & 0 \\ 50 & 255 & 161 \end{bmatrix}$$

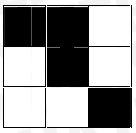
b)

Gambar 2.4. a) citra *grayscale* dari gambar 2.2. b) nilai piksel citra grayscale-nya

2.2.7. Citra Biner

Citra biner merupakan citra yang setiap pikselnya hanya terdiri dari warna hitam atau putih. Nilai pikselnya hanya terdiri dari 2 kemungkinan yaitu bernilai "0" yang berarti warna hitam atau bernilai "255" yang berarti warna putih (S.Jayaraman et al., 2011). Citra biner ini secara umum digunakan untuk menyatakan mana yang *background* dan manan yang *foreground*.

Gambar 2.5 dibawah ini merupakan contoh citra biner berukuran 3x3 piksel :



a)

$$\begin{bmatrix} 0 & 0 & 255 \\ 255 & 0 & 255 \\ 255 & 255 & 0 \end{bmatrix}$$

b)

Gambar 2.5. a) contoh citra biner 3x3, b) nilai setiap pikselnya

2.2.8. Aritmatika Citra

Aritmatika citra merupakan proses operasi aritmatika pada citra digital seperti penjumlahan, pengurangan, perkalian dan pembagian. Operasi tersebut dilakukan setiap antar piksel citra digital (Darma P., 2010). Akan tetapi, nilai minimum dan maksimum dari hasil operasi aritmatika adalah "0" dan "255". Tipe datanya tetap 8 bit bilangan bulat. Berikut ini adalah rumus setiap operasi aritmatika citra digital :

1) Penjumlahan

a. $h(x,y) = f(x,y) + g(x,y)$ .....(2.2)
b. $h(x,y) = f(x,y) + C$ .....(2.3)

2) Pengurangan

a. $h(x,y) = a \quad (f(x,y) - g(x,y))$ .....(2.4)
b. $h(x,y) = a \quad (f(x,y) - C))$ .....(2.5)

3) Perkalian

a. $h(x,y) = f(x,y) * g(x,y)$ .....(2.6)
b. $h(x,y) = f(x,y) + C$ .....(2.7)

4) Pembagian

a. $h(x,y) = \frac{f(x,y)}{g(x,y)}$ .....(2.8)
b. $h(x,y) = \frac{f(x,y)}{C}$ .....(2.9)

Keterangan :

1. $f(x,y)$ = variabel matriks 1 pada koordinat x dan y
2. $g(x,y)$ = variabel matriks 2 pada koordinat x dan y
3. $h(x,y)$ = variabel matriks hasil operasi aritmatika pada koordinat x dan y
4. C = konstanta yang nilainya tetap

2.2.9. Thresholding

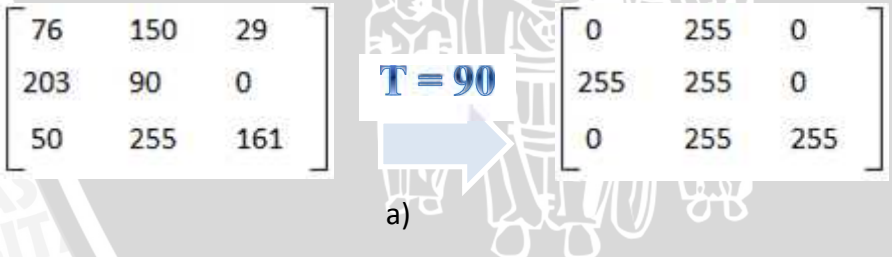
Thresholding merupakan suatu metode yang digunakan untuk segmentasi citra yaitu menghasilkan citra biner (Darma P., 2010). Metode ini umumnya digunakan untuk mengekstrak objek tertentu, misalnya memisahkan foreground dan background, proses deteksi tepi objek dan lain-lain. Secara umum rumus matematis *thresholding* dapat ditulis seperti rumus 2.10 dibawah ini :

$$g(x,y)= \begin{cases} 1 \; jika \; f(x,y) \geq T \\ 0 \; jika \; f(x,y) < T \end{cases} \quad \text{.....(2.10)}$$

Keterangan :
$g(x,y)$ = nilai piksel pada matriks output pada kolom ke- x dan baris ke-y
$f(x,y)$ = nilai piksel pada matriks input pada kolom ke- x dan baris ke-y
T = nilai variabel threshold

Tipe data dari nilai threshold adalah bilangan bulat positif 8 bit dengan rentang 0 – 255 karena nilai minimum citra digital adalah "0", sedangkan nilai maksimumnya adalah "255".

Jenis citra yang dapat di-threshold adalah citra *grayscale* dan citra warna. Dikarenakan setiap piksel citra *grayscale* hanya terdiri dari 1 array maka nilai threshold-nya juga 1 array, sedangkan setiap piksel citra warna terdiri dari 3 array, maka nilai threshold-nya juga terdiri dari 3 array. misalnya dari gambar 2.4 akan dithreshold dengan nilai T = 90.



a)

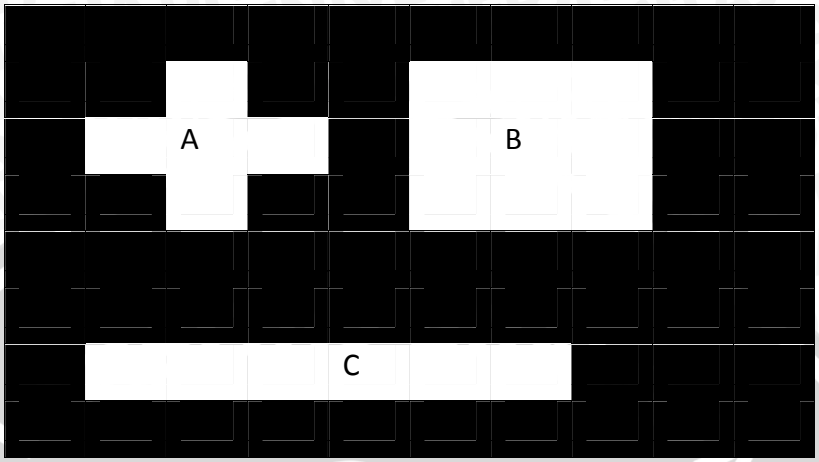
b)

Gambar 2.6. a) contoh proses thresholding dengan nilai T = 90. b) hasil matriksnya

2.2.10. Operasi Morfologi Citra (OMC)

OMC adalah suatu teknik pengolahan citra digital yang melibatkan struktur dan hubungan antar bagian suatu objek. Teknik ini digunakan untuk

menghilangkan *noise,* mengisi objek yang berlubang, menghaluskan gambar atau memisahkan satu objek dengan objek yang lain yang bersinggungan (Supianto A., 2013).



Gambar 2.7. Citra biner yang terdapat 3 himpunan objek A,B dan C

Pada gambar 2.7 terdapat 3 region objek A,B, dan C dengan asumsi warna putih sebagai latar depan (*foreground*) dan warna hitam sebagai latar belakang (*background*). Operasi morfologi citra terdiri dari 2 jenis input, yaitu citra digital sebagai input dan SE (*structuring elements*). SE merupakan suatu matriks berukuran relatif kecil yang digunakan untuk menentukan setiap nilai piksel dari matriks output.

a. b. c.

Gambar 2.8. a) SE berukuran 1x1. b) SE berukuran 3x1. c) SE berukuran 3x3

Gambar 2.8 diatas merupakan contoh-contoh SE. nilai piksel SE ditentukan sesuai keinginan *user*. Yang berwarna biru adalah bagian dari SE yang digunakan untuk operasi morfologi pada setiap piksel citra input. Bentuk-bentuk dan ukuran SE juga dapat disesuaikan oleh keinginan *user* seperti SE yang berbentuk lingkaran, oval, garis putus-putus, belah ketupat dan lain-lain. Secara umum operasi morfologi terdiri dari 2 jenis yaitu dilasi dan erosi.

2.2.10.1. Dilasi

Dilasi merupakan operasi morfologi yang digunakan untuk memperbesar ukuran region objek dengan menambah nilai piksel di sekeliling objek. Persamaannya :

$$h(x,y) = f(x,y) \oplus S \quad \text{.....(2.11)}$$

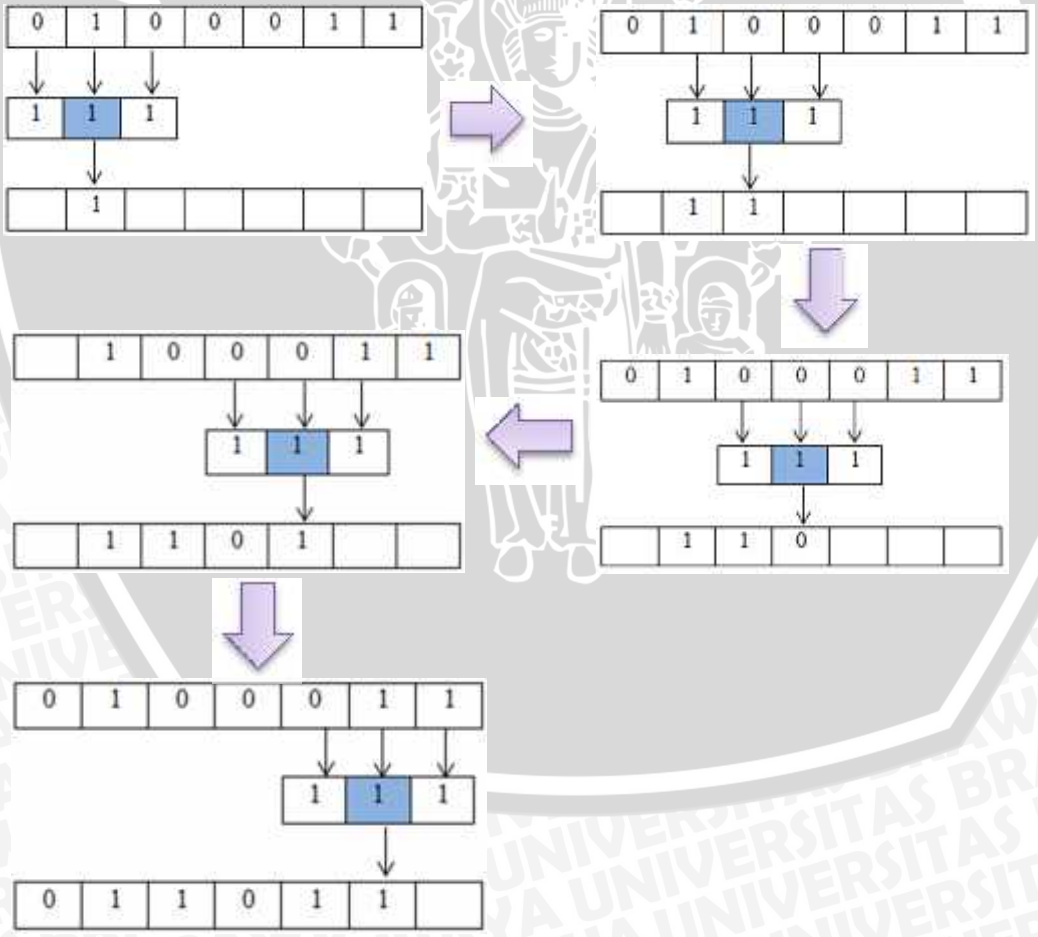
Keterangan :
$h(x,y)$ = nilai piksel matriks output pada kolom ke x dan baris ke y
$f(x,y)$ = nilai piksel matriks input pada kolom ke x dan baris ke y
SE = *structural elements* yang digunakan dalam operasi morfologi

Prinsip kerjanya adalah apabila terdapat nilai "1" dari SE yang sama dengan nilai input maka outputnya bernilai "1", sedangkan tidak ada yang sama dengan nilai input, maka outputnya bernilai "0". Misalnya, diberikan input dengan ukuran 7x1 dan SE bernilai 3x1 dengan nilai sebagai berikut:

input :	0	1	0	0	0	1	1

SE :	1	1	1

Maka urutan proses dilasi adalah sebagai berikut :



Gambar 2.9. urutan proses dilasi

Output dari proses dilasi yang ditunjukkan pada gambar 2.9 menunjukkan peningkatan jumlah nilai piksel putih dibandingkan input disekitar regionnya.

2.2.10.2. Erosi

Erosi adalah kebalikan dari dilasi. Erosi digunakan untuk memperkecil ukuran region objek dengan menghilangkan nilai piksel disekeliling objek. Rumus matematisnya adalah :

$$h(x,y) = f(x,y) \ominus \text{SE} \quad \text{..... (2.12)}$$

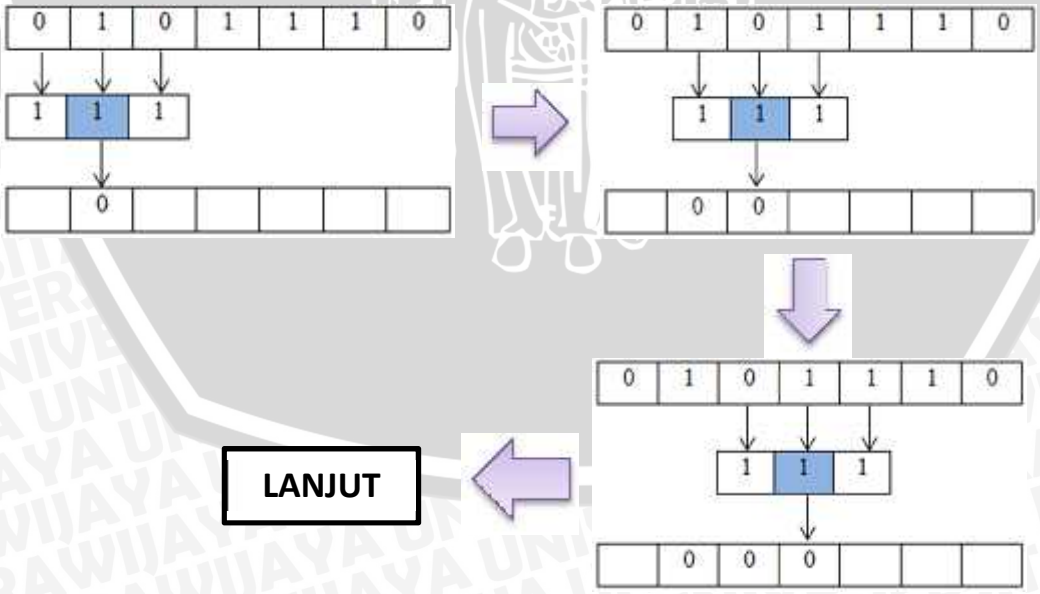
Keterangan :
$h(x,y)$ = nilai piksel matriks output pada kolom ke x dan baris ke y
$f(x,y)$ = nilai piksel matriks input pada kolom ke x dan baris ke y
SE = *structural elements* yang digunakan dalam operasi morfologi

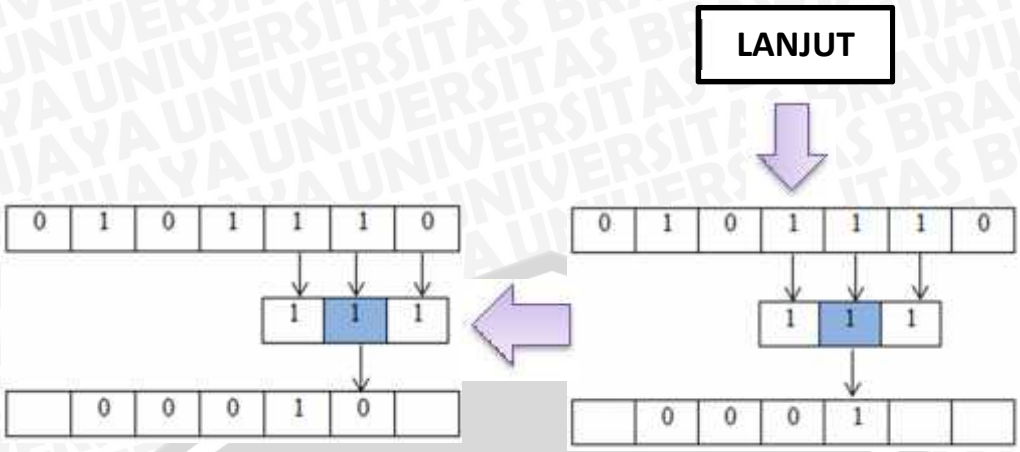
Jika semua nilai 1 dari SE sama dengan nilai input, maka outputnya bernilai 1 sedangkan ada salah satu nilai 1 dari SE tidak sama dengan input, maka outputnya bernilai 0. Misalnya, diberikan input dengan ukuran 7x1 dan SE bernilai 3x1 dengan nilai sebagai berikut:

input :	0	1	0	1	1	1	0

SE :	1	1	1

Maka urutan operasi erosinya adalah sebagai berikut :





Gambar 2.10. Urutan Proses Erosi

Output dari proses dilasi yang ditunjukkan pada gambar 2.10 menunjukkan pengurangan jumlah nilai piksel putih disekitar region objek sebagaimana fungsi dari erosi.

2.2.11. OpenCV

OpenCV adalah sekumpulan *library* khusus yang digunakan untuk pengolahan citra digital (*object oriented*). Opencv bersifat *open source* yang artinya gratis untuk semua *user* komputer. *Library* ini dapat diintegrasikan dengan bahasa pemrograman apapun seperti C/C++, Java, phyton dll. OpenCV mempunyai beberapa kelebihan antara lain : Tingkat komputasinya yang relatif ringan sehingga dapat diterapkan pada semua jenis komputer baik personal komputer atau *embedded system* atau platform lain (Baggio D.L. et al. 2015).

2.2.12. Video

Video merupakan serangkaian / sekumpulan gambar digital yang ditampilkan secara bergantian dengan kecepatan tertentu yang konstan. Sehingga, gambar tersebut terlihat bergerak. Video mempunyai beberapa parameter utama yang sering dijadikan sebagai spesifikasinya, antara lain : *frame rate / frame per second (fps)*, resolusi video, *bit rate* dan *file type* (Binanto I., 2010).

- *frame rate (fps)*
  frame rate adalah jumlah *frame* / gambar yang ditampilkan secara bergantian setiap detiknya. Misalnya, video dengan *frame rate* 30 fps, berarti terdapat 30 gambar yang ditampilkan secara bergantian setiap detiknya.
- Resolusi video
  Resolusi video adalah ukuran gambar yang terdapat pada video tersebut. Satuan yang digunakan adalah piksel. Semakin banyak jumlah pikselnya, maka semakin jelas pula gambarnya.
- *bit rate*

bit rate merupakan ukuran berkas video dalam setiap detiknya. Satuannya adalah bps (*bit per second*). Misalnya, suatu video dengan durasi 10 detik berukuran 1000 bit, berarti *bit rate*-nya adalah 100 bps.

- *file type*

  file type merupakan ekstensi berkas video / jenis berkas video. Ekstensi berkas video yang sering digunakan adalah AVI (Audio-Video Interleaved),MP4 (MPEG-4 Video), MKV (Matroska Video) dan FLV (Flash Video Format).

2.2.13. Background Substraction

Background Substraction merupakan salah satu metode dalam pengolahan citra digital sering digunakan untuk mendeteksi ada tidaknya gerakan dalam suatu gambar. *Background Substraction* terdiri dari 2 elemen yaitu *background* dan *foreground. Background* merupakan gambar yang dijadikan sebagai inisialisasi keadaan awal, sedangkan *foreground* merupakan gambar yang akan dibandingkan dengan *background*. Jika terdapat selisih nilai piksel dengan rentang tertentu (ditentukan oleh *user*) maka dianggap terdapat objek yang bergerak / melintas (Deepoy et al., 2014). Jika tidak terdapat selisih nilai piksel maka dianggap tidak ada objek yang melintas. Metode ini menggunakan operasi aritmatika pengurangan masing-masing nilai piksel matriks antara variabel *background* dan *foreground* seperti yang ditunjukkan rumus 2.4. Gambar 2.11 dan 2.12 dibawah ini adalah contoh operasi *background substraction* :

a) b) c)

Gambar 2.11. Contoh Implementasi *Background Substraction* bagian 1. a) background, b) foreground, c) hasil *background substraction.*

Pada gambar 2.11 terlihat tidak ada perbedaan gambar antara *background* dan *foreground*. Artinya, tidak ada selisih nilai piksel antara *background* dan *foreground*. Sehingga, hasil operasi *background substraction* tidak terlihat adanya suatu objek seperti yang ditunjukkan gambar 2.11 pada bagian c.

a) b) c)

Gambar 2.12. Contoh Implementas *Background Substraction* bagian 2. a) background, b) foreground, c) hasil *background substraction.*

Pada gambar 2.12 bagian a dan b terlihat adanya perbedaan gambar / nilai piksel antara *background* dan *foreground.* Perbedaan nilai piksel tersebut yang dijadikan sebagai objek gambar pada operasi *background substraction* seperti yang ditunjukkan gambar 2.12 bagian c. Metode ini cocok digunakan pada tempat yang posisinya tetap (tidak bergerak).

2.2.14. Gaussian Blur

Gaussian blur merupakan suatu metode dalam pengolahan citra digital yang digunakan untuk memperhalus atau mengurangi *noise* pada gambar dengan cara mengalikan nilai matriks kernel dengan input di koordinat yang sama, lalu dirata-rata sebagai nilai piksel outputnya (Juhari I., 2014). Gaussian blur terdiri atas 3 bagian:

a) Kernel

Kernel adalah suatu matriks berukuran kecil yang terdiri dari bilangan dengan pola-pola tertentu yang disebut dengan bobot (*weight*). Matriks ini digunakan untuk menentukan nilai piksel output dari setiap operasi *Gaussian* blur pada citra input.

b) Gaussian Filter

Gaussian filter merupakan fungsi Gaussian yang digunakan untuk menentukan nilai kernel. Berikut ini adalah rumus Gaussian filter :

$$g(x,y) = \frac{1}{k} * c * e^{-(x*x+y*y)/2(\sigma*\sigma)} \quad \text{.....(2.13)}$$

Keterangan :
$g(x,y)$ = nilai piksel Gaussian filter pada kolom ke- x dan baris ke-y
c = kontstanta normalisasi
e = kontanta euler yang bernilai 2,718281828
σ = Standard deviasi
k = nilai total bobot kernel

Secara umum, nilai standard deviasi yang digunakan pada pengolahan citra digital adalah “1”, dan secara umum ukuran kernel yang digunakan

berukuran 3x3 atau 5x5. Berikut ini adalah contoh pola nilai-nilai piksel kernel berukuran 3x3 dan 5x5 dengan nilai standard deviasi = 1 :

1	2	1
2	3	2
1	2	1

a)

1	2	3	2	1
2	4	6	4	2
3	6	7	6	3
2	4	6	4	2
1	2	3	2	1

b)

Gambar 2.13. Contoh pola nilai kernel dengan σ=1. a) ukuran kernel 3x3.b) ukuran kernel 5x5

Dari contoh kernel ukuran 3x3 yang ditunjukkan gambar 2.13.a, jumlah nilai-nilai matriksnya (nilai total bobot kernel) adalah 15. Sedangkan kernel ukuran 5x5 yang ditunjukkan gambar 2.13.b, jumlah nilai-nilai matriksnya (nilai total bobot kernel) adalah 79.

c) Konvolusi

Piksel output dari *Gaussian blur* dihasilkan dari operasi konvolusi. Konvolusi merupakan operator sentral pengolah citra yang menggunakan kernel yang diletakkan pada citra input untuk menghasilkan nilai piksel baru. Nilai piksel output dihitung dengan mengalikan setiap piksel bobot pada kernel dengan setiap piksel input yang posisinya sama dengan piksel kernel yang bersangkutan dan menjumlah hasil perkalian tersebut. Secara matematika dapat ditulis dalam bentuk persamaan 2.6, dimana $f(x, y)$ adalah piksel input ,$g(x, y)$ adalah nilai Gaussian filter dan $h(x, y)$ adalah outputnya.

2.2.15. Raspberry pi 2

Raspberry pi 2 adalah komputer yang berukuran kecil (mini-komputer) yang bersifat *portable* atau mudah dibawa. Karena ukurannya yang sangat kecil, maka spesifikasi sistemnya (seperti prosesor, RAM, GPU, konsumsi daya dan lain-lain) juga relatif lebih kecil dibandingkan komputer *desktop* atau komputer yang mempunyai ukuran lebih besar. Secara umum mini komputer digunakan sebagai *embedded system* yaitu suatu sistem yang menjalankan fungsi-fungsi tertentu. Misalnya pada bidang robotika, mini komputer digunakan untuk mengolah gambar yang berasal dari kamera yang digunakan sebagai mata, sehingga robot tersebut mengetahui bahwa didepannya ada objek atau tidak.

Gambar 2.14. Raspberry Pi 2

Seperti yang ditunjukkan gambar 2.14 bahwa raspberry pi 2 juga memiliki I/O (seperti USB, HDMI, ethernet dll), hanya saja lebih terbatas dibandingkan PC/komputer. Begitu juga dengan OS (*Operating System*) yang digunakan. OS yang dapat digunakan untuk raspberry pi 2 antara lain raspbian, minibian, hotknot dan windows 10 (Raspberry Pi Foundation, 2015).

2.2.16. Realtime

Realtime merupakan suatu keadaan komputasi / proses yang dilakukan oleh suatu sistem dengan batas maksimal waktu proses yang sudah ditentukan (Mall R.,2007). Misalnya, sistem mendeteksi gerakan benda dari kamera digital dengan *framerate* 30fps atau dengan *frametime* 33,33ms. Maka sistem dikatakan bekerja secara realtime apabila dapat mendeteksi gerakan benda dengan waktu proses (*delay*) kurang dari 33,33 ms. Dengan kata lain, sistem harus selesai melakukan proses pendeteksian gerakan benda sebelum datang *frame* berikutnya dari kamera. Jika waktu *delay* lebih dari 33,33 ms tersbut, maka sistem tersebut belum bisa dikatakan berjalan secara *realtime.*

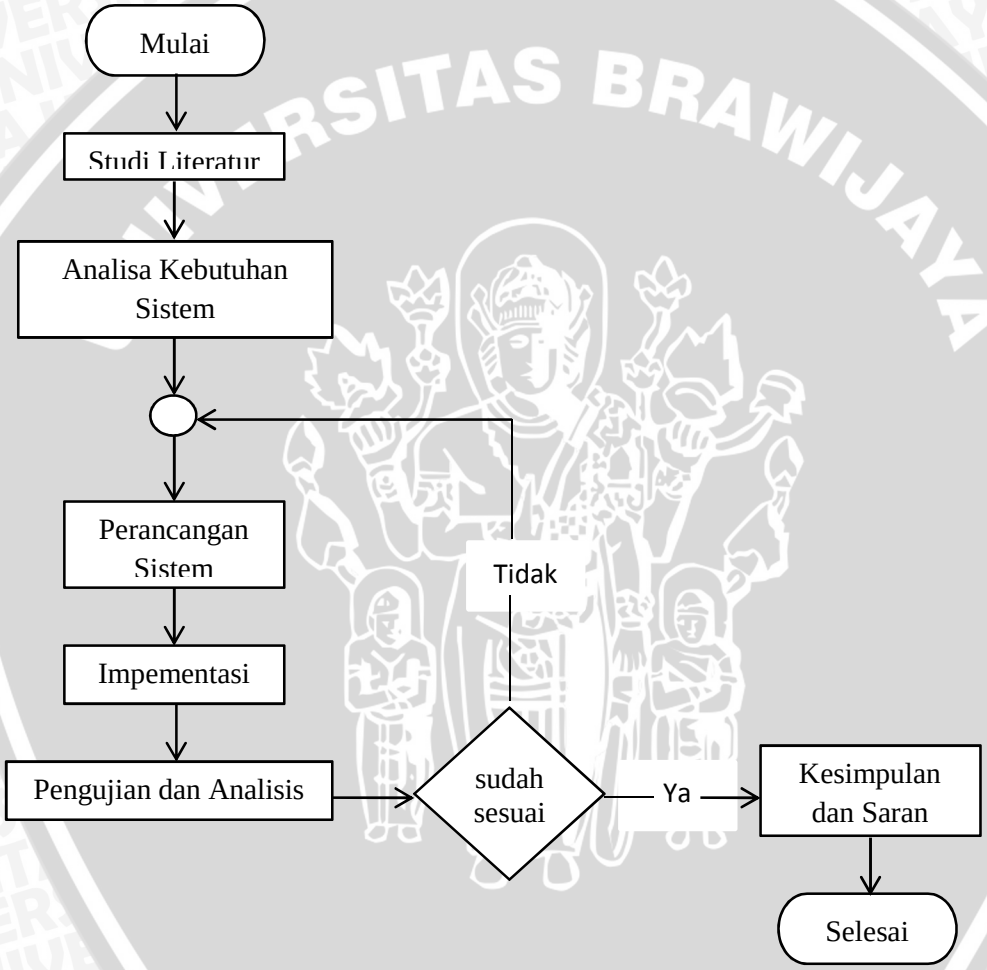
Suatu sistem yang akan diimplementasikan langsung di lapangan harus bisa berjalan secara *realtime*. Jika belum bisa berjalan secara *realtime* maka sistem tersebut masih dalam proses pengembangan atau simulasi atau perancangan.

BAB 3 METODOLOGI

Metodologi berisi tentang penjelasan mengenai metode atau langkah-langkah yang tersusun secara sistematis dalam merancang suatu sistem untuk menyelesaikan suatu masalah.

3.1. Urutan Metode Penelitian

Urutan / langkah-langkah metode penelitian secara umum dapat dilihat pada gambar 3.1 berikut :



Gambar 3.1 Diagram Alir Metode Penelitian

3.2. Studi Literatur

Studi literarur berisi tentang seluruh teori yang dibutuhkan yang mendukung dalam penelitian ini. Ada 3 hal pokok dalam studi literatur, antara lain :

a) Latar Belakang Masalah

Latar belakang masalah merupakan suatu proses identifikasi masalah tertentu seperti mencari suatu permasalahan yang berkembang di masyarakat dan menganalisanya. Bagian ini adalah hal yang paling fundamental dalam memulai pembuatan suatu sistem. Karena, hal ini yang menentukan seperti apa suatu akan dibuat dan untuk apa sistem akan dibuat. Proses identifikasi masalah diperoleh dari situs pemerintah dan berita melalui internet.

b) Tinjauan Pustaka

Tinjauan pustaka berisi tentang penelitian-penelitian sebelumnya mengenai studi kasus yang sedang dipelajari. Bagian ini bertujuan untuk menjadikan penelitian-penelitian tersebut sebagai referensi dalam pembuatan sistem ini. Tinjauan pustaka diperoleh dari jurnal internasional, jurnal nasional yang sudah di seminarkan melalui internet dan penelitian sebelumnya yang sudah dipublikasikan.

c) Dasar Teori

Dasar teori merupakan sekumpulan teori-teori dasar yang mendukung dalam penelitian ini. Dasar teori diperoleh dari buku, jurnal lokal dan internasional melalui internet.

3.3. Analisa Kebutuhan Sistem (ABS)

ABS adalah proses identifikasi kebutuhan peralatan apa saja yang dibutuhkan untuk membangun sistem. Proses identifikasi kebutuhan peralatan meliputi kebutuhan perangkat keras dan perangkat lunak.

3.4. Perancangan Sistem

Perancangan sistem berisi tentang pembuatan alur kinerja sistem secara menyeluruh baik bagian perangkat keras yang akan ditulis dalam bentuk diagram blok maupun perangkat lunak yang akan ditulis dalam bentuk flow-chart.

3.5. Implementasi Sistem

Implementasi sistem berisi tentang penerapan dari setiap perancangan yang sudah dibuat sebelumnya baik perancangan perangkat keras maupun perancangan perangkat lunak.

3.6. Pengujian Sistem

Pengujian sistem bertujuan untuk mengetahui seberapa baik sistem dapat berjalan, seberapa efektif dan seberapa efisien kinerja sistem yang sudah dibuat. Lebih detailnya pengujian ini meliputi :

1) Pengujian Integrasi PC dengan mini raspberry pi
2) Pengujian integrasi raspberry dengan *library* OpenCV
3) Menguji akurasi sistem dalam mendeteksi kendaraan
   Akurasi yang diuji meliputi :
   - Akurasi mendeteksi kendaraan yang melintas berdasarkan jenis kendaraan dengan kecepatan kendaraan yang sebenarnya, 2 kali dan 4 kali lebih cepat dari kecepatan yang sebenarnya.
   - Akurasi mendeteksi kendaraan yang melintas berdasarkan jumlahnya dengan kecepatan kendaraan yang sebenarnya, 2 kali dan 4 kali lebih cepat dari kecepatan yang sebenarnya.
4) Menguji dan membandingkan performa proses pengolahan gambar pada raspberry pi 2 dan komputer. resolusi gambar yang digunakan adalah 320 x 240 piksel.

3.7. Penutup

Kesimpulan dilakukan setelah melakukan tahap perancangan, implemenentasi, pengujian dan analisis sistem. Kesimpulan merupakan uraian hasil pengujian dan analisis yang berdasarkan rumusan masalah yang telah dibuat.

Sedangkan saran berisi tentang rekomendasi untuk memperbaiki kelemahan sistem ini dan mengembangkannya baik perangkat lunak dan atau perangkat keras agar sistem lebih sempurna.

BAB 4 PERANCANGAN DAN IMPLEMENTASI

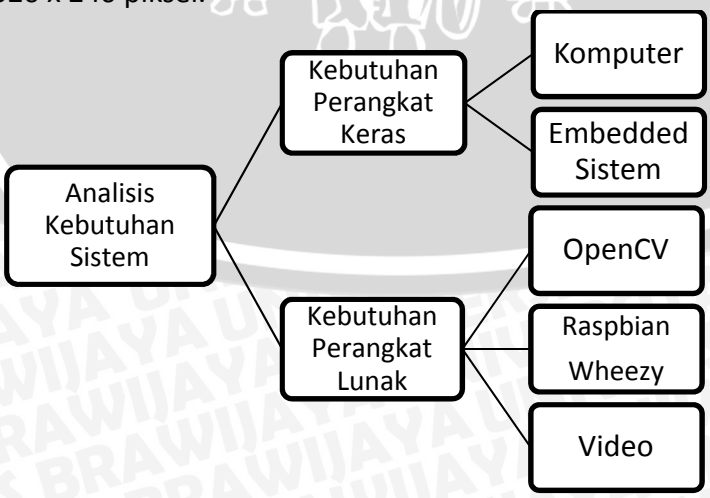
4.1. Perancangan

Perancangan berisi tentang pembuatan alur kinerja sistem baik secara keseluruhan. Bagian ini meliputi analisa kebutuhan sistem, diagram blok sistem untuk perancangan perangkat keras dan algoritma sistem untuk perancangan perangkat lunak.

4.1.1. Analisa Kebutuhan Sistem

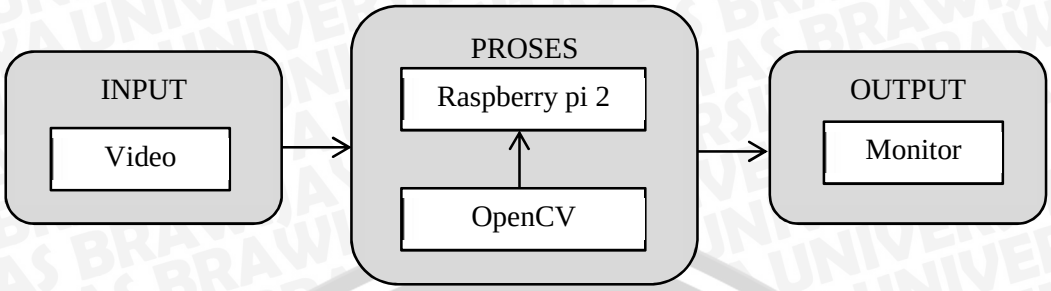
Bagian ini berisi penjelasan mengenai komponen-komponen apa saja yang dibutuhkan untuk merancang sistem. Analisa ini terdiri dari 2 bagian yaitu analisa kebutuhan perangkat keras dan analisa kebutuhan perangkat lunak, antara lain :

a) PC (personal computer)
   PC digunakan untuk menunjang proses pembuatan sistem, yaitu proses instalasi OS (*Operating System*) untuk mini komputer dan untuk memonitor aktivitas mini komputer.
b) Raspberry pi 2
   Mini komputer ini digunakan untuk mengolah citra digital.
c) OpenCV
   perangkat lunak ini dimasukkan kedalam raspberry pi 2 model B untuk mengolah setiap *frame* gambar yang diinputkan. Versi OpenCV yang digunakan dalam penelitian ini adalah OpenCV 2.4.10.
d) Raspbian-wheezy
   Raspbian-wheezy merupakan OS (*Operating System*) yang dirancang khusus untuk raspberry pi 2. OS ini diperlukan dalam membuat sistem karena mempunyai GUI (*Graphic User Interface*) dibandingkan OS lain yang tidak memiliki GUI.
e) Video
   Input untuk sistem ini adalah berupa gambar bergerak / video dengan resolusi 320 x 240 piksel.



Gambar 4.1. Analisis Kebutuhan Sistem

4.1.2. Diagram Blok Sistem



Gambar 4.2. Diagram Blok Sistem

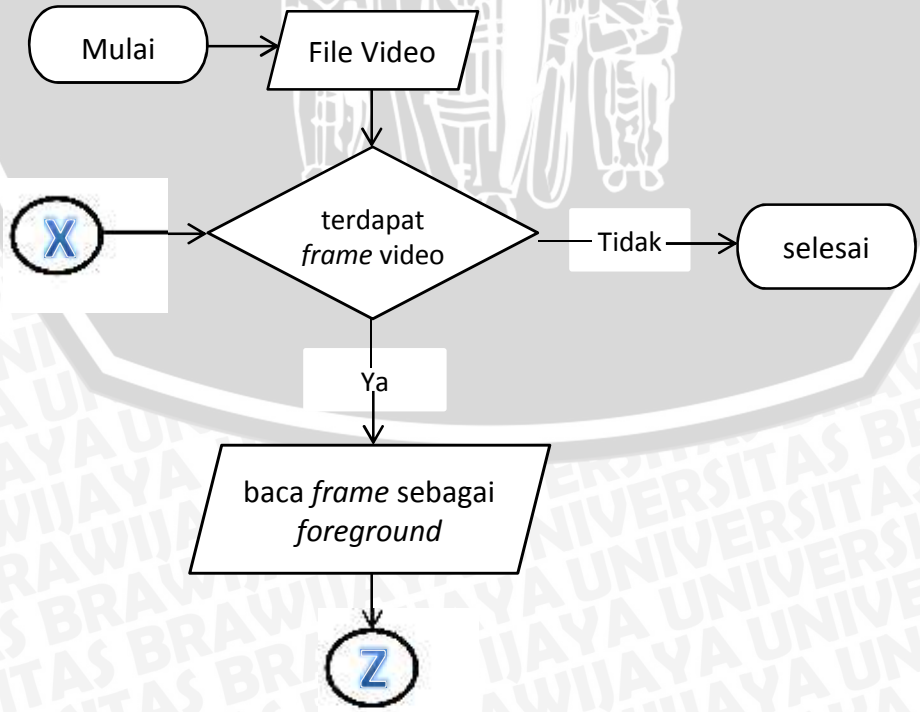
Pada gambar 4.2 input adalah video hasil rekaman kamera *handphone* cross A26 dengan *frame rate* 29 fps. Resolusi video yang digunakan adalah 320x240 piksel. Video dimasukkan kedalam *secondary storage* raspberry pi 2 untuk diolah menggunakan *library* OpenCV. Lalu, hasil olahannya ditampilkan di monitor komputer.

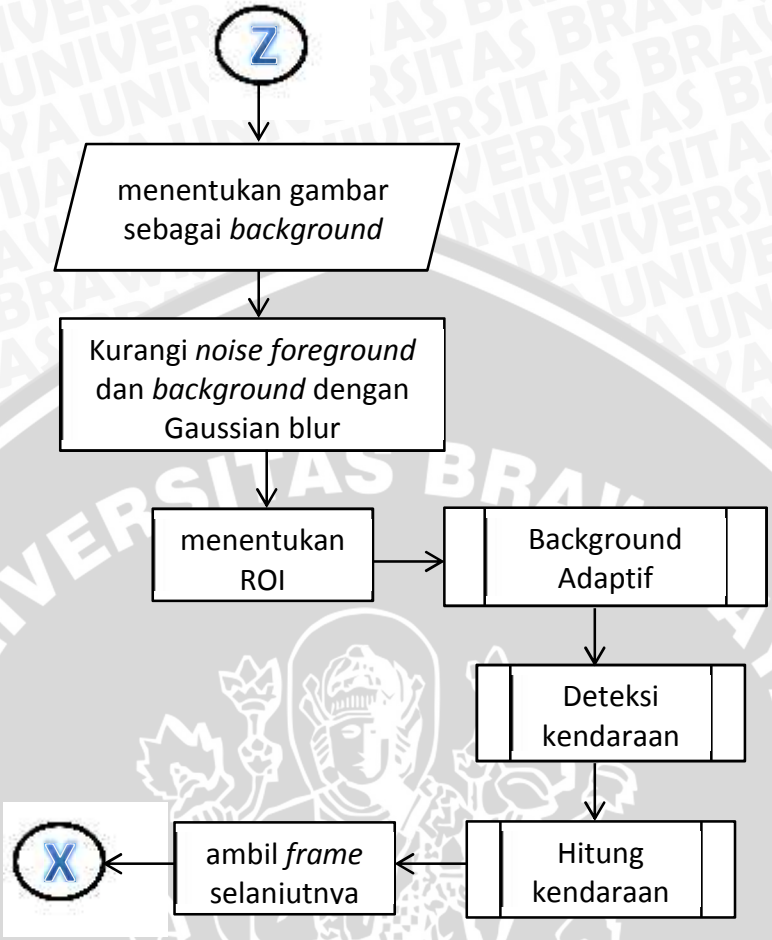
4.1.3. Perancangan Algoritma Sistem

Bagian ini menjelaskan tentang perancangan algoritma *source code* yang akan digunakan untuk mengolah gambar. Bahasa pemrograman yang digunakan adalah bahasa C++.

4.1.3.1 Perancangan Algoritma Sistem Secara Umum

Untuk mempermudah dalam proses perancangan algoritma sistem, maka langkah awal yang dilakukan adalah membuat diagram alir / algoritma sistem secara umum seperti yang ditunjukkan pada gambar 4.3 dibawah ini:



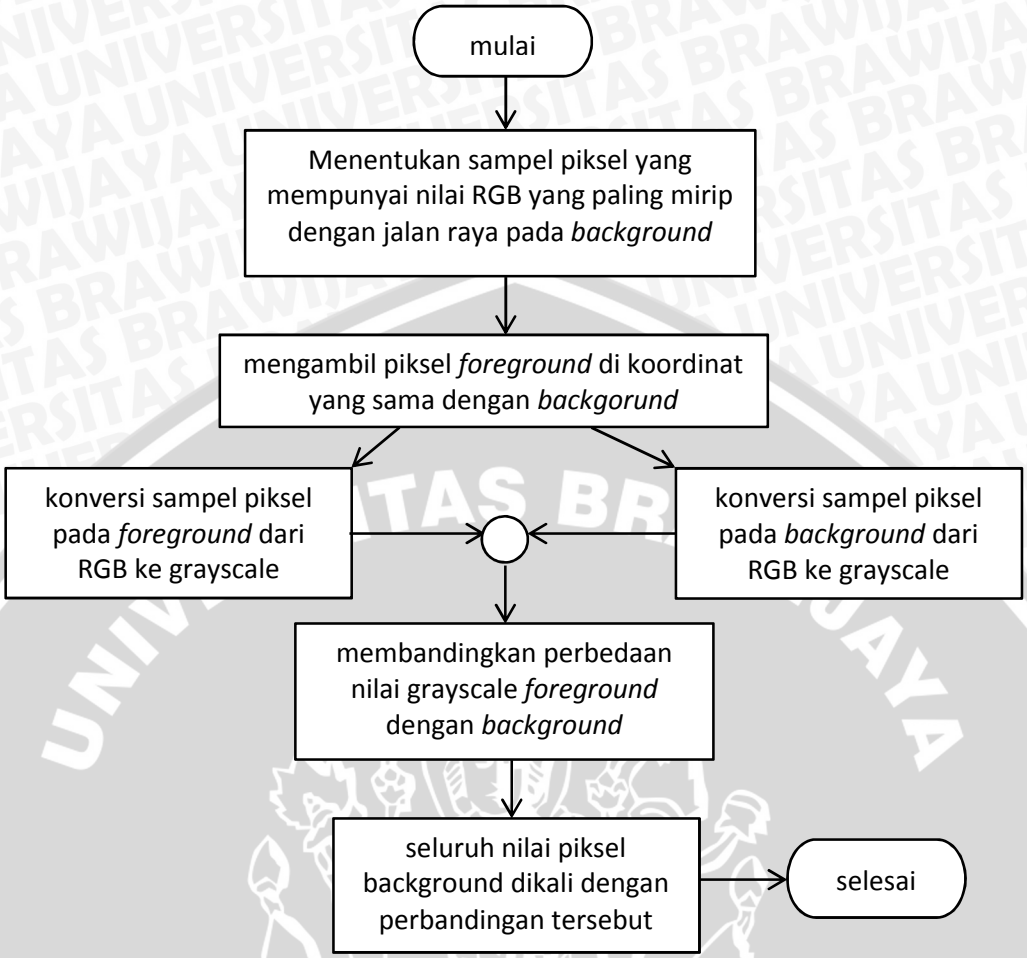


Gambar 4.3. diagram alir program secara umum

Mula-mula mengambil setiap *frame* yang ada pada video sebagai *foreground*, sedangkan gambar *background* sudah ditentukan sebelumnya. Kemudian, menghaluskan gambar *foreground* dan *background* dengan menggunakan *Gasussian Blur* yang bertujuan untuk mengurangi jumlah region *noise.* Setelah itu, menentukan ROI (*Region Of Interest*). ROI adalah suatu daerah / area tertentu yang diambil dari gambar tertentu untuk diproses. Sehingga, proses komputasinya lebih ringan karena tidak perlu seluruh *frame* gambar yang diolah, tapi hanya daerah tertentu yang diolah. Setelah itu, masuk ke subproses *Background Adaptif* agar cahaya pada Background dapat menyesuaikan dengan Foreground. Setelah itu, masuk ke proses deteksi kendaraan menggunakan metode *background substraction* dan kemudian masuk ke subproses hitung kendaraan yang terdeteksi.

4.1.3.2 Algoritma Background Adaptif

Langkah ini adalah proses menyesuaikan *brightness* dari gambar *background* dengan *brightness* dari gambar *foreground*.



Gambar 4.4. Diagram Alir menentukan nilai *background*

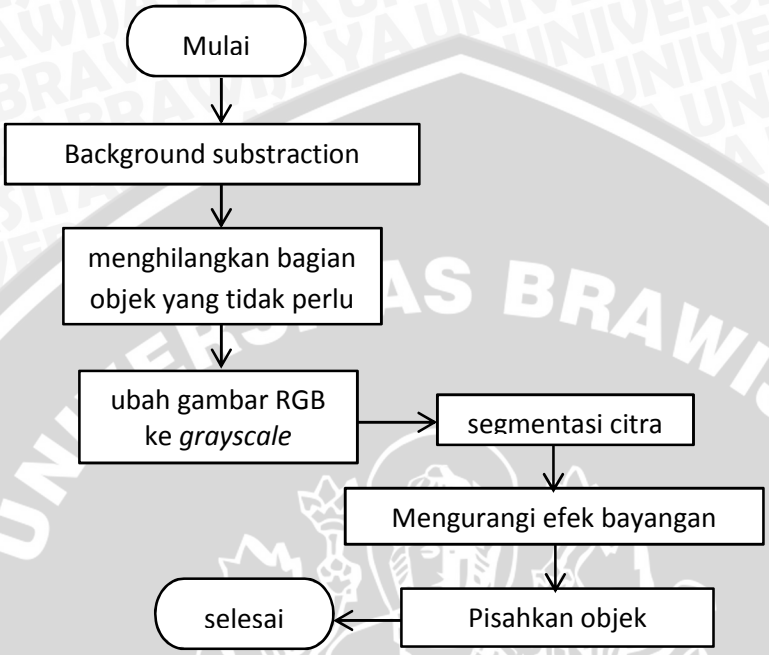
Tujuan menentukan nilai *background* yang bersifat adaptif ini adalah mengurangi gangguan yang diakibatkan perubahan cahaya (misalnya meredup). Hal ini meyangkut metode background substraction sangat sensitif terhadap perubahan pencahayaan. Cara menentukan nilai background yaitu menentukan sampel piksel dengan nilai RGB yang paling mirip dengan jalan raya pada *background*. mengambil piksel paling mirip dengan ROI bertujuan untuk mendapatkan nilai perbandingan yang paling sesuai dengan perubahan nilai intenitas cahaya pada *foreground*, sehingga sistem tetap dapat mendeteksi ada tidaknya kendaraan yang melintas. rumus matematis ditunjukkan oleh persamaan 4.1 berikut :

$$background_{(y)} = background_{(i)} \times \frac{h1}{h0} \quad \text{.....(4.1)}$$

Keterangan :
$background_{(i)}$ = background awal
$background_{(y)}$ = background baru
h1 = nilai *grayscale* piksel sampel *foreground*
h0 = nilai *grayscale* piksel sampel *background*

4.1.3.3 Algoritma Deteksi Objek

Setelah merancang diagram alir / algoritma sistem secara umum, langkah selanjutnya adalah merancang diagram alir subproses mendeteksi kendaraan. Diagram alir mendeteksi kendaraan ditunjukkan pada gambar 4.4 dibawah ini :

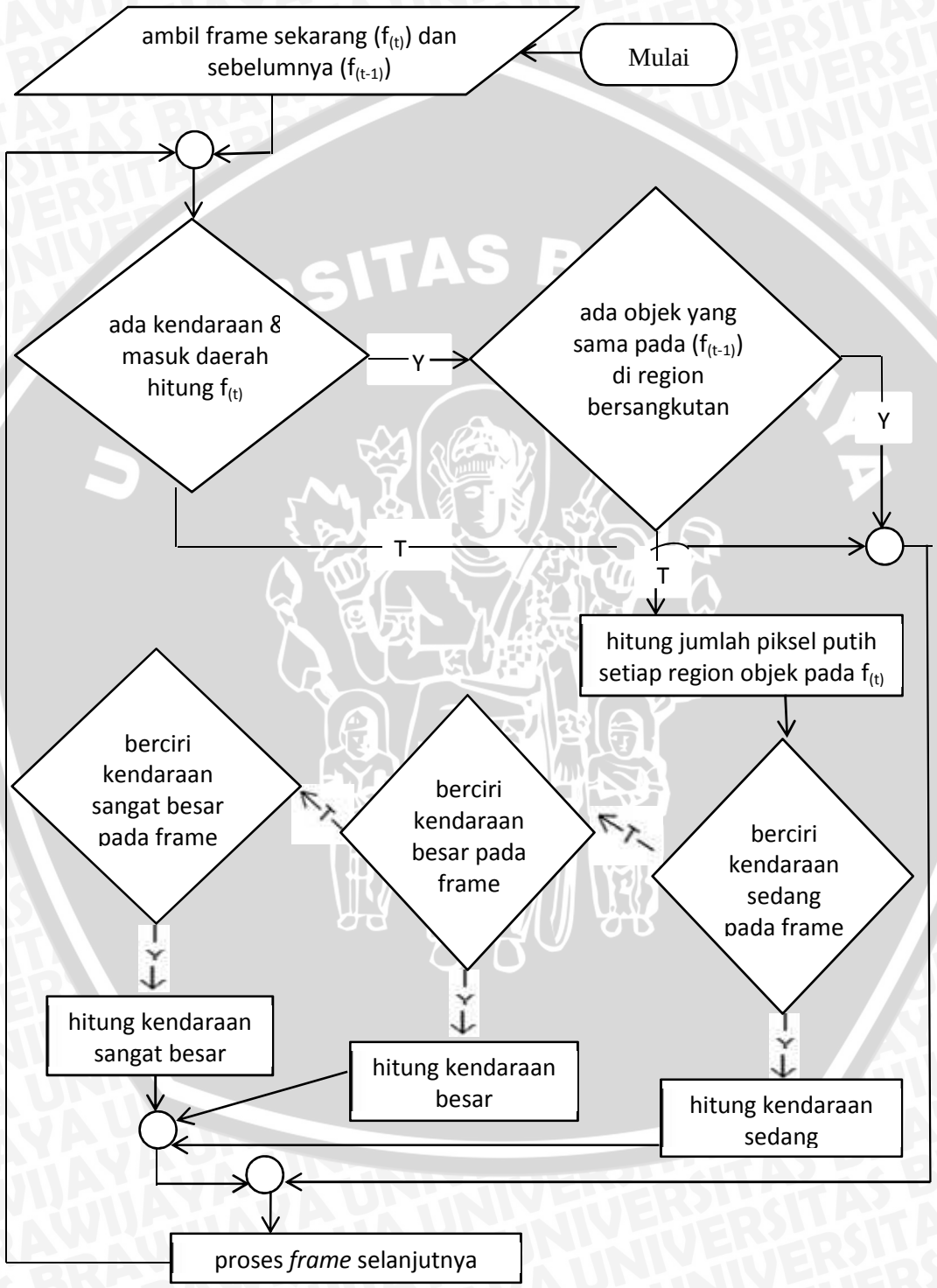


Gambar 4.5. Diagram Alir Mendeteksi Kendaraan

Metode yang digunakan dalam untuk mendeteksi objek adalah background substraction yaitu melakukan operasi pengurangan antara *foreground* dan *background.* setelah itu, menghilangkan bagian objek yang tidak pelu yaitu selain area jalan raya yang dijadikan objek penelitian. Hal ini bertujuan untuk mengurangi beban komputer selama proses pengolahan citra. Langkah selanjutnya adalah mengkonversi citra warna ke citra abu-abu (grayscale) menggunakan rumus 2.2. setelah itu, melakukan *thresholding* / segmentasi ke dalam bentuk citra biner. Proses segmentasi citra dalam hal ini, bertujuan untuk memisahkan objek (yaitu kendaraan) dengan latar belakangnya, dimana objek kendaraan akan diubah niali pikselnya dengan "255" sedangkan latar belakangnya diubah niali pikselnya dengan "0". Langkah berikutnya adalah mengurangi efek bayangan. Hal ini berkaitan dengan hanya ukuran / luas kendaraan yang dihitung, tidak termasuk bayangan. Berdasarkan analisa, rata-rata nilai RGB bayangan kendaraan adalah antara (0,0,0) hingga (25,15,15). Akan tetapi, bagian tepi bayangan niali RGB-nya lebih dari (25,15,15). Oleh karena itu, dilakukan proses erosi dan dilasi untuk menghilangkan objek bayangan tepi tersebut. Langkah selanjutnya, adalah melakukan erosi dan dilasi yang khusus digunakan mengisolasi / memisahkan objek yang bersambungan dan sekaligus menghilangkan noise.

4.1.3.4 Algoritma Hitung Kendaraan

Bagian ini bertujuan menghitung kendaraan dan mengklasifikasi apakah kendaraan tersebut masuk kategori kendaraan sedang, besar atau sangat besar. Urutannya ditunjukkan pada gambar 4.6 dibawah ini :



Gambar 4.6. Diagram Alir Hitung Kendaraan

Dalam subproses menghitung kendaraan, mula-mula deklarasi gambar yang sekarang ditangkap oleh sistem ($frame_{(t)}$) dan gambar tepat sebelum yang sekarang ditangkap oleh sistem ($frame_{(t-1)}$) ketika sistem pertama kali dijalankan. Sistem menangkap gambar pertama pada video ($frame_{(t)}$), lalu memeriksa apakah ada objek kendaraan dan masuk daerah hitung. Objek kendaraan dikenali melalui panjang dan lebar objek seperti yang ditunjukkan tabel 4.5, sedangkan daerah hitung ditunjukkan gambar 4.11. jika syarat terpenuhi tidak terpenuhi maka langsung proses *frame* selanjutnya pada video, sedangkan jika syarat terpenuhi maka akan masuk tahap selanjutnya yaitu memeriksa apakah terdapat objek yang sama pada $frame_{(t-1)}$ di region bersangkutan. Jika iya, maka tidak tidak akan dilakukan proses perhitungan kendaraan, sedangkan jika tidak maka akan dilakukan proses perhitungan kendaraan berdasarkan ukurannya yaitu kendaraan sedang (sejenis mobil), kendaraan besar (sejenis truk) atau kendaraan sangat besar (sejenis bus). Proses ini bertujuan agar kendaraan yang sama tidak dihitung kembali pada *frame* berikutnya. Setelah itu, *frame* yang sekarang ($frame_{(t)}$) dimasukkan ke $frame_{(t-1)}$ dan $frame_{(t)}$ mengambil data gambar selanjutnya dari video yang diolah dan kembali memroses langkah-langkah hitung kendaraan seperti yang ditunjukkan gambar 4.6.

4.2. Implementasi

Bagian ini membahas tentang implementasi sistem berdasarkan perancangan dan analisa kebutuhan sebelumnya. Pembahasan meliputi spesifikasi sistem, batasan-batasan implementasi, implementasi pengambilan video, implementasi integrasi mini PC dengan monitor, implementasi instalasi OpenCV pada mini PC, impementasi algoritma program.

4.2.1. Spesifikasi Sistem

spesifikasi sistem menjelaskan tentang rincian / spesifikasi setiap perangkat yang digunakan. Bagian ini terdiri dari 2 bagian yaitu spesifikasi perangkat keras dan spesifikasi perangkat lunak. Rincian spesifikasi perangkat keras terdiri dari komputer yang ditunjukkan tabel 4.1 dan mini komputer yang ditunjukkan tabel 4.2. Sedangkan rincian perangkat lunak terdiri dari video yang ditunjukkan tabel 4.3 dan in-raspberry (yang terdapat pada *micro SD* di *raspberry*) yang ditunjukkan tabel 4.4.

1. Perangkat keras
   a) Komputer

Tabel 4.1. Spesifikasi Komputer

System Model	AXIOO HNM MB40114
Processor	Intel Core i3-2310M CPU @2.10GHz
Memory	2048 MB RAM
Display	Intel® HD Graphic Family 816 MB
Harddisk	500 GB

b) Mini Komputer

Tabel 4.2. Spesifikasi raspberry pi 2

System Model	Raspberry Pi 2 model B
Processor	900 Mhz quad-core ARM Cortex-A7 CPU
Memory	1024 MB RAM
Display	VideoCore IV 3D graphics core
Storage	Micro SD vgen 16 GB
Expansion Port	4 USB, 40GPIO pins, HDMI, Ethernet, 4pole stereo output and composite video, CSI camera and DSI display, micro USB *power source*

2. Perangkat Lunak

a) Video

Tabel 4.3. Spesifikasi Video

Type File	AVI
Resolution	320 x 240
Refresh rate	29 fps

b) In-raspberry pi 2

Table 4.4 Spesifikasi Perangkat Lunak Pendukung

Operating System	Raspbian
Programming Tool	OpenCV 2.4.10

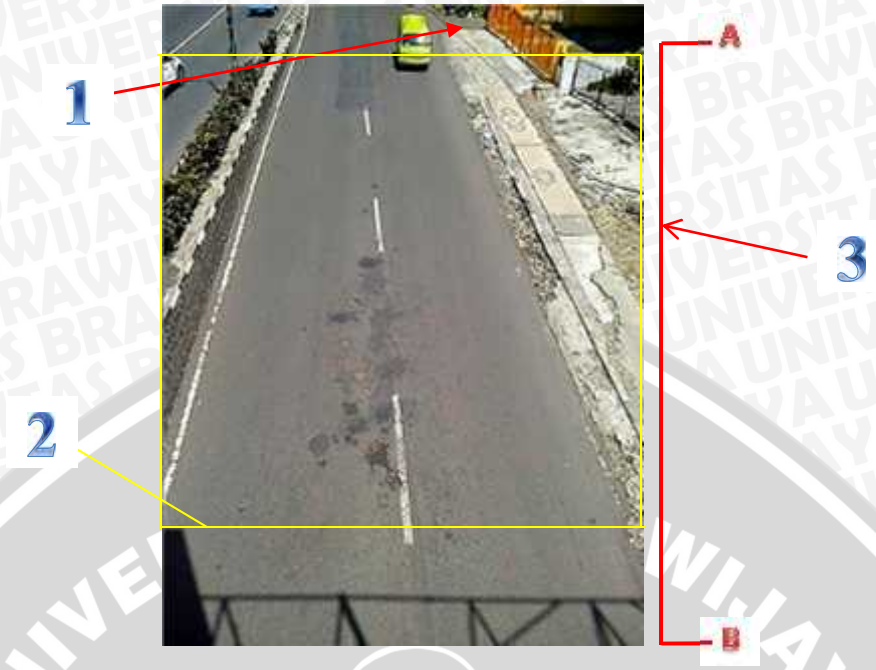
4.2.2. Batasan-batasan Implementasi

Ada beberapa batas-batasan dalam implementasi sistem ini, antara lain :

1. Implementasi dilakukan di siang hari yang cukup cahaya dari kamera *Hand Phone* di daerah Arjosari Malang pada pukul 11.20 WIB.
2. Aplikasi yang digunakan untuk mengolah gambar adalah OpenCV 2.4.10 dan bahasa pemrograman C++.
3. Objek yang diamati adalah kendaraan roda 3 atau di atasnya.
4. Terdapat area khusus sampel piksel yang digunakan untuk mengimplementasikan adaptif *background*

4.2.1. Pengambilan Data, Sampel Piksel dan ROI

Pengambilan data berupa video diambil di jalan raya menggnakan kamera handphone cross A26. Pengambilan video dilakukan di siang hari. Cara pengambilan video, sampel piksel dan ROI ditunjukkan pada gambar 4.7 dibawah ini :



Gambar 4.7. Menentukan Sample Piksel dan ROI

Keterangan :
1. sampel piksel
2. ROI (*Region Of Interest*)
3. Jarak AB adalah 40 Meter (sudah diinisialisasi)

Sampel piksel pada gambar 4.7 diasumsikan tidak ada objek yang melintas di area tersebut seperti yang sudah dijelaskan pada batasan penelitian. Objek yang dijadikan sampel diletakkan pada koordinat (155,11), karena di area tersebut mempunyai jenis warna RGB yang sama dengan jalan utama yang dijadikan objek penelitian seperti di keterangan 2. Sehingga, jika terjadi perubahan cahaya pada area tersebut maka, jalan utama juga akan mengalami perubahan nilai RGB dengan perbandingan yang relatif sama dengan nilai piksel yang dijadikan sampel.

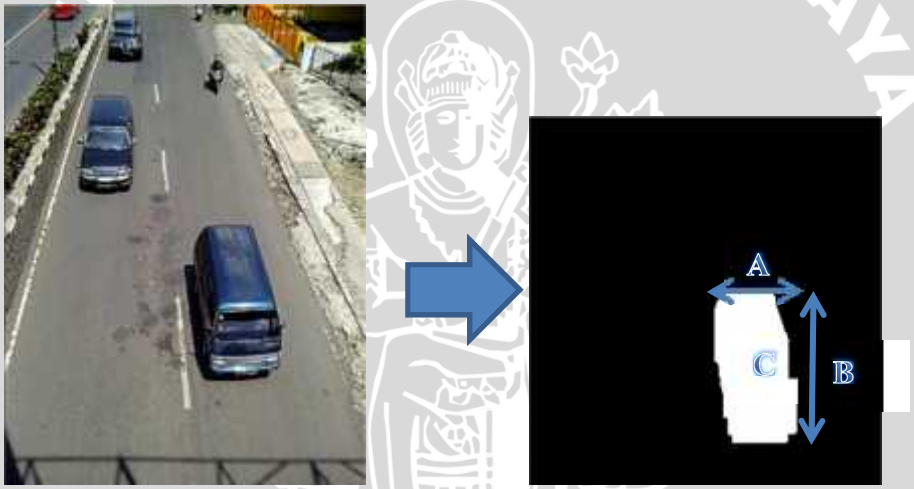
Sedangkan ROI yang diambil adalah baris ke-21 hingga baris ke-271 dan kolom ke-1 hingga kolom ke-240. Sehingga, resolusi ROI adalah 240x250. Pada resolusi tersebut, kendaraan dengan kategori sangat besar pun akan terlihat sepenuhnya. Sedangkan jarak AB dihitung secara manual.

4.2.4. Klasifikasikan kendaraan berdasarkan ukuran

Dalam penelitian ini kendaraan yang dijadikan objek dibagi menjadi 3 jenis, yaitu kendaraan dengan ukuran sedang, besar dan sangat besar. Kendaraan ukuran sedang yang dimaksud adalah mobil atau sejenisnya, ukuran besar adalah truk atau sejenisnya dan ukuran sangat besar adalah bus atau sejenisnya. Gambar 4.9 berikut ini adalah contoh gambar kendaraan berdasarkan jenisnya pada resolusi 320x240 piksel :

a) **b)** **c)**

Gambar 4.8. a) contoh kendaraan ukuran sedang, b) contoh ukuran besar dan c) contoh kendaraan ukuran sangat besar.



Gambar 4.9. Pengukuran kendaraan

Keterangan :
A : lebar kendaraan
B : Panjang Kendaraan
C : ukuran kendaraan (jumlah piksel putih)

Sedangkan luas / jumlah piksel masing-masing jenis kendaraan pada resolusi 320x240 piksel ditunjukkan oleh tabel 4.5 dibawah ini :

Tabel 4.5 Klasifikasi Jenis Kendaraan

Jenis kendaraan	Ukuran (piksel)	Panjang	Lebar
Sedang	3000 – 8400	75-143	50-72
	9000-10000*	130	96*
Besar	8400-13000	138-180	69-78
Sangat Besar	> 13000	>=200	>=90

Keterangan :
* : Ketika terdapat gangguan pencahayaan (pantulan cahaya matahari).

Berdasarkan tabel 4.5 gangguan pantulan cahaya matahari hanya terjadi pada kendaraan sedang / mobil. hal ini disebabkan karena kaca bagian depan mobil menghadap keatas, sehingga cenderung mudah terjadi efek refleksi cahaya matahari. Gangguan ini menyebabkan terjadinya perubahan ukuran kendaraan baik luas maupun lebar kendaraan.

4.2.5. Konfigurasi Kecepatan Kendaraan

Konfigurasi kecepatan kendaraan bertujuan untuk mensimulasikan jika terdapat kendaraan yang melaju dengan kecepatan tinggi dan mengetahui pengaruh kecepatan kendaraan terhadap akurasi sistem dalam mendeteksi kendaraan. Kecepatan kendaraan yang diuji meliputi kecepatan kendaraan yang sebenarnya, kecepatan kendaraan 2 kali lipat dan kecepatan kendaraan 4 kali lipat. Berikut ini adalah penjelasannya :

1. Kecepatan kendaraan sebenarnya
   kecepatan kendaraan sebenarnya artinya sistem mengolah setiap *frame* yang ada pada video secara berurutan dari *frame* pertama hingga *frame* terakhir. Sehingga, kecepatan kendaraan yang diolah sistem adalah kecepatan kendaraan yang sebenarnya.
2. Kecepatan kendaraan 2 kali lipat
   Kecpatan kendaraan 2 kali lipat artinya sistem mengolah *frame* video setiap kelipatan 2 secara berurutan dari awal hingga akhir yaitu *frame* ke-2,4,6,8,10 dan seterusnya, sedangkan *frame* yang tidak oleh sistem adalah *frame* ke-1,3,5,7,9 dan seterusnya. Sehingga, objek dapat dianggap bergerak dengan kecepatan 2 kali lipat.
3. Kecepatan kendarann 4 kali lipat
   Kecepatan Kecepatan kendaraan 4 kali lipat artinya sistem mengolah frame video setiap kelipatan 4 secara berurutan dari awal hingga akhir, yaitu pada *frame* ke-4,8,12,16,20 dan seterusnya, sedangkan frame selain kelipatan 4 tidak diolah. Sehingga, objek dapat dianggap bergerak dengan kecepatan 4 kali lipat.

Sedangkan untuk kecepatan kendaraan itu sendiri dihitung dari rumus :

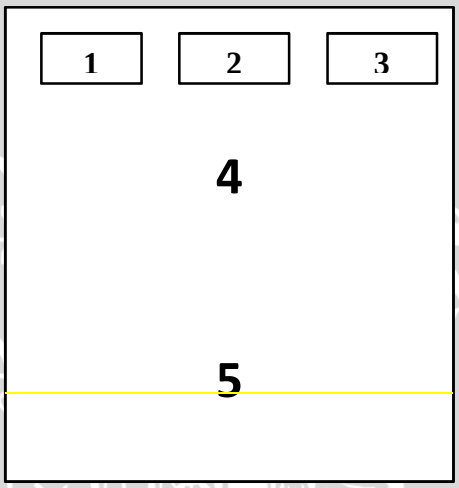
$$v = \frac{s}{t} \quad \text{.....(4.1)}$$

Keterangan :
v = kecepatan kendaraan
s = jarak yang ditempuh kendaraan
t = waktu yang dibutuhkan oleh kendaraan untuk menempuh suatu jarak

jarak yang ditempuh kendaraan sudah diinisialisasi dari baris A ke B adalah 40 meter seperti yang ditunjukkan gambar 4.7. Sedangkan kendaraan yang ada pada video membutuhkan waktu antara 2,5-5 detik, sehingga dengan menggunakan rumus 4.1 dapat dihitung kecepatan rata-rata kendaraan yaitu berkisar antara 29-60 km/jam. Jadi, pada konfigrasi kecepatan kendaraan 4 kali lipat, dapat dianggap bahwa kendaraan melaju dengan kecepatan 116-240 km/jam.

4.2.6. Perancangan Antar Muka

Perancangan antar muka diperlukan untuk mempermudah *user* mengenai hal-hal yang disampaikan oleh sistem, dalam hal ini adalah mengenai mampu tidaknya sistem dalam mendeteksi kendaraan saat mlintas, mengetahui jumlah kendaraan yang telah melintas saat ini dan posisi pendeteksian kendaraan. Gambar 4.10 dibawah ini adalah ilustrasi antar muka sistem:



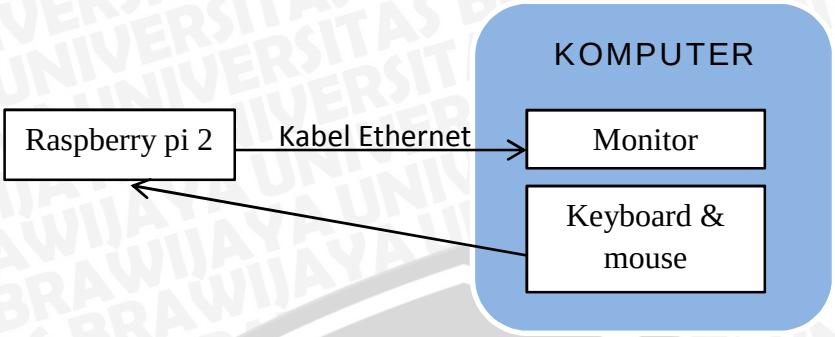
Gambar 4.10. Desain Antar Muka

Keterangan :

1. Informasi jumlah kendaraan ukuran sedang
2. Jumlah kendaraan ukuran besar
3. Jumlah kendaraan ukuran sangat besar
4. Gambar video
5. Garis penghitung pada posisi y = 271

4.2.7. Integrasi Raspberry pi 2 Dengan Komputer

Bagian ini berisi tentang cara menghubungkan raspberry pi 2 dengan PC / komputer. bagan integrasi komputer dengan raspberry pi 2 ditunjukkan gambar 4.12 berikut ini :



Gambar 4.11. Integrasi raspberry pi 2 Dengan Komputer

Menghubungkan raspberry pi 2 dengan komputer terdiri dari 2 bagian, yaitu me-remote raspberry pi 2 menggunakan input dari komputer dan menampilkan aktivitas raspberry pi 2 di monitor komputer seperti yang ditunjukkan gamba 4.11, raspberry pi 2 dihubungkan ke PC/komputer menggunakan kabel Ethernet.

4.2.8. Implementasi Algoritma

Setelah merancang algoritma dalam bentuk flowchart, maka langkah selanjutnya adalah diimplementasikan dalam bentuk code. Program mempunyai 2 bagian yaitu bagian inisialisasi dan bagian utama program. Bagian inisialisasi meliputi baca file video, menentukan background gambar, haluskan gambar dan ROI. Sedangkan bagian utama dalam algoritma deteksi kendaraan adalah background adaptif, deteksi kendaraan dan hitung kendaraan. bahasa pemrograman yang digunakan adalah C++.

4.2.8.1. Algoritma Background Adaptif

```
Mat hasil_0 = back(Rect(154,10,1,1));
Mat hasil_1= fore(Rect(154,10,1,1));
cvtColor(hasil_0,hasil_0,COLOR_BGR2GRAY);
cvtColor(hasil_1,hasil_1,COLOR_BGR2GRAY);
double h_0 = hasil_0.at<uchar>(0,0);
double h_1 = hasil_1.at<uchar>(0,0);
roi_back = roi_back * (h_1/h_0);
```

Gambar 4.12. Algoritma Background Adaptif

Koordinat sampel piksel adalah (155,11) untuk *background* sedangkan baris kedua untuk *foreground*. Kemudian, masing-masing piksel tersebut di-*grayscale*, lalu hasil perbandingan nilainya dikalikan dengan nilai matriks *background* yang lama dan menghasilkan nilai *background* yang baru.

4.2.8.2. Algoritma Deteksi Kendaraan

```
Point pointer_1[1][3];
Point pointer_2[1][3];
```

```
pointer_1[0][0] = Point(150,0);
pointer_1[0][1] = Point(238,219);
pointer_1[0][2] = Point(238,0);

const Point* ppt[1] = { pointer_1[0] };
int npt[] = { 3 };

pointer_2[0][0] = Point(62,0);
pointer_2[0][1] = Point(0,152);
pointer_2[0][2] = Point(0,0);

const Point* pps[1] = { pointer_2[0] };
```

a)

```
Mat cc; inRange(roi_fore,Scalar(0,0,0),Scalar(25,15,15),cc);
roi_back = roi_back * (h_1/h_0);
 hasil = abs(roi_fore-roi_back);
 fillPoly( hasil, ppt, npt, 1, Scalar( 0, 0, 0 ), 8 );
 fillPoly( hasil, pps, npt, 1, Scalar( 0, 0, 0 ), 8 );
  cvtColor(hasil,hasil,COLOR_BGR2GRAY);
threshold( hasil, hasil, 14, 255, THRESH_BINARY);
hasil = hasil -cc;
erode( hasil, hasil, getStructuringElement(MORPH_RECT,
Size(4, 1)));
dilate( hasil, hasil, getStructuringElement(MORPH_RECT,
Size(6, 5)));
findContours(hasil.clone(), contours, CV_RETR_EXTERNAL,
CV_CHAIN_APPROX_NONE);
Mat coba = Mat::zeros(250,239,CV_8UC1);
Mat coba2 = Mat::zeros(250,239,CV_8UC1);
for(int z = 0; z < contours.size();z++){
      int q = contourArea(contours[z]);
      if (q>2000){
      drawContours(coba, contours, z,Scalar(255),
CV_FILLED);}}
coba2 = coba.clone();
erode( coba2, coba2, getStructuringElement(MORPH_RECT,
Size(35, 35)) );
dilate( coba2, coba2, getStructuringElement(MORPH_RECT,
Size(35, 35)) );
```

b)

Gambar 4.13. a) Menentukan Koordinat yang tidak penting. b) Algoritma Deteksi Kendaraan

Dalam menghilangkan bagian gambar yang tidak perlu, mula-mula menentukan koordinatnya terlebih dahulu pada bagian awal program seperti yang ditunjukkan gambar 4.13 bagian a, lalu algoritma deteksi kendaraan diaplikasikan pada bagian setelah *background* adaptif seperti yang ditunjukkan gambar 4.13 bagian b.

Untuk nilainya thresholdnya adalah 14. Hal ini didapat dari nilai minimum threshold agar kendaraan tetap terdeteksi dengan semestinya pada saat kondisi cahaya redup. Sedangkan, nilai morfologinya adalah 35 piksel yang didapat dari nilai minimum agar kendaraan yang sedang sejajar dapat terhitung secara terpisah.

4.2.8.3. Algoritma Hitung Kendaraan

```
findContours(coba2.clone(), contours, CV_RETR_EXTERNAL,
CV_CHAIN_APPROX_NONE);
  vector<Rect> boundRect( contours.size() );int z;
  for (z = 0; z < contours.size(); z++)
  {   int qq = contourArea(contours[z]);
        boundRect[z] = boundingRect( Mat(contours[z]));

          if(boundRect[z].y + boundRect[z].height >=220 &&
boundRect[z].height > 80 && boundRect[z].height >
boundRect[z].width)
          {
             int pengecek = 0;
             for(int pencari = boundRect[z].x + 1;pencari <
boundRect[z].x + boundRect[z].width;pencari++)
             {

if(penanda_back.at<uchar>(0,pencari)==255)
                   {
                         pengecek++;
                   }
                   penanda_fore.at<uchar>(0,pencari)=255;
             }
             if (pengecek == 0)
             {
           //cout<<boundRect[z].height<<" "<<qq<<"
"<<boundRect[z].width<<endl;
rectangle(fore_asli,Point(boundRect[z].x,boundRect[z].y+20),
Point(boundRect[z].x + boundRect[z].width, boundRect[z].y +
boundRect[z].height+20),Scalar(255,0,0),2);
int qq = contourArea(contours[z]);
                   if(qq >= 2500 && qq <= 8000 &&
boundRect[z].height > 80)
                   {
                         hitung_mobil++;
                         cout<<"sedang =
"<<hitung_mobil<<endl;
                   }
                   else if(qq > 8000 && qq <=13000)
                   {
                          if(boundRect[z].height >= 140){
                         hitung_truk++;
                         cout<<"besar = "<<hitung_truk<<endl;
                         }
else{hitung_mobil++;cout<<"sedang = "<<hitung_mobil<<endl;}
```

```
                    }
                    else if(qq > 13000)
                    {
                          hitung_bus++;
                          cout<<"sangat besar =
"<<hitung_bus<<endl;
                    }
               }
          }
        }
penanda_back=penanda_fore;
 //menampilkan gambar
line(fore_asli,Point(0,240),Point(319,240),Scalar(0,255,255)
,2);
char text[255];
sprintf(text, "S : %d  B : %d  SB : %d",
(int)hitung_mobil,(int)hitung_truk,(int)hitung_bus);
putText(fore_asli,text,Point(4,14), FONT_HERSHEY_SIMPLEX,
0.6, Scalar(0,255,255), 1);
imshow("hasil_akhir",fore_asli);
```

Gambar 4.14. Hitung Kendaraan

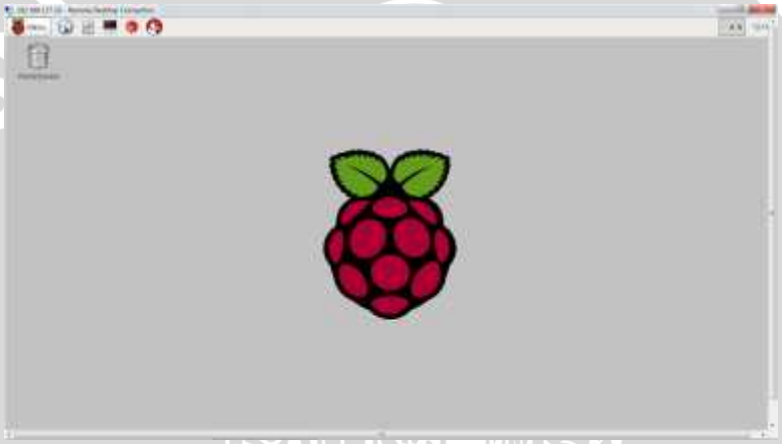
Daerah hitung kendaraan sudah ditentukan terlebih dahulu, yaitu pada koordinat y = 220. Sehingga, penghitungan jenis kendaraan yang berdasarkan luasnya yang ditunjukkan tabel 4.5 diukur pada saat kendaraan memasuki koordinat tersebut pada gambar video.

BAB V PENGUJIAN DAN ANALISIS

Bab ini membahas tentang pengujian dan analisis sistem penghitung kendaraan dengan metode *background substraction* adaptif untuk analisa kepadatan jalan. Pengujian dan analisis sistem bertujuan untuk mengetahui seberapa baik sistem dapat berjalan, seberapa efektif dan seberapa efisien kinerja sistem yang sudah dibuat. Pengujian dan analisis sistem meliputi integrasi *library* OpenCV dengan mini komputer, integrasi monitor (komputer) dengan mini komputer, algoritma program, kualitas performa pengolahan setiap *frame* gambar video.

5.1. Integrasi raspberry pi 2 dengan komputer

Integrasi raspberry pi 2 dengan komputer meliputi me-remote raspberry pi menggunakan input dari komputer dan me-monitor raspberry pi menggunakan monitor komputer. Hasilnya ditunjukkan gambar 5.1 berikut :



Gambar 5.1. memonitor aktifitas raspberry menggunakan komputer

Pada gambar 5.1 merupakan tampilan OS raspbian yang ditampilkan di monitor komputer. Dengan ini integrasi raspberry pi dengan komputer sudah berhasil.

5.2. Integrasi OpenCV dengan raspberry pi 2

Bagian ini menguji apakah *library* OpenCV dapat diintegrasikan dengan raspberry pi. Mula-mula mengetahui dimana *file* OpenCV yang sudah terpasang. Lalu melihat library-nya yang sudah terpasang. Setelah itu, menjalankan dan meninjau program yang sudah dirancang sebelumnya. Jika sudah bisa menjalankan program, maka integrasi OpenCV dengan raspberry pi 2 sudah berhasil.

Gambar 5.2. Letak berkas OpenCV yang telah terpasang

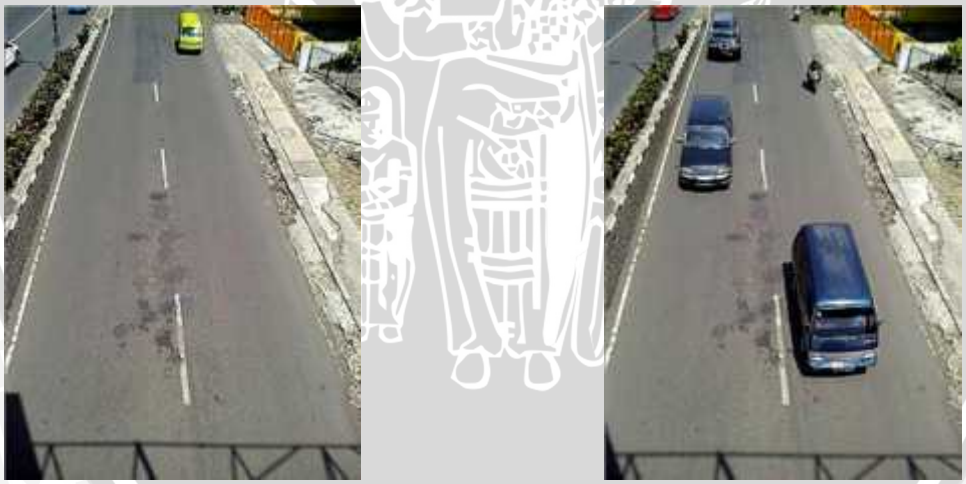
Dengan perintah “pkg-config –cflags opencv” akan terlihat dimana berkas opencv terpasang. seperti yang ditunjukkan gambar 5.2. langkah selanjutnya adalah memastikan *library-library* opencv juga sudah terpasang.

Gambar 5.3. Library opencv yang sudah terpasang

Dengan perintah “pkg-config –libs opencv” akan menunjukkan *library-library* opencv apa saja yang terpasang pada mini komputer di direktori usr/local/lib seperti yang ditunjukkan gambar 5.3. Dengan ini, opencv sudah terpasang dan dapat digunakan.

5.2.1. Proses Deteksi Kendaraan

Bagian ini merupakan proses deteksi kendaraan dari perancangan algoritma program yang sudah dibuat. Mula-mula mendeklarasikan gambar yang dijadikan sebagai *background* dan mengambil contoh *foreground* seperti gambar 5.4 berikut :



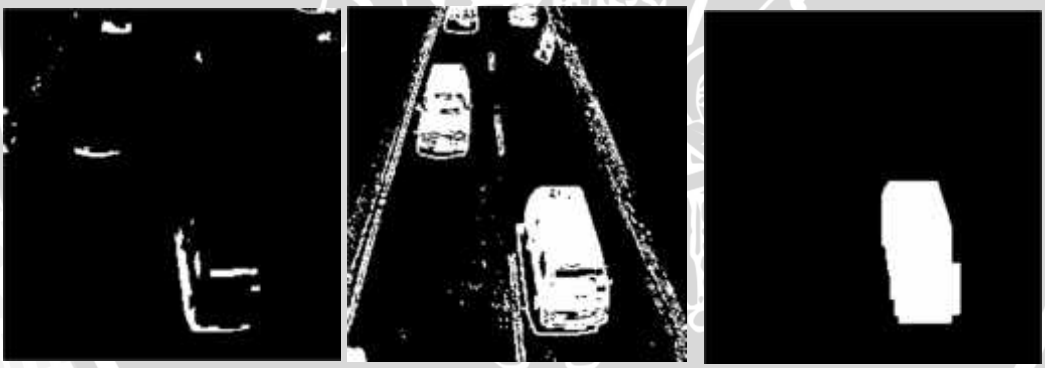
a) b)

Gambar 5.4. a) *frame background.* b) contoh *foreground*

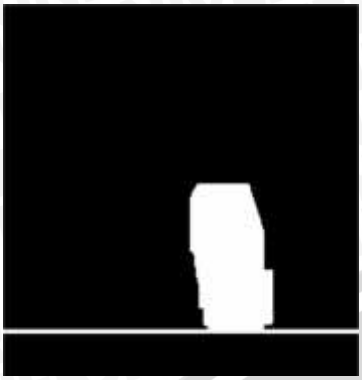
Berdasarkan background yang sudah ditentukan dan contoh foreground yang diambil, proses deteksi kendaraan ditunjukkan gambar 5.5 dibawah ini :

a) b) c)

d) e) f)



g) h) i)



j)

k)

Gambar 5.5. Proses deteksi kendaraan. a) ROI background dengan *Gaussian blur.* b) ROI *foreground* dengan *Gaussian blur.* c) hasil *background substraction*. d) menghilangkan daerah tidak perlu. e) *grayscale* citra.f) *thresholding.* g) *thresholding* efek bayangan. h) *background substraction* dengan citra efek bayangan. i) menghaluskan gambar. j) pemberian garis penghitungan. k) tampilan akhir.

Pada bagian thresholding yang digunakan adalah 14. Hal ini mengacu pada nilai thresholdnya yang dibuat rendah (mulai dari 0) karena semakin rendah nilai threshold maka semakin tinggi pula akurasi deteksi objek, akan tetapi jumlah *noise* juga semakin banyak. Oleh karena itu, dilakukan percobaan dalam menentukan nilai threshold dimana nilai tersebut adalah nilai minimum agar sistem dapat mendeteksi objek pada keadaan yang dapat menimbulkan *noise* terbesar yaitu pada saat kondisi palling redup.

a)

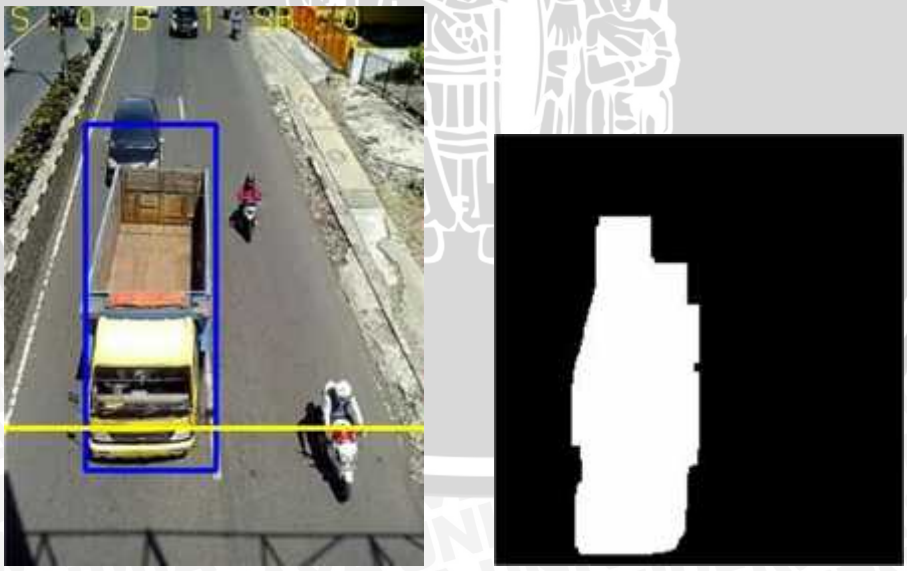
b)

c)

Gambar 5.6. Menentukan nilai threshold. a) Kondisi pencahayaan redup. b) segmentasi dengan nilai threshold = 10. c) segmentasi dengan nilai threshold =14.

Terlihat pada gambar 5.6.b merupakan hasil segmentasi dengan nilai threshold dibawah 14. Lalu, proses penentuan threshold dicari hingga hasil segmentasinya sesuai dengan objek yang melintas seperti yang ditunjukkan gambar 5.6.c.

Sedangkan pada bagian pisahkan objek, nilai morfologinya adalah 35. Nilai tersebut ditentukan dengan cara melakukan percobaan dimana nilai tersebut adalah nilai minimum agar kendaraan yang terlihat bergabung dapat dipisahkan.



a) b)

c)

d)

Gambar 5.7. Menentukan nilai morfologi. a) deteksi kendaraan dengan morfologi dibawah 35. b) segmentasi gambar dengan morfologi dibawah 35. c) deteksi kendaraan dengan nilai morfologi 35. d) segmentasi gambar dengan morfologi 35.

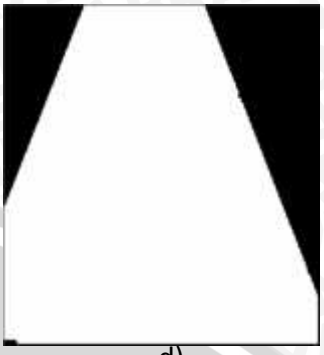
Ketika terdapat objek yang terlihat saling bergabung seperti yang ditunjukkan gambar 5.7, maka diperlukan operasi morfologi untuk memisahkannya. Untuk menentukan nilai morfologi diperlukan uji coba hingga hasilnya terlihat seperti gambar 5.7.c dan 5.7.d.

Untuk proses deteksi kendaraan yang ditunjukkan gambar 5.5 merupakan contoh kasus dimana *foreground* dalam keadaan terang. Sehingga, pengaruh algoritma *background* adaptif tidak terlihat. Pada gambar 5.8 dibawah ini mrupakan contoh hasil implementasi *background* adaptif, yaitu ketika *foreground* dalam keadaan redup :

a)

b)

c)



d)

Gambar 5.8. Contoh sistem background statis. a) background. b) contoh foreground. c) hasil background substraction d) hasil segmentasinya.

Gambar 5.8 diatas merupakan contoh masalah pada *background substraction* yaitu sensitif terhadap perubahan pencahayaan, sehingga ketika terjadi perubahan cahaya sistem tidak dapat mendeteksi ada tidaknya kendaraan. Oleh karena itu, untuk mengurangi dampak perubahan cahaya tersebut digunakan algoritma background adaptif. Hasilnya ditunjukkan gambar 5.9 berikut ini :

a)

b)

e)

c)

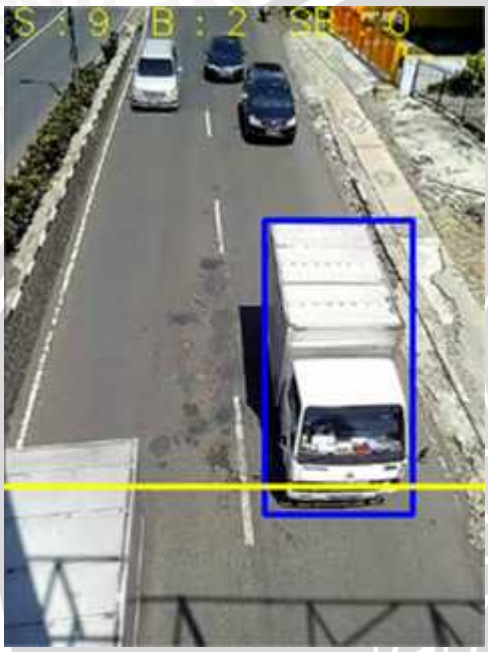
d)

Gambar 5.9. Contoh Sistem Background Adaptif. a) background asli. b) foreground. c) background baru. d) background substraction. e) hasil segmentasinya.

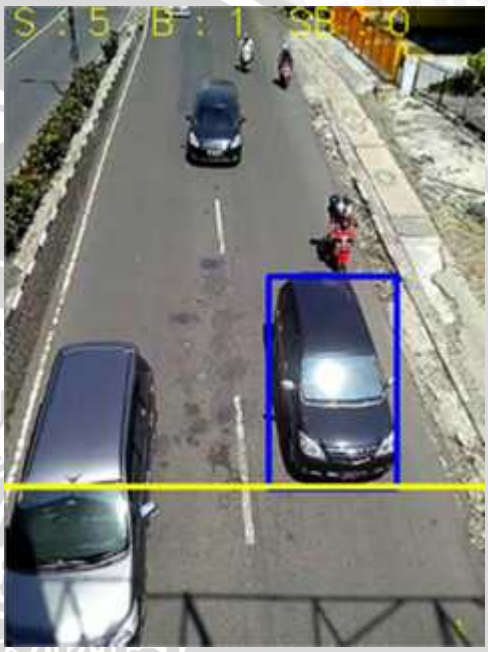
Dengan cara menerapkan *background* adaptif seperti gambar 5.9 diatas, maka sistem masih bisa mendeteksi kendaraan karena metode *background substraction* sangat sensitive terhadap intensitas cahaya.

5.2.2. Contoh Hasil Pendeteksi Kendaraan

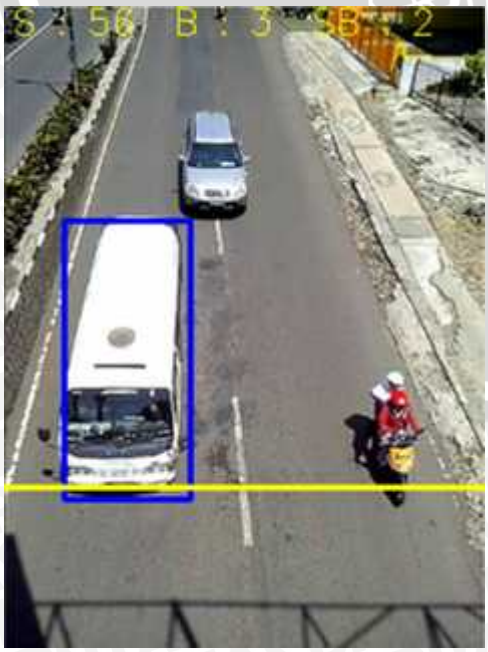
Adapun contoh-contoh hasil deteksi kendaraan seperti gambar 5.10 dibawah ini :



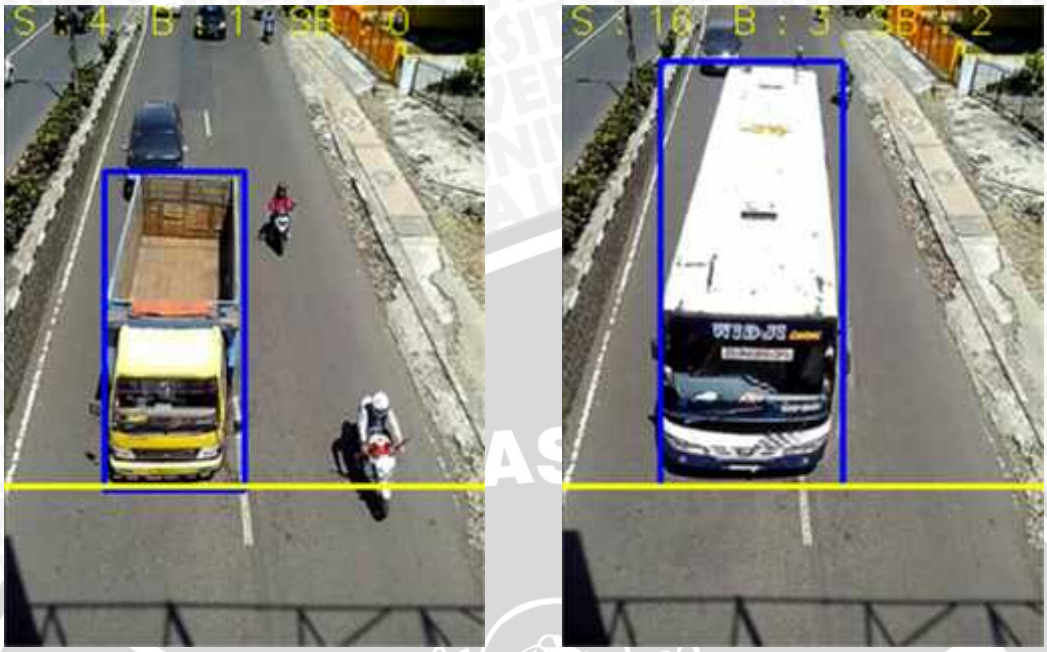
a)



b)



c) d)



e) f)

Gambar 5.10. Contoh-Contoh hasil pengujian sistem deteksi kendaraan

Tanda garis kuning menunjukkan sistem akan melakukan penghitungan kendaraan jika melewati garis kuning. Apabila sistem berhasil mendeteksi kendaraan tersebut maka akan ada tanda kotak berwarna biru seperti yang ditunjukkan gambar 5.10.

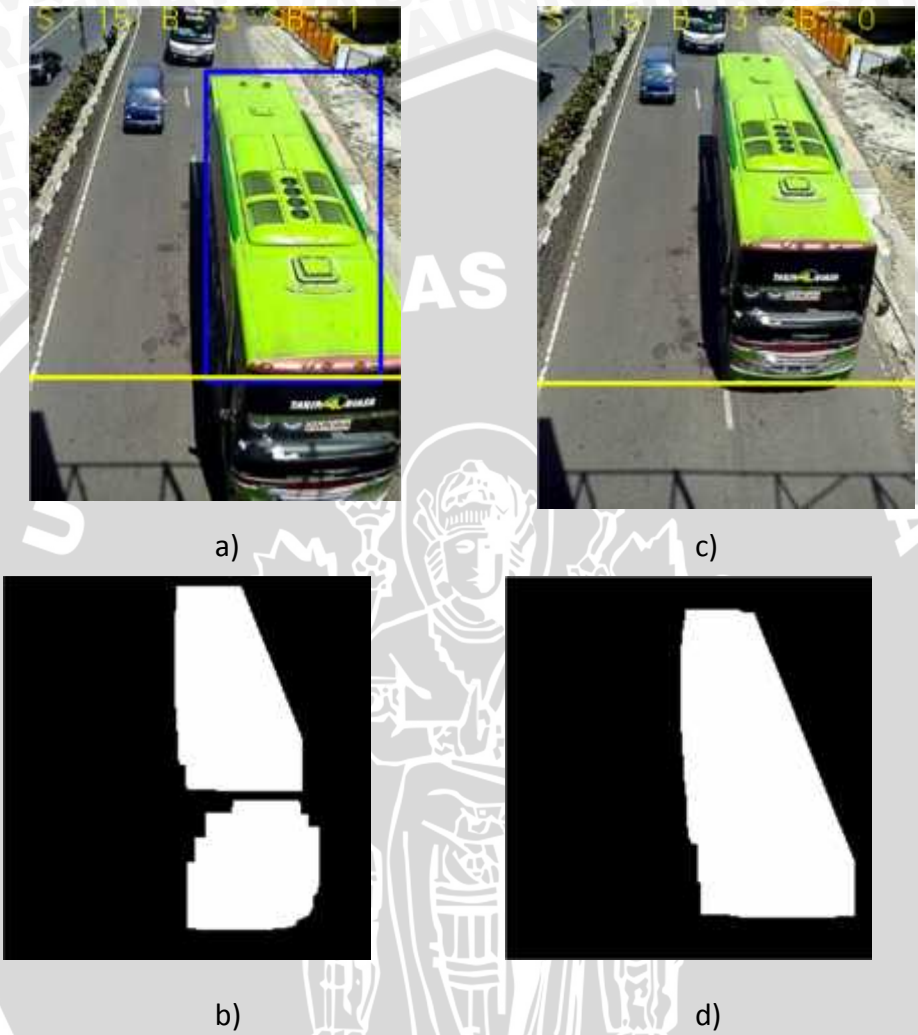
5.2.3. Gangguan Pantulan Cahaya

Akan tetapi, terdapat gangguan saat mendeteksi mobil pada waktu tertentu walaupun sistem dapat mendeteksi kendaraan tersebut seperti yang ditunjukkan gambar 5.11 dibawah ini :

Gambar 5.11. Contoh Gangguan Pantulan Cahaya

Khusus untuk tipe mobil tertentu dan pada posisi tertentu terjadi gangguan pantulan cahaya matahari, sehingga dapat mengakibatkan penambahan luas kendaraan (piksel putih) saat proses segmentasi.

5.2.4. Gangguan Pendeteksian Kendaraan Bus



a) c)

b) d)

Gambar 5.12. Contoh gangguan deteksi bus. a) bus yang melewati garis hitung. b) hasil segmentasi tidak sempurna. c) bus baru terdeteksi ketika berada beberapa jarak dari garis hitung. d) hasil segmentasi bus ketika terdeteksi

Khusus untuk kendaraan bus, dapat terjadi gangguan saat proses segmentasi seperti yang ditunjukkan gambar 5.12.b. Tidak semua bagian kendaraan terdeteksi karena ada bagian yang terpisah. Hal ini dikarenakan kaca gelap bagian depan bus mempunyai nilai yang sama dengan bayangan, sehingga pada saat memasuki proses mengurangi efek bayangan, bagian kaca gelap tersebut juga ikut terfilter. Bus akan terdeteksi setelah maju beberapa jarak kemudian. Akan tetapi, walaupun ada bagian bus yang tidak terdeteksi, sistem masih bisa mengenali kendaraan tersebut adalah bus seperti yang ditunjukkan gambar 5.12.d.

5.3. Pengujian Akurasi Sistem

Pengujian ini bertujuan untuk mengukur seberapa tepat sistem dalam mendeteksi kendaraan baik berdasarkan jenis kendaraan maupun berdasarkan jumlahnya pada kecepatan kendaraan yang sebenarnya, 2 kali dan 4 kali lipat dari kecepatan yang sebenarnya. Pengujian akurasi dapat ditulis seperti rumus 5.1 dan 5.2 dibawah ini :

1. $A = \frac{J}{J} x\ 100\%$ .....(5.1)
2. $A = \frac{J_t}{J} x\ 100\%$ .....(5.2)

Keterangan :
A : Akurasi
Jkt : Jumlah Kendaraan yang terdeteksi
Jks : Jumlah kendaraan yang sebenarnya
Jktsu : Jumlah kendaraan yang terdeteksi sesuai ukurannya

5.3.1. Akurasi sistem pendeteksian kendaraan berdasarkan jenisnya dengan kecepatan kendaraan yang sebenarnya

Tabel 5.1. Akurasi sistem pendeteksian kendaraan berdarkan jenisnya dengan kecepatan kendaraan yang sebenarnya

No	KS-t-KS	j-KS-s	KB-t-KB	J-KB-s	KSB-t-KSB	j-KSB-s	A
1	15	15	1	1	0	0	100 %
2	21	21	0	0	1	1	100 %
3	82	82	3	3	2	2	100 %
4	65	65	1	2	1	1	98,55 %

Keterangan :
KS-t-KS : Kendaraan ukuran sedang terdeteksi sebagai kendaraan ukuran sedang
j-KS-s : Jumlah kendaraan ukuran sedang yang sebenarnya
KB-t-KB : Kendaraan ukuran sedang terdeteksi sebagai kendaraan ukuran sedang
j-KB-s : Jumlah kendaraan ukuran besar yang sebenarnya
KSB-t-KSB : Kendaraan ukuran sangat besar terdeteksi sebagai kendaraan ukuran sangat besar
j-KSB-s : jumlah kendaraan ukuran sangat besar yang sebenarnya
A : Akurasi

Pada tabel 5.1 hanya data ke-4 yang tidak sesuai dengan keadaan sebenarnya karena ada kesalahan pembacaan jenis kendaraan ukuran besar. Penyebabnya adalah sebagian besar warna kendaraan besar tersebut hampir sama dengan warna jalan raya (*background*), sehingga bagian kendaraan tersebut tidak dianggap sebagai objek. Akibatnya, kendaraan besar tersebut terhitung sebagai kendaraan ukuran sedang. Jadi, jumlah kendaraan sedang

menjadi 66, sedangkan kendaraan ukuran besar terhitung 1, dan kendaraan sangat besar terhitung 1.

5.3.2. Akurasi deteksi kendaraan berdasarkan jenisnya dengan kecepatan kendaraan 2 kali lebih cepat dari yang sebenarnya

Tabel 5.2. Akurasi deteksi kendaraan berdasarkan jenisnya dengan kecepatan kendaraan 2 kali lebih cepat dari yang sebenarnya

No	KS-t-KS	j-KS-s	KB-t-KB	J-KB-s	KSB-t-KSB	j-KSB-s	A
1	15	15	1	1	0	0	100 %
2	21	21	0	0	1	1	100 %
3	82	82	3	3	2	2	100 %
4	65	65	1	2	1	1	98,55 %

Keterangan :
KS-t-KS : Kendaraan ukuran sedang terdeteksi sebagai kendaraan ukuran sedang
j-KS-s : Jumlah kendaraan ukuran sedang yang sebenarnya
KB-t-KB : Kendaraan ukuran sedang terdeteksi sebagai kendaraan ukuran sedang
j-KB-s : Jumlah kendaraan ukuran besar yang sebenarnya
KSB-t-KSB : Kendaraan ukuran sangat besar terdeteksi sebagai kendaraan ukuran sangat besar
j-KSB-s : jumlah kendaraan ukuran sangat besar yang sebenarnya
A : Akurasi

Untuk tingkat akurasi deteksi kendaraan berdasrkan jenisnya pada konfigurasi kecepatan kendaraan 2 kali lipat sama dengan tingkat akurasi pada kecepatan kendaraan yang sebenarnya.

5.3.3. Akurasi deteksi kendaraan berdasarkan jenisnya dengan kecepatan kendaraan 4 kali lebih cepat dari yang sebenarnya

Tabel 5.3. Akurasi deteksi kendaraan berdasarkan jenisnya dengan kecepatan kendaraan 4 kali lebih cepat dari yang sebenarnya

No	KS-t-KS	j-KS-s	KB-t-KB	J-KB-s	KSB-t-KSB	j-KSB-s	A
1	15	15	1	1	0	0	100 %
2	21	21	0	0	1	1	100 %
3	82	82	3	3	2	2	100 %
4	65	65	1	2	1	1	98,55 %

Keterangan :
KS-t-KS : Kendaraan ukuran sedang terdeteksi sebagai kendaraan ukuran sedang
j-KS-s : Jumlah kendaraan ukuran sedang yang sebenarnya
KB-t-KB : Kendaraan ukuran besar terdeteksi sebagai kendaraan ukuran besar

j-KB-s : Jumlah kendaraan ukuran besar yang sebenarnya
KSB-t-KSB : Kendaraan ukuran sangat besar terdeteksi sebagai kendaraan ukuran sangat besar
j-KSB-s : jumlah kendaraan ukuran sangat besar yang sebenarnya
A. : Akurasi

Untuk tingkat akurasi deteksi kedaraan berdasarkan jenisnya pada konfigurasi kecepatan kendaraan 4 kali lipat sama dengan tingkat akurasi dengan kecepatan kendaraan yang sebenarnya. Hal ini berarti, sistem masih bisa memepertahankan tingkat akurasi pada kecepatan kendaraan yang tinggi dalam hal mendeteksi kendaraan berdasarkan jenisnya.

5.3.4. Akurasi sistem pendeteksian kendaraan berdasarkan jumlahnya pada kecepatan kendaraan yang sebenarnya

Tabel 5.4. Akurasi sistem pendeteksian kendaraan berdasarkan jumlahnya pada kecepatan kendaraan yang sebenarnya

Data ke-	Jumlah kendaraan terdeteksi	Jumlah Kendaraan Sebenarnya	Akurasi
1	16	16	100 %
2	22	22	100 %
3	87	87	100 %
4	69	69	100 %

Pada tabel 5.4 akurasi sistem dalam mendeteksi kendaraan berdasarkan jumlahnya dengan kecepatan yang sebenarnya pada semua data video adalah 100%. Hal ini berarti jumlah kendaraan yang dideteksi oleh sistem sama dengan kondisi sebenarnya.

5.3.5. Akurasi sistem pendeteksian kendaraan berdasarkan jumlahnya pada kecepatan kendaraan 2 kali lebih cepat dari yang sebenarnya

Tabel 5.5. akurasi sistem pendeteksian kendaraan berdasarkan jumlahnya pada kecepatan kendaraan 2 kali lebih cepat dari yang sebenarnya

Data ke-	Jumlah kendaraan terdeteksi	Jumlah Kendaraan Sebenarnya	Akurasi
1	16	16	100 %
2	22	22	100 %
3	87	87	100 %
4	69	69	100 %

Pada tabel 5.5 akurasi sistem dalam mendeteksi kendaraan berdasarkan jumlahnya dengan kecepatan kendaraan 2 kali lipat pada semua data video

adalah 100%. Hal ini berarti jumlah kendaraan yang dideteksi oleh sistem sama dengan kondisi sebenarnya.

5.3.6. Akurasi mendeteksi kendaraan berdasarkan jumlahnya pada kecepatan kendaraan 4 kali lebih cepat dari yang sebenarnya

Tabel 5.6. akurasi mendeteksi kendaraan berdasarkan jumlahnya pada kecepatan kendaraan 4 kali lebih cepat dari yang sebenarnya

Data ke-	Jumlah kendaraan terdeteksi	Jumlah Kendaraan Sebenarnya	Akurasi
1	16	16	100 %
2	22	22	100 %
3	87	87	100 %
4	69	69	100 %

Pada tabel 5.6 akurasi sistem dalam mendeteksi kendaraan berdasarkan jumlahnya dengan kecepatan kendaraan 4 kali lipat pada semua data video adalah 100%. Hal ini berarti jumlah kendaraan yang dideteksi oleh sistem sama dengan kondisi sebenarnya.

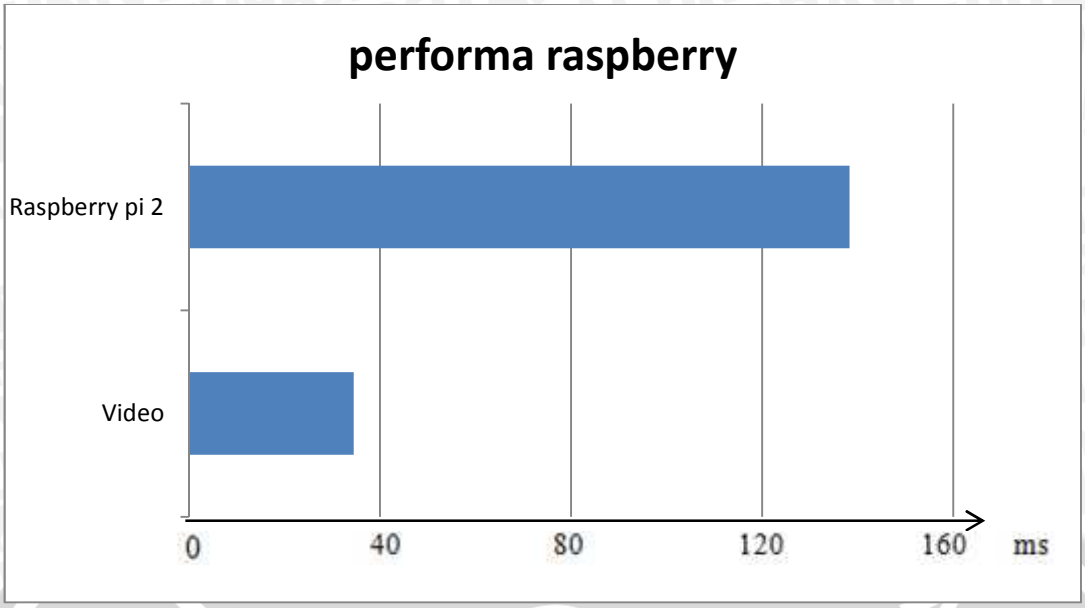
Sama seperti deteksi kendaraan berdasarkan jenisnya. Tingkat akurasi deteksi kendaraan berdasarkan jumlahnya pada konfigurasi kecepatan 4 kali lipat sama seperti tingkat akurasi pada kecepatan yang sebenarnya. Hal ini berarti sistem masih bisa mendeteksi kendaraan berdasarkan jumlahnya pada kecepatan kendaraan 120-240 km/jam.

5.4. Pengujian Performa

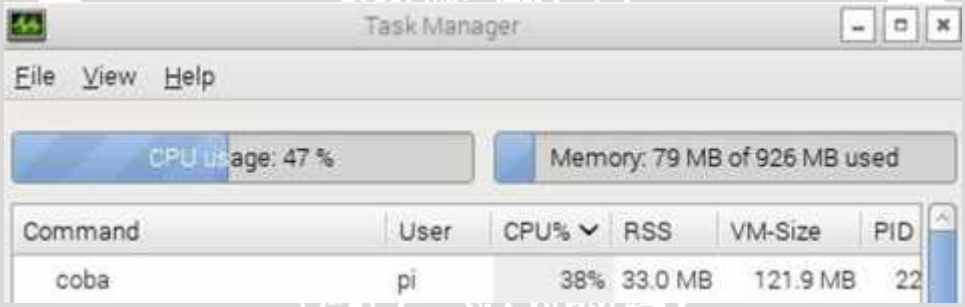
Pengujian performa bertujuan untuk mengetahui apakah sistem dapat bekerja secara *real-time* atau belum. Sistem dapat dikatakan bekerja secara real-time apabila sistem dapat mengolah setiap frame video dalam kurun waktu kurang dari 34.48 ms, berhubungan dengan *frame rate* video yang digunakan adalah 29 fps yang artinya rentang waktu pergantian antar *frame* adalah 34.48 ms. Pengujian performa dilakukan pada 2 *device* yaitu raspberry pi 2 dan komputer sebagai perbandingan.

5.4.1. Pengujian Performa Pada Raspberry pi 2

- Jumlah *frame* yang diolah = 1421 *frame.*
- Total waktu yang diperlukan = 179,7 detik.
- Waktu yang diperlukan untuk mengolah setiap *frame* = $\frac{1\ ,5}{1} =$ 138,325 m .
- 138,325 ms > 34,48 ms. Waktu yang diperlukan untuk mengolah setiap gambar lebih besar daripada *frametime* video. Hal ini berarti embedded system belum bisa berjalan secara *realtime*.



a)



b)

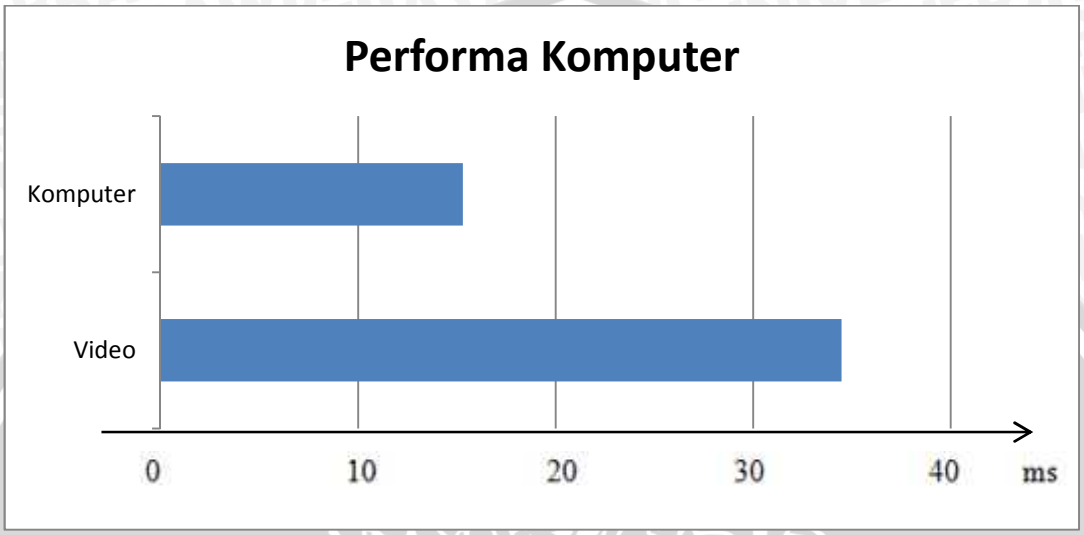
Gambar 5.13. a) performa kecepatan pengolahan gambar pada raspberry pi 2. b) performa CPU raspberry pi 2

Data hasil pengujian menunjukkan bahwa raspberry pi 2 membutuhkan waktu 138,325 ms untuk memproses setiap *frame* video atau dengan *framerate* 7,22 fps, sementara *frame time* video adalah 34,48 ms atau dengan *framerate* 29 fps. Hal ini berarti raspberry pi 2 butuh kecepatan minimum dalam memproses gambar sekitar 4,02 kali lipat agar dapat berjalan secara *real-time.*

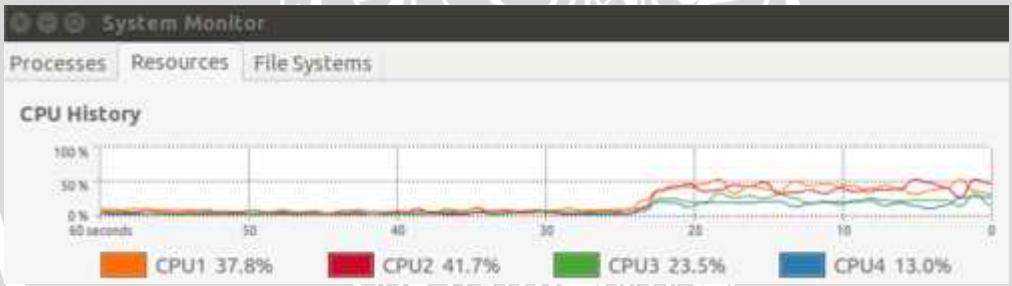
Selain itu, penyebab lain raspberry pi 2 tidak dapat mengolah gambar secara *real-time* adalah library OpenCV belum dapat mengoptimasi *quad core* CPU dengan maksimal. Hal ini ditunjukkan oleh gambar 5.13.b bahwa total utilisasi prosessor saat menjalankan OpenCV adalah 47%, artinya hanya sekitar 2 core saja yang digunakan (jumlah *core* prosesor raspberry pi 2 ada 4). Hal ini dapat disimpulkan bahwa kecepatan pengolahan gambar pada raspberry pi 2 dapat ditingkatkan dengan cara meningkatkan utilisasi prossesor.

5.4.2. Pengujian Performa pada komputer

- Jumlah *frame* yang diolah = 1421 *frame.*
- Total waktu yang diperlukan = 21,75 detik
- Waktu yang diperlukan untuk mengolah setiap *frame* = $\frac{2\ ,7}{1} = 15{,}3\ m$.
- 15,3 ms < 34,48 ms. Sudah bisa berjalan secara *real-time*



a)



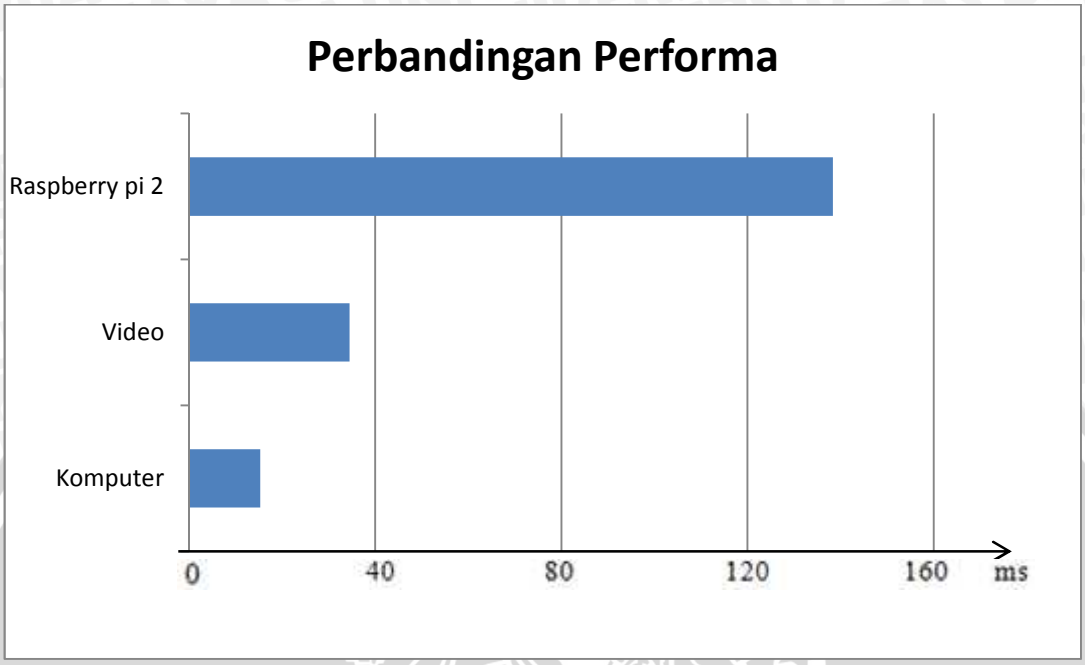
b)

Gambar 5.14. a) performa kecepatan pengolahan gambar pada komputer. b) performa CPU pada komputer

Data hasil Pengujian membuktikan komputer dengan spsesifikasi yang ditunjukkan tabel 4.1 membutuhkan waktu 15,3 ms untuk mengolah setiap *frame* video dengan utilisasi prosessor sebesar 29% seperti yang ditunjukkan oleh gambar 5.14.b. Hal ini berarti komputer dapat mengolah file video secara *real-time* karena waktu yang dibutuhkan untuk mengolah setiap *frame* video lebih cepat daripada *frame time* video. Komputer masih mempunyai waktu bebas 19,18 ms (seperti yang ditunjukkan oleh gambar 5.14.a dari *delay* maksimum untuk memproses video secara *real-time.*

5.4.3. Perbandingan Performa

Perbandingan performa raspberry pi 2 dan komputer ditunjukkan pada gambar 5.13 berikut :



Gambar 5.15. perbandingan performa komputer dengan raspberry pi 2.

Dari gambar 5.15 menunjukkan bahwa ada perbedaan yang signifikan antara kecepatan pengolahan gambar pada raspberry pi 2 dengan komputer. komputer hanya memerlukan waktu 15,3 ms untuk mengolah setiap *frame* video,sedangkan raspberry pi 2 memerlukan waktu 138,325 ms untuk mengolah setiap *frame* video. Padahal, waktu *delay* maksimum untuk mengolah data secara real-time adalah 34,48 ms. sehingga, dapat disimpulkan bahwa raspberry pi 2 belum dapat mengolah gambar video secara real-time.

BAB 6 PENUTUP

6.1. Kesimpulan

Berdasarkan perancangan, implementasi dan hasil pengujian dan analisis sistem maka dapat disimpulkan sebagai berikut :

1. Untuk dapat merancang sistem deteksi kendaraan pada *embedded* sistem diperlukan OS yang mempunyai GUI (*Graphic User Interface*) agar hasilnya dapat muncul, OpenCV sebagai perangkat lunak yang khusus digunakan untuk pengolahan citra digital (*Object Oriented*) dan komputer yang digunakan untuk menjalankan aktivitas raspberry pi 2 dan *monitoring* hasil pendeteksian kendaraan. Metode yang digunakan dalam pengolahan citra dalam penelitian ini adalah *background substraction* yaitu mendeteksi ada tidaknya objek yang melintas dengan cara ada tidaknya perubahan nilai piksel antara *background* dan *foreground.* Bahasa pemrogramana yang digunakan dalam penelitian ini adalah C++.

2. Berdasarkan hasil pengujian dan analisis, sistem dapat mendeteksi adanya kendaraan dengan akurasi 100% pada kondisi cahaya terang. pada saat intensitas cahaya (*brightness*) tetap ataupun berubah-ubah. Lalu, sistem diuji pada kondisi kecepatan kendaraan 2 kali lipat dan kemudian diuji pada kondisi kecepatan kendaraan 4 kali lipat. Hal ini bertujuan untuk mengukur / mengetahui kehandalan sistem dalam mendeteksi kendaraan yang berkecepatan tinggi. Setelah diuji ternyata sistem masih dapat mendeteksi adanya kendaraan yang melintas dengan akurasi 100% atau sesuai dengan jumlah kendaraan yang sebenarnya.

3. Berdasarkan hasil pengujian dan analisis, sistem dapat mendeteksi kendaraan berdasarkan jenisnya sebesar 100% untuk data video ke-1 dan ke-2, dan 98,55% untuk data video ketiga. Adanya kesalahan deteksi jenis kendaraan disebabkan oleh kendaraan truk yang sebagian besar warnanya mirip dengan warna background. hal ini menyebabkan bagian yang warnanya sama dengan *background* tidak terdeteksi sebagai objek. Sehingga, truk tersebut terbaca sebagai mobil.

4. Kecepatan pengolahan gambar pada raspberry pi 2 belum dapat dieksekusi secara real-time. Raspberry pi 2 membutuhkan waktu sekitar 138,3 ms untuk mengolah setiap *frame* video atau setara dengan 7,22fps, sedangkan *frame rate* video yang digunakan adalah 29 fps. Artinya, *embedded system* belum bisa berjalan secara *realtime.* Salah satu penyebabnya adalah OpenCV belum dapat menggunakan prosessor raspberry pi 2 dengan maksimal yaitu sebesar 47%. Sehingga, kecepatan pengolahan gambar pada raspberry pi 2 dapat ditingkatkan dengan cara menggunakan kemampuan prosessornya dengan maksimal (100%).

Sedangkan pengolahan gambar pada komputer dapat dieksekusi secara *realtime* walaupun utilisasi prosessor 29 %. Komputer hanya membutuhkan waktu 15,3 ms untuk mengolah setiap *frame* video atau setara dengan 61,53fps. Hal ini menunjukkan ada perbedaaan performa yang signifikan antara prosessor raspberry pi 2 yang berbasis ARM cortex A7 @900 Mhz dengan komputer yang menggunakan intel core i3 generasi kedua @2,1Ghz. Sehingga, proses pengolahan data secara *real-time* masih hanya dapat dilakukan pada komputer.

6.2. Saran

1. Diperlukan utilisasi CPU dan optimasi program yang lebih baik untuk meningkatkan performa, terutama pada embedded sistem karena belum mampu berjalan secara *realtime*.
2. Diperlukan kamera digital jika ingin diimplementasikan di lapangan secara langsung. Kualitas gambar setidaknya sama dengan gambar yang digunakan pada penelitian ini.
3. Diperlukan modul komunikasi jarak jauh antara embedded sistem dan server agar lebih efisien / hemat biaya.

DAFTAR PUSTAKA

Ahmad, S., Joko, L.B, Bilqis, A., Silvester, T., Jani, F.M, 2013. Perangkat Lunak Untuk Deteksi Jumlah Kendaraan Di Jalan Dengan Transceiver SRF02. [pdf] Tersedia di : <http://satek.unila.ac.id/wp-content/uploads/2014/03/2-260.pdf> [diakses 25 Februari 2015]

Baggio,D.L., Escriva,D.M., Mahmood,N., Shilkrot,R., Emami,S., Levgen,K., Saragih,J., 2012. Mastering OpenCV with Practical Computer Vision Projects. UK : Packt Publishing Ltd.

Binanto,I., 2010. Multimedia Digital – Dasar Teori dan Pengembangannya. Yogyakarta : C.V. ANDI OFFSET.

Darma,P., 2010. Pengolahan Citra Digital. Yogyakarta : Andi

Deepoy., Saharia,S., 2014. Implementation And Performance Evaluation Of Background Subtraction Algorithms. [pdf] Tersedia di : <http://arxiv.org/ftp/arxiv/papers/1405/1405.1815.pdf> [diakses 5 Juni 2015]

DITRENDUK., 2010. Kepadatan Penduduk 2010. [online] Tersedia di : <http://www.bkkbn.go.id/kependudukan/Pages/DataSensus/Sensus_Penduduk/Penduduk/Kepadatan_Penduduk/Nasional.aspx> [diakses 23 Juni 2015]

Jayaraman,S., Esakkirajan,S., Veerakumar,T., 2011. Digital Image Processing. New Delhi : Tata McGraw Hill

Juhari,I., 2014. Perancangan Aplikasi Pengurangan Noise Pada Citra Digital Menggunakan Metode Filter Gussian. [pdf] Tersedia di : <http://inti-budidarma.com/berkas/jurnal/12.pdf> [diakses 23 Juni 2015]

Litbang, 2013. Hasil Hitung Lalu lintas, Tolak Ukur Tangani Jalan. [online] Tersedia di : <http://pu.go.id/berita_satminkal/go/801/badan-penelitian-dan-pengembangan> [diakses 25 Frebruari 2015]

Mall,R., 2007. Real-Time Systems: Theory and Practice. India : Dorling Kindersley Pvt. Ltd.

Meshram,S.A., Malviya,A.V., 2013. Traffic Surveillance by Counting and Classification of Vehicles from Video using Image Processing. [pdf] Tersedia di : <http://www.ijarcsms.com/docs/paper/volume1/issue6/V1I6-0026.pdf> [diakses 5 Juni 2015]

Permana,M.,I., 2015. Implementasi Pengenalan Kendaraan Menggunakan Embedded System. S1. Universitas Brawijaya.

Raspberry Pi Foundation, 2015., Raspberry pi 2 model b. [online] Tersedia di : <http://www.bkkbn.go.id/kependudukan/Pages/DataSensus/Sensus_Penduduk/Penduduk/Kepadatan_Penduduk/Nasional.aspx> [diakses27 April 2015]

Sifuentes,E., Casas, O., Pallas-Areny, R., 2011. Wireless Magnetic Sensor Node for Vehicle Detection With Optical Wake-Up. [pdf] Tersedia di : <http://www.researchgate.net/profile/Ramon_Pallas-Areny/publication/224209597_Wireless_Magnetic_Sensor_Node_for_Vehicle_Detection_With_Optical_Wake-Up/links/0912f512cb3fab2aa3000000.pdf> [diakses 25 Februari 2015]

Silvanus, A., 2013. Kisah Edi Warsito : Hitung Kendaraan Mudik Secara Manual. [online] Tersedia di : <http://news.liputan6.com/read/662163/kisah-edi-warsito-hitung-kendaraan-mudik-secara-manual> [diakses 25 Februari 2015]

Supianto,A., 2013. Morfologi Citra. [pdf] Tersedia di : <http://afif.lecture.ub.ac.id/files/2013/10/Slide-06-Morfologi-Citra.pdf> [diakses 23 Juni 2015]