LAPORAN PROYEK AKHIR

PURWARUPA SISTEM KLASIFIKASI DAN PENGHITUNG JUMLAH KENDARAAN BERMOTOR MENGGUNAKAN KAMERA *WEBCAM* BERBASIS CITRA DIGITAL

Di PT. Industri Telekomunikasi Indonesia (Persero)

Disusun Oleh:

WISNU RIZKY KURNIAWAN

NIM: 2012/327839/SV/00015

PROGRAM DIPLOMA TEKNIK ELEKTRO

SEKOLAH VOKASI

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2015

LAPORAN PROYEK AKHIR

PURWARUPA SISTEM KLASIFIKASI DAN PENGHITUNG JUMLAH KENDARAAN BERMOTOR MENGGUNAKAN KAMERA *WEBCAM* BERBASIS CITRA DIGITAL

Di PT. Industri Telekomunikasi Indonesia (Persero)

Diajukan sebagai salah satu syarat Lulus
Program Studi Diploma Teknik Elektro
Sekolah Vokasi
Universitas Gadjah Mada
Yogyakarta

Oleh

Wisnu Rizky Kurniawan
2012/327839/SV/00015

PROGRAM DIPLOMA TEKNIK ELEKTRO
SEKOLAH VOKASI
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2015

LEMBAR PENGESAHAN

Judul	: Purwarupa sistem klasifikasi dan penghitung kendaraan bermotor menggunakan kamera *webcam* berbasis citra digital di PT. INTI (Persero)
Nama	: Wisnu Rizky Kurniawan
Konsentrasi	: Teknik Telekomunikasi
Pembimbing	: Budi Bayu Murti, S.T., M.T
Waktu / Tempat Ujian	: Jum'at, 6 Juli 2015 Pukul 08.00 WIB/ Ruang 234

Sudah disetujui oleh Program Diploma Teknik Elektro, Sekolah Vokasi, Universitas Gadjah Mada, sebagai bagian dari syarat kelulusan untuk memperoleh gelar Ahli Madya (A. Md).

Ketua Program Diploma :

Ir. Lukman Subekti, M. T
NIP. 196210301993031002

Pembimbing PA :

Budi Bayu Murti, S. T., M. T.
NIP. 197212231999031001

TIM PENGUJI

Ketua :

Nur Sulistyawati, S.T., M.Eng.
NIP. 19730932005012001

Sekretaris :

Ir. Rizal
NIP. 195207271988031001

Penguji Utama :

Muhammad Arrofiq, S.T., M.T., Ph.D.
NIP.197911271999031001

PERNYATAAN

Dengan ini saya menyatakan bahwa Proyek Akhir ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar ahli madya di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 29 Juni 2015

Wisnu Rizky Kurniawan

MOTTO

Ketika tulisan ini berhasil maka saya adalah penulis pertama dari kampus saya yang menulis tentang sistem berbasis pengolah citra, dan penulis pertama yang menggunakan bahasa pemrograman berbeda diantara para pendahulu saya yang menulis tentang sistem berbasis pengolah citra tingkat universitas versi ETD UGM Juni 2015. [Wisnu Rizky Kurniawan]

Kelak buatlah hal yang sederhana... yang tanpa itu, hal penting tidak akan pernah ada, seperti jarum yang merajut pakaian yang kini kamu pakai [Yuni Sukarsana]

Bangunlah dari tidurmu se-dini mungkin, karena engkau perlu merencanakan hari ini, bangunlah potensi diri karena itu akan menolongmu nanti [Purdiyaningsih]

Dalam hidup, seperti permainan catur, orang yang berpikir panjanglah yang menang [Charles Buxton]

Ubahlah batu sandungan anda, menjadi batu-batu loncatan [Anonim]

Laksanakan kewajiban anda sebaik-baiknya, selebihnya serahkan kepada Tuhan [Pierre Corneille]

Perkembangan terjadi saat seseorang melewati batasannya, mengetahuinya pun termasuk dalam tugasnya [Itachi Uchiha]

Kebanggaan kita yang terbesar adalah bukan tidak pernah gagal, tetapi bangkit kembali setiap kali kita jatuh [Konfusius]

Aku adalah pejalan kaki yang lambat, tetapi aku tak pernah mundur.

Kesuksesan besar adalah akumulasi dari kesuksesan-kesuksesan kecil, Kesuksesan-kesuksesan kecil adalah akumulasi dari kesulitan, halangan dan kegagalan yang mampu diatasi.

HALAMAN PERSEMBAHAN:

Laporan Proyek Akhir berjudul "Purwarupa sistem klasifikasi dan penghitung kendaraan bermotor menggunakan kamera *webcam* berbasis citra digital di PT. Industri Telekomunikasi Indonesia (Persero)" ini saya persembahkan untuk:

1. Kedua orang tua yang telah sedari dini memberi arti hidup dan menyokongku,
2. Untuk Almamaterku dan nama baiknya.
3. Pada para Dosen Kampus Program Diploma Teknik Elektro, Sekolah Vokasi Universitas Gadjah Mada,
4. Bapak Budi Bayu Murti, S.T., M.T. selaku pembimbing saya,
5. Bapak Parmono Raharjo. Ssi selaku pembimbing di PT. Industri Telekomunikasi Indonesia (Persero)
6. Bapak M. Rachmat Gunawan, S.T., M.T. selaku Kepala Bagian Divisi Pengembangan Produk PT. PT. Industri Telekomunikasi Indonesia (Persero)
7. Bapak Kasnanta Suwita selaku bagian urusan diklat PT. Industri Telekomunikasi Indonesia (Persero)
8. Semua Civitas Akademik Universitas Gadjah Mada

PRAKATA

Alhamdulillah, alhamdulillah, alhamdulillah segala pujian kepada Allah SWT atas berkah nikmat sehat dan nikmat waktu luang yang diberikan kepada penulis sehingga dapat menyelesaikan Laporan Proyek Akhir ini.

Proyek Akhir merupakan salah satu mata kuliah yang harus ditempuh oleh mahasiswa Program Diploma Teknik Elektro Sekolah Vokasi Universitas Gadjah Mada dan merupakan syarat bagi mahasiswa untuk memperoleh gelar ahli madya. Proyek Akhir sangat bermanfaat bagi saya, dan semoga memberi manfaat bagi mahasiswa, diantaranya mahasiswa dapat belajar secara *literature* lewat laporan Proyek Akhir yang langsung membahas sebuah permasalahan dan pemecahan permasalahan tersebut.

Penulis ingin mengucapkan terima kasih kepada para pihak yang telah membantu dalam pelaksaan Proyek Akhir ini maupun dalam penyusunan laporannya, diantaranya :

1. Bapak dan Ibu-ku yang selalu menjadi motivasiku, tiada henti hentinya mendoakan dan memberi nasihat. Aku bersyukur mempunyai orang tua seperti kalian
2. Bapak Ir. Lukman Subekti, M.T. selaku Ketua Program Diploma Teknik Elektro Sekolah Vokasi Universitas Gadjah Mada.
3. Bapak Budi Bayu Murti, S.T., M.T., selaku dosen pembimbing, terima kasih atas saran-saran dan bimbingannya, maaf kalau saya banyak membuat bapak kerepotan.
4. Bapak Kasnanta Suwita selaku bagian urusan diklat PT. Industri Telekomunikasi Indonesia (Persero).
5. Bapak Parmono Raharjo. SSi selaku pembimbing di Divisi Pengembangan Produk PT. Industri Telekomunikasi Indonesia (Persero).

6. Bapak Rachmat, Bapak Budi dan seluruh karyawan divisi Pengembangan Produk PT. Industri Telekomunikasi Indonesia (Persero) yang telah banyak membantu saya.
7. Apis, Fuji, Sidiq, Rizky, Wisnu Fajri dan teman-teman Program Diploma Teknik Elektro tahun angkatan 2012 seperjuangan dari awal hingga akhir.
8. Prima, Vicky, Zahrul, Rosman, Wildan, Yoga, Riki dan rekan Departemen IPTEK HMTE terima kasih atas semua hal dari awal berkarya sampai hari ini.
9. Akbar, Icang, Romzul, Satrio, Rico, Robit, Gusnaidi dan teman-teman konsentrasi Telekomunikasi 2012 *matur nuwun*.. TELKOM TETAP ISTIMEWA!
10. Mbak ulfa, Mas bangun, Mas Elfa, Mbak Har terima kasih semangat inspiratif dan motto bersama Agustus 2015 nya, selamat menempuh level selanjutnya.
11. Aji, Zaka, Fauzi, Syari, Vicky, Amira, Nanda, Nadia dan *dulur* IMAGE terima kasih dan terus berkarya untuk daerah tercinta.
12. Mas Tanto dan Mas Sugeng terima kasih atas *wejangan*-nya semoga kembali kepada anda manfaatnya.
13. Kakak-ku Dian terima kasih atas bantuannya.

Penulis menyadari bahwa laporan ini jauh dari kesempurnaan. Oleh karena itu, sangat mengharap kritik dan saran yang sifatnya membangun guna penulisan laporan-laporan selanjutnya. Semoga laporan ini dapat bermanfaat bagi rekan-rekan yang membacanya.

Yogyakarta, 29 Juni 2015

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

INTISARI

PURWARUPA SISTEM KLASIFIKASI DAN PENGHITUNG KENDARAAN BERMOTOR MENGGUNAKAN KAMERA *WEBCAM* BERBASIS CITRA DIGITAL

Oleh:

Wisnu Rizky Kurniawan

2012/327839/SV/15

Telah dibuat purwarupa sistem klasifikasi dan penghitung kendaraan bermotor menggunakan kamera *webcam*. Sistem ini dibuat dengan memanfaatkan *computer vision*. Metode yang digunakan dalam sistem ini adalah *Background Subtractor*, deteksi tepi dan kontur, *size thresholding*, serta *line intersect counting*. Sebelum dilakukan proses pengolahan citra, tahapan yang dilakukan pertama kali adalah proses pengambilan video dengan latar belakang jalan raya. Pada pengolahan citra, video jalan raya akan diproses dengan menggunakan *background subtractor*, sehingga *foreground* objek dapat dipisahkan dari latar belakangnya dengan syarat objek tersebut bergerak. *Foreground* tersebut kemudian di proses menggunakan deteksi tepi untuk menentukan *size thresholding* proses klasifikasi jenis kendaraan. Lalu pada *foreground* diberi indikator *tracking*. Saat indikator *tracking* tersebut menyentuh garis yang telah dibuat, penghitugan sistem akan memberikan nilai penambahan satu.

Pembuatan sistem ini memanfaatkan OpenCV 2.4.9 dengan menggunakan bahasa pemrograman Python. Pengujian dilakukan dengan menggunkan video jalan raya yang diambil secara langsung dan hasil *download* dari Internet. Kondisi jalan raya yang digunakan adalah kondisi jalan searah dengan kepadatan jalan yang berbeda dan perbedaan kondisi siang serta malam hari. Sedangkan proses ini akan bekerja secara optimal saat inputan video memiliki *framerate* 24 fps atau lebih dengan ukuran *frame* 320x240 *pixel*. Hal-hal yang berpengaruh terhadap hasil akhir dari sistem adalah keberadaan bayangan dari objek.

Kata kunci: *sistem klasifikasi dan penghitung kendaraan bermotor, webcam, video processing, Background Subtractor, deteksi tepi dan OpenCV*

ABSTRACT

VEHICLE CLASSIFICATION AND COUNTING SYSTEM PROTOTYPE USING WEB CAMERA BASED ON DIGITAL IMAGE

By:

Wisnu Rizky Kurniawan

2012/327839/SV/15

This is a prototype of a vehicle classification and counting system using web camera based on digital image. This prototype system created by using computer vision. The methods used in this prototype system are background subtractor, edge detection and contour, size thresholding, and also line intersect counting. Prior to image processing, first step is capture a video of highway. After that, that video will be execute using background subtractor. So, the image processing can capture its foreground from a background as we know as a highway. This algorithm will only capture a foreground if they moved, if not it will known as a background. After foreground detected, next step is edge detection. In there, we can decided the size thresholding for classification each vehicle. Finnaly we can label it like the classification of vehicle and also mark a point tracking. If the point intersect the line counting that the counting system will count it one and the next one will be same.

This prototype system using Open CV 2.4.9 and Python programming language. Tests carried out using video of highway taken from the webcam and from some page web. The highway condition is an one way highway with a different volume of the highway. Tests also trying to make it as adaptive as it can in different condition, like a sunny day and night day. From there except in adaptive condition, system have the best accurate counting in framerate 24 fps and 320x240 pixel size. The materials that affect the outcome of vehicle counting system are the presence of the shadows from the vehicle on the highway.

Keywords : *Vehicle classification and counting system, web camera, video processing, background subtractor, edge and contour detection, and OpenCV*

BAB I
PENDAHULUAN

1.1 Latar Belakang dan Permasalahan

Jalan raya adalah prasarana transportasi darat yang meliputi segala bagian jalan, termasuk bangunan pelengkap dan perlengkapannya yang diperuntukan bagi lalu lintas, yang berada pada permukaan tanah, di atas permukaan tanah, di bawah permukaan tanah dan/atau air, serta di atas permukaan air, kecuali jalan kereta api, jalan lori, dan jalan kabel (UU Nomor 38 Tahun 2004).

Seiring meningkatnya jumlah kendaraan bermotor setiap tahun yang berbanding terbalik dengan pembangunan infrastruktur jalan menjadi penyebab umum kemacetan lalu lintas yang sering berkelanjutan dan semakin buruk tiap tahunnya. Selain ketidakseimbangan jumlah kendaraan dan ketersediaan fasilitas jalan, kemacetan juga sering terjadi pada waktu-waktu tertentu dan di wilayah tertentu. Misalnya kemacetan sering terjadi di waktu pagi hari saat hari-hari kerja dimana karyawan menuju ke tempat kerja masing-masing, ruas jalan yang banyak kemungkinan macet adalah ruas-ruas jalan menuju tempat pekerjaan tersebut. Hal ini berlaku tidak hanya untuk karyawan, bagi pedagang khususnya pedagang barang-barang hasil agraris juga akan timbul hal yang sama terutama di tengah sebuah kota.

Akibat dari kemacetan yang berkepanjangan itu sendiri menyebabkan kerugian negara skala besar, dimulai dari rugi waktu, rugi bahan bakar dan keterlambatan distribusi kebutuhan daerah yang diangkut oleh kendaraan dari desa ke kota maupun sebaliknya.

Setiap permasalahan tentu memiliki sebuah jalan keluar, sama seperti kemacetan. Kemacetan dapat ditanggulangi, salah satunya dengan cara menciptakan sistem pengaturan lampu lalu lintas yang terdistribusi. Sistem ini akan membuat sebuah sistem lampu lalu lintas yang adaptif terhadap jalan raya. Selain adaptif terhadap kondisi tiap jalan raya, sistem ini juga dapat

membantu pihak dinas terkait untuk melakukan riset jangka panjang sehingga dapat membuat pengaturan rute kendaraan yang adaptif.

Konsep sistem tersebut telah direncanakan oleh PT. Industri Telekomunikasi Indonesia (INTI) untuk mengganti sistem sebelumnya yang telah diuji oleh Dinas Pusjatan Pekerjaan Umum (Pusjatan PU) khususnya wilayah kota Bandung.

1.2 Tujuan Penelitian

Tujuan yang ingin dicapai pada penelitian ini adalah membuat purwarupa 1 sistem penghitungan dan klasifikasi otomatis kendaraan yang melintas pada suatu jalan raya menggunakan kamera webcam berbasis citra digital yang akan dikembangkan untuk menciptakan *Intelligent Transportation System* (ITS) berupa rambu lalu lintas yang adaptif.

1.3 Manfaat Penelitian

Dari Proyek Akhir ini diharapkan dapat memberi manfaat sebagai berikut:

1. Mampu membedakan objek kendaraan bemotor dengan objek lain.
2. Mampu mengklasifikasikan sepeda motor dan mobil.
3. Mendapatkan informasi jumlah kendaraan yang melintas.
4. Mendapatkan spesifikasi optimal untuk tiap komponen sistem berdasarkan hasil penelitian.
5. Dapat dikembangkan dan diaplikasikan ke dalam sistem *Intelligent Transportation System* (ITS)

1.4 Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah merancang dan melakukan pengujian terhadap sistem klasifikasi dan penghitung kendaraan bermotor berbasis pengolah citra yang nantinya akan dikembangkan menuju sistem *Intelligent Transportation System*

.

1.5 Batasan Masalah

Batasan masalah dalam penelitian Proyek Akhir:

a. Data yang diolah berupa ekstrak hasil rekaman video.
b. Data citra berupa video yang diperoleh dari pengambilan data menggunakan kamera *webcam* maupun hasil *download* dari media youtube dengan latar belakang kondisi jalan raya.
c. Metode deteksi dan penghitung dilakukan pada kendaraan yang bergerak.
d. Deteksi kendaraan roda empat atau lebih dilakukan pada saat siang hari atau pada saat intensitas cahaya matahari besar.
e. Data masih terbatas pada bayangan yang mengikuti benda asli yang bergerak, bayangan yang menyambungkan dua objek atau lebih dan objek besar yang menghalangi objek lain.
f. Pada Purwarupa 1 ini sistem penghitung dan klasifikasi masih belum terintegrasi atau masih terpisah.

1.6 Metodologi Pelaksanaan Proyek Akhir

Dalam pelaksanaan pembuatan Proyek Akhir ini, penulis menggunakan beberapa metode penelitian, yaitu :

1. **Studi *literature* dan referensi**

   Mencari dan mempelajari referensi tentang deteksi, *tracking*, penghitung objek bergerak dan referensi tentang OpenCV, pengolah citra, *video processing*, bahasa pemrograman *python* yang sesuai dengan penelitian tugas akhir yang dilakukan.

2. **Konsultasi dan diskusi**

   Melakukan konsultasi dengan dosen pembimbing, pembimbing lapangan ataupun dengan orang-orang yang berkompeten di bidang yang sesuai dengan judul penelitian menggunakan pengolahan citra digital. Hasil yang akan didapatkan berupa metode yang tepat untuk melalakukan deteksi, *tracking* dan penghitungan kendaraan di jalan raya maupun tol.

3. **Rancangan metode penghitung kendaraan di jalan raya atau jalan tol**

   Rancangan metode penghitung kendaraan meliputi rancangan logika untuk mendeteksi objek terkait, melakukan *tracking* dan menghitung kendaraan. Dalam hal pendeteksian objek digunakan metode *Background Subtraction*, deteksi tepi dan deteksi kontur.

4. **Implementasi**

   Implementasi metode pendeteksi objek, *tracking* objek dan penghitungan jumlah objek menggunakan OpenCV 2.4.9. Pengambilan video dilakukan menggunakan kamera *webcam* dan juga mengambil video dari internet.

5. **Pengujian dan pembahasan**

   Pengujian metode klasifikasi dan penghitung kendaran bermotor dilakukan dengan memproses empat sampel video kondisi jalan raya untuk diketahui keterkaitan keakuratan hasil hitung dan klasifikasi dengan hasil manual, serta pengujian kemampuan deteksi sistem pada kondisi malam hari.

1.7 Sistematika Penulisan Laporan

Dalam penyusunan laporan proyek akhir ini, penulis menggunakan sistematika sebagai berikut :

BAB I PENDAHULUAN

Berisi tentang latar belakang masalah, tujuan, batasan masalah, waktu dan tempat pelaksanaan, metodologi pelaksanaan kerja praktek dan sistematika penulisan.

BAB II LANDASAN TEORI

Pada bagian ini dijelaskan mengenai metode-metode yang digunakan serta algoritma sistem yang dibuat.

BAB III PERANCANGAN DAN IMPLEMENTASI SISTEM

Bab ini berisi tentang perancangan dan implementasi sistem penghitung jumlah kendaraan roda empat lebih.

BAB IV HASIL PENELITIAN DAN PEMBAHASAN

Pada bab ini akan dijelaskan mengenai proses pengujian dan hasil purwarupa sistem klasifikasi dan penghitung jumlah kendaraan bermotor serta pembahasan hasil pengujian.

BAB IV PENUTUP

Pada bab ini membahas tentang kesimpulan dan saran, sehingga purwarupa sistem ini dapat diperbaiki dapat dikembang lebih lanjut.

DAFTAR PUSTAKA

Berisi tentang referensi-referensi yang telah dipakai oleh penulis sebagai acuan dan penunjang serta parameter yang pendukung penyelesaian masalah ini baik secara praktis maupun teoritis.

BAB II

LANDASAN TEORI

2.1 Kemacetan

Definisi kata “macet” menurut Kamus Besar Bahasa Indonesia memiliki arti tidak dapat berfungsi dengan baik, terhenti, atau tidak lancar, sedangkan kata “kemacetan” artinya hal (keadaan) macet [1]. Kondisi kemacetan lalu lintas merupakan suatu keadaan kondisi jalan jika tidak ada keseimbangan antara kapasitas jalan dengan jumlah kendaraan yang melalui jalan tersebut. Gejala ini ditandai dengan kecepatan kendaraan yang semakin lamban bahkan berhenti, jarak antar kendaraan yang satu dengan yang lain rapat, dan pengemudi tidak dapat menjalankan kendaraan dengan kecepatan yang diinginkannya[2].

Beberapa upaya yang dapat dilakukan untuk mengurangi tingkat kemacetan yaitu [3]:

A. Peningkatan koordinasi transportasi,
B. Penambahan kapasitas jalan,
C. Meningkatkan infrastuktur jalan,
D. *Supply and demand*,
E. Mengajak masyarakat untuk hidup *go green*,
F. Anjuran kepada para pengendara,
G. *Intelligent Transportation System*,

2.2 *Intelligent Transportation System*

Saat sistem transportasi di seluruh dunia mulai dirasa meningkatkan peluang tingkat ekonomi dan sosial, hal tersebut juga menimbulkan berbagai masalah seperti kemacetan, keamanan, dan kerusakan lingkungan. *Intelligent Transportation System* (ITS) adalah aplikasi dari teknologi informasi dan komunikasi dalam sistem transportasi. *Intelligent Transportation System* (ITS) digunakan untuk meningkatkan mobilitas, mengurangi angka kecelakaan dan kematian, serta melestarikan lingkungan.

Sasaran dari *Intelligent Transportation System* (ITS) ini sendiri meliputi:

A. Memantau lalu lintas dan kondisi lingkungan di jalan,
B. Memaksimalkan keamanan operasional dan efisiensi dari jaringan,
C. Meminimalisir dampak negatif yang disebabkan dari kemacetan yang sering terjadi maupun kemacetan yang tidak sering terjadi akibat insiden tertentu,
D. Menyediakan pengguna jalan informasi yang dibutuhkan untuk membantu pengambilan keputusan di perjalanan dan meringankan beban mental dan *stress* ketika berkendara.

Tujuan tersebut dapat dicapai apabila komponen berikut diidentifikasi sebagai fungsi utama, yaitu:

A. *Maintaining road serviceability and safety*,
B. *Traffic control,*
C. *Travel aid and user information,*
D. *Demand management,*
E. *Network monitoring.*

2.3 *Computer Vision*

Computer vision adalah sebuah disiplin ilmu yang mempelajari bagaimana merekonstruksi, mengintrepretasikan, dan memahami sebuah tampilan 3 dimensi dari tampilan 2 dimensinya dalam hal sifat dari struktur tampilan tersebut. *Computer Vision* berkaitan dengan pemodelan dan meniru penglihatan manusia dengan menggunakan perangkat lunak dan perangkat keras pada komputer. *Computer Vision* menggabungkan ilmu pengetahuan dalam bidang ilmu komputer, teknik elektro, matematika, fisiologi, biologi, dan ilmu kognitif. Diperlukan ilmu dari semua bidang tersebut untuk memahami dan menyimulasikan pengoperasian penglihatan manusia.

Perlunya metode untuk mendekati kemampuan manusia dalam menangkap informasi, sebuah teknologi *Computer Vision* harus terdiri dari banyak fungsi pendukung yang berfungsi secara penuh. Fungsi-fungsi pendukung tersebut antara lain:

A. Proses penangkapan citra atau gambar (*image acquisition*)

Image Acquisition pada manusia dimulai dari mata, kemudian informasi visual diterjemahkan ke dalam suatu format yang kemudian dapat dimanipulasi oleh otak. Seperti halnya proses tersebut, *Computer Vision* membutuhkan sebuah mata untuk menangkap sebuah sinyal visual. Kamera akan menerjemahkan sebuah *scene* atau *image.* Keluaran dari kamera berupa sinyal analog, dimana frekuensi dan amplitudonya merepretasikan detail ketajaman (*brightness*) pada *scene* (frekuensi berhubungan dengan jumlah sinyal dalam satu detik, sedangkan amplitudo berkaitan dengan tingginya sinyal listrik yang dihasilkan). Kamera mengamati sebuah kejadian pada satu jalur dalam satu waktu, memindainya dan membaginya menjadi ratusan garis horizontal yang sama. Tiap-tiap garis membuat sinyal analog yang amplitudo menjelaskan perubahan *brightness* sepanjang garis sinyal tersebut. Komputer tidaklah bekerja dengan sinyal analog, oleh karena itu diperlukan *Analog to Digital Converter* (ADC), dibutuhkan untuk memproses semua sinyal tersebut oleh komputer. ADC ini akan mengubah sinyal analog yang direpresentasikan dalam bentuk informasi sinyal tunggal ke dalam sebuah aliran (*stream*) sejumlah bilangan biner. Bilangan biner ini kemudian disimpan di dalam memori dan akan menjadi data *raw* yang akan diproses.

B. Proses pengolahan citra (*image processing*)

Pada proses ini *Computer Vision* akan melibatkan sejumlah manipulasi utama (*initial manipulation*) dari data *binary* yang dihasilkan pada proses *image acquisition. Image Processing* membantu peningkatan dan perbaikan kualitas *image,* sehingga dapat dianalisa dan diolah lebih jauh secara efisien. *Image processing* akan meningkatkan perbandingan sinyal terhadap *noise*, rumus dari *signal to ratio* ditunjukkan pada persamaan 2.1 berikut:

$$Signal\ to\ noise\ ratio = \frac{s}{n} \quad \textbf{(2.1)}$$

Keterangan:

Signal to noise ratio = Daya derau(dB)

S = Kekuatan sinyal

N = kekuatan derau (*noise*)

Sinyal-sinyal tersebut adalah informasi yang akan merepresentasikan objek yang ada dalam *image*, sedangkan *noise* adalah segala bentuk interferensi dan pengaburan yang terjadi pada sebuah objek.

C. Analisa data citra (*Image Analysis*)

Pada tahap ini, *scene* akan dieksplorasi ke dalam bentuk karakteristik utama dari objek melalui suatu proses investigasi. Sebuah program komputer akan mulai melihat melalui bilangan biner yang merepresentasikan informasi visual untuk mengidentifikasikan fitur-fitur spesifik dan karakteristiknya. Pada proses yang lebih khusus lagi, program *image analysis* digunakan untuk mencari tepian batas-batas objek dalam *image.* Sebuah tepian (*edge*) terbentuk antara objek dan latar belakangnya atau antara dua objek yang spesifik. Tepi ini akan terdeteksi sebagai akibat dari perbedaan level *brightness* pada sisi yang berbeda dengan salah satu batasnya.

D. Proses pemahaman data citra (*Image Understanding*)

Proses ini merupakan langkah terakhir dalam proses *Computer Vision,* dimana spesifik objek dan hubungannya diidentifikasi. Pada bagian ini akan melibatkan kajian tentang teknik-teknik *artificial intelligent. Understanding* berkaitan dengan *template matching* yang ada dalam sebuah *scene.* Metode ini menggunakan program pencarian (*search program*) dan teknik penyesuaian pola (*pattern matching techniques*).

2.4 Citra

Citra merupakan representasi (gambaran), kemiripan, atau imitasi dari suatu objek. Citra sebagai keluaran suatu sistem perekaman data yang dapat bersifat optik maupun foto, bersifat analog berupa sinyal-sinyal video gambar seperti pada suatu media penyimpanan. (Sutoyo, dkk, 2009)

Sebuah citra adalah kumpulan piksel-piksel yang disusun dalam larik dua dimensi. Indeks baris dan kolom (x,y) dari sebuah piksel dinyatakan dalam bilangan bulat. Piksel (0,0) terletak pada sudut kiri atas pada citra, indeks x bergerak ke kanan dan indeks y bergerak ke bawah. Konvensi ini merujuk pada cara penulisan larik yang digunakan dalam pemrograman komputer. (Ahmad, 2005)

Sebuah citra digital dapat diwakili oleh sebuah matriks yang terdiri dari M kolom dan N baris, dimana perpotongan antara kolom dan baris disebut piksel (piksel = *picture element*), yaitu elemen terkecil dalam sebuah citra.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & f(0,M-1) \\ f(1,0) & \ddots & f(1,M-1) \\ f(N-1,0) & f(N-1,1) & f(N-1,M-1) \end{bmatrix} \quad \textbf{(2.2)}$$

Piksel mempunyai dua parameter, yaitu koordinat dan intensitas atau warna. Nilai yang terdapat pada koordinat (x,y) adalah f(x,y), yaitu besar intensitas atau warna dari piksel di titik itu. Oleh sebab itu, sebuah citra digital dapat ditulis dalam bentuk matriks di atas.

Berdasarkan gambaran tersebut, secara sistematis citra digital dapat dituliskan sebagai fungsi intensitas f(x,y), dimana harga x (baris) dan y (kolom) merupakan koordinat posisi dan f(x,y) adalah nilai fungsi pada setiap titik (x,y) yang menyatakan besar intensitas citra atau tingkat keabuan atau warna dari piksel di titik tersebut. Pada proses digitalisasi (sampling dan kuantitas) diperoleh besar baris M dan kolom N hingga citra membentuk matriks M x N dan jumlah tingkat keabuan piksel G. Jenis-jenis citra digital:

a. Citra Biner (Monokrom)

   Banyaknya warna hanya dua, yaitu hitam dan putih. Dibutuhkan 1 bit di memori untuk menyimpan kedua warna ini.

b. Citra *Grayscale*

   Banyaknya warna tergantung pada jumlah bit yang disediakan di memori untuk menampung kebutuhan warna ini. Semakin besar jumlah bit warna yang disediakan di memori, semakin halus gradasi warna yang terbentuk.

c. Citra Warna (*True Color*)

   Setiap piksel pada citra warna mewakili warna yang merupakan kombinasi dari tiga warna dasar yaitu merah, hijau dan biru. Setiap warna dasar menggunakan penyimpanan 1 byte (8 bit), yang berarti setiap warna mempunyai kombinasi sebanyak 16 juta warna lebih. Itulah sebabnya format ini dinamakan true color karena mempunyai jumlah warna yang cukup besar sehingga bisa dikatakan hampir mencakup semua warna di alam. Satu piksel citra true color diwakili oleh 3 byte, dimana masing-masing byte mempresentasikan warna merah, hijau dan biru.

2.5 Pengolahan Citra Digital

Pengolahan Citra digital adalah sebuah disiplin ilmu yang mempelajari hal-hal yang berkaitan dengan perbaikan kualitas gambar (peningkatan kontras, transformasi warna, restorasi citra). Transformasi gambar (rotasi, translasi, skala transformasi geometrik), melakukan pemilihan citra ciri (*feature images*) yang optimal untuk tujuan analisis, melakukan proses penarikan informasi atau deskripsi objek atau pengenalan objek yang terkandung pada citra, melakukan kompresi atau reduksi data untuk penyimpanan data, transmisi data, dan waktu proses input. Input dari pengolah citra adalah citra, sedangkan outputnya adalah citra hasil pengolahan.

2.6 *Thresholding*

Threshold adalah metode di mana kita menetapkan suatu nilai, kemudian kita hanya mengambil nilai di atasnya saja atau di bawahnya saja, sedangkan nilai selainnya diabaikan (Bradski dan Kaehler, 2008). Hasil dari *treshold* adalah citra yang tampak nyata perbedaan intensitasnya, biasanya hitam dan putih. Metode ini biasanya digunakan untuk segmentasi atau pemisahan suatu objek dengan selainnya.

2.7 OpenCV

OpenCV adalah sebuah *library* yang berisi fungsi-fungsi pemrograman untuk teknologi *Computer Vision* secara *real time.* OpenCV bersifat *open source,* bebas digunakan untuk hal-hal yang bersifat akademis maupun komersial.Di dalam OpenCV, terdapat *interface* untuk bahasa pemrograman C, C++, Python, dan nantinya Java yang dapat berjalan pada Windows, Linux, Android dan Mac. Terdapat lebih dari 2500 algoritma dalam OpenCV, digunakan di seluruh dunia, telah lebih dari 2.5 juta kali diunduh, dan digunakan lebih dari 40 ribu orang. Penggunaanya antara lain pada seni interaktif, inspeksi tambang, menampilkan peta di *web* melalui teknologi robotik.

Pada awalnya OpenCV ditulis dengan menggunakan bahasa C namun sekarang secara menyeluruh sudah menggunakan antarmuka bahasa C++ dan seluruh pengembangannya terdapat dalam format bahasa C++. Contoh aplikasi dari OpenCV yaitu interaksi manusia dengan komputer: identifikasi, segmentasi, pengenalan objek, pengenalan wajah, pengenalan gerakan, penelusuran gerakan, gerakan diri, dan pemahaman gerakan, struktur dari gerakan, kalibrasi *stereo* dan beberapa kamera serta komputasi mendalam, dan robotik.

Fitur-fitur yang terdapat pada OpenCV antara lain:

A. Manipulasi data *image* (alokasi, rilis, duplikasi, pengaturan, konversi),

B. *Image* dan I/O video (masukan berbasis *file* dan kamera, keluaran *image*/*video file*),
C. Manipulasi matriks dan vektor serta aljabar linear (produk, solusi, *eigenvalues*, SVD),
D. Beragam struktur data dinamis (daftar, baris, grafik),
E. Dasar pengolahan citra (filter, deteksi tepi, deteksi sudut, pengambilan sampel, dan interpolasi, konversi warna, operasi morfologi, *histogram*),
F. Analisis struktur (komponen yang berhubungan, pengolah kontur, transformasi jarak, variasi momen, transformasi *Hough*, perkiraan *polygonal*, penyesuaian garis, Delaunay *triangulation*),
G. Kalibrasi kamera (menemukan dan menelusuri pola kalibrasi, dasar estimasi matriks, estimasi homografi, korespondensi *stereo*),
H. Analisis gerakan (*otical flow*,segmentasi gerakan, penelusuran),
I. Pengenalan objek (metode *eigen*, HMM),
J. Dasar *Graphical User Interface* atau GUI (menampilkan *image*/video, penanganan *mouse* dan *keyboard*, *scroll-bars*),
K. Pelabelan *image* (garis,poligon,gambar teks).

Modul-modul terdapat pada OpenCV antara lain:

A. *cv*-fungsi utama OpenCV,
B. *cvaux*-fungsi penolong OpenCV,
C. *cxcore*-pendukung struktur data dan aljabar linear,
D. *highui*-fungsi *Graphical User Interface* (GUI).

2.8 *Background Substraction*

Background subtraction banyak digunakan pada proyek-proyek berbasis pengolah citra. Salah satu aplikasi yang sering menggunakan fungsi dari *background substraction* ini adalah aplikasi penghitung jumlah pengunjung yang memasuki maupun meninggalkan ruangan ataupun kendaraan yang melewati suatu jalur dalam sistem informasi lalu lintas. Metode ini

memisahkan manusia atau kendaraan dengan cara pembedaan latar belakang (*background*) dan manusia atau kendaraan (*foreground*) yang bergerak.

Jika kondisi yang akan diamati oleh *background subtraction* hanya berupa latar belakang dan objek bergerak yang akan diamati, hal tersebut sangat mudah. Namun, ketika terdapat objek lain yang juga berpindah dari titik satu ke titik yang lain misalnya bayangan dari objek tersebut hal ini akan diproses juga sebagai *foreground* atau masuk ke dalam objek yang diklasifikasikan. *Background Subtraction* pada OpenCV memiliki tiga algoritma yang sering diimplementasikan, diantaranya:

A. *Background Subtractor MOG*

Algoritma ini menggunakan metode *Gaussian Mixture* berdasarkan pemisahan latar belakang objek dan objek yang akan diproses (*background/foreground segmentation*). Algoritma ini pertama kali diperkenalkan lewat sebuah *paper* berjudul "*An improved adaptive background mixture model for real-time tracking with shadow detection*" oleh P. Kadew TraKupong dan R. Bowden di tahun 2001.

Metode yang digunakan adalah memodelkan setiap *pixel background* oleh campuran distribusi K Gaussian. Dengan nilai K berkisar dari 3 sampai 5. Nilai K diambil dari campuran warna yang ada di setiap waktu (*scene*). Perbedaan warna akan semakin besar ketika terdapat dua indikator yaitu indikator terlalu sedikit bergerak atau bahkan tidak bergerak sama sekali dengan indikator yang selalu bergerak.

B. *Background Subtractor MOG 2*

Sama seperti halnya *Background Subtractor MOG*, *Background Subtractor MOG 2* juga berdasarkan pemisahan latar belakang objek dan objek yang akan diproses (*background/ foreground segmentation*). Algoritma ini diambil dari dua *paper* yaitu "*Improved adaptive Gaussian mixture model for background*

subtraction" di tahun 2004 dan "*Efficient adaptive density estimation per image pixel for the task of background subtractor*" di tahun 2006, kedua *paper* itu ditulis Z. Zivkovic.

Salah satu hal atau fitur yang penting disini adalah bagaimana setiap warna *pixel* memiliki nilai distribusi Gaussian, berbeda dengan *Background Subtractor MOG* yang memilih nilai distribusi berdasarkan nilai k. Hal tersebut membuat *Background Subtractor MOG 2* lebih adaptif pada kondisi perubahan pencahayaan atau iluminasi.

C. *Background Subtractor GMG*

Algoritma *Background Subtractor GMG* menggabungkan estimasi stastik latar belakang (*statistical background image estimation*) dan segmentasi Bayesian per piksel (*per-pixel bayesian segmentation*). Algoritma ini pertama kali diperkenalkan oleh Andrew B. Godbehere, Akihiro Matsukawa, Ken Goldberg pada *paper* mereka yang berjudul "*Visual Tracking of human visitors under variable-light conditions for a responsive audio art installation*" pada tahun 2012.

Algoritma ini memungkinkan segmentasi *foreground* yang akan mengidentifikasi *foreground* pada kondisi yang lebih kompleks atau lebih sulit membedakan latar belakang dan *foreground*-nya melalui inferensi Bayesian (*Bayesian Inference*). Estimasi dapat adaptif juga, pengamatan atau pendeteksian yang baru akan lebih observatif dibanding pengamatan atau pendeteksian yang sebelumnya. Selain hal tersebut, *Background Subtractor GMG* juga dapat menghilangkan *noise* atau gangguan seperti bayangan dari objek yang bergerak.

2.9 Deteksi Tepi

Detektor tepi merupakan proses untuk menemukan perubahan intensitas yang berbeda nyata dalam sebuah citra (Sutoyo, dkk, 2009). Deteksi tepi merupakan pendekatan yang paling umum untuk pendeteksian diskontinuitas nilai intensitas (Prasetyo, 2011).

2.10 Contoh Sistem pemantau lalu lintas yang telah diimplementasikan

Sistem pemantau lalu lintas sebenarnya sudah pernah ditemukan oleh orang lain sebelumnya. Namun, metode yang digunakan dapat berbeda-beda. Tujuan penulisan contoh sistem pemantau lalu lintas yang ditemukan oleh orang lain ini adalah sebagai perbandingan dengan sistem yang dibuat.

A. Sistem Pengaturan lampu lalu lintas terdistribusi

Sistem Pengaturan lampu lalu lintas terdistribusi ini bertujuan untuk membuat sebuah sistem lampu lalu lintas yang adaptif terhadap kondisi jalan di sekitarnya. Sistem ini memerlukan sistem pemantau lalu lintas sebagai input untuk mengukur tingkat kemacetan jalan di sekitarnya. Sistem ini menggunakan metode *blob tracking, haar training* dan algoritma *Principal Component Analysis* dalam mendeteksi kendaraan. Sistem ini juga menggunakan OpenCV untuk memakai algoritma tersebut.

Kelemahan dari sistem ini antara lain:

i. Kesalahan intepretasi dua objek yang berdekatan, dan sistem menganggapnya satu objek.
ii. Kesalahan intepretasi satu objek menjadi dua objek berbeda
iii. Sistem memiliki tingkat akurasi yang berbeda-beda antara kondisi waktu pagi, siang dan malam.
iv. Sistem mengalami penurunan akurasi pengenalan pada saat kondisi jalan macet.

B. *Traffic Jam Detection System*

Sistem ini bertujuan untuk mendeteksi kendaraan dan kecepatan dari kendaraan tersebut. Sistem ini menggunakan *blob detection* dengan latar belakang jalan yang kosong untuk mendeteksi kendaraan tersebut. Sistem ini menggunakan bahasa pemrograman .Net dan C. Secara umum tahapan yang dilakukan yaitu:

i. *Image analysis,* Menganalisis tiap citra dengan syarat citra haruslah berupa jalan yang kosong.
ii. *Object detection,* citra jalan tersebut kemudian diproses melalui metode *lane masking,* dimana jalan akan diubah warnanya menjadi warna *grayscale,* kemudian *blob detection* akan menjadi indikator deteksi objek.
iii. *Counting detected object*, Tiap *blob* akan dianggap sebagai nilai satu kemudian masing-masig dijumlahkan.
iv. *Motion detection,* metode ini akan memperkirakan nilai kecepatan dari kendaraan dengan menggunakan algoritma *image differentiation*.
v. *Display*, menunjukan pada pengguna jumlah objek terdeteksi dan nilai estimasi kecepatan.

Sistem ini telah diuji pada beberapa kondisi. Hasil pengujian menunjukan bahwa sistem ini membutukan waktu tiga sampai empat detik untuk mendeteksi objek dan kecepatannya dalam citra berukuran 1024x768 piksel.

C. *Image processing in road traffic analysis*

Sistem ini bertujuan untuk mendeteksi kemacetan, kecepatan, dan plat nomor kendaraan. Sistem ini menggunakan metode lane masking, metode blob detection menggunakan citra jalan yang kosong dan juga algoritma edge detection. Secara umum tahapan yang dilakukan yaitu:

i. *Lane masking*, Proses pemisahan citra dari citra lain yang tidak diperlukan, sehingga citra yang akan diproses hanyalah citra jalan raya.

ii. *Background elimination*, Proses pemisahan citra kendaraan dari citra jalan raya atau yang tidak terkait dengan kendaraan.

iii. *Noise & blob filtration,* proses menghilangkan *noise* yang berada pada sekitar kendaraan

iv. *Contour extraction,* menggunakan metode *edge detection* sehingga mendapatkan komponen penyusun *contour labelling*.

v. *Contour labelling*, Proses memberi tanda pada objek yang terdeteksi

vi. *Vehicle tracking,* Proses memberi tanda objek dan tanda tersebut tetap mengikuti objek yang bergerak sekalipun.

D. *The implementation of a vision sensor for traffic surveillance*

Sistem ini menggunakan metode *open model for network-wide heterogeneous intersection-based transport management* (OMNI). Metode ini menggunakan sudut pandang jalan secara vertikal dan memanfaatkan lebar tiap lajur pada jalan untuk menghitung jumlah kendaraan. Sistem ini selain berfungsi memantau kondisi lalu lintas, juga dapat berfungsi sebagai pendeteksi plat nomor kendaraan.

BAB III
PERANCANGAN DAN IMPLEMENTASI SISTEM

3.1 Rancangan Sistem Klasifikasi dan Penghitung Kendaraan

Sistem klasifikasi dan penghitung kendaraan bermotor dengan berbasis citra digital ini memiliki metode perancangan logika dan perancangan perangkat keras. Perancangan logika meliputi kemampuan dalam mendeteksi kendaraan. Perancangan perangkat keras digunakan untuk mengambil data. Sistem yang dibuat menggunakan perangkat keras dan perangkat lunak, sebagi berikut:

a. Laptop ASUS A450L *Series* Intel Core i5 1,6 GHz RAM 4 GB
b. Open CV 2.4.9
c. Sistem Operasi Ubuntu 14.04 LTS

Perancangan Sistem Penghitung Kendaraan ditunjukan pada gambar berikut:

Gambar 3.1 Perancangan sistem klasifikasi dan penghitung kendaraan

Pada Gambar 3.1 di atas dijelaskan bahwa sistem terdiri dari tahap pengambilan video, pengolahan citra video, dan proses deteksi, *tracking*, klasifikasi dan penghitung tiap kendaraan bermotor. Langkah yang pertama dilakukan adalah pengambilan video. Pada langkah ini, video diambil dengan cara men-*download* dari internet maupun pengambilan video dengan dengan menggunakan kamera *webcam* Logitech HD 720p dengan fitur Autofokus yang ditunjukkan oleh Gambar 3.2 di bawah ini.

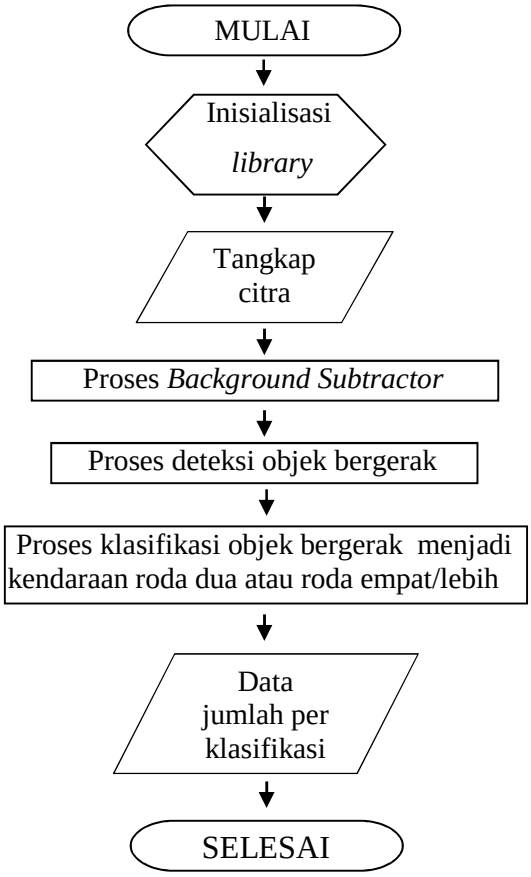
Gambar 3.2 Kamera *webcam* yang digunakan

Setelah tahap pengambilan video, tahap selanjutnya adalah pengolahan citra video dengan menggunakan laptop atau *Personal Computer* (PC). Dalam tahap *Video Processing* ini menggunakan OpenCV 2.4.9 pada sistem operasi Ubuntu 14.04 LTS. Pada proses pengolahan citra video ini dilakukan deteksi, *tracking*, klasifikasi dan penghitung kendaraan. Sehingga output yang didapatkan adalah klasifikasi kendaraan bermotor dan penghitung kendaraan bermotor.

3.2 Deskripsi Sistem

Sistem secara menyeluruh yang akan dibangun merupakan sebuah sistem yang digunakan untuk mengitung jumlah klasifikasi kendaraan bermotor. Dimana sistem yang dibangun merupakan perangkat lunak yang mampu mengambil dan memasukkan video, kemudian mendeteksi kendaraan bermotor, mengklasifikasinya dan kemudian menghitung jumlah tiap klasifikasi kendaraan bermotor tersebut. Hanya saja pada purwarupa 1 ini, antara sistem klasifikasi dan sistem penghitung belum terintegrasi. Proses pengambilan dan memasukkan video sebagai citra yang akan diolah ini merupakan *preprocessing* citra. Selanjutnya akan dilakukan proses *Backgound Subtractor*, sehingga objek (*foreground*) akan terpisah dengan latar belakang citra (*background*). Ketika objek yang berupa *foreground* mulai dapat dipisahkan dari latar belakangnya, akan lebih mudah menandai objek sesuai *Region of Interest* (ROI) yang kita inginkan. *Region of Interest* (ROI) pada sistem ini ditentukan melalui ukuran-ukuran sesuai *threshold* objek yang akan dideteksi dan sesuai klasifikasi kendaraan bermotor. Setelah satu/lebih objek

telah terdeteksi ditempakanlah tanda(*labelling*) klasifikasi kendaraan roda dua dan kendaraan roda lebih dari dua. Selain setiap objek diklasifikasi, masing-masing objek tersebut kemudian diberi tanda berupa titik yang mengikuti pergerakan objek (*tracking*). Saat objek dan titik yang terbuat sebelumnya memenuhi nilai sebuah koordinat y maka proses penghitungan akan bertambah satu. Proses secara keseluruhan ini dijelaskan dalam Gambar 3.3 berikut:



Gambar 3.3 *Flowchart* kerja sistem

3.3 Perancangan Logika Algoritma

Algoritma yang akan dipakai dalam sistem ini dibuat menggunakan bahasa pemrograman Python. Bahasa Pemrograman Python memiliki banyak keunggulan dibandingkan dengan bahasa pemrograman lain yaitu cocok digunakan untuk membuat sebuah *prototype*, bahasa pemrograman ini juga cocok digunakan pada *embedded system* sehingga dapat dilakukan pengembangan ke depannya.

Algoritma yang akan dibuat menggunakan fasilitas *video processing* dari Open CV (Open Computer Vision). Video yang akan diolah oleh algoritma diambil dari video-video pantauan lalu lintas secara *real time*.

a. *Preprocessing Image*

Awal kerja algoritma sistem dimulai dari pembukaan video, kemudian masuk ke tahap *preprocessing image* dimana tiap *frame* video yang terwakili oleh gambar akan diproses di sini. Setiap Gambar pada *frame* video tersebut diproses berdasarkan perbedaan antara latar belakang objek (*background*) dengan objek (*foreground*). Pemisahan latar belakang objek (*background*) dengan objek *(foreground)* akan menggunakan metode *Background Subtraction* dari Open CV.

b. Deteksi Objek

Proses Deteksi Objek ini terdiri atas 4 sub proses sebagai berikut:

i. *Background Substraction*

Metode *Background Subtraction* akan mengubah citra gambar tiap *frame*. Tiap citra gambar akan diubah dari gambar berwarna menjadi gambar hitam putih (*grayscale*). Perbedaan warna hitam dan putih dari citra diambil dari indikator bergerak atau tidaknya suatu objek tiap *frame* gambar pada video. Jika objek bergerak tiap *frame* gambar pada video maka objek tersebut akan berwarna putih, begitu pula sebaliknya jika objek tidak bergerak maka objek akan berwarna hitam.

ii. Deteksi tepi objek

Setelah didapatkan pembeda antara objek yang bergerak dengan objek yang tidak bergerak lewat Metode *Background Subtraction*, akan lebih mudah menempatkan tanda (*region of interest* atau ROI) pada objek tersebut. Penempatan tanda pada objek yang bergerak dilakukan dengan metode deteksi tepi. Metode deteksi tepi ini akan memberikan tanda titik-titik yang membentuk garis pada sisi terluar objek.

iii. Penentuan koordinat titik terluar

Setelah terbentuk tanda pada tiap tepi objek, akan lebih mudah mengetahui nilai-nilai koordinat untuk titik koordinat tanda (*region of interest* atau ROI). Koordinat titik yang diperlukan untuk membuat sebuah tanda (*region of interest* atau ROI) adalah koordinat maksimum tepi (diwakili oleh nilai koordinat titik terluar objek, nilai tinggi dan lebar pada objek.

iv. Menggambar kotak atau persegi pada objek

Setelah diketahui nilai koordinat titik terluar objek, nilai tinggi dan lebar pada objek. Langkah Proses selanjutnya adalah menggambarkan tanda kotak atau persegi pada *Region of Interest* berdasarkan ketiga nilai tersebut.

v. Klasifikasi kendaraan

Proses klasifikasi kendaraan dapat dilakukan berdasarkan nilai *threshold* tinggi dan/atau nilai lebar objek sehingga dapat diklasifikasikan minimal dua macam klasifikasi yaitu kendaraan roda dua dan kendaraan roda empat.

c. *Tracking Objek*

Setelah objek diberi tanda berupa kotak atau *Region of Interest* (ROI), selanjutnya dapat diketahui nilai koordinat titik tengah persegi tersebut. Nilai koordinat titik tengah tersebut didapatkan dari hasil penambahan hasil bagi dua nilai tinggi dengan hasil bagi dua nilai lebar objek. Setelah koordinat titik tengah diketahui, kemudian pada koordinat titik tersebut digambar atau ditandai dengan lingkaran kecil yang nantinya mewakili indikator *tracking* objek.

d. Penghitung Kendaraan

Setelah indikator *tracking* objek didapat, akan lebih mudah melakukan penghitugan tiap objek. Metode penghitungan objek dilakukan dengan menggambar garis berupa koordinat (x_1,y_1) dan (x_2,y_2) dengan nilai y_1 sama dengan y_2 . Garis tersebut harus melintang dari sisi kanan jalan sampai sisi kiri jalan. Pengitungan akan dilakukan apabila indikator *tracking* objek (berupa titik atau lingkaran kecil) memotong koordinat garis yang dibuat.

3.4 *User Requirement*

Spesifikasi perangkat yang digunakan pada penelitian ini adalah:

a. Kamera

Spesifikasi kamera yang digunakan dalam penelitian ini menggunakan kamera *webcam* Logitech HD 720p dengan fitur Autofokus.

b. Laptop

Spesifikasi laptop yang digunakan dalam penelitian ini adalah:

i. *Processor* laptop : Intel Core i-5
ii. RAM laptop : 4 GB
iii. VGA *Card* : NVDIA GEFORCE 720M
iv. *Hard Disk Drive* (HDD) : 500 GB

3.5 *Background Subtractor*

Setelah langkah Pemrosesan awal atau *preprocessing image* yang membuka *file* video, selanjutnya adalah proses pemisahan objek (*foreground*) dengan latar belakang objek (*background*). Proses pemisahan objek dengan latar belakangnya disebut *Background Subtractor*, *Background Subtractor* sendiri merupakan fitur yang telah disediakan oleh OpenCV. Program *preprocessing image* ditunjukan pada *listing* program berikut dan hasil dari program tersebut ditunjukan oleh Gambar 3.4.

```
bgsMOG = cv2.BackgroundSubtractorMOG2()
cap=  cv2.VideoCapture("/home/wisnu/Projek/video1/25
fps.mp4")
counter    =0
if cap:
    while True:
        ret, frame = cap.read()
        if ret:
            fgmask = bgsMOG.apply(frame, None, 0.01)
```

Gambar 3.4 Hasil eksekusi program *Background Subtractor*

3.6 Deteksi Objek Bergerak

Langkah-langkah untuk mendapatkan *Region of Interest* (ROI) pada objek yang diinginkan adalah sebagai berikut:

a. Deteksi tepi objek

Deteksi tepi ini menggunakan *Background Subtractor* sebagai input. Output dari *Background Subtractor* yang sebelumnya berupa gambar berwarna putih pada objek akan dibuat berwarna hitam kecuali setiap daerah tepi pada objek tersebut. Deteksi tepi ini juga dapat dibilang menampilkan garis terluar yang membentuk objek. Program untuk deteksi tepi ditunjukan oleh *listing* program berikut dan hasil program tersebut ditunjukan oleh Gambar 3.5.

```
contours, hierarchy = cv2.findContours(fgmask,
cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
```

Gambar 3.5 Hasil program deteksi tepi

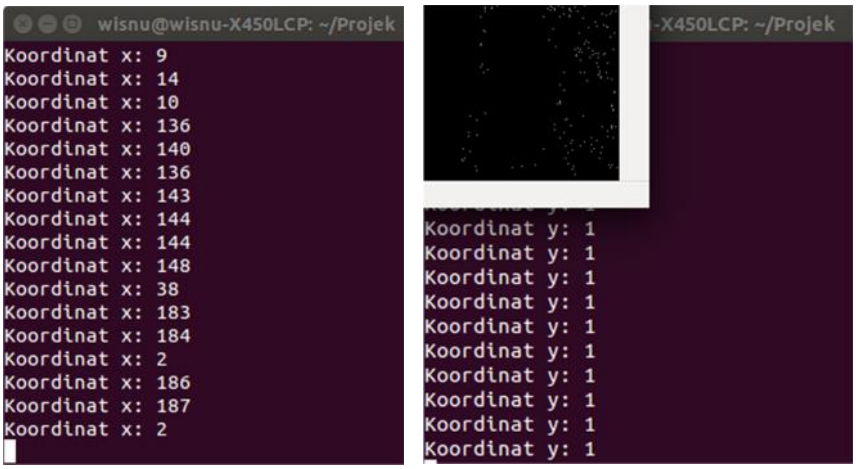
b. Penentuan koordinat titik terluar, nilai tinggi dan lebar

Data-data yang harus diperoleh sebelum memperoleh *Region of Interest* (ROI) adalah data nilai koordinat titik terluar (x,y), nilai tinggi dan nilai lebar. Ketiga nilai tersebut dapat diperoleh dengan perintah cv2.contour yang merupakan *library* dari OpenCV.

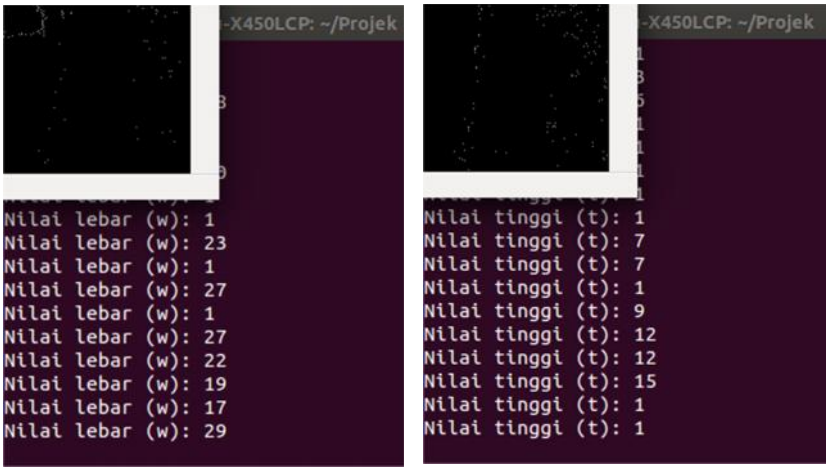
Sama seperti pada langkah sebelumnya, output dari program sebelumnya akan menjadi input untuk program selanjutnya. Dalam hal ini, output dari program deteksi tepi akan diproses untuk mendapatkan ketiga nilai tersebut. Program untuk mendapatkan ketiga nilai tersebut

ditunjukan pada *listing* program berikut dan hasil program ditunjukan oleh Gambar 3.6 dan Gambar 3.7.

```
try: hierarchy = hierarchy[0]
except: hierarchy = []
for contour, hier in zip(contours, hierarchy):
      (x,y,w,h) = cv2.boundingRect(contour)
```



Gambar 3.6 Hasil eksekusi program penampil nilai koordinat titik terluar (x,y)

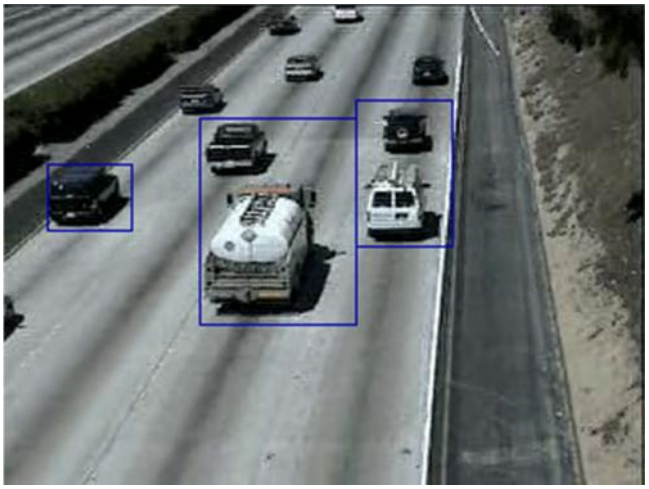


Gambar 3.7 Hasil eksekusi program penampil nilai lebar (w) dan nilai tinggi (t) objek

c. Mendapatkan *Region of Interest* (ROI)

Setelah didapat data koordinat terluar (x,y), koordinat nilai tinggi (y_{max}) dan koordinat nilai lebar (x_{max}), maka Region of Interest (ROI) dapat digambar secara otomatis di setiap tepi objek. Program untuk menempatkan *Region of Interest* (ROI) ditunjukan pada *listing* program berikut dan hasil program tersebut ditunjukan pada Gambar 3.8.

```
if w > 20 and h > 25:
  cv2.rectangle(frame, (x,y), (x+w,y+h), (180, 1, 0), 1)
```



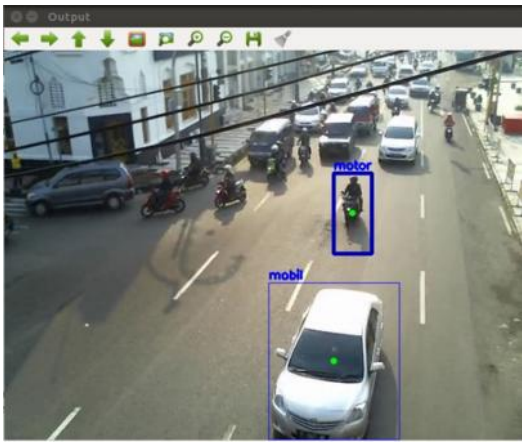
Gambar 3.8 Hasil eksekusi program menampilkan *Region of Interest* (ROI)

d. Klasifikasi Kendaraan

Selain dapat digunakan untuk menggambar kotak atau persegi, data koordinat terluar (x,y), koordinat nilai tinggi (y_{max}) dan koordinat nilai lebar (x_{max}) juga dapat dipakai untuk klasifikasi jenis kendaraan sesuai *threshold* yang diinginkan. Klasifikasi jenis kendaraan sendiri menggunakan data koordinat nilai tinggi dan/atau data koordinat nilai lebar. Program untuk mengklasifikasikan jenis kendaraan ditunjukan oleh *listing* program berikut dan hasil program tersebut ditunjukan oleh Gambar 3.9.

```
if w > 20 and h > 25:
  cv2.rectangle(frame, (x,y), (x+w,y+h), (255, 1, 1), 1)
  cv2.putText(frame,'mobil',(x,y-
5),cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,0,0),2)

elif w > 80 and h > 80:
  cv2.rectangle(frame, (x,y), (x+w,y+h), (180, 1, 0), 1)
  cv2.putText(frame,'motor',(x,y-
5),cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,0,0),2)
```



Gambar 3.9 Hasil eksekusi program klasifikasi kendaraan

3.7 *Tracking* Objek Bergerak

Tracking objek dapat dilakukan hanya dengan menempatkan *Region of Interest* (ROI) sama seperti langkah sebelumnya. Namun, algoritma penghitungan yang dipakai menggunakan konsep pemotongan suatu garis maka *tracking* menggunakan *Region of Interest* (ROI) sebagai indikator akan memiliki banyak kemungkinan yang dapat menurunkan tingkat akurasi sistem.

Oleh karena itu, *tracking* pada tugas akhir ini digunakan tanda titik sebagai fungsi *tracking*. Konsep tanda titik sebagai indikator *tracking* ini menempatkan tanda titik pada titik tengah kotak atau persegi yang telah digambar sebelumnya. Data yang diperlukan untuk menggambar titik tersebut

adalah koordinat nilai titik tengah (x,y) pada kotak atau persegi. Koordinat nilai titik tengah dapat diperoleh dengan rumus berikut:

Rumus nilai tengah koordinat sumbu x:

$$X = \frac{X}{2}$$

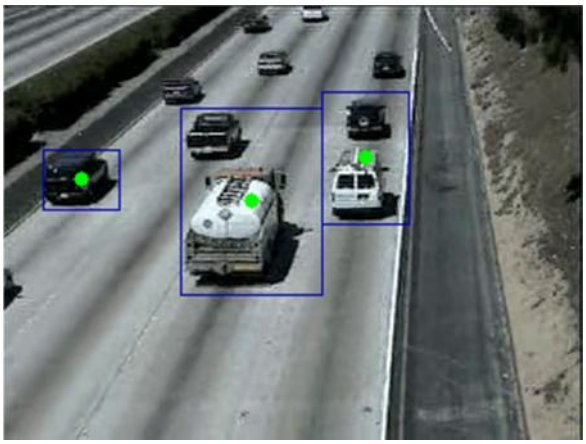
Rumus nilai tengah koordinat sumbu y:

$$Y = \frac{Y}{2}$$

Program untuk menampilkan *tracking* berupa tanda titik pada objek ditunjukan pada *listing* program berikut dan hasil dari program tersebut ditunjukan pada Gambar 3.10.

```
if w > 20 and h > 25:
   cv2.rectangle(frame, (x,y), (x+w,y+h), (180, 1, 0), 1)
    x1=w/2
    y1=h/2
    cx=x+x1
    cy=y+y1
    centroid=(cx,cy)

    titik = cv2.circle(frame,(int(cx),int(cy)),4,(0,255,0),-1)
```



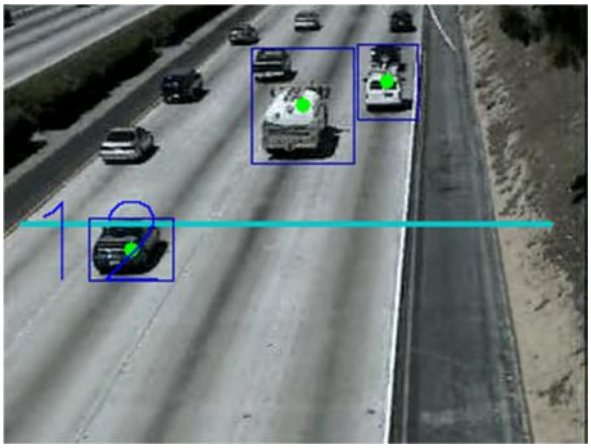
Gambar 3.10 Hasil eksekusi program *tracking* menggunakan tanda titik

3.8 Penghitung Kendaraan

Algoritma yang dipakai untuk menghitung tiap kendaraan menggunakan konsep titik yang memotong koordinat garis. Koordinat garis yang dipakai menggunakan modul cv2.line pada OpenCV kemudian koordinat titik menggunakan koordinat *tracking* dari proses sebelumnya. Program untuk membuat koordinat garis dan program penghitung kendaraan ditampilkan pada *listing* program berikut serta hasil dari kedua program tersebut ditunjukan pada Gambar 3.11.

```
garis = cv2.line(frame,(10,130),(300,130),(200,200,0),2)

  if cy==210:
      counter=counter+1
  if cy==205:
      counter=counter+1
  if cy==200:
      counter=counter+1
  if cy==190:
      counter=counter+1
  else:
      counter=counter+0
```



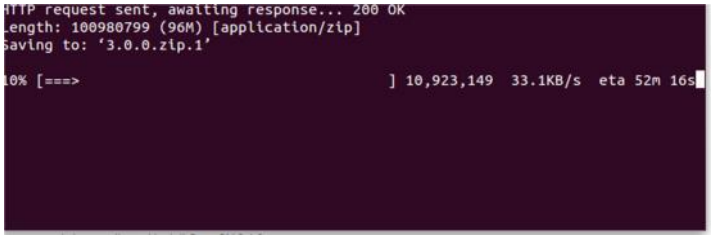
Gambar 3.11 Hasil eksekusi program penghitung

3.9 Implementasi Perangkat OpenCV pada Ubuntu 14.04 LTS

Pada pembahasan ini akan dibahas langkah-langkah mengintegrasikan *library* Open CV dengan sistem operasi yang kita miliki. Bahasan berikut menggunakan Open CV 3.0 yang saat itu masih dalam versi *Release Candidate* 1 (RC 1), namun secara keseluruhan langkah mengintegrasikan *library* Open CV 2.4.9 yang dipakai penulis tidaklah jauh berbeda. Langkah pertama yang harus dilakukan untuk mengimplementasi Open CV pada sistem operasi Ubuntu 14.04 LTS yaitu me-*download* versi Open CV yang diinginkan. Dalam hal ini, penulis menggunakan Open CV versi 3.0. Namun, keseluruhan langkah-langkah yang disebutkan oleh penulis dapat diimplementasikan pada macam-macam versi Open CV.

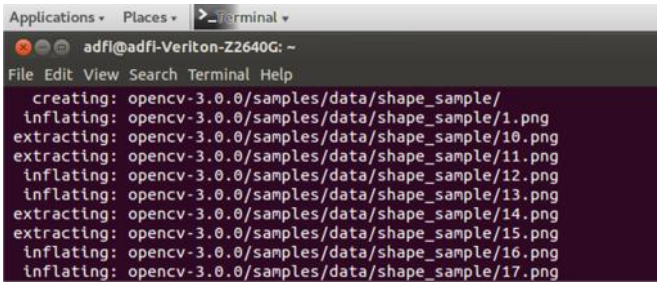
Langkah-langkah pengimplementasian Open CV pada Ubuntu 14.04 LTS:

1. Unduh versi Open CV yang akan digunakan. Proses pengunduhan pada ubuntu dilakukkan dengan cara menuliskan “wget situs download” (tanpa tanda petik) pada terminal. Contoh: wget http:opencv.org/downloads/open-cv-3.zip. Jika berhasil maka akan muncul output seperti Gambar 3.12 berikut:



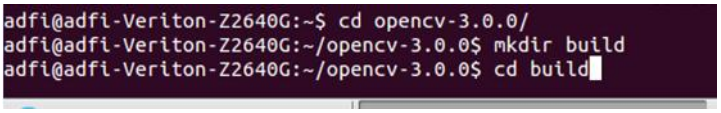
Gambar 3.12 Eksekusi perintah *download* Open CV

2. Setelah file telah terunduh (100%), selanjutnya tulis perintah “unzip open-cv-3.zip” (tanpa tanda petik) pada terminal. Jika penulisan perintah benar maka akan muncul proses unzip seperti Gambar 3.13. Setelah perintah unzip telah selesai, arahkan terminal ke direktori terkait dengan cara menuliskan “cd open-cv-3”.

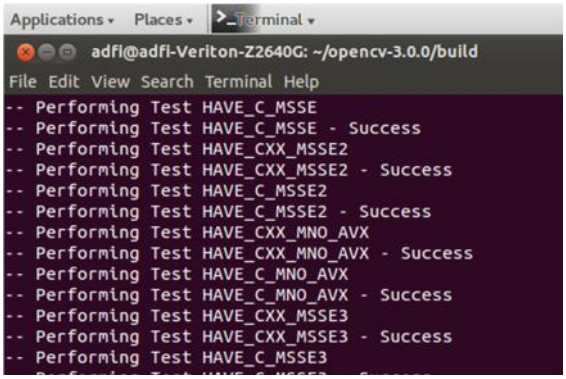


Gambar 3.13 Hasil eksekusi perintah unzip

3. Setelah kedua langkah tersebut telah dilakukan, langkah selanjutnya adalah memasang (*build*) Open CV pada sistem operasi ubuntu. Cara mengintegrasikannya ditunjukkan pada Gambar 3.14 kemudian ditambah dengan "cmake -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON -D INSTALL_C_EXAMPLES=ONDINSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=ON -D WITH_QT=ON -D WITH_OPENGL=ON -D WITH_VTK=ON .." hasil dari perintah tersebut ditunjukkan pada Gambar 3.15 dan jika Open CV telah berhasil terintegrasi akan memiliki hasil seperti Gambar 3.16



Gambar 3.14 Perintah *build library* Open CV pada sistem operasi Ubuntu 14.04 LTS

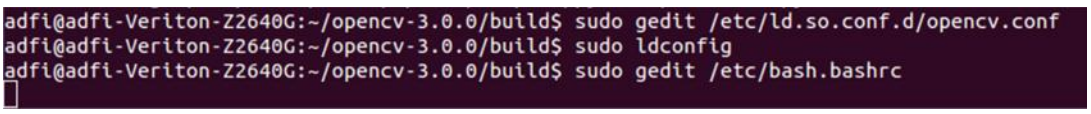


Gambar 3.15 Hasil eksekusi perintah Gambar 3.14

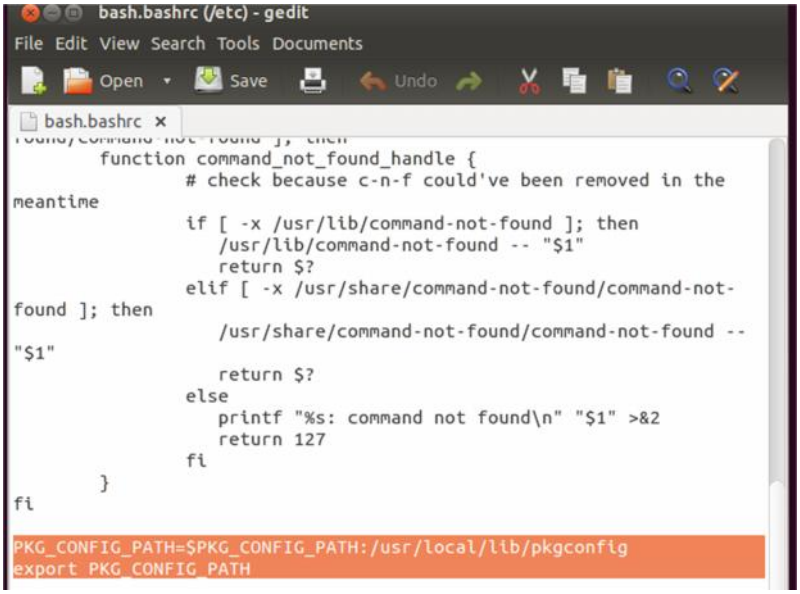
Gambar 3.16 Indikator *build* Open CV berhasil

4. Setelah Open CV telah terpasang (*builded*), langkah selanjutnya adalah mengintegrasikan Open CV. Sebelumnya sistem operasi harus memiliki aplikasi "make", semacam aplikasi untuk menuliskan kode misalnya gedit. Aplikasi tersebut dapat diinstall dengan cara menuliskan "sudo make install" (tanpa tanda petik). Langkah pertama mengkonfigurasikan Open CV yaitu dengan menuliskan "sudo gedit /etc/ld.so.conf.d/opencv.conf" (tanpa tanda petik) maka akan muncul kotak dialog gedit selanjutnya tuliskan "/usr/local/lib" (tanpa tanda petik) seperti Gambar 3.17 berikut, lalu simpan. Setelah langkah tersebut telah terlaksana, selanjutnya adalah menuliskan perintah sesuai Gambar 3.18, kemudian tambahkan perintah seperti pada Gambar 3.19 (yang diblok pointer) lalu simpan.

Gambar 3.17 Menambahkan *library* Open CV



Gambar 3.18 Perintah *build library* Open CV



Gambar 3.19 Penambahan perintah konfigurasi *library* Open CV pada gedit

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

4.1 Pengujian Sistem klasifikasi dan penghitung jumlah kendaraan

Pada bab ini akan menjelaskan tentang pengujian purwarupa sistem klasifikasi dan penghitung kendaraan bermotor berbasis pengolah citra. Pengujian metode ini dilakukan dengan cara mengaplikasikan metode kepada video-video jalan raya yang sudah di dapat dari pengambilan data sendiri maupun video yang diambil dari internet.

Tabel 4.1: Tabel Video yang akan diuji beserta sumbernya

No.	Nama Video	Format Video	Sumber
1	Video 1	.avi	https://github.com/paritosh-gupta/streetView-Mario/blob/master/data/vehicle_detection_haarcascades/video1.avi
2.	Video 2	.avi	Jalan Asia-Afrika Kota Bandung (direkam di Jembatan penyeberangan tempat terkait)
3.	Video 3	.mp4	https://www.youtube.com/watch?v=q6KyiYZBKps
4.	Video 4	.mp4	https://www.youtube.com/watch?v=WVagFjxJrYI

Pengujian sistem klasifikasi dan penghitung jumlah kendaraan ini dilakukan tidak secara *real-time* namun diuji dengan memproses video yang didapatkan baik dari hasil rekaman video menggunakan kamera *webcam* maupun video hasil *download* dari internet. Pengambilan video dengan menggunakan kamera *webcam* dilakukan di jembatan penyebrangan jalan Asia-Afrika kota Bandung, tinggi jembatan penyebrangan ini kurang lebih 5 meter. Kamera *webcam* yang digunakan untuk merekam menggunakan jenis kamera *webcam* Logitech HD 720p dengan fitur Autofokus.

Sistem ini akan diuji seberapa akurat sistem mengklasifikasi kendaraan roda dua (motor) dan kendaraan roda empat (mobil) berdasarkan ukuran *threshold* yang dibuat pada video. Sistem ini juga akan diuji seberapa akurat sistem menghitung jumlah tiap klasifikasi kendaraan. Kondisi tiap-tiap video yang digunakan dalam proses pengujian dibuat pada variasi fokus kamera yaitu fokus pada jalan satu arah (*one-way*) . Selain itu, penulis juga melakukan pengujian pengaturan kamera meliputi ketinggian pemasangan kamera, nilai *frame* yang digunakan, dan nilai *framerate*. Selain dilakukan pengujian pada sistem penghitung dan sistem klasifikasi, sistem juga akan diuji kemampuan mendeteksi kendaraan saat kondisi malam.

4.2 Langkah-langkah Pengujian

Pengujian pertama yang dilakukan adalah pengujian seberapa akurat sistem dapat mengklasifikasi jenis kendaraan. Langkah-langkah dalam pengujian:

1. Dijalankan program pemrosesan video.
2. Video yang akan diproses dimasukkan yaitu video 2 karena memiliki variasi kendaraan roda dua dan lebih.
3. Komputer atau laptop dijalankan dengan sistem operasi (*operating system*) ubuntu. Dalam hal ini, penulis menggunakan ubuntu versi 14.04 LTS.
4. Pada aplikasi terminal buka folder tempat program dan video terkait berada, misalnya: cd Home/Projek.
5. Eksekusi program dengan perintah “python nama berkas program” (tanpa tanda petik) dengan cara menuliskannya pada terminal. Misalnya python projek.py.
6. Tampak jendela dengan nama FGMASK dan output
7. Program akan terus berjalan selama durasi video belum habis, dan akan keluar jika tombol “Q” ditekan.

Setelah pengujian sistem klasifikasi kendaraan dilakukan, selanjutnya adalah pengujian seberapa akurat sistem menghitung jumlah kendaraan berdasarkan klasifikasi jenis kendraan tersebut. Langkah-langkah:

1. Menentukan video yang akan diuji.
2. Komputer atau laptop dijalankan dengan sistem operasi (*operating system*) ubuntu. Dalam hal ini, penulis menggunakan ubuntu versi 14.04 LTS.
3. Pada aplikasi terminal, kemudian buka folder tempat program dan video terkait berada, misalnya: Home/Projek.
4. Eksekusi program dengan perintah “python nama berkas program” (tanpa tanda petik) dengan cara menuliskannya pada terminal. Misalnya python projek.py.
5. Dilakukan perbandingan hasil penghitungan sistem berdasarkan klasifikasi jenis kendaraan dengan penghitungan manual.

Selain diuji pada tingkat keakuratan penghitungan dan klasifikasi kendaraan bermotor. Sistem juga akan diuji seberapa akurat sistem membedakan objek yang ingin dideteksi dengan objek yang tidak ingin dideteksi saat kondisi malam hari. Langkah-langkah yang dilaksanakan:

1. Video yang akan diuji adalah Video 3 (Pencahayaan jalan raya minimal) dan Video 4 (Pencahayaan jalan raya baik)
2. Komputer atau laptop dijalankan dengan sistem operasi (*operating system*) ubuntu. Dalam hal ini, penulis menggunakan ubuntu versi 14.04 LTS.
3. Pada aplikasi terminal buka folder tempat program dan video terkait berada, misalnya: Home/Projek.
4. Eksekusi program dengan perintah “python nama berkas program” (tanpa tanda petik) dengan cara menuliskannya pada terminal. Misalnya python projek.py.

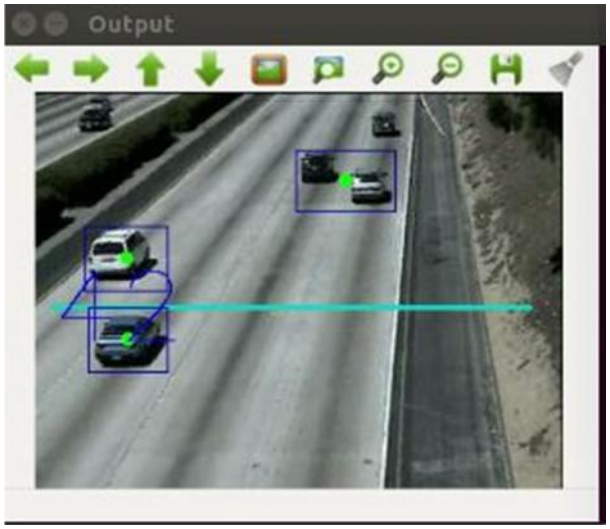
5. Dilakukan pengamatan pada kedua video secara bergantian. Pengamatan difokuskan pada kemampuan deteksi kendaraan bermotor pada kondisi malam hari.

4.3 Hasil Pengujian

A. Contoh Kondisi Pertama

Pengujian pertama ini akan meninjau seberapa akurat sistem penghitungan kendaraan bermotor. Pengujian ini dilakukan pada Video 1, karena video tersebut merekam kondisi jalan searah (*one way*). Kondisi jalan pada video 1 dapat dilihat pada Gambar 4.1. Video tersebut berdurasi 33 detik dan memiliki ukuran piksel 320 x 240 piksel. Sedangkan output sistem penghitung ditunjukkan pada Gambar 4.2.

Gambar 4.1 Kondisi jalan pada video 1

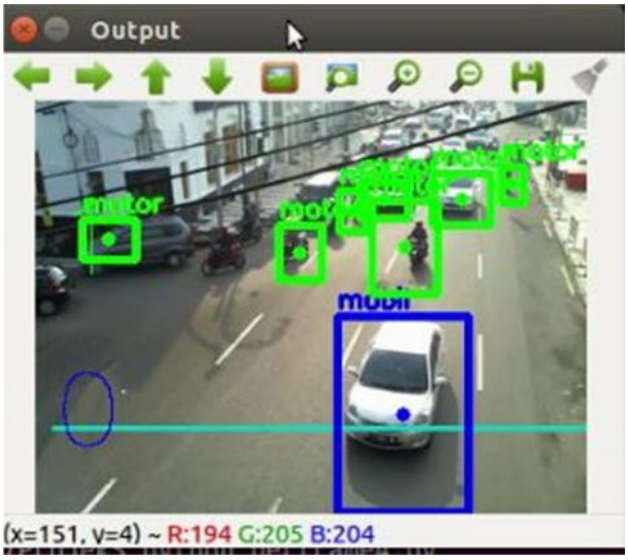


Gambar 4.2 Hasil eksekusi sistem penghitung

B. Contoh Kondisi Kedua

Pengujian pertama ini akan meninjau seberapa akurat sistem klasifikasi kendaraan bermotor. Pengujian ini dilakukan pada Video 2, karena video tersebut merekam kondisi jalan searah (*one way*) dan memiliki variasi kendaraan bermotor dari kendaraan roda dua dan kendaraan roda lebih dari dua. Diharapkan sistem ini dapat mengklasifikasi sepeda motor dengan mobil. Kondisi jalan pada video 1 dapat dilihat pada Gambar 4.3. Video tersebut berdurasi 30 menit 31 detik dan memiliki ukuran piksel 320 x 240 piksel. Sedangkan output sistem penghitung ditunjukkan pada Gambar 4.4.

Gambar 4.3 Kondisi jalan video 2

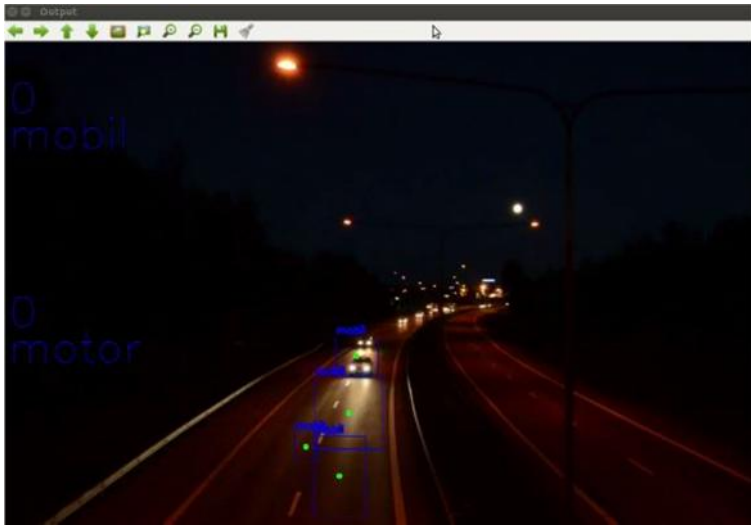


Gambar 4.4 Hasil eksekusi sistem klasifikasi

C. Contoh kondisi ketiga

Pengujian kondisi ketiga ini akan meninjau seberapa akurat sistem pendeteksi kendaraan bermotor saat kondisi malam. Pada video 3 yang akan diuji memberikan latar belakang jalan raya dua arah dalam kondisi malam dengan penerangan jalan yang minimal. Kondisi jalan dapat dilihat pada Gambar 4.5 sedangkan output dari sistem dapat dilihat pada Gambar 4.6 berikut:

Gambar 4.5 Kondisi jalan video 3



Gambar 4.6 Output sistem pada kondisi video 3

D. Contoh kondisi keempat

Pengujian kondisi keempat ini akan meninjau seberapa akurat sistem pendeteksi kendaraan bermotor saat kondisi malam. Pada video 4 yang akan diuji memberikan latar belakang jalan raya dua arah dalam kondisi malam dengan penerangan jalan yang baik. Kondisi jalan dapat dilihat pada Gambar 4.7 sedangkan output dari sistem dapat dilihat pada Gambar 4.8 berikut:

Gambar 4.7 Kondisi jalan video 4

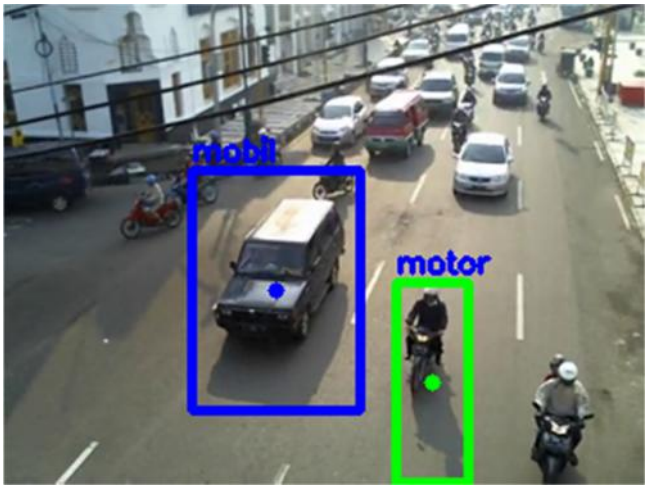
Gambar 4.8 Output sistem pada kondisi jalan video 4

4.4 Pembahasan

Pada pembahasan ini, sistem akan diuji dengan beberapa pengujian, diantaranya sistem akan diuji seberapa akurat sistem mengklasifikasi kendaraan bermotor, sistem akan diuji seberapa akurat melakukan penghitungan kendaraan bermotor, sistem juga akan diuji mendeteksi kendaraan di malam hari saat kondisi jalan raya memiliki pencahayaan minimal dan pencahayaan jalan raya yang baik. Pengujian untuk seluruh kondisi tersbut juga disertai analisis yang dapat mendukung untuk pengembangan sistem selanjutnya untuk mewujudkan *Intelligent Transportation System*.

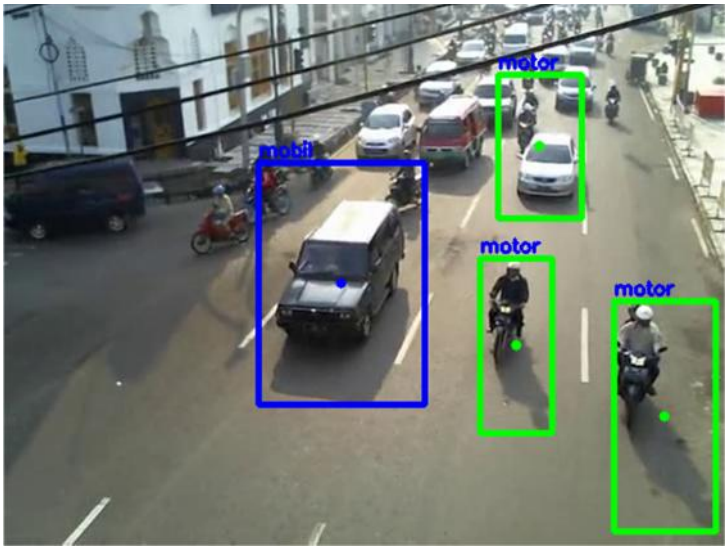
A. Analisis keakuratan sistem klasifikasi kendaraan

Analisis keakuratan sistem dalam hal mengklasifikasi kendaraan bermotor ini menggunakan video pertama. Algoritma yang dipakai untuk mengklasifikasikan kendaraan bermotor ini berdasarkan ukuran kendaraan bermotor itu sendiri. Hasil output yang diinginkan dari sistem ini ditunjukan oleh Gambar 4.5.

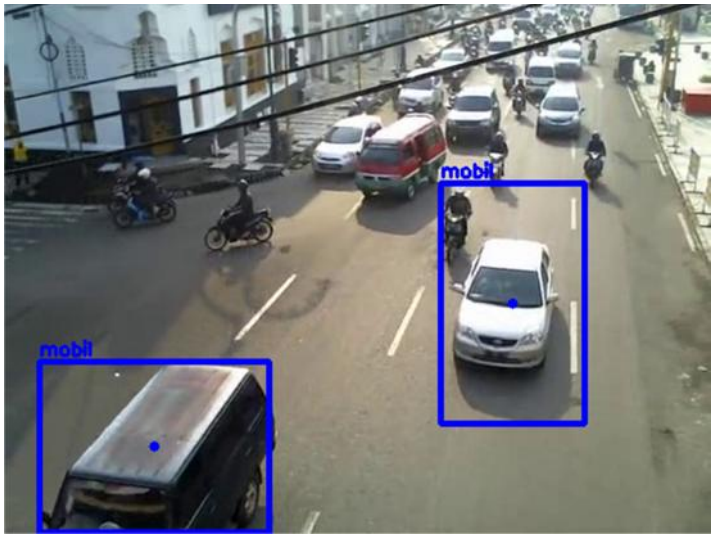


Gambar 4.9 Hasil output sistem klasifikasi yang diinginkan

Pada awal mula sistem klasifikasi kendaraan bermotor ini akan terjadi kesalahan klasifikasi sesaat sistem dalam hal mengklasifikasikan mobil seperti pada Gambar 4.10. Hal ini terjadi karena kondisi video yang dipakai memiliki objek yang bergerak mendekati kamera. Saat objek jauh dari fokus kamera maka objek akan terlihat lebih kecil daripada ketika objek berada atau mendekati fokus kamera. Kesalahan klasifikasi sesaat sistem akan berakhir ketika objek berada atau hampir mendekati titik fokus kamera, seperti terlihat pada Gambar 4.11.



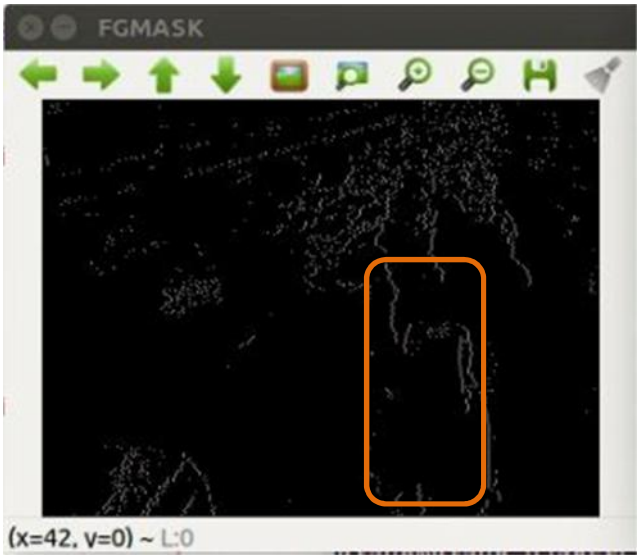
Gambar 4.10 Kesalahaan klasifikasi sesaat sistem.



Gambar 4.11Kesalahan klasifikasi sesaat sistem berakhir

Kesalahan lain dari sistem klasifikasi terjadi ketika dua atau lebih objek saling berdekatan. Saat kondisi tersebut, sistem akan mengartikan objek yang lebih dari satu tersebut menjadi satu objek. Kesalahan ini timbul karena algoritma *background subtractor* pada Open CV 2.4.9 belum dapat menghilangkan *noise* berupa bayangan dari objek. Gambar kesalahan dari kondisi ini ditunjukan oleh Gambar 4.12 kemudian gambar output algoritma *background subtractor* ditunjukan oleh Gambar 4.13.

Gambar 4.12Kesalahan sistem mendeteksi objek yang saling berdekatan



Gambar 4.13 Penyebab kesalahan deteksi objek yang saling berdekatan

Kesalahan sistem pada Gambar 4.13 tersebut terjadi karena bayangan objek menyentuh objek lain sehingga sistem menganggapnya sebagai satu objek. Kesalahan tersebut juga akan berdampak pada kesalahan sistem klasifikasi karena selain sistem menganggapnya sebagai satu objek, objek tersebut juga memiliki ukuran yang lebih besar. Sistem klasifikasi yang dibuat telah memiliki rentang nilai ukuran objek secara tetap, jika terjadi penambahan ukuran tersebut hal yang mungkin terjadi adalah sistem menganggap dua atau lebih objek yang berdekatan tersebut menjadi sebuah mobil.

Kesalahan sistem tersebut dapat diatasi dengan meng-*upgrade* versi OpenCV yang memiliki fitur BackgroundSutractorGMG. Fitur tersebut mampu menghilangkan *noise* sistem yang berupa bayangan. Fitur tersebut menurut referensi tersedia pada OpenCV versi 3.0.0. Sejauh penulis mengikuti perkembangan, OpenCV versi 3.0.0 masih berupa *Release Candidate* (RC) 1 pada bulan Mei 2015. Versi tersebut masih belum memiliki fitur BackgroundSubtractorGMG.

B. Analisis keakuratan sistem penghitung kendaraan

Analisis keakuratan sistem penghitung kendaraan bermotor ini menggunakan kondisi video 1, karena video 1 tidak memiliki *noise* berupa bayangan objek. Berbeda dengan video 1, video 2 cenderung memiliki *noise* yang komplek karena saat perekaman kondisi matahari membentuk sudut lancip sehingga bayangan yang terbentuk lebih lebar daripada video 1.Penghitungan tingkat akurasi sistem menggunakan rumus berikut:

$$Presentase\ akurasi\ sistem = 100\% - \%\ error \quad \textbf{(Persamaan 4.1)}$$

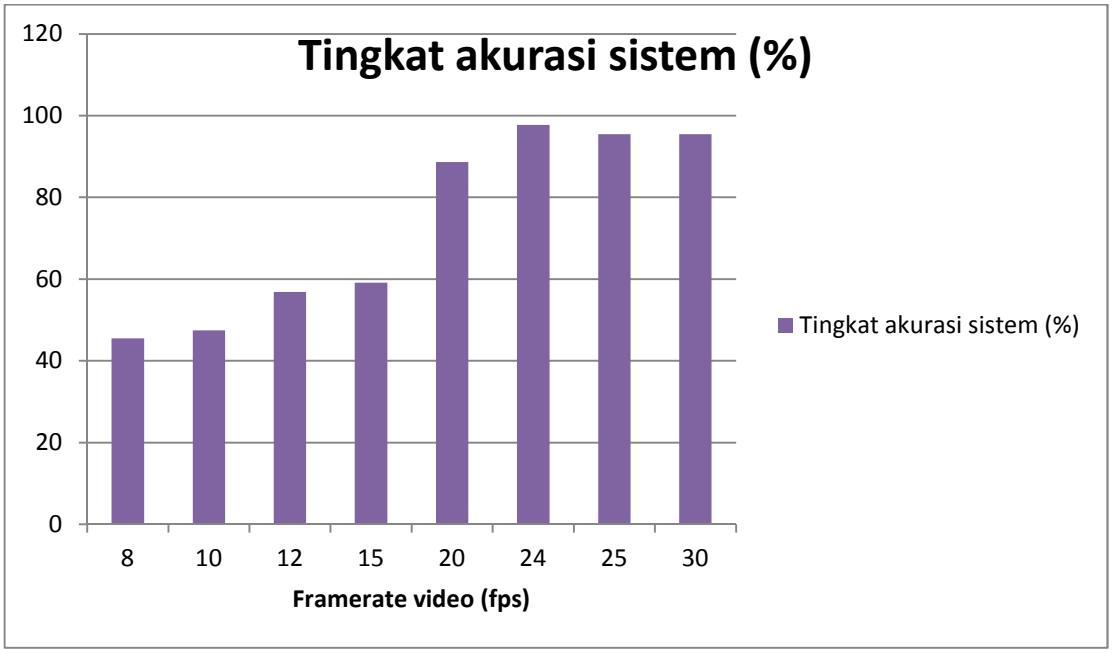
Dengan rumus Presentase error sistem dihitung melalui rumus:

$$\%\ error = \frac{selisih\ penghitungan\ manual\ dengan\ \ sistem}{penghitungan\ manual}\ x\ 100\% \quad \textbf{(Persamaan 4.2)}$$

Parameter yang diuji dalam analisis keakuratan sistem penghitung kendaraan ini menggunakan faktor nilai *framerate* video Analisis pengaruh nilai *framerate* video diperlukan untuk menentukan dalam rentang berapa nilai *framerate* video memiliki tingkat akurasi yang optimal. Berikut hasil pengujian hubungan *framerate* dengan tingkat akurasi sistem untuk video 1:

Tabel 4.2 Hubungan nilai *framerate* video terhadap tingkat akurasi

Nilai *framerate* video (fps)	Nilai penghitungan sistem	Nilai penghitungan manual	Presentase error (%)	Presentase akurasi sistem (%)
8	20	44	54,54	45,46
10	21	44	52,27	47,43
12	25	44	43,18	56,82
15	26	44	40,9	59,1
20	39	44	11,36	88,64
24	45	44	2,27	97,73
25	46	44	4,54	95,46
30	46	44	4,54	95,46



Gambar 4.14: Grafik hubungan tingkat akurasi sistem dengan nilai *framerate* video

Grafik 4.14 di atas menunjukkan hubungan nilai *framerate* video akan memiliki hasil maksimal pada nilai 24 fps. Nilai *framerate* maksimal yang diambil adalah 30 fps (*framerate per seconds*) karena

kamera *webcam* yang dipakai akan mengeluarkan nilai *framerate* maksimal 30 fps.

Nilai *framerate* video berbanding lurus terhadap tingkat akurasi yang akan dicapai karena semakin tinggi *framerate* sebuah video maka video tersebut akan lebih detail begitu pula sebaliknya. Video adalah frame gambar yang bergerak, misalnya nilai *framerate* video adalah 10 fps (*frame per seconds*) maka dalam video tersebut tiap detiknya menampilkan 10 frame gambar. Jadi, semakin besar nilai *framerate* video akan semakin banyak gambar yang ditampilkan setiap detiknya atau dapat disebut lebih detail.

Namun, teori tersebut nampaknya tidak terjadi pada purwarupa ini. Hal tersebut dikarenakan faktor penghitungan objek yang berkelipatan. Artinya satu objek dapat terhitung lebih daripada satu saat nilai *framerate* tinggi, sedangkan saat nilai *framerate* rendah indikator *tracking* akan lebih cepat berpindah sehingga terjadi kemungkinan bahwa indikator tersebut tidak melewati nilai koordinat garis penghitung.

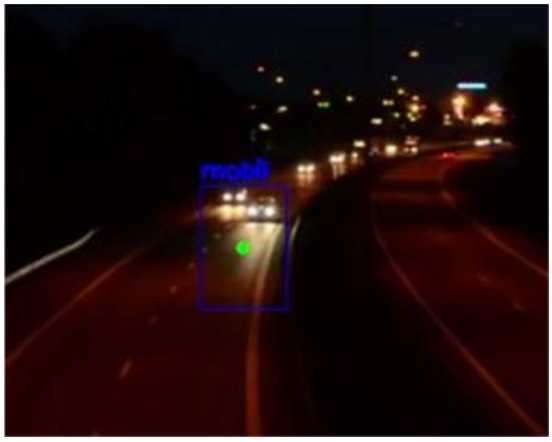
Jadi secara teoritis semakin detail sebuah video memberikan frame gambarnya setiap maka semakin mudah *image processing* mengolah data koordinatnya. Namun, khusus untuk sistem ini terdapat kelemahan jika *framerate* terlalu besar. Solusi dari permasalahan ini adalah mengembangkan metode *tracking* pada purwarupa selanjutnya.

C. Analisis keakuratan sistem pendeteksi saat kondisi malam hari

Pengujian pada poin ini akan membahas tingkat keakuratan sistem mendeteksi kendaraan bermotor pada kondisi malam hari. Pengujian ini dilakukan dengan menggunakan dua kondisi penerangan (iluminasi) jalan raya yang berbeda sehingga analisis pada bagian ini-pun juga terbagi menjadi dua yaitu:

i. Kondisi penerangan jalan minimal

Pada pengujian ini digunakan video 3 sebagai video yang akan diproses. Video 3 memiliki latar belakang jalan raya di waktu malam hari dengan penerangan jalan yang sangat minim. Pencahayaan yang minimal tersebut membuat sistem pengolah citra kesulitan membedakan latar belakang (*background*) dengan objek (*foreground*). Sistem akan menganggap ruang lingkup yang terkena cahaya lampu kendaraan bermotor sebagai objek (*foreground*) seperti Gambar 4.15. Hal ini terjadi karena *Background Subtractor* kurang efektif pada kondisi pencahayaan minimal terihat dari output algoritma *Background Subtractor* yang hanya dapat menangkap pergerakan cahaya mobil seperti pada Gambar 4.16. Pada Gambar 4.16 tersebut, output algoritma *Background Subtractor* hanya dapat mendeteksi nyala lampu dan ruang tembak lampu kendaraan bermotor sebagai objeknya (*foreground*-nya). Bagian yang ditunjuk dengan tanda persegi berwarna ungu tersebut adalah *foreground* dari ruang tembak lampu sebuah mobil.



Gambar 4.15 Kesalahan sistem pendeteksi pada kondisi pencahayaan minimal

Gambar 4.16 Penyebab kesalahan sistem pendeteksi pada kondisi pencahayaan minimal

ii. Kondisi penerangan jalan baik

Pada kondisi video 3 didapatkan sebuah permasalahan, yaitu sistem harus tetap bisa beroperasi pada kondisi tersebut. Terdapat minimal dua pilihan untuk mengatasi permasalahan yang terjadi pada kondisi video 3. Pertama, kelemahan sistem ini terdapat pada kondisi pencahayaan yang minimal. Saat objek yang tak diinginkan juga terdeteksi maka tingkat kesalahan (*error rate*) akan semakin tinggi. Oleh karena itu, kondisi jalan raya harus direkayasa agar memiliki pencahayaan yang cukup. Kondisi tersebut akan lebih lanjut dijelaskan pada poin ini. Pilihan ke dua adalah menggunakan fitur kamera infra merah (*infra red*) yang dapat mendeteksi panas walaupun kondisi pencahayaan minimal.

Pada pengujian kali ini dilakukan dengan video 4 yang memiliki pencahayaan yang lebih baik dari pada video 3. Hasil pengujian tersebut memberikan hasil yang jauh lebih baik daripada hasil pengujian video 3. Hasil pengujian tersebut dapat mendeteksi kendaraan bermotor seperti yang terlihat pada Gambar 4.17 untuk posisi kamera berada di belakang objek.

Ketika kamera berada di posisi depan objek, cahaya lampu kendaraan dan ruang tembak kendaraan akan kembali terdeteksi sebagai *foreground* seperti pada Gambar 4.18. Oleh karena itu, posisi kamera yang baik yaitu pada posisi di belakang objek.

Gambar 4.17 Posisi kamera berada di belakang objek

Gambar 4.18 Posisi kamera di depan objek mendeteksi ruang tembak cahaya lampu sebagai *foreground*

BAB V
PENUTUP

5.1 Kesimpulan

Setelah melakukan perancangan sistem, dan membuat pengimplementasian purwarupa sistem klasifikasi dan penghitung Kendaraan bermotor berbasis pengolah citra ini, maka dapat diambil kesimpulan sebagai berikut :

1. Berdasarkan hasil pengujian yang telah dilakukan, sistem klasifikasi telah mampu mengklasifikasi jenis kendaraan sepeda motor dan mobil dengan baik.
2. Berdasarkan hasil pengujian yang telah dilakukan, sistem penghitung jumlah kendaraan mampu menghitung kendaraan dengan tingkat akurasi tertinggi adalah 90% pada video 1.
3. Pemasangan kamera pemantau yang lebih tinggi dari 5 meter sampai batas ketinggian tertentu memberikan tingkat akurasi penghitungan lebih daripada pemasangan dengan ketinggian kurang dari 5 meter.
4. Nilai *framerate* video berbanding lurus dengan tingkat akurasi yang dihasilkan untuk sistem ini.
5. Nilai *framerate* video yang memberikan tingkat akurasi paling optimal terjadi pada rentang nilai 24 fps.
6. Kondisi jalan satu arah memiliki keakuratan hitung yang lebih baik dari kondisi jalan dua arah karena metode hitung yang menempatkan koordinat nilai absis (x) maupun ordinat (y) pada video.

5.2 Saran

Dari hasil pengujian penghitung otomatis ini masih terdapat beberapa kekurangan dan dimungkinkan untuk pengembangan lebih lanjut. Oleh karenanya, penulis merasa perlu untuk memberi saran demi peningkatan kualitas dan pengembangannya yaitu sebagai berikut :

1. Sistem *tracking* pada purwarupa ini masih sangat sederhana sehingga dapat ditingkatkan.
2. Untuk pengembangan lebih lanjut pada purwarupa ini, bayangan dari objek asli yang bergerak dapat dieliminasi.
3. Untuk kondisi malam hari, kinerja purwarupa ini akan lebih optimal dengan dibantu pencahayaan jalan raya yang baik khusus untuk daerah liputan kamera.
4. Algoritma penghitung pada purwarupa ini masih memiliki kemungkinan menghitung satu objek menjadi lebih dari satu dalam kondisi tertentu, sehingga algoritma tersebut dapat dikembangkan.

DAFTAR PUSTAKA

Barandiaran, J., Cortez, A., Nieto, M., Otaegui, O., Sanchez, P., Unzueta, L., 2011. *Adaptive Multi-Cue Background Subtraction for Robust Vehicle Counting and Classification*. IEEE Transactions on Intelligent Transportation System Journal.

Bradski, Gary., Kaehler, Adrian., 2008. *Learning Open CV*. O'Reilly Media, Amerika Serikat.

Erik, Jan Solem, 2012. *Programming Computer Vision with Python*.

Github, https://github.com/paritosh-gupta/streetView-Mario/blob/master/data/vehicle_detection_haarcascades/video1.avi (diakses 6 April 2015, Pukul 10.00)

Helmiriawan, 2012. Rancang bangun dan analisis sistem pemantau lalu lintas menggunakan OpenCV dengan algoritma canny dan blob detection. *Skripsi*, Departemen Teknik Elektro, Fakultas Teknik, Universitas Indonesia, Depok.

Jie, Kim-Sung., Liu, Ming., 2004. *Computer Vision based Real-Time Traffic Monitoring System*. Thesis Presentation, Departement of Electrical and Computer System Engineering, Monash University.

Phisca, A.R., 2014. Purwarupa sistem visual tracking jalan raya menggunakan unmanned aerial vehicle berbasis citra digital.*Skripsi*, Jurusan Ilmu Komputer dan Elektronika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Gadjah Mada, Yogyakarta.

Rossum, Guido van., 2012. *Python Tutorial Release 3.2.3*. Python Software Foundation.

Solem, Jan Erik., 2012. *Programming Computer Vision with Python*. Creative Commons.

Stackoverflow, http://stackoverflow.com/review/suggested-edits/8575035 (diakses 22 April 2015, pukul 07.55)

Stackoverflow, http://stackoverflow.com/questions/30434651/algorithm-counting?noredirect=1#comment48953767_30434651 (diakses 25 Mei 2015, pukul 09.02)

Yotube, https://www.youtube.com/watch?v=q6KyiYZBKps (diakses 15 Mei 2015, pukul 08.00)

Youtube, https://www.youtube.com/watch?v=WVagFjxJrYI (diakses 15 Mei 2015, Pukul 08.30)

Lampiran 1. *Listing program*

```
import cv2
bgsMOG = cv2.BackgroundSubtractorMOG2()
cap        = cv2.VideoCapture("/home/wisnu/Projek/video1/25 fps.mp4")
counter    =0
flag = 0
flaga = 0
flagb = 0
flagc = 0
flagd = 0
flage = 0
flagf = 0
flagg = 0
flagh = 0
flagi = 0
flagj = 0
flagk = 0
if cap:
    while True:
        ret, frame = cap.read()
        if ret:
            fgmask = bgsMOG.apply(frame, None, 0.01)
```

```
    garis = cv2.line(frame,(10,130),(300,130),(200,200,0),2)

    contours, hierarchy = cv2.findContours(fgmask,
cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

    try: hierarchy = hierarchy[0]

    except: hierarchy = []

    for contour, hier in zip(contours, hierarchy):

      (x,y,w,h) = cv2.boundingRect(contour)


      if w > 20 and h > 25:

        cv2.rectangle(frame, (x,y), (x+w,y+h), (180, 1, 0), 1)

        x1=w/2

            y1=h/2

        cx=x+x1

        cy=y+y1

        centroid=(cx,cy)

        titik = cv2.circle(frame,(int(cx),int(cy)),4,(0,255,0),-1)

            if cy == 130:

                if cy==130 and flag == flaga :

                        flaga= flag +1

                    counter=counter+1

                elif cy == 130 and flag < flaga:

                        flaga = flag

                        counter = counter + 0
```

```
elif cy == 129:
        if cy==129 and flag == flagb :
            flagb= flag +1
          counter=counter+1
        elif cy == 129 and flag < flagb:
            flagb = flag
            counter = counter + 0
elif cy == 127:
        if cy==127 and flag == flagd :
            flagd= flag +1
          counter=counter+1
        elif cy == 127 and flag < flagd:
            flagd = flag
            counter = counter + 0
elif cy == 126:
      if cy==126 and flag == flage :
            flage= flag +1
          counter=counter+1
        elif cy == 126 and flag < flage:
            flage = flag
            counter = counter + 0
elif cy == 124:
        if cy==124 and flag == flagg:
            flagg= flag +1
```

```
                counter=counter+1
            elif cy == 124 and flag < flagg:
                flagg = flag
                counter = counter + 0
        elif cy == 123:
            if cy==123 and flag == flagh :
                flagh= flag +1
                counter=counter+1
            elif cy == 123 and flag < flagh:
                flagh = flag
                counter = counter + 0
        elif cy == 121:
            if cy==121 and flag == flagj:
                flagj= flag +1
                counter=counter+1
            elif cy == 121 and flag < flagj:
                flagj = flag
                counter = counter + 0
        elif cy == 120:
            if cy==120 and flag == flagk:
                flagk= flag +1
                counter=counter+1
            elif cy == 120 and flag < flagk:
                flagk = flag
```

```
                    counter = counter + 0
                else:
                    counter=counter+0
        cv2.putText(frame, str(counter),(10,150),
cv2.FONT_HERSHEY_SIMPLEX,2, (255, 0, 0), 1, True)



            print (cy)
            print (flaga)


        cv2.imshow('Output', frame)
        cv2.imshow('FGMASK', fgmask)


        key = cv2.waitKey(100)
        if key == ord('q'):
            break
cap.release()
cv2.destroyAllWindows()
```