SISTEM PENDETEKSIAN KECEPATAN KENDARAAN YANG DIKLASIFIKASIKAN *HAAR CASCADE CLASSIFIER* UNTUK PENEGAKAN *SPEED BUMP*

TESIS

PROGRAM MAGISTER TEKNIK ELEKTRO

MINAT SISTEM KOMUNIKASI DAN INFORMATIKA

Ditujukan Untuk Memenuhi Persyaratan

Memperoleh Gelar Magister Teknik

MUHAMMAD ZULFIKRI

NIM. 166060300111004

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2019

JUDUL TESIS:

SISTEM PENDETEKSIAN KECEPATAN KENDARAAN YANG DIKLASIFIKASIKAN *HAAR CASCADE CLASSIFIER* UNTUK PENEGAKAN *SPEED BUMP*

Nama Mahasiswa	:	Muhammad Zulfikri
NIM	:	166060300111004
Program Studi	:	Teknik Elektro
Kekhususan / Minat	:	Sistem Komunikasi dan Informatika

KOMISI PEMBIMBING

Ketua	:	Dr. Ir. Erni Yudaningtyas, M.T.
Anggota	:	Rahmadwati, S.T., M.T., Ph.D

TIM DOSEN PENGUJI

Dosen Penguji I	:	Dr.Eng. Panca Mudjirahardjo, S.T., M.T.
Dosen Penguji II	:	Ir. Ponco Siwindarto, M.Eng.Sc
Tanggal Ujian	:	14 Mei 2019
SK Penguji	:	937 Tahun 2019

Karya ilmiah ini ditunjukkan kepada

Ayahanda dan Ibunda tercinta,

dan kekasihku Aryama Ulfa Firdausya

RIWAYAT HIDUP

Muhammad Zulfikri, dilahirkan di Penujak, Lombok Tengah 29 Februari 192 dari pasangan Drs. H. Sulaiman. dan Hj. Maisarah. Pendidikan SD diselesaikan di SDN 3 Penujak, Praya Barat. Pendidikan SMP diselesaikan di SMPN 1 Praya, dan SMA diselesaikan di SMA Darul 'Ulum 1 Jombang. Lulus SMA pada tahun 2009, kemudian melanjutkan studi Diploma 3 (D3) di Jurusan Teknik Elektro, Politeknik Negeri Malang dengan bidang konsentrasi Teknik Elektronika. Setelah menyelesaikan studinya, penulis melanjutkan studi Strata 1 (S1) di Jurusan Teknik Elektro, Universitas Muhammadiyah Malang pada tahun 2013, dan dilanjutkan dengan studi Strata 2 (S2) di Jurusan Teknik Elektro, Universitas Brawijaya Malang pada tahun 2016 dengan bidang minat Sistem Komunikasi dan Informatika.

UCAPAN TERIMA KASIH

Segala puji bagi Allah SWT, atas segala karunia yang telah diberikan sehingga penulis dapat menyelesaikan karya ilmiah ini. Dalam pelaksanaan pembuatan tesis ini, penulis juga banyak menerima bantuan dari berbagai pihak. Oleh karena itu, dengan rasa hormat penulis mengucapkan terima kasih kepada pihak-pihak yang telah membantu antara lain:

1. Kedua orang tua dan keluarga besar penulis atas dukungan yang sudah diberikan.
2. Ibu Dr. Ir. Erni Yudaningtyas, M.T. dan Ibu Rahmadwati, S.T., M.T., Ph.D., selaku Pembimbing, terima kasih atas semua arahan dalam penyelesaian penelitian tesis ini.
3. Bapak Dr.Eng. Panca Mudjirahardjo, S.T., M.T. dan Bapak Ir. Ponco Siwindarto, M.Eng.Sc. selaku dewan penguji, terima kasih atas saran dan masukannya yang membuat penelitian ini menjadi lebih detail.
4. Bapak Dr. Eng. Panca Mudjirahardjo, S.T., M.T. selaku Ketua Program Studi Magister Teknik Elektro, beserta seluruh dosen pengajar dan staf di Program Magister Teknik Elektro Universitas Brawijaya atas sistem pembelajaran dan ilmu pengetahuan yang sudah diberikan.
5. Seluruh rekan-rekan di SKI 2016, serta rekan-rekan mahasiswa Magister Teknik Elektro, Universitas Brawijaya.

Dan semua pihak yang tidak bisa disebutkan satu persatu, yang secara langsung maupun tidak langsung turut membantu menyelesaikan Tesis ini, terima kasih atas segala dukungan dan doa yang diberikan kepada penulis, semoga Allah SWT melimpahkan rahmat dan hidayat sebagai imbalan atas kebaikannya.

RINGKASAN

Muhammad Zulfikri, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya, Juli 2019, *Sistem Pendeteksian Kecepatan Kendaraan yang Diklasifikasikan Haar Cascade Classifier untuk Penegakan Speed Bump*, Dosen Pembimbing: Erni Yudaningtyas dan Rahmadwati.

Perkembangan teknologi yang pesat saat ini memberikan kemudahan bagi kinerja manusia. Salah satunya dalam bidang *Intelligent Transport System* (ITS) yang merupakan integrasi antar sistem informasi dan teknologi komunikasi dengan infrastruktur transportasi, kendaraan dan pengguna jalan. Fasilitas *speed bump* atau polisi tidur merupakan salah satunya yang berfungsi memperingatkan pengendara agar memperlambat laju kendaraan dan memperhatikan daerah sekitarnya. Namun pemasangan *speed bump* dibeberapa tempat menjadi permasalahan tersendiri yaitu dapat memberikan ketidaknyamanan bagi pengemudi kendaraan yang melaju dengan kecepatan rendah.

Mekanisme secara selektif dalam menegakkan speed bump sesuai dengan kecepatan kendaraan yang masuk diperlukan dengan memanfaatkan kamera dengan menggunakan pengolahan citra dalam proses deteksi kendaraan yang menggunakan metode learning yang efektif dalam pendeteksian objek yaitu *Haar cascade classifier*. Hasil deteksi objek mobil yang didapatkan akan dihitung kecepatannya dan menjadi parameter dalam penegakan *speed bump*.

Total data yang digunakan dalam deteksi objek pada pengujian ini berjumlah 12 objek mobil dengan jenis dan warna yang berbeda. Pendeteksian dilakukan pada area khusus atau *Region of Interest* (ROI) dalam satu arah dan jalur, sehingga proses deteksi lebih akurat untuk mendeteksi satu objek mobil dalam satu waktu. Dalam pengukuran kecepatan, dilakukan verifikasi hasil kecepatan secara *real time* dengan kecepatan pada video yang dilakukan dengan melihat kecepatan yang ditampilkan pada *speedometer* pada saat mobil melintasi ROI dan dengan menggunakan sampel mobil dengan kecepatan yang berbeda.

Berdasarkan hasil uji coba pada proses deteksi objek mobil, didapatkan hasil akurasi deteksi yang baik dari berbagai jenis dan warna mobil. Sedangkan dalam proses perhitungan kecepatan, didapatkan nilai *Mean Square Error* (MSE) yaitu 0,6. Hasil yang masih belum baik yang disebabkan oleh proses komputasi yang berat dan mengakibatkan sistem berjalan lambat. Sehingga perlunya device dengan spesifikasi yang lebih tinggi, untuk menghasilkan perhitungan kecepatan yang lebih baik.

Kata Kunci: *speed bump, haar cascade classifier*, deteksi, kecepatan kendaraan

SUMMARY

Muhammad Zulfikri, Department of Electrical Engineering, Faculty of Engineering, University of Brawijaya, July 2019, *Vehicle Speed Detection System Classified by Haar Cascade Classifier for Speed Bump Enforcement*, Academic Supervisor: Erni Yudaningtyas & Rahmadwati.

Rapid technological developments now make it easy for human performance. One of them is in the field of Intelligent Transport System (ITS) which is an integration between information systems and communication technology with transportation, vehicle and road user infrastructure. Speed bump or police sleep facility is one of them that serves to warn motorists to slow down the vehicle and pay attention to the surrounding area. But the installation of speed bumps in some places is a problem in itself, which can provide inconvenience for drivers who drive at low speed.

The mechanism selectively in enforcing speed bump according to the speed of the incoming vehicle is needed by utilizing the camera by using image processing in the vehicle detection process that uses learning methods that are effective in detecting objects namely Haar cascade classifier. The car object detection results obtained will be calculated speed and become a parameter in enforcing speed bump.

The total data used in object detection in this test is 12 car objects of different types and colors. Detection is carried out in special areas or Region of Interest (ROI) in one direction and path, so that the detection process is more accurate for detecting one car object at a time. In speed measurement, verification of speed results in real time is carried out at the speed of the video performed by looking at the speed displayed on the speedometer when the car crosses the ROI and by using samples of cars at different speeds.

Based on the results of trials on the car object detection process, the results of good detection accuracy obtained from various types and colors of the car. While in the speed calculation process, the value of Mean Square Error (MSE) is 0.6. Results that are still not good are caused by a heavy computational process and cause the system to run slowly. So the need for devices with higher specifications, to produce better speed calculations.

Keywords: speed bump, haar cascade classifier, detection, vehicle speed

KATA PENGANTAR

Penelitian dan penyusunan Tesis ini didasari oleh banyak ditemukan fasilitas *speed bump* atau polisi tidur di beberapa ruas jalan yang berfungsi memperingatkan pengendara agar memperlambat laju kendaraan dan memperhatikan daerah sekitarnya. Namun pemasangan *speed bump* dibeberapa tempat menjadi permasalahan tersendiri, seperti yang terpasang di lingkungan sekolah atau persimpangan tertentu dapat memberikan ketidaknyamanan bagi pengemudi kendaraan yang melaju dengan kecepatan rendah. Dengan penelitian ini diharapkan ada mekanisme secara selektif dalam menegakkan *speed bump* sesuai dengan kecepatan kendaraan yang masuk.

Penulis sadar bahwa penelitian ini masih tidak sempurna, oleh karena itu segala saran, masukan dan pertanyaan terkait penelitian ini dapat diungkapkan langsung kepada penulis melalui *email*: mzulfikri@student.ub.ac.id.

Malang, Juli 2019

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Di zaman modern saat ini, bidang teknologi mengalami perkembangan yang sangat pesat. Berbagai terobosan dilakukan oleh para ahli guna mempermudah kinerja manusia dalam berbagai bidang. Salah satunya dalam bidang *Intelligent Transport System* (ITS) yang merupakan integrasi antar sistem informasi dan teknologi komunikasi dengan infrastruktur transportasi, kendaraan dan pengguna jalan. Sistem ini diterapkan untuk mengelola dan mengendalikan lalu lintas, dsitribusi kendaraan dan infrastruktur untuk mencapai sistem transportasi yang lebih aman dan nyaman.

Di beberapa ruas jalan, banyak ditemukan fasilitas *speed bump* atau polisi tidur yang berfungsi memperingatkan pengendara agar memperlambat laju kendaraan dan memperhatikan daerah sekitarnya. Namun pemasangan *speed bump* dibeberapa tempat menjadi permasalahan tersendiri, seperti yang terpasang di lingkungan sekolah atau persimpangan tertentu dapat memberikan ketidaknyamanan bagi pengemudi kendaraan yang melaju dengan kecepatan rendah. Diharapkan ada mekanisme secara selektif dalam menegakkan *speed bump* sesuai dengan kecepatan kendaraan yang masuk.

Umumnya dalam mendeteksi kecepatan kendaraan, metode yang sering digunakan adalah menggunakan pistol radar. Pistol radar bekerja atas dasar efek *dopler*, yaitu dimana pistol radar memancarkan suatu gelombang radar yang diarahkan pada suatu objek yang bergerak dan dipantulkan kembali untuk kemudian dihitung kecepatan objek tersebut. Meskipun teknologi radar memberikan hasil yang menjanjikan dan sangat akurat, namun teknologi radar masih memiliki kelemahan, dimana teknologi radar sangat mahal dan juga jika ada perangkat yang menghasilkan gelombang radio di sekitarnya, hasilnya deteksinya akan terpengaruh (Jianping dkk, 2009).

Secara keseluruhan penelitian ini pernah dilakukan sebelumnya oleh Cheng-Yu dkk (2016), yang menyajikan sistem *speed bump* digabungkan dengan deteksi kecepatan dan teknik pengenalan plat nomor. Penelitian ini memiliki kelebihan di dalam menonaktifkan sistem *speed bump* sesuai dengan plat nomor yang terdaftar dalam database. Namun perlunya pengembangan dari aspek metode di dalam mendeteksi kecepatan kendaraan yang

masih menggunakan dua sensor tekanan untuk mengetahui perbedaan waktu pemicu antara kedua sensor tersebut.

Sensor tekanan yang digunakan pada penelitian Cheng-Yu dkk, memiliki prinsip kerja yang sama dengan sensor piezoelektrik. Sensor tersebut pada dasarnya terdiri dari dua bidang yang berdempet, yang akan menghasilkan tegangan listrik ketika bidang piezo mengalami tekanan. Menurut Mimbela & Lawrence (2000), sensor piezoelektrik memiliki kemampuan yang baik untuk menentukan dan memantau bobot dari kendaraan, sehingga baik jika digunakan dalam mendeteksi kendaraan. Meskipun begitu sensor tersebut memiliki kelemahan tersendiri, dimana dalam proses instalasi di permukaan jalan yang buruk dan jika terdapat adanya perbaikan dan pelapisan kembali pada jalan raya, dapat membutuhkan instalasi ulang pada sensor. Selain itu, sensor piezoelektrik sensitif terhadap temperatur jalan yang dapat membatasi penggunaan dari sensor.

Penelitian terkait dengan metode deteksi objek (kendaraan) di jalan raya telah dilakukan oleh berberapa peneliti, diantaranya penelitian yang dilakukan oleh Thadagoppula & Vikas (2016) yang menggunakan metode *adaptive background subtraction*. Metode tersebut digunakan untuk memisahkan antara gambar latar (*background*) dengan objek yang akan di deteksi atau kendaraan (*foreground*). Terdapat proses pembaharuan secara *self-adapting* pada *background* yang terjadi jika terjadi perubahan intensitas cahaya atau terdapat benda stasioner yang telah ada sejak lama dan bergerak keluar pada *background*, hal tersebut akan meninggalkan lubang dibelakangnya yang akan terdeteksi sebagai objek palsu. Metode tersebut bekerja dengan baik dan tahan terhadap perubahan cuaca dan penerangan, namun masih diperlukan perluasan penelitian pendeteksian dalam situasi berkabut.

Menurut Xiaobin dkk (2016), metode *background subtraction* merupakan metode berbasis gerakan yang dapat dilakukan secara *real time* dan umumnya tingkat pendeteksian yang cukup tinggi karena tidak diperlukan perhitungan yang rumit dan sebagian besar area gerakan dapat dengan mudah dideteksi. Namun, metode ini biasanya hanya mempertimbangkan gerak kendaraan dan mengabaikan fitur kendaraan lainnya. Oleh karena itu, metode berbasis gerakan sensitif terhadap *noise* dan biasanya dapat mencapai tingkat kesalahan deteksi yang tinggi, berbeda dengan metode berbasis *appearance*/tampilan yang umumnya lebih kuat. Selain itu, metode *learning* untuk deteksi kendaraan saat ini telah mengikuti tren dalam *machine learning*.

Keberadaan pendeteksi kecepatan kendaraan yang masih sepenuhnya belum sempurna dari berbagai aspek inilah yang mendorong peneliti untuk mencari alternatif yang lebih baik dalam hal performa dan biaya. Dalam penelitian ini, peneliti menawarkan sistem deteksi

kendaraan menggunakan kamera dengan menggunakan metode pengolahan citra yang dapat memberikan hasil lebih handal dan dengan biaya yang lebih rendah. Pengolahan citra saat ini banyak dikembangkan untuk mendeteksi kecepatan kendaraan. Dengan pengembangan tersebut, dapat membantu penegak hukum dalam mengetahui aktivitas kecurangan yang dilakukan oleh pengemudi. Proses deteksi kendaraan menggunakan *Haar cascade classifier*, yang merupakan salah satu metode *learning* yang efektif dalam pendeteksian objek. Metode ini menggabungkan beberapa konsep yaitu *Haar Features*, *Integral Image*, *AdaBoost Learning,* dan *Cascade Classifier* menjadi sebuah metode utama untuk mendeteksi objek. Selanjutnya hasil deteksi yang didapatkan akan dihitung kecepatannya dan menjadi parameter penegakan *speed bump*.

1.2. Persamaanan Masalah

Pada penelitian ini dapat dipersamaankan suatu permasalahan sebagai berikut :

1. Bagaimana penerapan metode *Haar Cascade Classifier* untuk mendeteksi objek kendaraan?
2. Bagaimana membangun sistem penghitung kecepatan kendaraan pada *Region of Interest* (*ROI*) yang menjadi parameter penegakan *speed bump*?
3. Bagaimana analisa efektivitas kerja sistem yang dirancang berdasarkan tingkat akurasi deteksi dan perhitungan kecepatan.

1.3. Batasan Masalah

Pada penelitian diberikan batasan-batasan berupa :

1. Video yang digunakan yaitu dengan resolusi 320x240 piksel dengan kondisi siang hari (intensitas cahaya tinggi).
2. Menggunakan *library* OpenCV versi 3.1.0.
3. Menggunakan *interpreter* Python 3.5.6.
4. Kendaraan yang di deteksi adalah kendaraan roda empat dengan dimensi maksimal panjang 5,5 meter, lebar 2 meter dan tinggi 2 meter.
5. Pemantauan dilakukan pada jalur lurus dalam satu arah dengan sudut pandang kendaraan terhadap kamera adalah tampak dari samping jalan.
6. Tidak mebahas mengenai perancangan *speed bump*.

1.4. Tujuan Penelitian

Penulis berharap pada proses pembelajaran maupun proses pengujian pada sistem pendeteksian kecepatan kendaraan yang diklasifikasikan *haar cascade classifier* untuk penegakan *speed bump* ini dapat dijalankan dengan baik dan metode *Haar Cascade Classifier* yang digunakan dapat mendeteksi berbagai jenis kendaraan serta hasil perhitungan kecepatan kendaraan yang didapatkan dapat menentukan penegakan *speed bump*.

1.5. Manfaat Penelitian

Penelitian ini bermanfaat bagi pihak perguruan tinggi dalam mengembangkan istem pendeteksian kecepatan kendaraan yang diklasifikasikan *haar cascade classifier* untuk penegakan *speed bump*. Selain itu, penelitian ini juga bermanfaat bagi pihak penegak hukum yang ingin menerapkan sistem tersebut pada lokasi atau jalan yang rawan kecelakaan lalu lintas akibat kendaraan yang melaju dengan kencang.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Relevan

Penjelasan secara lengkap mengenai beberapa penelitian yang terkait ditunjukkan sebagai berikut :

Penelitian yang dilakukan oleh Cheng-Yu dkk (2016) dengan membuat prototipe yang menyajikan sistem *speed bump* cerdas yang digabungkan dengan deteksi kecepatan dan teknik pengenalan plat nomor yang dinamis. Idenya adalah untuk mengetahui penegakan *speed bump* berdasarkan kecepatan dan deteksi kendaraan. Sebagai bagian dari sistem transportasi cerdas untuk menyediakan kontrol arus lalu lintas, teknik yang diusulkan membuat berkendara lebih aman dan lebih nyaman. Pada tahap pengenalan plat lisensi dinamis, pelacakan adaptif dilakukan untuk alokasi plat nomor, diikuti oleh segmentasi karakter dan identifikasi. "*Tesseract OCR*" kemudian digunakan untuk membuat gambar sampel dan melakukan tugas pengenalan karakter. Percobaan dilakukan pada sistem prototipe dengan *embedded computation* (Cheng-Yu dkk, 2016).

Penelitian selanjutnya dilakukan oleh Bhargava dan Drdinesh (2014) dengan mengembangkan sistem deteksi kecepatan kendaraan dan penegakan hukum menggunakan pengenalan plat nomor otomatis. Pengolahan citra telah digunakan untuk sistem pengawasan lalu lintas cerdas. Kfeseluruhan bekerja untuk sistem terdiri dari pengembangan perangkat lunak dan pengembangan perangkat keras. Pendekatan menggunakan satu sistem kamera yang dipasang di tiang atau lampu lalu lintas yang mendeteksi kendaraan yang melaju kencang dan mengekstraksi plat nomornya. Sistem ini bekerja dengan menangkap bingkai video dari objek yang bergerak dengan menggunakan pengolahan citra untuk mendapatkan *differencing image, binary image*. Algoritma kemudian melanjutkan untuk memberi label pada semua komponen yang terhubung, *bounding box* dan pusat gambar yang diekstraksi untuk menghitung kecepatannya. Algoritma ini dirancang untuk memperkirakan kecepatan kendaraan pada jalur linier dan melingkar. Algoritma ini juga menangkap bingkai kendaraan yang bergerak untuk merekamnya. Pada algoritma pelanggaran kecepatan dengan mendeteksi informasi plat nomor menggunakan teknik pengenalan karakter optik (Bhargava dan Drdinesh, 2014).

Penelitian yang dilakukan oleh Thadagoppula dan Vikas (2016) dengan mengembangkan sistem deteksi kecepatan kendaraan menggunakan metode pengolahan citra. Metode tersebut merupakan metode alternatif untuk mengatasi kelemahan senjata radar. Dengan menggunakan *live video stream* dari kamera cctv untuk menghitung kecepatan kendaraan. Kecepatan kendaraan diupdate setiap setengah detik, sehingga menjaga track pada akselerasi dan perlambatan kendaraan di bidang pandang kamera. Setiap pelanggaran dapat diamati dan diberitahukan kepada petugas hukum. Hal tersebut membantu melacak pelanggaran kecepatan dan menghemat usaha seorang petugas yang memegang pistol radar di jalan raya. Hasil peritungan kecepatan menunjukkan kesalahan hanya 3% secara *real time* dan diuji dalam banyak percobaan. Hasil yang didapatkan terbukti telah mencapai performa yang memuaskan (Thadagoppula dan Vikas, 2016).

Pada penelitian yang dilakukan oleh Lazaro dkk (2017) yang menegmbangkan sistem deteksi dan menghitung kendaraan yang terdeteksi berdasarkan jenisnya. Tujuannya yaitu sistem mampu mendeteksi jenis kendaraan pada suatu video dan menghitung jumlah kendaraan berdasarkan jenisnya. Pengguna cukup memasukkan video rekaman lalu lintas sistem akan memproses video menghasilkan file .txt sebagai keluaran. File tersebut berisi jenis dan jumlah kendaraan yang terdeteksi berdasarkan jenisnya. Dari hasil pengujian, program memiliki akurasi rata-rata 77.8% untuk kondisi jalan sepi, 47.5% untuk kondisi jalan normal, dan 28.2% untuk kondisi jalan padat. Hal-hal yang mempengaruhi akurasi deteksi adalah sudut pandang kendaraan terhadap kamera, jarak antar kendaraan, serta waktu tempuh kendaraan dalam memasuki hingga keluar dari wilayah deteksi (Lazaro dkk, 2017).

Pada tahun 2016, Sudirman dan Nur, melakukan penelitian tentang traffic monitoring dengan dengan memanfaatkan kamera pengawas jalan dan metode *optical flow* sebagai metode dalam memproses video untuk mendapatkan kecepatan dan jumlah kendaraan yang melintas. Sistem diujikan pada jalan raya dengan lajur jalan sebanyak tiga. Berdasarkan hasil pengujian dengan membandingkan selisih hasil antara perhitungan kecepatan secara manual dan secara sistem yang tidak signifikan (kecil), didapatkan sistem yang mampu menghitung kecepatan kendaraan yang bergerak dengan cukup akurat selama kendaraan tersebut tetap berada dalam lajur jalan yang ditandai. Sedangkan pada penghitungan jumlah kendaraan terdapat perbedaan yang diakibatkan masuknya objek lain atau berpindahnya suatu objek secara tiba-tiba di daerah *ROI* (Sudirman dan Nur, 2016).

Tabel 2.1 *Penelitian Relevan*

No.	Penulis	Judul	Metode	Hasil
1	Cheng-Yu Ho, Huei-Yung Lin, & Lu-Ting Wu (2016)	*Intelligent Speed Bump System With Dynamic License Plate Recognition*	- Deteksi kecepatan kendaraan memanfaatkan perbedaan waktu pemicu antara dua sensor tekanan. - Pengenalan plat nomor menggunakan *image processing* yang dibagi menjadi lima tahap, (i) akuisisi citra dinamis dan *pre-processing*, (ii) identifikasi lokasi plat, (iii) segmentasi karakter, (iv) *adaptive license plate localization* dan *adaptive character labeling*, (v) pengenalan karakter.	Hasil penelitian yang menggunakan sistem berbasis PC dan *embedded platform* didapatkan hasil yang baik dalam konsep *quick bump control control* yang diusulkan.
2	Kritika Bhargava & Dinesh Goyal (2014)	*A Video Surveillance System for Speed Detection of Vehicles and Law Enforcement using Automatic Number Plate Recognition*	Menggunakan *image processing* yang dibagi menjadi, - *Object Detection* – menggunakan kombinasi algoritma *adaptive background subtraction* dengan *three-frame differencing algorithm*. - *Object Tracking* – menggunakan *object segmentation*, *object labeling*, dan *center extraction*. - *Speed Calculation* - menggunakan berbagai fungsi MATLAB - Pengenalan Plat Nomor - menggunakan teknologi OCR.	Sistem bekerja dengan baik dalam penentuan kecepatan lalu lintas dan dapat menjadi solusi *real time* untuk sistem kontrol lalu lintas.
3	Pranith Kumar Thadagoppula & Vikas Upadhyaya (2016)	*Speed Detection using Image Processing*	Menggunakan *image processing* yang dibagi menjadi, - *Object Detection* - menggunakan *adaptive background subtraction* untuk memisahkan *background* dengan *foreground* (kendaraan). - *Object tracking* - melibatkan *shadow removal*, *convex hull*, *morphological operations*, merencanakan *bounding box* di sekitar benda yang terdeteksi, menemukan *centroid*dan melacak benda-benda di setiap bingkai video. - *Speed calculation* – menghitng kecepatan akhir dalam km / jam. -	Metode yang digunakan memiliki kesalahan maksimum sebesar 3% pada kecepatan yang terdeteksi.

No.	Penulis	Judul	Metode	Hasil
4	Alvin Lazaro, Joko Lianto Buliali, Bilqis Amaliah (2017)	Alvin Lazaro - Deteksi Jenis Kendaraan di Jalan Menggunakan OpenCV	Deteksi jenis kendaraan menggunakan metode *haar cascade classifier* serta menghitung kendaraan yang terdeteksi berdasarkan jenisnya.	Tingkat akurasi rata-rata untuk tiga kondisi jalan, 77.8% untuk kondisi jalan sepi, 47.5% untuk kondisi jalan normal, 28.2% untuk kondisi jalan padat.
5	Dirvi Eko Juliando Sudirman, & Nur Asyik Hidayatullah (2016)	Traffic Monitoring : Sistem Penghitung Jumlah dan Pengukur Laju Kecepatan Kendaran Bermotor pada Jalan Tiga Lajur Berbasis Optical Flow	Memanfaatkan kamera pengawas jalan dan menggunakan metode *optical flow* sebagai metode untuk memproses video untuk mendapatkan kecepatan dan jumlah ken-daraan yang melintas.	Hasil pengujian didapatkan selisih hasil antara perhitungan kecepatan secara manual dan secara sistem yang tidak signifikan (kecil), sehingga didapatkan sistem yang mampu menghitung kecepatan kendaraan yang bergerak dengan cukup akurat selama kendaraan tersebut tetap berada dalam lajur jalan yang ditandai. Sedangkan pada penghitungan jumlah kendaraan terdapat perbedaan yang diakibatkan masuknya objek lain atau berpindahnya suatu objek secara tiba-tiba di daerah *ROI*.

2.2 Kecelakaan Lalu Lintas

Menurut Peraturan Pemerintah Nomor 43 tahun 1993 pasal 93 tentang Prasarana dan Lalu Lintas Jalan, kecelakaan lalu lintas adalah suatu peristiwa di jalan yang tidak disangkasangka dan tidak disengaja melibatkan kendaraan dengan atau tanpa pemakai jalan lainnya, mengakibatkan korban manusia atau kerugian harta benda. Korban kecelakaan lalu lintas sebagaiman dimaksud dalam hal ini adalah terbagi menjadi 3 (tiga), yaitu : Korban Mati, korban luka berat dan korban luka ringan.

Lalu lintas ditimbulkan oleh adanya pergerakan dari alat-alat angkutan, karena adanya kebutuhan perpindahan manusia dan atau barang. Karena itu, dampak yang tidak mungkin ditolak karena adanya pergerakan tersebut adalah terjadinya kecelakaan. Kecelakaan dapat disebabkan oleh faktor pemakai jalan (pengemudi dan pejalan kaki), faktor kendaraan dan faktor lingkungan (Pignataro, 1973). Pignataro juga menyatakan bahwa kecelakaan diakibatkan oleh kombinasi dari beberapa faktor perilaku buruk dari pengemudi ataupun pejalan kaki, jalan, kendaraan, pengemudi ataupun pejalan kaki, cuaca buruk ataupun pandangan yang buruk.

Hobbs (1979) mengelompokkan faktor-faktor penyebab kecelakaan menjadi tiga kelompok, yaitu :

1. Faktor manusia

Faktor manusia merupakan faktor yang paling dominan dalam kecelakaan. Hampir semua kejadian kecelakaan didahului dengan pelanggaran rambu-rambu lalu lintas. Pelanggaran dapat terjadi karena sengaja melanggar, ketidaktahuan terhadap arti aturan yang berlaku ataupun tidak melihat ketentuan yang diberlakukan atau pula pura-pura tidak tahu.Selain itu manusia sebagai pengguna jalan raya sering sekali lalai bahkan ugal ugalan dalam mengendarai kendaraan, tidak sedikit angka kecelakaan lalu lintas diakibatkan karena membawa kendaraan dalam keadaan mabuk, mengantuk, dan mudah terpancing oleh ulah pengguna jalan lainnya yang mungkin dapat memancing gairah untuk balapan.

2. Faktor kendaraan

Faktor kendaraan yang paling sering adalah kelalaian perawatan yang dilakukan terhadap kendaraan. Contoh nya seperti rem blong, setir macet, dll.

Untuk mengurangi faktor kendaraan perawatan dan perbaikan kendaraan diperlukan, disamping itu adanya kewajiban untuk melakukan pengujian kendaraan bermotor secara reguler.

3. Faktor jalan dan lingkungan

Faktor jalan terkait dengan kecepatan, rencana jalan, geometrik jalan, pagar pengaman di daerah pegunungan,ada tidaknya median jalan, jarak pandang dan kondisi permukaan jalan. Jalan yang rusak/berlobang sangat membahayakan pemakai jalan terutama bagi pemakai sepeda dan sepeda terbang

2.3 Kecepatan

2.3.1. Pengertian Kecepatan

Kecepatan adalah besaran yang menunjukkan jarak yang ditempuh kendaraan dibagi waktu tempuh, biasanya dinyatakan dalam km/jam. Kecepatan ini menggambarkan nilai gerak dari kendaraan.Perencanaan jalan yang baik tentu saja harus berdasarkan kecepatan yang dipilih dari keyakinan bahwa kecepatan tersebut sesuai dengan kondisi dan fungsi jalan yang diharapkan (Sukirman, 1994).

Menurut Hobbs (1979), menyatakan bahwa, kecepatan umumnya dibagi menjadi tiga jenis, yaitu :

1. Kecepatan setempat (*spot speed*) adalah kecepatan kendaraan pada suatu saat diukur dari suatu tempat yang ditentukan.
2. Kecepatan begerak (*running speed*) adalah kecepatan kendaraan rata-rata pada suatu jalur saat kendaraan bergerak dan didapat dengan membagi panjang jalur dengan lawa waktu kendaraan bergerak menempuh jalur tersebut.
3. Kecepatan perjalanan (*journey speed*) adalah kecepatan efektif kendaraan yang sedang dalam perjalanan Antara dua tempat dan merupakan jarak Antara dua tempat dibagi dengan lama waktu bagi kendaraan untuk menyelesaikan Antara dua tempat tersebut, dengan lama waktu yang termasuk di dalamnya waktu berhenti yang ditimbulkan oleh hambatan (penundaan) lalu intas.

2.3.2. Batas Kecepatan

Pada Manual Kapasitas Jalan Indonesia (MKJI) 1997, disebutkan bahwa batas kecepatan, jika secara tepat dilaksanakan, dapat mengurangi tingkat kecelakaan sesuai dengan faktor (Vsesudah / Vsebelum). Di dalam Undang-Undang Republik Indonesia Nomor 22 Tahun 2009 Tentang Lalu Lintas dan Angkutan Jalan pada pasal 21 ayat 1 dan 2 disebutkan bahwa setiap jalan memiliki batas kecepatan paling tinggi yang ditetapkan secara nasional dan ditentukan bedasarkan kawasan permukiman, kawasan perkotaan, jalan antar kota, dan jalan bebas hambatan. Selanjutnya berdasarkan Peraturan Pemerintah Republik

Indonesia Nomor 79 Tahun 2013 Tentang Lalu Lintas dan Angkutan Jalan pada pasal 23 ayat 4 huruf a sampai d ditetapkan batas kecepatan sebagaimana berikut :

a. paling rendah 60 (enam puluh) kilometer per jam dalam kondisi arus bebas dan pling tinggi 100 (seratus) kilometer per jam untuk jalan bebas hambatan;
b. paling tinggi 80 (delapan puluh) kilometer per jam untuk jalan antar kota;
c. paling tinggi 50 (lima puluh) kilometer per jam untuk kawasan perkotaan; dan
d. paling tinggi 30 (tiga puluh) kilometer per jam untuk kawasan permukiman.

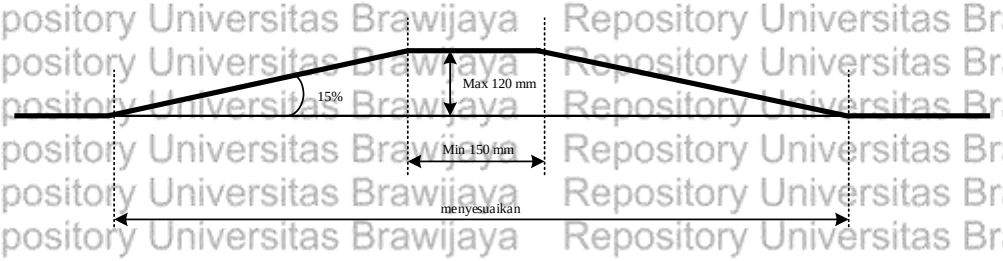
2.4. Speed Bump

2.4.1 Pengertian *Speed Bump*

Menurut Keputusan Menteri Perhubungan No 3 Tahun 1994 tentang Alat Pengendali dan Pengaman Pemakai Jalan, alat pembatas kecepatan (*speed bump*) adalah kelengkapan tambahan pada jalan yang berfungsi untuk membuat pengemudi kendaraan bermotor mengurangi kecepatan kendaraannya. Kelengkapan tambahan dapat berupa peninggian sebagian badan jalan yang melintang terhadap sumbu jalan dengan lebar, tinggi, dan kelandaian tertentu.

2.4.2 Pengaturan *Speed Bump*

Di Indonesia, ketentuan yang mengatur tentang desain polisi tidur diatur dalam Keputusan Menteri Perhubungan No 3 Tahun 1994 tentang Alat Pengendali dan Pengaman Pemakai Jalan, dimana sudut kemiringan adalah 15% dan tinggi maksimum tidak lebih dari 120mm dan lebar 150mm seperti pada Gambar 2.1. Selain itu *speed bump* harus diberikan 2 warna berbeda dengan lebar warna : 20 cm dan 30 cm seperti pada Gambar 2.2.



Gambar 2.1 Desain standar Alat Pembatas Kecepatan (Polisi Tidur) Berdasarkan KM Menhub No. 3 Tahun 1994

Sumber: Kemenhub No. 3 Tahun 1994

Gambar 2.2 Desain standar Alat Pembatas Kecepatan (Polisi Tidur) Berdasarkan KM Menhub No. 3 Tahun 1994
Sumber: Kemenhub No. 3 Tahun 1994

2.4.3 *Speed Bump* dinamis

Speed bump dinamis berbeda dari *speed bump* konvensional dimana hanya akan aktif jika kendaraan yang melintas di atasnya melaju melebihi batas kecepatan tertentu. Kendaraan yang melaju dengan kecepatan yang tidak melebihi batas, tidak akan mempengaruhi *speed bump* tersebut. *Speed bump* dinamis memungkinkan lewatnya kendaraan-kendaraan darurat pada kecepatan tinggi.

Dalam satu desain, sebuah karet dilengkapi dengan katup tekanan yang mampu mengetahui kecepatan dari sebuah kendaraan. Jika kendaraan tersebut melintas dibawah batas kecepatan maka katup tersebut akan terbuka dan *speed bump* akan menjadi datar, tetapi katup tetap tertutup bila kendaraan tersebut melaju terlalu cepat. Katup tersebut juga dapat diatur untuk memungkinkan kendaraan berat, seperti mobil pemadam kebakaran, ambulans, dan bus untuk lewat pada kecepatan yang tinggi.

2.5 Citra Digital

Citra adalah suatu representasi (gambaran), kemiripan, atau imitasi dari suatu objek. Citra terbagi 2 yaitu citra yang bersifat analog dan ada citra yang bersifat digital. Citra analog adalah citra yang bersifat continue seperti gambar pada monitor televisi, foto sinar X, dan lain-lain. Sedangkan pada citra digital adalah citra yang dapat diolah oleh komputer (Sutoyo, 2009).

Citra dapat didefinisikan sebagai fungsi f(x,y) berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitudo f di titik koordinat (x,y) dinamakan intensitas atau tingkat keabuan dari citra pada citra tersebut (Putra, 2010:19).

2.5.1 Pengolahan Citra

Pengolahan citra adalah segala kegiatan yang dilakukan untuk melakukan perubahan pada citra tersebut seperti memanipulasi dan memodifikasi citra dengan berbagai cara (Sutoyo, 2009). Pengolahan citra berdasarkan tujuan operasi pengolahan citra dapat dikatagorikan sebagai berikut:

a. Peningkatan Kualitas Citra (*Image Enhancement*)

Operasi yang bertujuan untuk melakukan peningkatan terhadap kualitas citra dengan meningkatkan fitur tertentu pada citra.

b. Pemulihan Citra (*Image Restoration*)

Operasi pemulihan citra bertujuan untuk mengembalikan kondisi citra pada kondisi yang diketahui sebelumnya, yang diakibatkan oleh faktor pengganggu yang menyebabkan penurunan kualitas citra.

2.5.2 Segmentasi Citra

Segmentasi citra merupakan proses untuk membagi citra menjadi bagian-bagian yang mengandung nilai koherenitas dari property yang dianalisis. Segmentasi sering dianalogikan sebagai proses pemisahan latar depan dan latar belakang dengan memilih piksel-piksel dalam suatu nilai yang memiliki nilai koherenitas sebagai latar depan dan menolak sisanya sebagai latar belakang. Dengan demikian, citra terbagi atas dua bagian yang dinyatakan dengan warna tertentu yang membatasi setiap wilayah (Sutoyo, 2009).

2.5.3 Citra *Threshold*

Citra *threshold* digunakan untuk mempertegas citra dengan mengubah citra yang memiliki derajat keabuan, menjadi 2 warna yaitu hitam dan putih berdasarkan nilai ambang (*threshold*). Pada proses *threshold* di haruskan untuk menentukan nilai *threshold* (T) dimana piksel yang bernilai dibawah nilai *threshold* akan presentasikan dengan warna hitam dan piksel yang bernilai diatas nilai *threshold* akan dipresentasikan dengan warna putih. Nilai *Threshold* secara umum dihitung dengan menggunakan persamaan:

$$T = \frac{fmaks + fmin}{2} \quad \text{.....(2-1)}$$

dengan,

T = nilai *threshold*

fmaks = nilai intensitas maksimum pada citra

fmin = nilai intensitas minimum pada citra

2.5.4 Resolusi Citra

Resolusi citra merupakan istilah untuk menerangkan ukuran citra digital yang digambarkan dengan banyaknya piksel yang digunakan untuk menvisualisasikan citra digital. Resolusi dalam citra juga menjadi acuan untuk menentukan tingkat detailnya suatu citra. Semakin tinggi resolusinya semakin tinggi pula tingkat detail dari citra tersebut begitu juga sebaliknya. Resolusi dalam citra digital memiliki dua jenis resolusi, yaitu:

1. Resolusi Sparsial

Resolusi spasial ini merupakan ukuran halus atau kasarnya visualisasi citra dengan melakukan pembagian piksel pada baris dan kolom. Resolusi ini dipakai untuk menentukan banyaknya piksel per satuan panjang, umumnya satuan resolusi yang digunakan adalah dpi (*dot per inch*). Resolusi ini berpengaruh pada detail pada suatu citra digital.

2. Resolusi Kecemerlangan

Resolusi intensitas kecemerlangan (*brightness*) juga disebut dengan kedalaman bit warna (*Bit Depth*), kedalaman bit warna (*Bit Depth*) adalah ukuran nilai banyaknya pembagian tingkat gradasi warna yang mempengaruhi halus kasarnya tingkat gradasi saat dilakukan kuantisasi. *Bit Depth* menentukan informasi warna untuk divisualisasikan dalam setiap piksel. Semakin besar nilai dari *Bit Depth*, semakin banyak tingkat gradasi yang dapat dihasilkan dan mempengaruhi tingkat kualitas gambar yang dihasilkan (Sutoyo, 2009).

2.5.5. Citra Biner

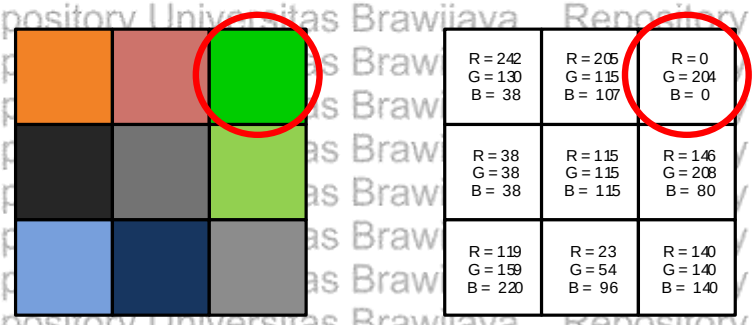
Citra biner adalah jenis dari citra digital yang hanya mempunyai 2 warna yaitu hitam dan putih, pada ini hanya dibutuhkan 1 bit di memori untuk menyimpan informasi warna (Sutoyo, 2009). Pada Gambar 2.4 diperlihatkan perubahan Citra True Color menjadi Citra Biner.

(a) (b)

Gambar 2.4 (a) Citra T*rue Color*, (b) Citra Biner

2.5.6. Citra Warna (*True Colour*)

Citra warna adalah sebuah citra yang memiliki nilai warna pada setiap piksel yang dibentuk dari nilai kombinasi dari warna dasar (RGB = *Red green blue*). Setiap warna dasar disimpan dalam memori dengan ukuran 1 byte, yang berarti dalam setiap 1 warna dasar mempunyai tingkat gradasi sebanyak 255 warna. Berbeda dengan citra grayscale, pada penyimpanan didalam memori setiap 1 piksel citra true color diwakili oleh 3 byte yang masing - masing byte merepresentasikan warna merah (*Red*), hijau (*Green*), biru (*Blue*) , Sedangkan setiap 1 piksel dari citra grayscale memiliki 256 gradasi warna diwakili oleh 1 byte (Sutoyo, 2009). Contoh seperti pada Gambar 2.5.(a) terdapat warna hijau yang dilingkari, merupakan kombinasi warna merah, hijau dan biru sehingga nilai RGB-nya adalah 146 208 80 (Gambar 2.5.(b)).



(a) (b)

Gambar 2.5 (a) Citra Warna, (b) Penyimpanan warna di memori

2.5.7. Citra Skala Keabuan (*Grayscale*)

Skala keabuan adalah model penyederhanaan dari citra warna yang memiliki nilai warna yang direpresentasikan oleh warna merah (*Red*), hijau (*Green*), biru (*Blue*) yang akan diubah dengan mengubah 3 bagian warna dasar RGB menjadi 1 bagian warna *grayscale* sehingga dihasilkan citra *grayscale* seperti pada Gambar 2.6. Citra *grayscale* tidak mempunyai nilai RGB melainkan nilai dari derajat keabuan. Dalam mengubah citra warna menjadi citra grayscale dilakukan dengan mendapatkan nilai rata rata dari setiap bagian warna dasar yaitu bagian warna merah, warna hijau dan warna biru yang ditunjukkan pada formula (2-2)

$$S = (r + g + b)/3 \quad \text{.....(2-2)}$$

dengan,

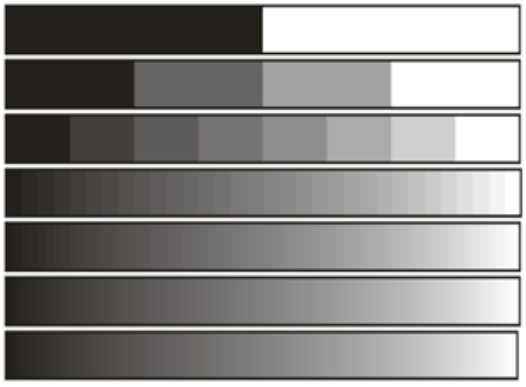
S = skala keabuan

r, g, b = nilai warna dari warna merah (*Red*), hijau (*Green*), biru (*Blue*)

(a) (b)

Gambar 2.6 (a) Citra *True Color*, (b) Citra Skala Keabuan (*Grayscale*)

Banyaknya gradasi warna diwakili oleh banyaknya memory yang di sediakan, semakin besar memori semakin banyak warna di tampung semakin banyak warna gradasi yang terbentuk (Gambar 2.7)



Citra 1 bit = 2^1 = 2 gradasi warna

Citra 2 bit = 2^2 = 4 gradasi warna

Citra 3 bit = 2^3 = 8 gradasi warna

Citra 5 bit = 2^5 = 32 gradasi warna

Citra 6 bit = 2^6 = 64 gradasi warna

Citra 7 bit = 2^7 = 128 gradasi warna

Citra 8 bit = 2^8 = 256 gradasi warna

Gambar 2.7 Perbandingan Gradasi Warna Berdasarkan Gradasi

Sumber: www.kompasiana.com

2.6. Video

Video adalah kumpulan dari beberapa *frame* yang merupakan citra digital yang ditampilkan dengan kecepatan tententu sesuai nilai dari banyaknya *frame* yang bisa ditampilkan pada setiap satuan detik atau biasa disebut dengan istilah *frame rate* dengan satuan fps (*frame per second)*. Karakteristik dari suatu video akan menentukan kualitas video yang ditentukan oleh resolusi, kedalam bit dan *frame* rate (Sutoyo, 2009), yang akan dijabarkan sebagai berikut:

a. Resolusi adalah ukuran dari sebuah *frame* yang dinyatakan dengan mengkalikan antara banyaknya nilai piksel lebar dan nilai piksel panjang, nilai yang didapat

perpengaruh terhadap kualitas video yang dihasilkan, dalam artian semakin tinggi resolusi maka semakin baik kualitas dan semakin detail video yang dihasilkan.

b. Kedalaman bit (*Bit Depth*) digunakan untuk menerangkan nilai dari jumlah bit yang digunakan untuk merepresentasikan tiap piksel pada sebuah *frame* yang dinyatakan dalam bit/piksel. Bit Depth mempengaruhi kualitas dari video dalam artian semakin tinggi kedalaman pikselnya maka semakin baik kualitas video yang dihasilkan.
c. *Frame* rate adalah banyaknya *frame* dari citra yang bisa ditampilkan pada setiap satuan detik. *Frame* rate mempengaruhi kualitas dari video dalam artian semakin banyak *frame* yang ditampilkan per detik maka semakin baik kualitas video yang dihasilkan.

2.7. Computer Vision

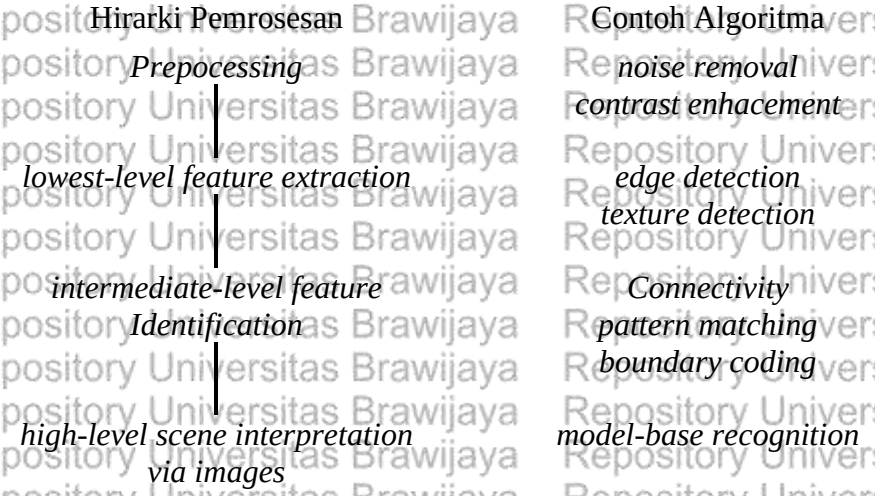
Computer vision merupakan sejumlah proses yang terintegrasi secara otomatis yang diperuntukan sebagai persepsi visual, seperti akuisisi citra, pengolahan citra, pengenalan, dan pembuatan keputusan. *Computer vision* mempunyai tujuan untuk meniru atau menjelaskan mengenai cara kerja sistem visual manusia. Hal yang ingin ditiru oleh computer vision adalah cara kerja dimana mata manusia menangkap objek dan merepresentasikan objek tersebut menuju retina, yang kemudian retina akan merubah objek tersebut menjadi sinyal-sinyal yang dimengerti oleh otak dan pada akhirnya otak akan memutuskan untuk mengenali objek apakah yang ditangkap oleh mata (Munir, 2004).

Computer vision terdiri dari teknik-teknik untuk mengestimasi ciri-ciri objek yang terdapat dalam citra, geometri objek yang berkaitan dengan pengukuran ciri yang kemudian menginterpretasi informasi yang dapat diambil dari geometri tersebut (Munir, 2004).

Computer vision memiliki tiga buah aktivitas proses yang meliputi:

1. Memperoleh atau mengakuisisi citra digital;
2. Melakukan teknik komputasi untuk memperoses dan memodifikasi data citra (operasi-operasi pengolahan citra);
3. Menganalisis dan menginterpretasi citra menggunakan hasil pemrosesan untuk tujuan tertentu.

Jika proses-proses didalam *computer vision* di klasifikasikan maka akan terbentuk hirarki sebagai berikut:



Gambar 2.8 Hirarki dari *Computer Vision*

Dari Gambar 2.8, pengolahan citra dan pengenalan pola merupakan salah satu hirarki yang merupakan bagian dari computer vision, tetapi untuk menlakukan pengolahan citra maupun pengenalan pola, terlebih dahulu citra harus melalui tahap awal (preprocessing) pada computer vision. Untuk mengenali jenis dari objek yang ditangkap oleh komputer, maka komputer harus melakukan proses pengenalan pola yang merupakan teknik yang penting yang ada didalam computer vision (Munir, 2004).

2.8. OpenCV (Open Computer Vision)

OpenCV adalah sebuah pustaka perangkat lunak yang ditujukan untuk pengolahan citra dinamis secara real time. Pustaka yang ada ditulis dalam bahasa C dan C++ dan dapat dijalankan pada Linux, Windows dan Mac OS X. *OpenCV* dirancang untuk komputasi yang efisien dan terfokus pada aplikasi real time. *OpenCV* telah ditulis secara optimal dalam bahasa C dan memiliki keuntungan pada proseseor multicore. Salah satu tujuan *OpenCV* adalah untuk menyediakan infrastruktur penggunaan *computer vision* yang sederhana. Pustaka *OpenCV* berisi lebih dari 500 fungsi yang meliputi banyak hal dalam visi, termasuk pemeriksaan produk pabrik, pencitraan medis, keamanan, dan robotika (Bradski dan Kaehler, 2008).

Fitur-fitur yang terdapat pada OpenCV antara lain:

1. Manipulasi data *image* (alokasi, rilis, duplikasi, pengaturan, konversi).
2. *Image* dan I/O video (masukan berbasis *file* dan kamera, keluaran *image/ video file*).
3. Manipulasi matriks dan vektor serta aljabar linear (produk, solusi, *eigenvalues*, SVD).

4. Beragam struktur data dinamis (daftar, baris, grafik).
5. Dasar pengolahan citra (filter, deteksi tepi, deteksi sudut, pengambilan sampel dan interpolasi, konversi warna, operasi morfologi, *histogram*).
6. Analisis struktur (komponen yang berhubungan, pengolahan kontur, transformasi jarak, variasi momen, transformasi Hough, perkiraan polygonal, menyesuaikan garis, Delaunay *triangulation*.
7. Kalibrasi kamera (menemukan dan menelusuri pola kalibrasi, kalibrasi, dasar estimasi matriks, estimasi homografi, korespondensi stereo).
8. Analisis gerakan (*optical flow*, segmentasi gerakan, penelusuran).
9. Pengenalan objek (metode eigen, HMM).
10. Dasar *Graphical User Interface* atau GUI (menampilkan *image*/ video, penanganan mouse dan keyboard, scroll-bars).
11. Pelabelan *image* (garis, poligon, gambar teks).

Modul-modul yang terdapat pada OpenCV antara lain:

1. *cv* – fungsi utama OpenCV.
2. *cvaux* – fungsi penolong OpenCV.
3. *cxcore* – pendukung struktur data dan aljabar linear .
4. *highgui* – fungsi GUI.

2.9. *Haar Cascade Classifier*

Haar Cascade Classifier merupakan metode yang digunakan dalam pendeteksian obyek. Metode ini juga dikenal dengan *Haar Cascade Classifier* yang diperkenalkan oleh Paul Viola dan Michael Jones untuk mendeteksi wajah. Metode ini mendeteksi objek dalam gambar dengan manggabungkan empat kunci utama yaitu *Haari**Feature*, *Integrali**Image*, *Adabossti**Learning* dani*Cascade Classifier* (P. Viola dan M. Jones, 2004).

2.9.1 *Haar-like Feature*

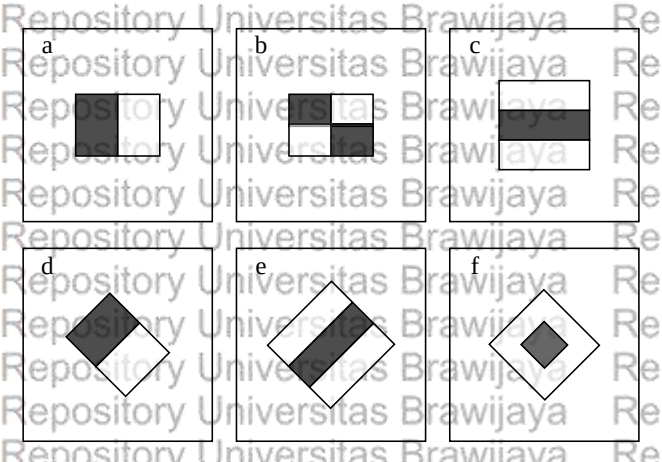
Haar-like feature merupakan *rectangular* (persegi) *features*, yang memberikan indikasi secara spesifik pada sebuah gambar atau image. Ide dari *Haar-like feature* adalah untuk mengenali objek berdasarkan nilai sederhana dari fitur. Metode ini memiliki kelebihan yaitu komputasinya sangat cepat, karena hanya bergantung pada jumlah piksel dalam persegi bukan setiap nilai piksel dari sebuah image. Metode ini merupakan metode yang menggunakan statistika model (*classifier*). Pendekatan untuk mendeteksi objek dalam

gambar menggabungkan empat konsep utama yaitu training data, fitur segi empat sederhana yang disebut fitur haar, integral image untuk pendeteksian fitur secara cepat dan pengklasifikasian bertingkat (*cascade classifier*) untuk menghubungkan banyak fitur secara efisien. *Haar-like feature* digunakan dalam mendeteksi objek pada citra digital. Namai*haar* merujukipada suatuifungsiimatematika (*Haari**Wavelet*) yangiberbentukikotak. Awalnya pengolahan gambar hanya dengan melihat dari nilai RGB setiap piksel, namun metode ini ternyata tidaklah efektif. Kemudian dikembangkan sehingga terbentuk *Haar-like feature* (P. Viola dan M. Jones, 2004).

Deteksi obyek kendaraan pada penelitian ini merupakan pemodifikasian sistem Haar-like features dari deteksi wajah yang pertama kali dilakukan oleh oleh Violaidan Jones kemudian dikembangkaniolehiLienhart (Lienhart, Rainer, & Maydt, 2002)

Menurut Putro dkk (2012), prosedur deteksi objek menggunakan *Haar Cascade Classifier* yaitu dengan mengklasifikasikan gambar berdasarkan pada nilai fitur sederhana. Terdapat banyak alasan untuk menggunakan fitur dari pada piksel secara langsung. Alasan yang paling umum adalah bahwa fitur dapat digunakan untuk mengkodekan informasi domain *ad-hoc* yang sulit dalam pembelajaran terhadap data latih yang terbatas jumlahnya. Alasan penting kedua untuk menggunakan fitur adalah sistem fitur berbasis operasi jauh lebih cepat daripada sistem berbasis pixel.

Dalam *Haar Cascade Classifier* terdapat beberapa macam variasi fitur *haar*, seperti yang ditunjukkan pada Gambar 2.9.



Gambar 2.9 Berbagai bentuk *Haar-like Feature*

Sumber: Jurnal Lienhart

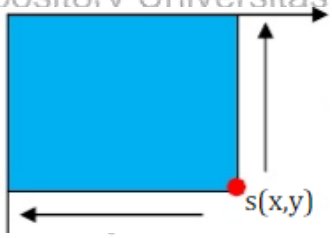
Pada Gambar 2.9 menjelaskan bahwa fitur a, b) Fitur *Haar* yang diusulkan Papageogiou dkk, (c) Fitur *Haar* yang diusulkan Viola dan Jones dan (d, e, f) Variasi fitur *Haar* yang

diusulkan Leinhart. Dalam penelitian ini, fitur haar yang digunakan yaitu fitur *Haar* yang diusulkan Leinhardt. Hal tersebut sesuai dengan objek penelitian yang dilakukan yaitu tentang objek mobil.

2.9.2 *Integral Image*

Integral image adalah reperesentasi citra baru yang digunakan untuk untuk menentukan ada atau tidaknya dari fitur haar pada sebuah gambar secara efisien. pada umumnya, pengintegrasian tersebut berarti menambahkan nilai piksel bersamaan. Nilai integral untuk masing-masing piksel adalah jumlah dari semua piksel-piksel dari atas sampai bawah. Dimulai dari kiri atas sampai kanan bawah, nilai dari citra dapat dijumlahkan dengan beberapa operasi bilangan bulat per piksel (Nugraha dan Muljono, 2015).

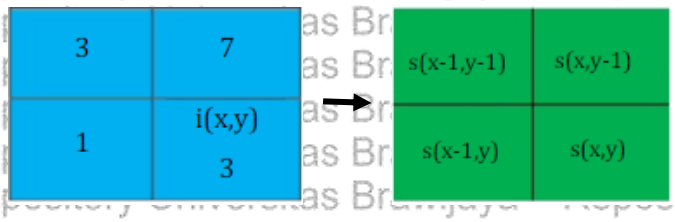
Integral Image dapat diilustrasikan pada Gambar 2.10 dan dihitung menggunakan persamaan 2-3:



Gambar 2.10 Ilustrasi piksel

$$s(x,y) = i(x,y) + s(x,y-1) + s(x-1,y1) - s(x-1,y-1) \quad \text{.....(2-3)}$$

s(x,y) merupakan jumlah nilai pada baris lokasi (x,y), i(x, y) merupakan masukkan nilai citra asli pada lokasi (x,y), s(x,y-1) merupakan jumlah nilai area dari nilai piksel tetangga y, dan s(x-1,y) merupakan jumlah nilai area dari nilai piksel tetangga x, serta s(x-1,y-1) merupakan jumlah nilai area dari nilai piksel tetangga diagonalnya. Hal tersebut dapat dicontohkan pada Gambar 2.11.



Gambar 2.11 Perhitungan *integral image*

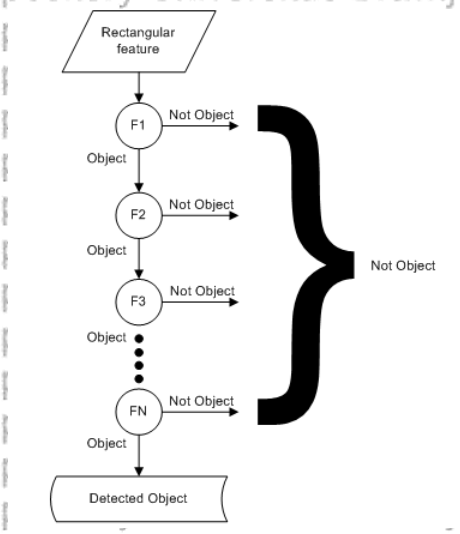
2.9.3 *Adaboost Learning*

Setelah mendapatkan nilai dari *Integral Image* maka akan dipilih fitur-fitur yang lebih spesifik untuk di gunakan. Untuk pemilihan fitur menggunakan metode *machine learning* yang disebut *Adaboost*. *Adaboost* akan menentukan fitur-fitur yang spesifik yang akan digunakan untuk mengatur nilai ambang (*threshold*), Dalam proses pembentukan *classifier* kuat dilakukan penambahan bobot terhadap *classifier* lemah dan digabungkan untuk mengevaluasi bagian citra mengandung objek atau tidak. Adapun yang maksud *classifier* lemah adalah suatu hasil prediksi dengan tingkat kebenaran yang kurang akurat (Howse, 2013). Berikut adalah tahapan yang dilakukan *adaboost* dalam membentuk *classifier* kuat:

1. Normalisasikan bobot untuk mendapatkan distribusi probabilitas atau kandidat *classifier* lemah.
2. Evaluasi setiap kandidat *classifier* lemah tersebut.
3. Pilih kandidat *classifier* lemah dengan kesalahan yang paling sedikit, tetapkan sebagai *classifier* lemah.
4. Klasifikasi semua data training menggunakan *classifier* lemah yang telah didapatkan, dan lakukan pemberian bobot ulang terhadap data-data tersebut. Perbesar bobot semua data yang mengalami kesalahan klasifikasi, dan kurangi bobot (kembalikan ke bobot awal) semua data yang telah diklasifikasi dengan benar. Hal ini ditujukan agar setiap kesalahan klasifikasi yang terjadi dapat terlihat dan diatasi oleh *classifier* lemah yang terpilih pada tahapan selanjutnya.
5. *Classifier* akhir yang didapatkan merupakan gabungan dari semua *classifier* lemah yang didapatkan dari setiap tahapan boosting.

2.9.4 *Cascade Classifier*

Cascade classifier adalah metode klasifikasi bertingkat dengan menggunakan fitur yang diseleksi dengan menggunakan algoritma *Adaboost*. *Cascade classifier* memiliki beberapa tingkatan dalam melakukan klasifikasi, pada setiap tingkatan dipisahkan antara subcitra yang mengandung gambar positif (gambar yang memiliki objek yang diinginkan) dengan gambar negatif (gambar yang tidak memiliki objek yang diinginkan), dimana bagian subcitra yang tidak mengandung objek positif akan ditolak sedangkan subcitra yang mengandung objek akan dijadikan input pada tingkatan klasifikasi berikutnya dengan kriteria penyaringan yang lebih spesifik hingga didapatkan subcitra yang merupakan objek yang terdeteksi (P. Viola dan M. Jones, 2004).



Gambar 2.12 Skema Pendeteksian Objek Bertingkat

Sumber: www.jati.stta.ac.id

Pada Gambar 2.12 dijelaskan skema untuk mendapatkan bagian citra yang dikehendaki sebagai objek deteksi, mulanya pada tahapan F1 yang merupakan tahapan awal melakukan penyaringan awal terhadap bagian dari subcitra dengan tujuan memisahkan antara sub-citra yang mengandung objek atau tidak, untuk sub citra yang tidak mengandung objek pada setiap tahapan penyaringan akan dievaluasi atau ditolak, sedangkan pada sub-citra yang mengandung objek akan dijadikan inputan tahap selanjutnya dengan tingkat penyaringan yang lebih spesifik seiring banyaknya tahapan yang dilalui hingga didapatkan hasil yang berupa sub-citra yang diyakini sebagai objek deteksi yang dikehendaki.

BAB III
KERANGKA KONSEP PENELITIAN

3.1 Kerangka Konsep Penelitian

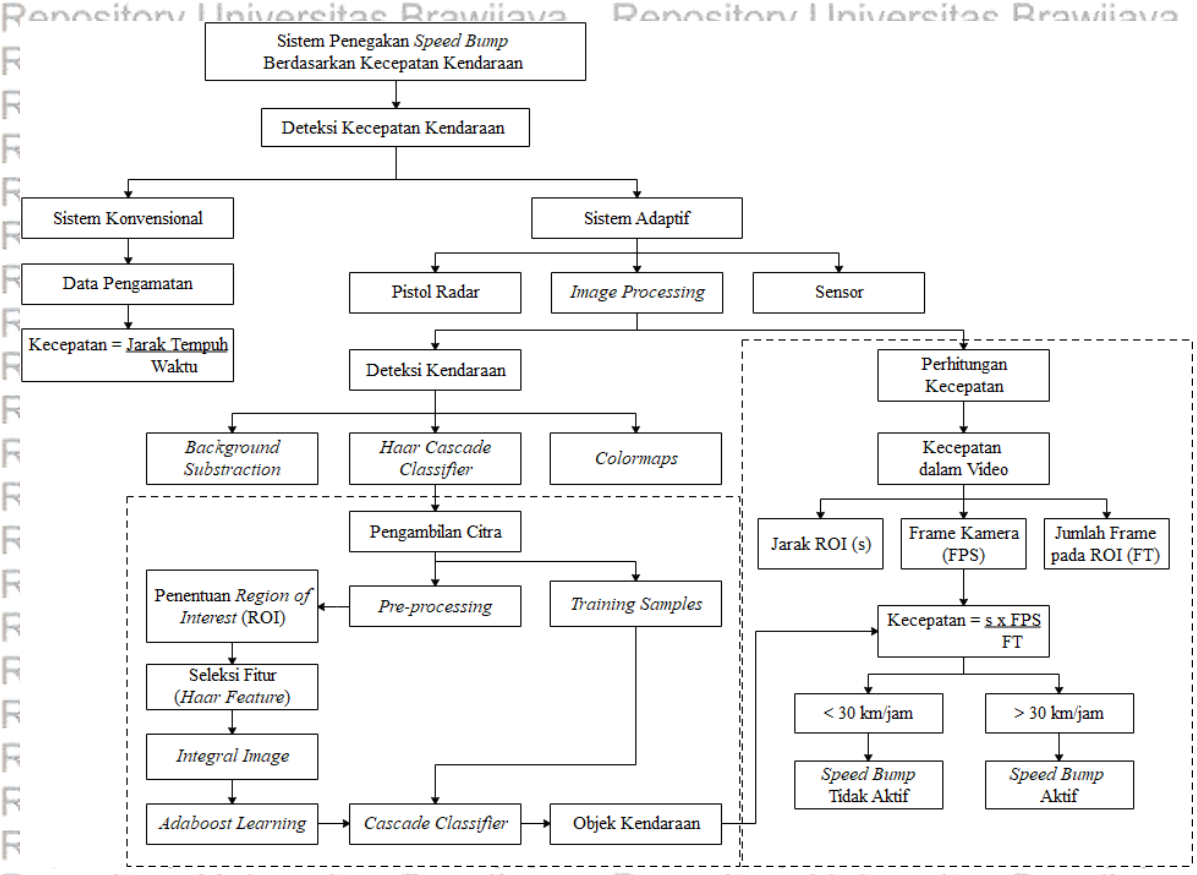
Saat ini untuk mengetahui kecepatan kendaraan, masih banyak menggunakan sistem konvensional, dimana parameter yang harus diketahui yaitu jarak tempuh dibagi dengan waktu tempuh kendaraan. Sistem tersebut kemudian dikembangkan menjadi sistem adaptif, dengan beberapa metode yang dapat digunakan yaitu pistol radar, *image processing*, dan sensor elektronik. Penelitian ini menggunakan *image processing* dengan memanfaatkan kamera dalam pengambilan citra, sehingga dapat menyelesaikan permasalahan dalam deteksi dan menghitung kecepatan kendaraan.

Dalam menghitung kecepatan kendaraan, proses deteksi kendaraan perlu dilakukan. Ada beberapa metode yang dapat digunakan dalam deteksi objek yaitu *Background Substraction, Haar Cascade Classifier*, dan sebagainya. Penelitian ini menggunakan metode *Haar Cascade Classifier* yang merupakan metode *learning* efektif dalam pendeteksian objek (Viola-Jones, 2004). Metode deteksi ini menggunakan beberapa tahapan untuk membuat keputusan. Tahap pertama untuk *training samples* dari citra yang sudah diambil dan terdiri dari citra positif (kendaraan) dan citra negatif (*background*) yang menghasilkan sebuah database citra dan digunakan pada *source* pendeteksian objek. Tahap kedua yaitu *pre-processing* (*rescale* dan *grayscale*). *Rescale* yaitu mengubah resolusi video yang didapatkan menjadi resolusi 320x240 piksel, dengan tujuan untuk mengurangi penggunaan memori yang digunakan dan komputasi sistem dapat bekerja lebih cepat. Untuk mengurangi resiko kesalahan dan proses deteksi menjadi lebih akurat, diperlukan membatasi area deteksi dengan menentukan *Region of Interest* (*ROI*) pada video.

Tahap selanjutnya proses deteksi kendaraan menggunakan *Haar Cascade Classifier*, yang terdiri dari beberapa tahap untuk mendapatkan hasil deteksi kendaraan. Tahap-tahap tersebut yaitu *Haar-Like Feature, Integral image, Adaboost (Adaptive Boosting),* dan *Cascade Classifier*. *Haar-Like Feature* yaitu memilih fitur *Haar* atau mendeteksi objek pada citra yang memproses citra dalam kotak-kotak. Dalam satu kotak terdapat beberapa pixel yang di proses dan didapatkan perbedaan nilai (*threshold*) yang menandakan daerah hitam

dan putih. Dalam citra (video), perhitungan dan penjumlahan pixel terjadi secara terus-menerus dan membutuhkan waktu yang lama, sehingga penjumlahan diganti menggunakan *integral image* sehingga didapatkan hasil lebih cepat. Hasil deteksi dari *Haar-Like Feature* kurang akurat jika hanya menggunakan satu fungsi, sehingga tahap ketiga yaitu *Adaboost* berfungsi untuk melakukan pemilihan *Haar-Like Feature* dalam jumlah banyak dengan hanya memilih *Haar-Like Feature* tertentu. Tahap terakhir yaitu penggabungan *cascade classifier* supaya kecepatan proses pendeteksian dapat meningkat, dengan memusatkan perhatian pada daerah yang berpeluang dalam *image*. Setelah semua tahap dilakukan, maka didapatkan sebuah hasil pendeteksian yang bisa berupa mobil atau objek lain.

Objek yang terdeteksi sebagai kendaraan akan dihitung kecepatannya, dengan mengalikan panjang daerah yang ditandai (*ROI*) dengan *frame rate* dari video dan hasilnya dibagi dengan jumlah *frame* yang dihitung sejak kendaraan masuk sampai keluar *ROI*. Hasil perhitungan kecepatan tersebut akan menjadi parameter untuk menentukan *speed bump* aktif atau tidak, dimana kendaraan yang terdeteksi melaju dengan kecepatan diatas batas maksimal yang ditentukan (30 km/jam) akan mengakibatkan *speed bump* aktif. Seperti pada Gambar 3.1 menunjukkan kerangka konsep pada penelitian ini.



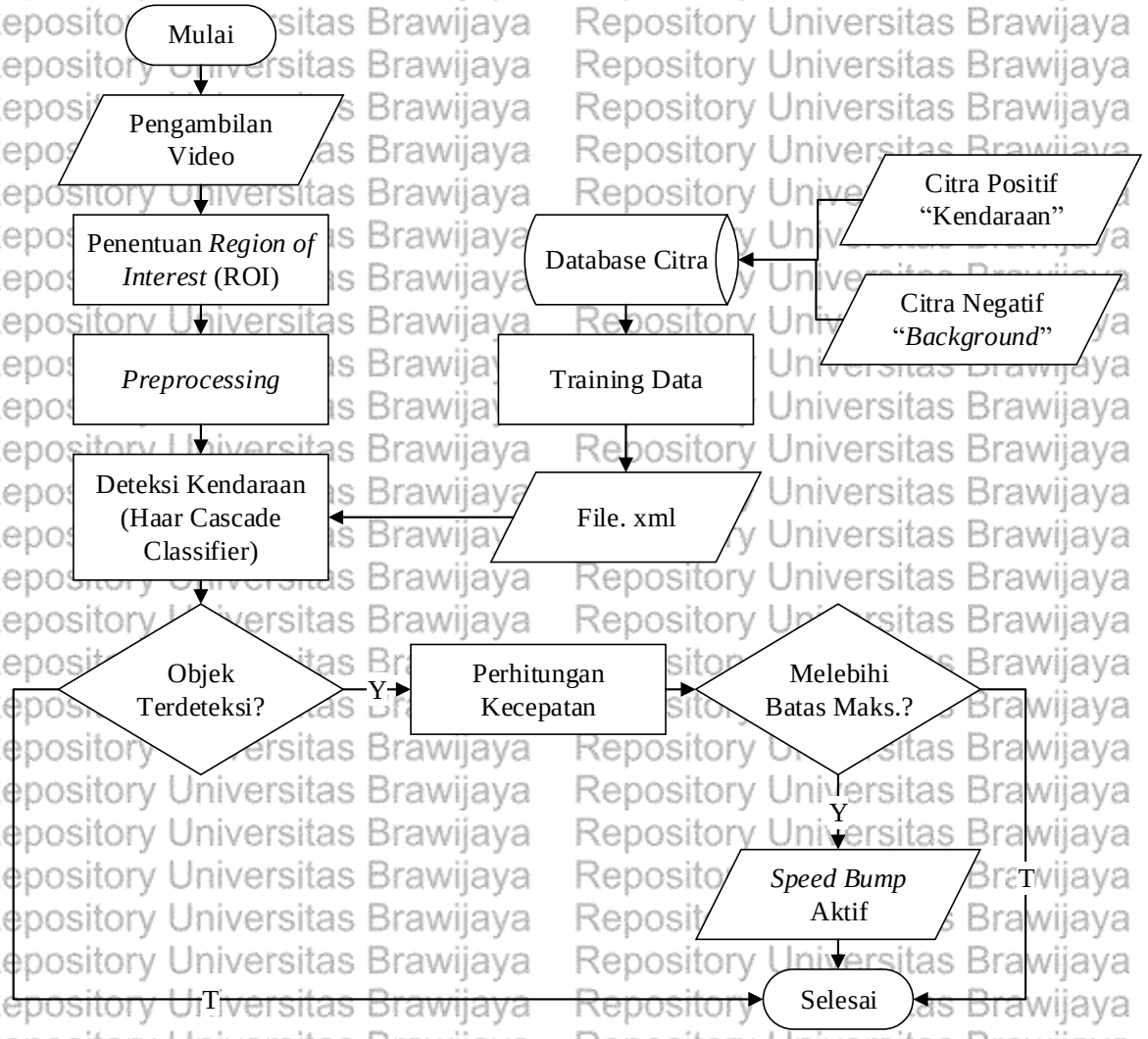
Gambar 3.1 Kerangka konsep penelitian

3.2. Analisis Masalah

Kecelakaan lalu lintas merupakan sebuah peristiwa yang terjadi tanpa disengaja dan melibatkan antar kendaraan atau dengan objek lainnya. Terdapat banyak faktor yang menyebabkan kecelakaan terjadi, salah satunya adalah faktor manusia dalam hal ini pengemudi itu sendiri. Salah satu kesalahan yang sering terjadi adalah pengemudi berkendara dengan melebihi batas kecepatan yang telah ditetapkan. Di beberapa ruas jalan raya atau jalan kecil (gang) banyak ditemukan fasilitas *speed bump* atau polisi tidur yang berfungsi memperingatkan pengendara agar memperlambat laju kendaraan dan memperhatikan daerah sekitarnya. Namun pemasangan *speed bump* dibeberapa tempat menjadi permasalahan tersendiri, seperti yang terpasang di lingkungan sekolah atau jalan tertentu yang dapat memberikan ketidaknyamanan bagi pengemudi kendaraan yang melaju dengan kecepatan rendah. Sehingga diperlukan mekanisme secara selektif dalam menegakkan *speed bump* sesuai dengan kecepatan kendaraan. Batas kecepatan yang bisa diberikan pada jalan untuk kawasan permukiman yaitu paling tinggi 30 km/jam sehingga kecepatan tersebut menjadi batas maksimal untuk mengaktifkan *speed bump*.

3.3. Konsep Solusi

Data yang dibutuhkan dalam penelitian ini yaitu citra dari kendaraan, citra selain kendaraan atau *background* dan citra (video) yang diambil dari samping jalan dan menghadap kebawah. Pada Gambar 3.2 langkah pertama ialah membuat database citra dari citra positif (kendaraan) dan citra negatif (*background*). Database tersebut di *training* sehingga mendapatkan database dalam bentuk file .xml, yang digunakan dalam pencocokan pada proses deteksi kendaraan (*haar cascade classifier*). Selanjutnya proses pengambilan video menggunakan kamera dan hasilnya di proses untuk mendapatkan hasil deteksi dan hasil perhitungan kecepatan dari kendaraan yang terdeteksi. Hasil perhitungan akan menjadi parameter untuk menggerakkan motor servo yang digunakan sebagai *speed bump*. Berikut Gambar 3.2 menjelaskan kerangka solusi pada penelitian ini.



Gambar 3.2 Kerangka solusi

3.4. Hipotesis

Berdasarkan perancangan skema penegakan *speed bump* yang dilakukan, maka metode *Haar Cascade Classifier* digunakan dalam deteksi kendaraan, karena memiliki kelebihan dalam komputasinya sangat cepat, dan hanya bergantung pada jumlah piksel dalam persegi bukan setiap nilai piksel dari sebuah citra. Sistem yang dibuat dapat diterapkan secara *real time* dalam mendeteksi kecepatan kendaraan yang berkecepatan tinggi pada *Region of Interest* (*ROI*) yang telah ditentukan. Efektivitas kerja sistem juga bekerja dengan baik dengan kondisi lalu lintas pada siang hari atau intensitas cahaya tinggi.

BAB IV
METODE PENELITIAN

4.1 Alat dan Bahan

Pada penelitian ini digunakan beberapa peralatan untuk merancang program, mengevaluasi, dan mengkomparasi kinerja sistem deteksi dan perhitungan kecepatan kendaraan:

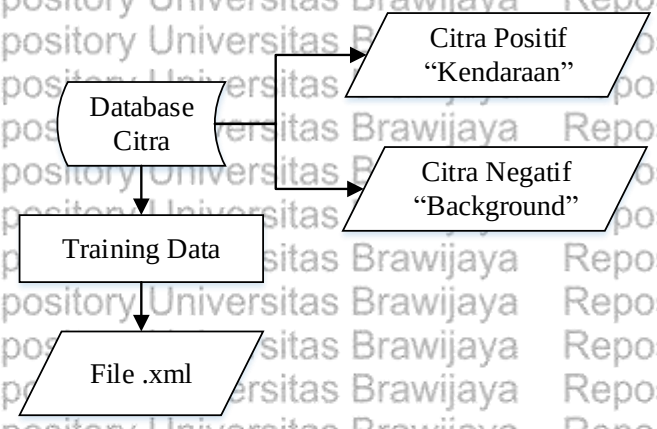
1. Processor : Intel(R) Core (TM) i3-4030U 1.90GHz
2. RAM : 6,00 GB DDR3
3. Sistem Operasi : Windows 10 Pro
4. Interpreter : Python 3.5.6
5. Library : OpenCV 3.1.0

4.2 Perancangan Sistem

Perancangan sistem penegakan *speed bump* pada penelitian ini terdiri dari beberapa tahap. Disetiap hasil dari tahap tersebut menghasilkan *output* yang menjadi parameter dari tahap selanjutnya.

4.2.1. Perancangan Database Citra

Pada tahap ini dijelaskan mengenai perancangan *database* citra dalam bentuk file .xml yang digunakan untuk mendeteksi objek kendaraan pada video.

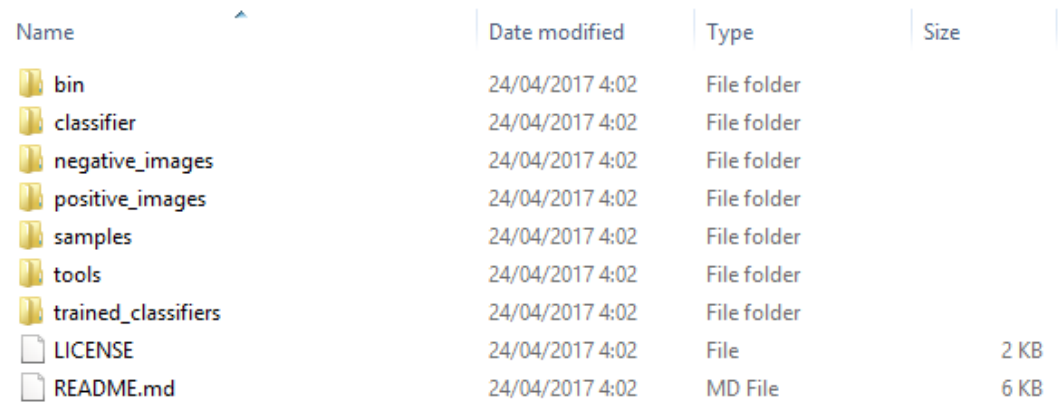


Gambar 4.1 Skema Perancangan *Classifier*

Dari Gambar 4.1, perancangan *classifier* memiliki beberapa tahap, diantaranya :

1. Persiapan Direktori

Tahap pertama adalah mempersiapkan direktori dengan susunan sebagai berikut. Susunan direktori ini menggunakan contoh dari repository milik github Thorsten Ball (2017) (https://github.com/mrnugget/opencv-haar-classifier-training).



Gambar 4.2 Susunan Direktori

2. Persiapan Citra Positif dan Negatif

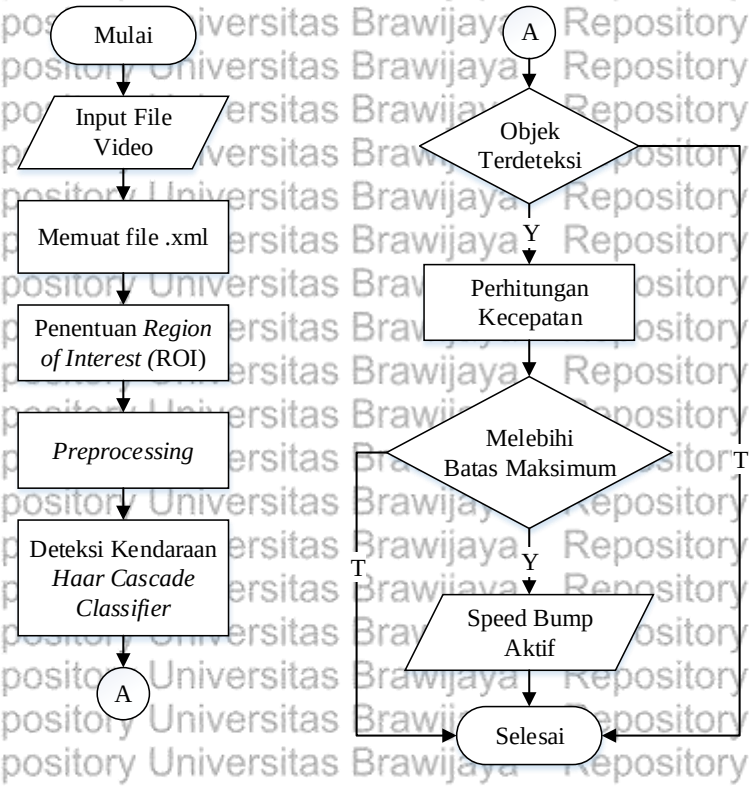
Tahap selanjutnya adalah menyiapkan citra positif (kendaraan) dan citra negatif (background). Tahap ini juga menghasilkan sampel positif dan negatif menggunakan tools OpenCV (opencv_createsamples) dari citra yang dimasukkan. Tools ini melakukan transformasi dan distorsi pada gambar. Proses ini menghasilkan sebuah file .vec yang nantinya digunakan untuk proses training classifier.

3. *Training Classifier*

Tahap berikutnya adalah melakukan training classifier. Proses ini membutuhkan waktu yang cukup lama, tergantung spesifikasi perangkat yang digunakan. Saat proses selesai, akan dihasilkan sebuah file .xml di direktori *classifier*. File tersebut nantinya akan digunakan dan dipanggil pada source pendeteksian objek.

4.2.2. Perancangan Pendeteksian Objek

Dalam melakukan implementasi dari *classifier* yang akan dibuat dan diujikan kedalam sebuah sistem yang ditambahkan beberapa fitur yang meliputi deteksi dan perhitungan kecepatan kendaraan.

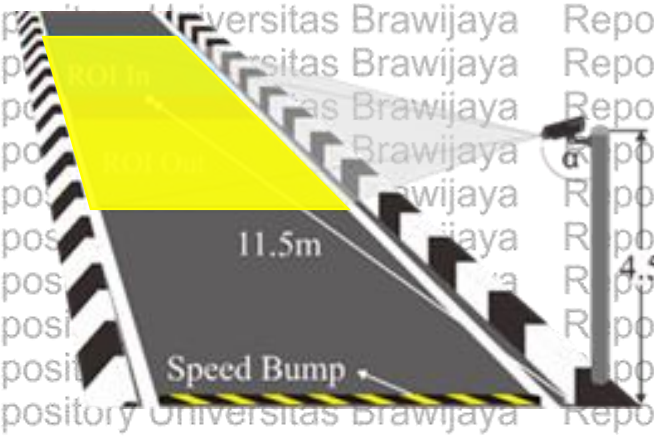


Gambar 4.3 Alur Proses Sistem

Untuk penjelasan secara rinci mengenai proses dari alur proses sistem pada Gambar 4.3, akan dijelaskan sebagai berikut :

1. Akuisisi Citra

Proses awal dari proses utama penelitian ini adalah proses memasukkan citra dalam program. Data citra yang digunakan adalah video jalan raya pada satu jalur yang diambil menggunakan kamera dengan posisi kamera dari samping jalan dan menghadap kebawah, seperti pada Gambar 4.4.



Gambar 4.4 Posisi pengambilan video

Pada Gambar 4.4, tata letak kamera dimiringkan lurus dengan titik tengah *ROI* untuk mendapatkan citra yang mencakup bagian *ROI* yang sudah ditentukan. Sudut kemiringan kamera dalam pengambilan video ditentukan menggunakan fungsi untuk menghitung invers tangen ($\tan^{-1}$) yang merupakan sebuah fungsi trigonometri, ditunjukkan pada persamaan 4-1.

$$\alpha = tan^{-1}(L/H) \quad \text{.....(4-1)}$$

dimana,

L = jarak dari tiang kame ra dengan titik tengah dari *ROI*

H = tinggi tiang penempatan kamera

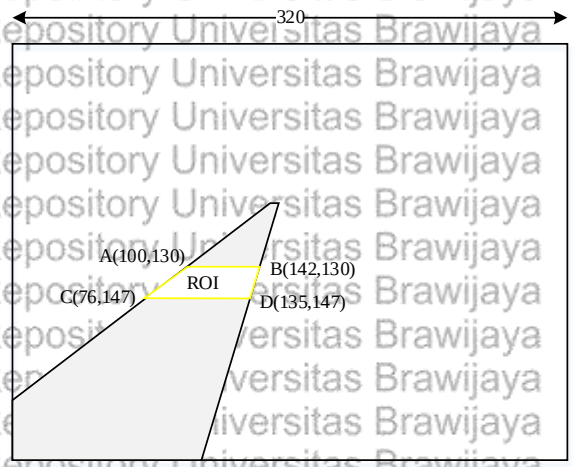
Sehingga didapatkan,

$\alpha = tan^{-1}(8{,}25m/5m)$

$\alpha = tan^{-1}0{,}8 = 50{,}53°$

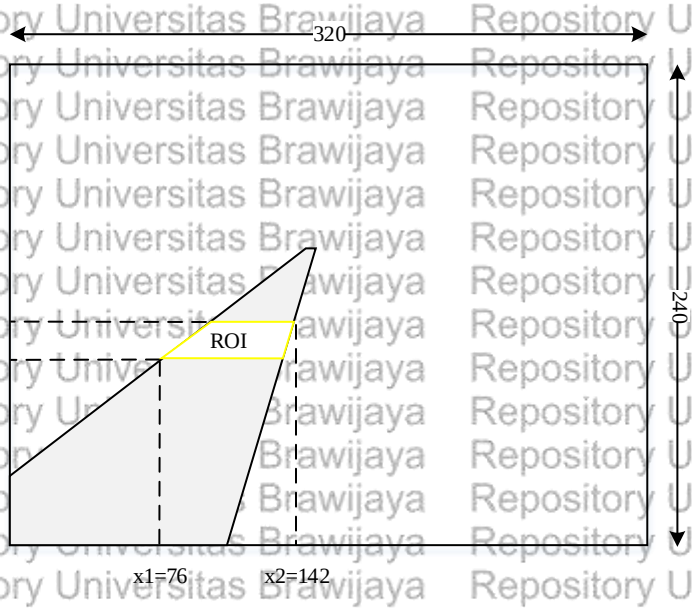
Resolusi video yang didapatkan dan dimasukkan dalam program akan diatur kembali menjadi 320x240 piksel, sehingga lebih proporsional dan proses komputasi dapat berjalan lebih cepat. Setelah itu, video dibaca secara *frame by frame*.

Penelitian ini dilakukan dengan menggunakan satu jalur dan lajur dari jalan, dengan tujuan dalam penegakan *speed bump* hanya untuk satu mobil pada satu kali penegakan. *ROI* ditentukan dengan membatasi area deteksi, sehingga mengurangi resiko kesalahan deteksi dan proses deteksi menjadi lebih fokus dan akurat. Penentuan *ROI* dilakukan dengan menentukan titik-titik koordinat (x,y), sehingga didapatkan *ROI* dalam bentuk persegi Panjang. Penentuan *ROI* ditunjukkan pada Gambar 4.5.



Gambar 4.5 Tampilan pengambilan citra dan penentuan titik koordinat *ROI*

Untuk menentukan resolusi dari *ROI* (x,y), ditentukan terlebih dahulu koordinat (x,y), seperti pada Gambar 4.6.



Gambar 4.6 Penentuan koordinat (x,y) *ROI*

Setelah penentuan koordinat (x,y) *ROI* , resolus ditentukan dengan persamaan 4-2.

$$ROI\ (x,y) = ((x2 - x1), (y2 - y1)) \quad \text{.....(4-2)}$$

dan didapatkan,

$$ROI\ (x,y) = \big((142 - 76), (147 - 130)\big)$$

$$ROI\ (x,y) = (66,17)$$

Sehingga didapatkan resolusi dari *ROI* yaitu 66x17 pixel.

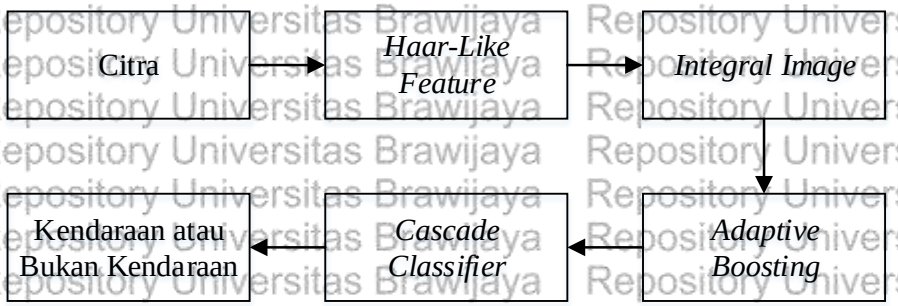
2. Preprocessing

Citra video selanjutnya akan dikonversi dari warna asal *Red Green Blue* (RGB) menjadi *grayscale* menggunakan fungsi *cv2.cvtcolor* yang telah disediakan pada *OpenCV library*.

Pada proses *preprocessing* ini akan menghasilkan *output* berupa *Region of Interest subcitra grayscale*, yang merupakan hasil dari konversi *grayscale* pada daerah *ROI*.

3. Processing

Pada proses pendeteksian kendaraan dengan menggunakan *Haar Cascade Classifier*, terdapat beberapa proses yang dilakukan sebelum akhirnya menghasilkan sebuah *output* mobil yang terdeteksi pada sebuah citra. Proses-proses tersebut yaitu Haar-*Like Featrure, Integral image, Adaboost (Adaptive Boosting),* dan *Cascade Classifier*. Skema proses dari tiap-tiap tahap yang dilalui oleh sebuah citra untuk memperoleh hasil pendeteksian citra mobil dapat dilihat pada Gambar 4.7.



Gambar 4.7 Skema deteksi *Haar Cascade Classifier*

Untuk detail dari tiap tahap yang dilalui oleh sebuah citra pada saat proses pendeteksian kendaraan seperti pada Gambar 4.7 adalah sebagai berikut :

1. Pemilihan fitur

a. Haar-like feature

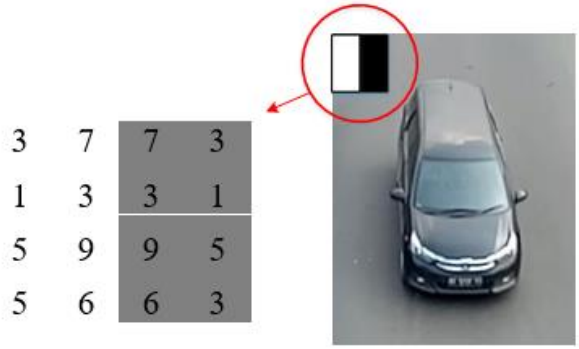
Setelah *citra image* dirubah menjadi *citra grayscale*, proses selanjutnya yaitu memilih fitur *Haar* yang ada pada *image* tersebut yang dalam menggunakan *Haar Cascade Classifier* disebut dengan *Haar-Like feature*. Teknik yang dilakukan yaitu dengan cara mengkotak-kotakkan setiap daerah pada *image* dari mulai ujung kiri atas sampai kanan bawah. Proses ini dilakukan untuk mencari apakah ada fitur kendaraan pada area tersebut. Dalam *Haar Cascade Classifier*, ada beberapa jenis fitur yang bisa digunakan seperti *Edge-feature, Line feature,* dan *Four-rectangle feature*. Pada setiap kotak-kotak fitur tersebut terdiri dari beberapa pixel dan akan dihitung selisih antara nilai pixel pada kotak putih dengan nilai pixel pada kotak hitam. Apabila nilai selisih antara daerah putih dengan daerah hitam di atas nilai ambang (*threshold*), maka daerah tersebut dinyatakan memiliki fitur.

Pada citra kendaraan yang akan dideteksi, kondisinya menghadap ke depan. Untuk mempermudah dan mempercepat proses perhitungan nilai *Haar* pada sebuah *image, Haar Cascade Classifier* menggunakan sebuah perhitungan yang disebut dengan *Integral Image*.

b. Integral Image

Integral image adalah reperesentasi citra baru yang digunakan untuk menentukan keberadaan fitur *haar* pada sebuah gambar secara efisien. Dengan menggunakan *integral image* proses perhitungan bisa dilakukan hanya dengan satu kali scan dan memakan waktu yang cepat dan akurat. *Integral image* digunakan untuk menghitung hasil penjumlahan nilai pixel pada daerah yang dideteksi oleh fitur haar.

Nilai-nilai pixel yang akan dihitung merupakan nilai-nilai pixel dari sebuah citra masukan yang dilalui oleh fitur *haar* pada saat pencarian fitur kendaraan. Pada setiap jenis fitur yang digunakan, pada setiap kotak-kotaknya terdiri dari beberapa pixel. Apabila ada sebuah citra masukan yang dilalui oleh fitur *haar* dapat dilihat pada Gambar 4.8.



Citra Masukan

Gambar 4.8 Nilai Pixel-Pixel Pada Sebuah Fitur

2. Klasifikasi bertingkat

a. Adaboost (Adaptive Boosting)

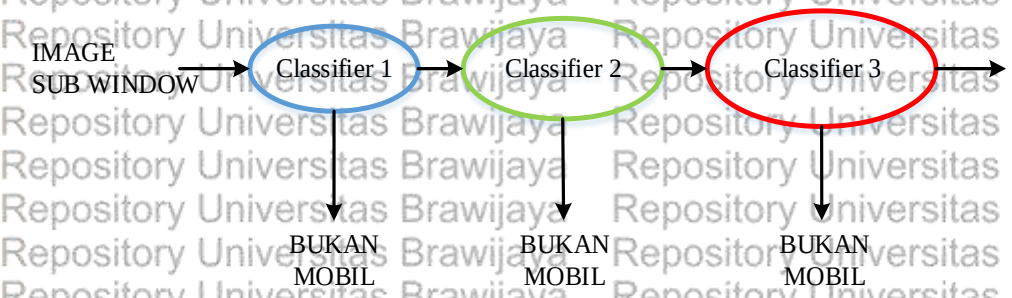
Adaptive boosting merupakan teknik yang digunakan untuk mengkombinasikan banyak *classifier* lemah untuk membentuk suatu gabungan *classifier* yang lebih baik. Proses dari *adaptive boosting* akan menghasilkan sebuah *classifier* yang kuat dari *classifier* dasar. Satuan dari *classifier* dasar tersebut disebut dengan *weak learner*. Setelah sebelumnya dilakukan pemilihan fitur *Haar*, pada proses selanjutnya dalam deteksi kendaraan menggunakan *Haar Cascade Classifier*, dengan menggunakan algoritma *adaboost*, fitur pada sebuah *image* akan dideteksi kembali. Tujuannya untuk mengetahui apakah ada fitur kendaraan pada daerah dengan klasifikasi fitur yang lemah. Pada *classifier* lemah akan dilakukan perhitungan dan dibandingan dengan *classifier* lainnya secara acak. Selanjutnya dilakukan kombinasi atau penggabungan pada *classifier* lemah untuk membentuk suatu kombinasi yang *linier*.

b. Cascade classifier

Cascade classifier melakukan proses dari banyak fitur fitur dengan mengorganisir dengan bentuk klasifikasi bertingkat. Terdapat tiga buah klasifikasi untuk menentukan apakah benar atau tidak ada fitur kendaraan pada fitur yang sudah dipilih.

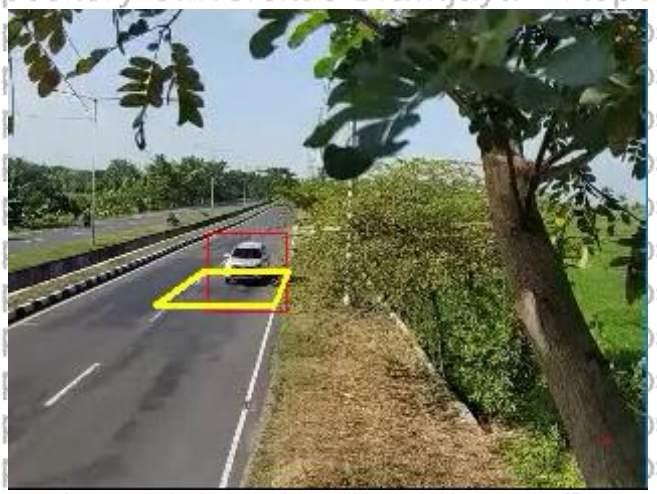
Pada klasifikasi *filter* pertama, tiap *subcitra* akan diklasifikasi menggunakan satu fitur. Jika hasil nilai fitur dari filter tidak memenuhi kriteria yang diinginkan, hasil tersebut akan ditolak.

Algoritma kemudian bergerak ke *sub window* selanjutnya dan menghitung nilai fitur kembali. Jika didapat hasil sesuai dengan *threshold* yang diinginkan, maka dilanjutkan ke tahap filter selanjutnya. Hingga jumlah *sub window* yang lolos klasifikasi akan berkurang hingga mendekati image yang dideteksi. Pada Gambar 4.9 di bawah ini merupakan proses rangkaian *filter* yang dilalui oleh setiap *classifier*.



Gambar 4.9 Alur *Cascade Classifier*

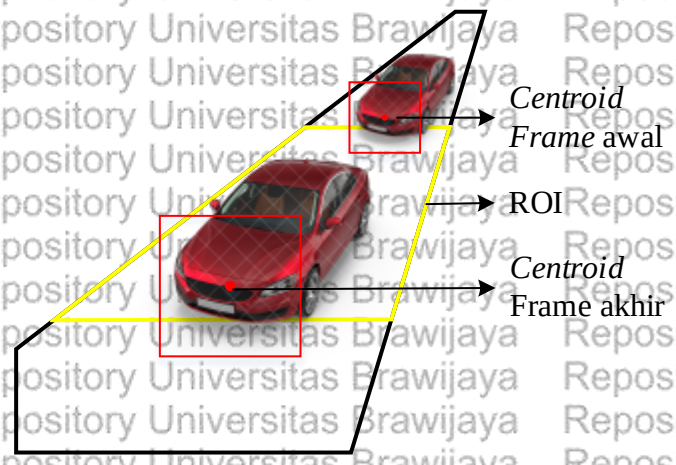
Setelah dilakukan serangkaian proses seperti pemilihan fitur dan klasifikasi bertingkat maka akan didapatkan sebuah hasil pendeteksian. Hasil pendeteksian bisa berupa kendaraan atau bukan kendaraan. Pada saat proses klasifikasi bertingkat dilakukan maka, pada *image* tersebut akan ditandai dengan sebuah *bounding box* atau *rectangle* pada daerah kendaraan yang terdeteksi dan apabila tidak ada kendaraan tedeteksi, maka *image* tersebut tidak akan ditandai oleh sebuah rectangle. Pada Gambar 4.10 di bawah ini merupakan contoh hasil pendeteksian dari proses akhir deteksi kendaraan menggunakan *Haar Cascade Classifier*.



Gambar 4.10 Hasil pendeteksian

4.4.3. Perancangan Perhitungan Kecepatan

Langkah selanjutnya adalah melakukan perhitungan kecepatan mobil yang terdeteksi saat memasuki *ROI*, dengan menggunakan posisi mobil di setiap *frame*. Setiap mobil yang terdeteksi diberikan *bounding box* dan *centroid* (titik tengah). *Centroid* berfungsi sebagai titik referensi untuk lokasi suatu objek dan membantu dalam mengetahui jarak mobil yang bergerak dalam *frame* secara berturut-turut, sehingga jumlah *frame* dari setiap pergerakan yang direkam akan diketahui, dan memungkinkan perhitungan kecepatan. *Centroid* yang diperoleh pada *frame* awal *ROI* dan *frame* akhir *ROI* digunakan untuk mencari nilai perpindahan jarak antar *frame*, dengan cara mencari selisih resultan antara koordinat *frame* akhir dengan koordinat *frame* awal, seperti pada Gambar 4.11.



Gambar 4.11 Proses perhitungan jumlah *frame*

Adapun untuk menghitung Setiap mobil diberikan Kecepatan objek dihitung dengan cara membagi panjang daerah yang ditandai, dengan waktu kendaraan. Dalam hal ini daerah yang ditandai atau *Region of Interest* (*ROI*) diukur panjangnya pada keadaan yang sesungguhnya. Pada Gambar 4.12, ditunjukkan tampilan penempatan kamera pada ketinggian 4.5 meter dari permukaan dan bidang pandang deteksi (*ROI*) warna abu-abu.

Gambar 4.12 Tampilan dari kamera pada daerah *ROI*

Untuk menghitung kecepatan kendaraan, persamaan yang digunakan adalah:

$$V_m = \frac{s}{t} \quad \text{(4-3)}$$

dengan,

Vm = kecepatan kendaraan (meter/detik)

s = panjang daerah yang ditempuh (meter)

t = waktu tempuh kendaraan (detik)

Untuk menghitung kecepatan dalam video, waktu tempuh kendaraan dapat dikembangkan menjadi :

$$t = \frac{ft}{fps} \quad \text{(4-4)}$$

dengan,

ft = jumlah *frame* yang dihitung sejak kendaraan masuk sampai keluar *ROI*

fps = jumlah *frame* yang dihasilkan oleh kamera dalam satu detik

Sehingga dalam perhitungan kecepatan pada sistem dilakukan dengan langkah saat menghitung jumlah *frame* yang dihasilkan dari sejak mobil terdeteksi pada *frame* awal *ROI* sampai pada mobil terdeteksi pada *frame* akhir *ROI*. Untuk mencari jumlah *frame* kendaraan, dapat dihitung dengan menggunakan persamaan 4-4. Seperti yang sudah diketahui, dimana panjang *ROI* (s) = 8 meter dan *frame* rate kamera = 30 fps. Sehingga untuk mencari ft, dapat dimisalkan jika diketahui dalam sistem mendeteksi kendaraan masuk sampai keluar *ROI* dengan waktu (t) = 1.5s, maka:

$$ft = t\ x\ fps \quad \text{..... (4-4)}$$

$$ft = 1.5s\ x\ 30fps$$

$$ft = 70\ frame$$

Selanjutnya dengan mensubtitusi persamaan 4-3 ke persamaan 4-4 maka akan diperoleh algoritma untuk menghitung kecepatan kendaraan pada video dan sistem yaitu :

$$V = \frac{s\ .\ fps}{ft} \quad \text{..... (4-5)}$$

4.4 Pengujian (*Testing*)

Setelah tahapan perancangan sistem berhasil dilakukan maka tahapan selanjutnya yaitu pengujian sistem. Pada tahapan ini dilakukan pengujian terhadap fungsionalitas sistem yang dibangun. Pengujian dan analisa akan dilakukan dengan mengetahui kesesuaian dengan perencanaan. Tahap ini juga dilakukan untuk menganalisa jalannya sistem, mengetahui masalah yang mungkin terjadi dan melakukan perbaikan jika terdapat kesalahan (*error*).

Pengujian kemampuan sistem dimulai dengan menghitung akurasi hasil pendeteksian kendaraan yang didapatkan dari perbandingan citra kendaraan yang terdeteksi dengan benar, dengan kesulurah citra yang digunakan pada data. Akurasi data dihitung pada proses pengujian, lalu dibandingkan hasilnya.

$$Akurasi = \frac{jumlah\ data\ benar}{keseluruhan\ data} \text{x } 100\% \quad \text{..... (4-6)}$$

Nilai error pada hasil pengenalan didapat melalui selisih dari keseluruhan data dan hasil deteksi kendaraan yang dikenali dengan baik. Nilai error meliputi jumlah kendaraan yang dikenali dengan hasil yang salah, dan tidak dapat dikenali.

Pengujian lainnya dalam menghitung kecepatan dilakukan dengan membandingkan hasil perhitungan kecepatan pada sistem, dengan hasil perhitungan kecepatan dari video yang didapatkan atau hasil *capture*. Waktu tempuh perhitungan kecepatan pada video hasil *capture* mengacu pada jarak (s) *ROI* yang telah ditentukan dibagi dengan waktu tempuh kendaraan untuk melintasi *ROI* (t), yang mengacu pada persamaan 4-3. Sedangkan hasil kecepatan pada sistem mengacu pada persamaan 4-5, dimana kecepatan diperoleh dengan hasil perkalian jarak (s) dengan jumlah *frame* dari video (fps) dibagi dengan jumlah *frame* yang dihitung sejak kendaraan masuk sampai keluar *ROI* (ft).

Untuk mengukur kinerja atau efektivitas dalam perhitungan kecepatan kendaraan, digunakan perhitungan tingkat kesalahan rata-rata atau *Mean Square Error* (MSE). MSE digunakan untuk mengetahui perbedaan antara estimator dengan hasil estimasi. MSE digunakan sebagai tolok ukur suatu estimator. Hasil yang diperoleh selalu berupa angka positif. Semakin mendekati nol maka semakin baik kinerja estimator tersebut (Makridakis, 1999). Persamaan MSE sebagai berikut:

$$MSE = \frac{1}{N}\sum_{i=h}^{N}(y_t - \hat{y}_t)^2 \quad \text{..... (4-7)}$$

dengan, N adalah jumlah sampel, y_t adalah nilai aktual indeks, dan $\hat{y}_t$ adalah nilai prediksi indeks.

4.5. Evaluasi Hasil

Melakukan perbandingan dari hasil pengujian, untuk mendapatkan tingkat akurasi dari pendeteksian dan perhitungan kecepatan kendaraan.

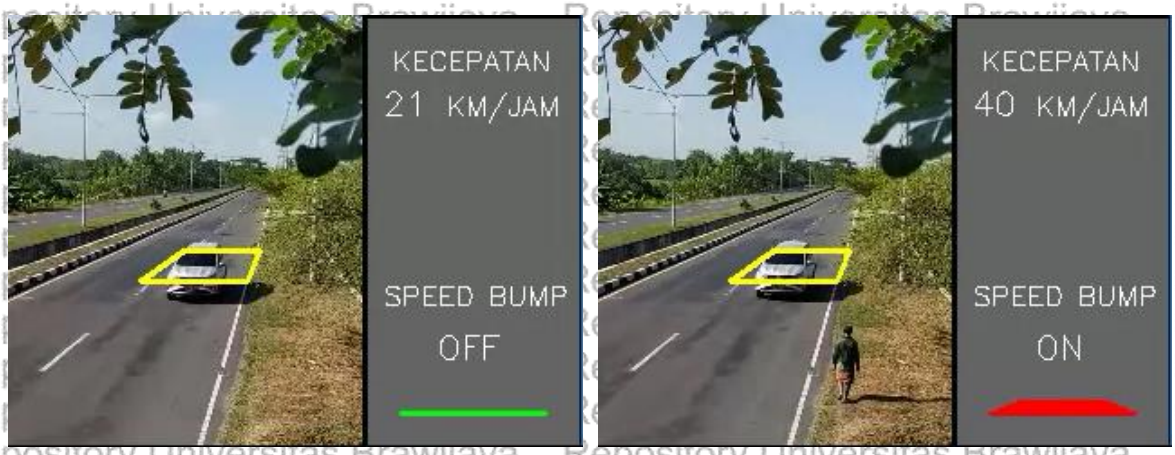
4.6. Analisis

Pada tahap ini dilakukan analisis terhadap sistem yang telah dibuat dengan cara membandingkan hasil perancangan dan pembuatan dengan parameter-parameter hasil pengujian. Hasil analisis ini akan digunakan sebagai bahan dalam pengambilan kesimpulan dan saran.

BAB V
HASIL DAN PEMBAHASAN

5.1. Tampilan Antarmuka Sistem

Sistem dalam penelitian ini ditampilkan dalam bentuk simulasi yang dapat disesuaikan dengan sistem secara *real time*. Sistem *real time* dapat diterapkan pada *embedded system* yang terdiri dari *raspberry pi* dan terintegrasi dengan *pi camera* atau *webcam* sebagai sensor visual untuk mengambil citra dan juga terhubung dengan motor servo sebagai *speed bump*. Tampilan hasil perancangan sistem ditampilkan pada Gambar 5.1.



(a) (b)

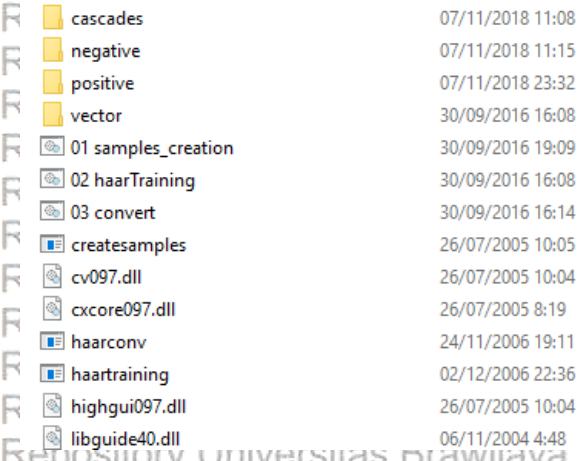
Gambar 5.1 Tampilan hasil perancangan sistem, (a) kondisi kecepatan dibawah 30 km/jam, (b) kondisi kecepatan diatas 30 km/jam.

5.2 Proses Perancangan Database Citra

Pada penelitian ini, proses pertama yang dilakukan adalah perancangan database citra. Terdapat beberapa langkah-langkah untuk mendapatkan sebuah database citra dengan format file .xml, yang dijelasakan sebagai berikut:

5.2.1 Persiapan Direktori

Dalam proses perancangan database citra, diperlukan sebuah direktori dengan susunan menggunakan contoh dari *repository* milik Rezaei [13], https://www.cs.auckland.ac.nz/~m.rezaei/ Tutorials/Haar-Training.zip.



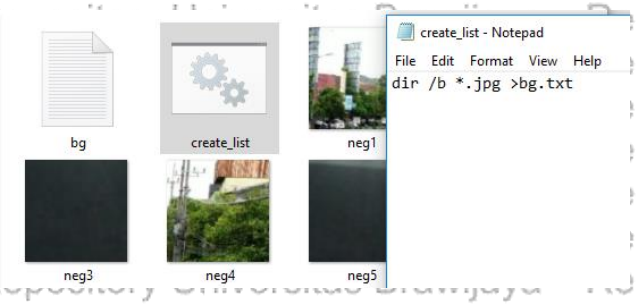
Gambar 5.2 Susunan direktori

5.2.2 Pembuatan Sampel Citra

Pembuatan sampel citra tediri dari 600 citra positif dan 1000 citra negatif. Citra positif adalah citra yang mengandung objek mobil dan citra negatif adalah citra yang tidak mengandung objek mobil atau *background*. Resolusi untuk sample positif dan sample negatif bernilai sama dengan resolusi video yaitu 320x240 piksel. Ini dilakukan untuk mengurangi *error* dalam proses pengklasifikasiannya. Berikut adalah tahap-tahap dalam membuat sampel citra:

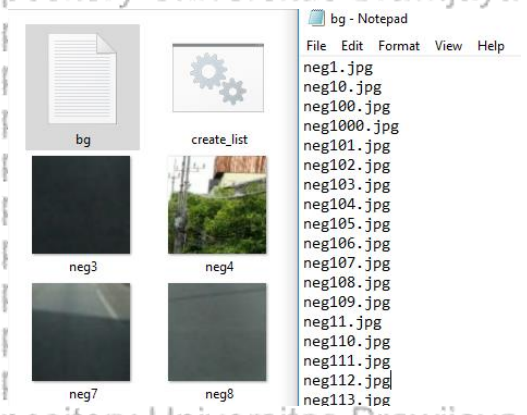
1. Pembuatan List Citra Negatif

Letakkan citra negatif (*background*) pada folder …\training\negative dan jalankan file batch create_list.bat yang berisi berkas teks dengan perintah *command line*.



Gambar 5.3 File batch

Setelah menjalankan file batch, akan didapatkan hasil file berupa teks yang berisi nama-nama citra negatif seperti pada Gambar 5.4.

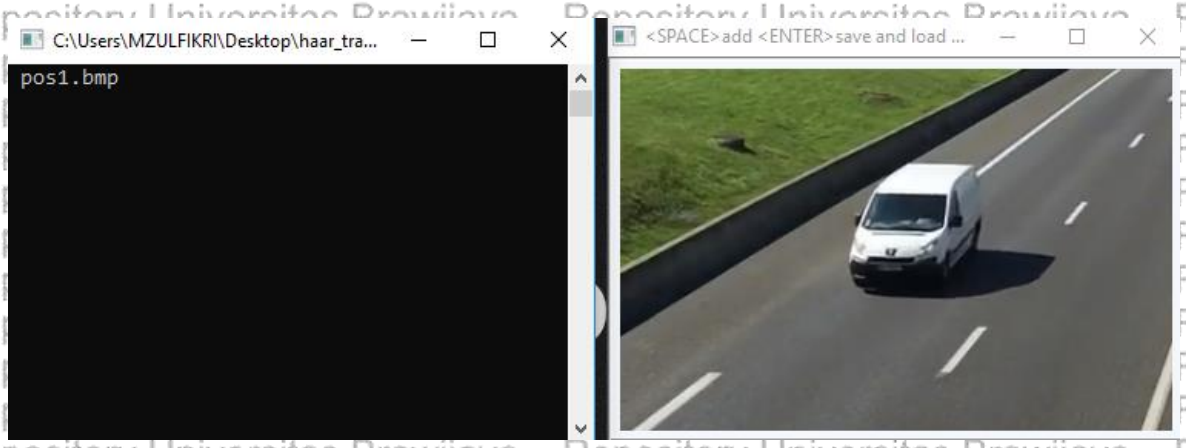


Gambar 5.4 Daftar nama-nama citra negatif

2. Pembuatan List Citra Positif

Pada langkah ini, diperlukan sebuah file data (file vector) yang berisi nama-nama citra positif disertai lokasi lokasi objek dalam setiap gambar. File dibuat menggunakan sebuah *utilities Objectmaker*.

Letakkan citra positif yang sudah disiapkan pada folder …training\ positive\rawdata dan file objectmaker.exe pada folder …training\positive. Saat menjalankan file objectmaker.exe, akan terlihat dua tampilan jendela berupa citra yang dimuat (Gambar 5.5a) dan citra positif yang akan tandai (Gambar 5.5b).

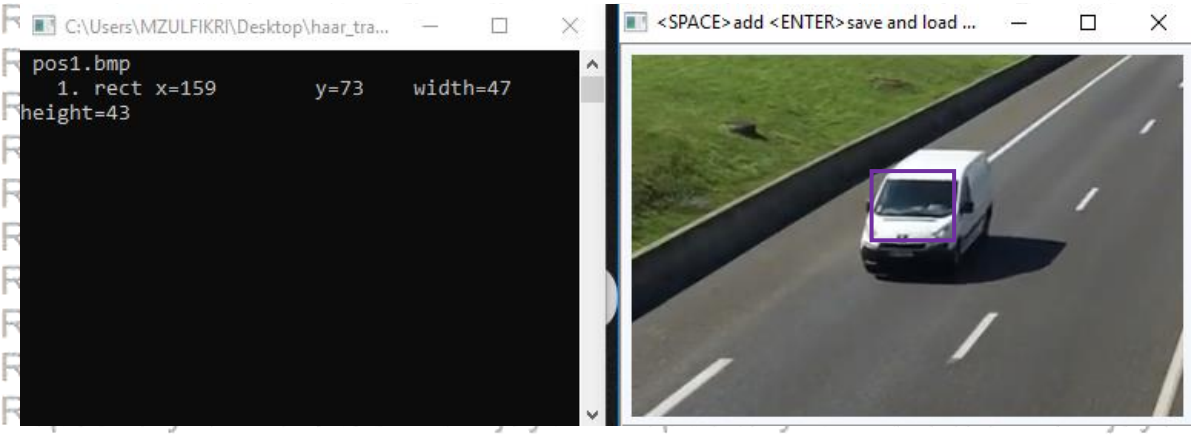


(a) (b)

Gambar 5.5 (a) Citra yang dimuat (b) Citra positif yang akan ditandai

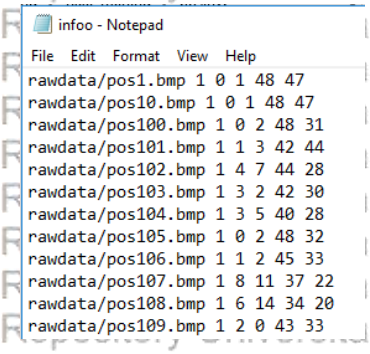
Proses selanjutnya adalah menandai atau *cropping* untuk menentukan objek-objek yang akan digunakan dalam proses *training* (Gambar 5.5b), untuk mendapatkan lokasi/area objek yang akan dideteksi dengan menarik mouse dari sudut kiri ke sudut kanan seperti pada Gambar 5.6b, dan hasilnya otomatis akan tercatat pada jendela citra yang dimuat seperti pada

Gambar 5.6a. Pada *training*, resolusi objek yang di *cropping* akan dikonversikan menggunakan teknik *resize* (umumnya menggunakan resolusi citra 24x24 piksel) pada keseluruhan objek dan mencari keberadaan objek mobil. Proses *resize* menunjukkan bahwa setiap citra yang akan diklasifikasi dengan resolusi berapapun, akan dirubah menjadi ukuran 24x24 piksel. Proses tersebut dilakukan sampai semua citra positif ditandai sampai akhir dengan menghilangnya dua tampilan jendela tersebut, dan akan menghasilkan file info.txt yang berisi *rawdata* dari citra yang sudah ditandai, seperti pada Gambar 5.7.



(a) (b)

Gambar 5.6 (a) Citra yang dimuat (b) Citra positif yang ditandai



Gambar 5.7 File info.txt yang berisi *rawdata* citra positif

3. Pembuatan Vector Citra Positif

Dalam folder `..\training\` terdapat *file batch* bernama `samples_creation.bat`. Isi dari *file batch* tersebut adalah:

```
createsamples.exe -info positive/info.txt -vec vector/vector.vec -
num 200 -w 24 -h 24
```

dengan parameter utamanya yaitu:

`-info positive/info.txt` : Path untuk file info positif

`-vec vector/vector.vec` : Path untuk output file vektor

`-num 200` : Jumlah file positif yang akan dikemas dalam file vektor

`-w 24` : Lebar objek

`-h 24` : Ketinggian objek

File batch akan memuat `info.txt` dan mengemas citra objek ke dalam file vektor dengan nama `vector.vec`. Setelah *file batch* dijalankan, akan didapatkan sebuah file `vector.vec` di dalam folder `..training\vector`.

4. Pembuatan *Classifier*

Tahap selanjutnya yaitu pembuatan *classifier* atau *haar-training*, yang merupakan tahap untuk membuat classifier .xml yang masih terpisah-pisah atau belum disatukan dengan jumlah *stage* yang ditentukan. Dalam folder `..\training`, dilakukan proses modifikasi *file batch* `haartraining.bat` :

```
haartraining.exe -data cascades -vec vector/vector.vec -bg
negative/bg.txt -npos 200 -nneg 200 -nstages 15 -mem 1024 -
mode ALL -w 24 -h 24 -nonsym
```

dengan parameter utamanya yaitu:

`-data cascades` : Path dan untuk menyimpan *cascade* dari *classifier*

`-vec data/vector.vec` : Path yang menunjukkan lokasi file *vector*

`-bg negative/bg.txt` : Path yang menunjukkan *file background*

`-npos 200` : Jumlah sample positif

`-nneg 200` : Jumlah sample negatif

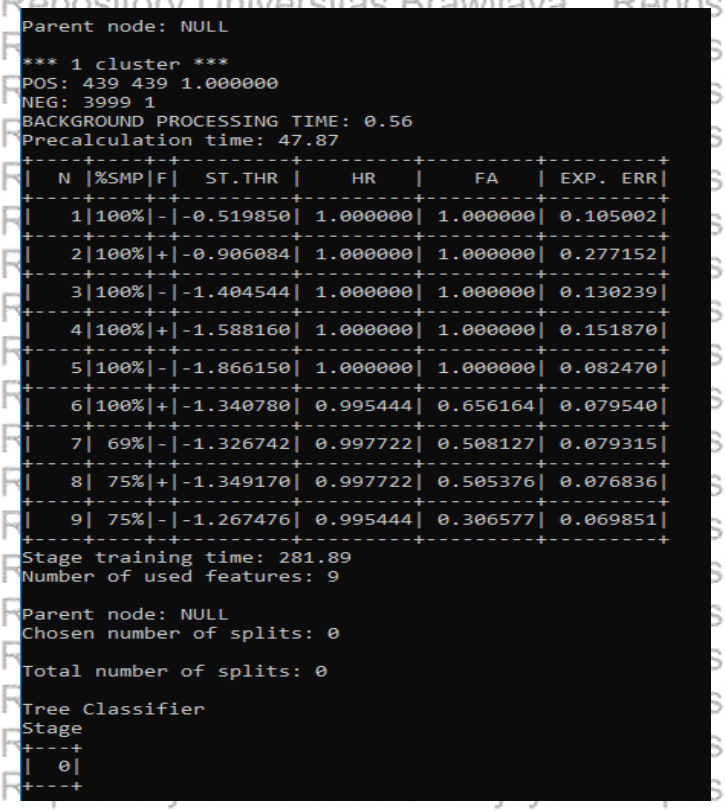
`-nstages 15` : Jumlah tahapan dalama prose *training*

`-mem 1024` : Kuantitas dari memori yang ditetapkan dalam MB

`-w 24 -h 24` : Resolusi sample (harus sama dengan resolusi saat didefinisikan pada `sample-creation.bat`.

`-nonsym` : Path digunakan jika subjek tidak simetris atau horizontal

Saat menjalankan `haartraining.bat`, akan terlihat beberapa informasi dari proses *training* seperti pada Gambar 5.7.



```
Parent node: NULL

*** 1 cluster ***
POS: 439 439 1.000000
NEG: 3999 1
BACKGROUND PROCESSING TIME: 0.56
Precalculation time: 47.87
+----+----++---------+---------+---------+---------+
|  N |%SMP|F|  ST.THR |    HR   |    FA   | EXP. ERR|
+----+----++---------+---------+---------+---------+
|   1|100%|-|-0.519850| 1.000000| 1.000000| 0.105002|
+----+----++---------+---------+---------+---------+
|   2|100%|+|-0.906084| 1.000000| 1.000000| 0.277152|
+----+----++---------+---------+---------+---------+
|   3|100%|-|-1.404544| 1.000000| 1.000000| 0.130239|
+----+----++---------+---------+---------+---------+
|   4|100%|+|-1.588160| 1.000000| 1.000000| 0.151870|
+----+----++---------+---------+---------+---------+
|   5|100%|-|-1.866150| 1.000000| 1.000000| 0.082470|
+----+----++---------+---------+---------+---------+
|   6|100%|+|-1.340780| 0.995444| 0.656164| 0.079540|
+----+----++---------+---------+---------+---------+
|   7| 69%|-|-1.326742| 0.997722| 0.508127| 0.079315|
+----+----++---------+---------+---------+---------+
|   8| 75%|+|-1.349170| 0.997722| 0.505376| 0.076836|
+----+----++---------+---------+---------+---------+
|   9| 75%|-|-1.267476| 0.995444| 0.306577| 0.069851|
+----+----++---------+---------+---------+---------+
Stage training time: 281.89
Number of used features: 9

Parent node: NULL
Chosen number of splits: 0

Total number of splits: 0

Tree Classifier
Stage
+---+
|  0|
+---+
```

Gambar 5.8 Tampilan proses *training*

Data yang ditampilkan pada Gambar 5.8 terkait untuk tahap pertama proses *training* yaitu:

`Parent node` : Menentukan tahap saat ini dalam proses pelatihan

`N` : Jumlah fitur yang digunakan dalam tahap ini

`% SMP` : Persentase Sampel (Persentase sampel yang digunakan untuk fitur ini)

`F` : "+" jika dibalik (saat simetri diterapkan) dan "-" jika tidak

`ST.THR` : Tahap *Threshold*

`HR` : *Hit Rate* berdasarkan tahap *threshold*

`FA` : Salah Alarm berdasarkan tahap *threshold*

`EXP. ERR` : Exponential Error dari *strong classifier*

Gambar 5.9 menunjukkan data untuk tahap *classifier* ke-14, yang menampilkan jumlah fitur yang digunakan dalam node lebih tinggi dari node sebelumnya. Deteksi salah secara keseluruhan (*false alarm*) yang dihasilkan menurun, dan waktu komputasi dalam proses *training* semakin meningkat.

```
Parent node: 13

*** 1 cluster ***
POS: 437 480 0.910417
NEG: 3981 3.07042e-005
BACKGROUND PROCESSING TIME: 26770.59
Precalculation time: 36.13
+----+----++---------+---------+---------+---------+
|  N |%SMP|F|  ST.THR |    HR   |    FA   | EXP. ERR|
+----+----++---------+---------+---------+---------+
|   1|100%|-|-0.441172| 1.000000| 1.000000| 0.570167|
+----+----++---------+---------+---------+---------+
|   2|100%|+|-0.727451| 1.000000| 1.000000| 0.166818|
+----+----++---------+---------+---------+---------+
|   3| 89%|-|-0.863296| 1.000000| 1.000000| 0.253735|
+----+----++---------+---------+---------+---------+
|   4| 93%|+|-0.698193| 0.997712| 0.946998| 0.253735|
+----+----++---------+---------+---------+---------+
|   5| 93%|-|-0.697238| 0.995423| 0.951771| 0.223857|
+----+----++---------+---------+---------+---------+
|   6| 93%|+|-0.675568| 0.995423| 0.929666| 0.208692|
+----+----++---------+---------+---------+---------+
|   7| 91%|-|-0.709074| 0.995423| 0.853806| 0.372793|
+----+----++---------+---------+---------+---------+
|   8| 84%|+|-0.692284| 0.995423| 0.871640| 0.208239|
+----+----++---------+---------+---------+---------+
|   9| 88%|-|-0.827348| 0.997712| 0.903039| 0.219104|
+----+----++---------+---------+---------+---------+
|  10| 82%|+|-0.805083| 0.995423| 0.854308| 0.203938|
+----+----++---------+---------+---------+---------+
|  11| 86%|-|-0.938284| 0.995423| 0.889475| 0.174287|
+----+----++---------+---------+---------+---------+
|  12| 81%|+|-0.978119| 0.997712| 0.809093| 0.192847|
+----+----++---------+---------+---------+---------+
|  13| 82%|-|-0.730136| 0.995423| 0.739261| 0.159348|
+----+----++---------+---------+---------+---------+
|  14| 78%|+|-0.885146| 0.995423| 0.761366| 0.228384|
+----+----++---------+---------+---------+---------+
|  15| 78%|-|-0.973502| 0.995423| 0.800804| 0.182888|
+----+----++---------+---------+---------+---------+
|  16| 80%|+|-1.115595| 0.995423| 0.818136| 0.203486|
+----+----++---------+---------+---------+---------+
|  17| 78%|-|-0.909138| 0.995423| 0.681236| 0.156406|
+----+----++---------+---------+---------+---------+
|  18| 78%|+|-0.855477| 0.995423| 0.620698| 0.145088|
+----+----++---------+---------+---------+---------+
|  19| 78%|-|-0.975686| 0.995423| 0.638784| 0.118606|
+----+----++---------+---------+---------+---------+
|  20| 77%|+|-0.967594| 0.995423| 0.661140| 0.143051|
+----+----++---------+---------+---------+---------+
|  21| 77%|-|-0.868225| 0.995423| 0.593821| 0.129244|
+----+----++---------+---------+---------+---------+
|  22| 76%|+|-1.032130| 0.995423| 0.654861| 0.111363|
+----+----++---------+---------+---------+---------+
|  23| 75%|-|-0.806423| 0.995423| 0.486812| 0.118832|
+----+----++---------+---------+---------+---------+
Stage training time: 368.31
Number of used features: 23

Parent node: 13
Chosen number of splits: 0

Total number of splits: 0

Tree Classifier
Stage
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--
|  0|  1|  2|  3|  4|  5|  6|  7|  8|  9| 10| 11| 12| 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--

   0---1---2---3---4---5---6---7---8---9--10--11--12--1
```

Gambar 5.9 Tampilan proses training stage ke-14

5. Pembuatan File .XML

Setelah menyelesaikan langkah *haar-training*, dalam folder `../training/cascades/` akan ditemukan daftar nama dari "`0`" sampai "`N-`

1″ yang mana N adalah jumlah *stages* yang telah didefinisikan dalam haartraining.bat. Di setiap daftar nama tersebut harus ada file AdaBoostCARTHaarClassifier.txt yang nantinya harus digabungkan menjadi satu dalam sebuah *file* .xml yang merupakan *file* akhir dalam proses *training*.

Langkah yang dilakukan yaitu menjalankan *file batch* convert.bat dalam folder ..\training, yang berisi teks :

```
haarconv.exe data myhaar.xml 24 24
```

myhaar.xml adalah nama *file output* dan 24 24 adalah W dan H. File .xml yang telah dihasilkan agan digunakan menjadi *classifier* dalam *source code* sistem deteksi kendaraan.

5.3. Proses Pendeteksian Objek

5.3.1. Akuisisi Citra

Pada penelitian ini, digunakan data citra (video) jalan raya dengan kondisi jalan pada siang hari. Posisi pengambilan yaitu dengan sudut pandang dari atas jembatan penyeberangan dan dari sisi samping dengan posisi kamera menghadap ke bawah, yang ditunjukkan pada Gambar 5.10. Penentuan *ROI* dilakukan pada sistem dengan membatasi area deteksi, sehingga mengurangi resiko kesalahan deteksi dan proses deteksi menjadi lebih fokus dan akurat, seperti ditunjukkan pada Gambar 5.11.

Gambar 5.10 Tampilan pada jalan raya

Gambar 5.11 Penentuan *Region of Interest* (*ROI*)

5.3.2. Hasil *Preprocessing*

Pada sistem yang telah dibuat, selanjutnya citra video akan dikonversi dari warna asal *Red Green Blue* (RGB) menjadi *grayscale* menggunakan fungsi *cv2.cvtcolor* yang telah disediakan pada *OpenCV library*. Untuk merubah RGB menjadi *grayscale*, dapat menggunakan persamaan 5-1 dengan mengunakan salah satu citra sebagai contoh, yang ditampilkan pada Gambar 5.12. Pada proses *preprocessing* ini akan menghasilkan *output* berupa *Region of Interest subcitra grayscale*, yang merupakan hasil dari konversi *grayscale* pada daerah *ROI*.

$$I(x,y) = \alpha.R + \beta.G + \gamma.B \quad (5\text{-}1)$$

(a) (b)

Gambar 5.12 Contoh konversi citra (a) citra asli dengan RGB, dan (b) citra hasil konversi dengan *grayscale*

Dari Gambar 5.10 didapatkan nilai RGB yaitu, R (122), G (120), dan B(117), sehingga nilai *grayscale* dapat dihitung menggunakan persamaan 5-1.

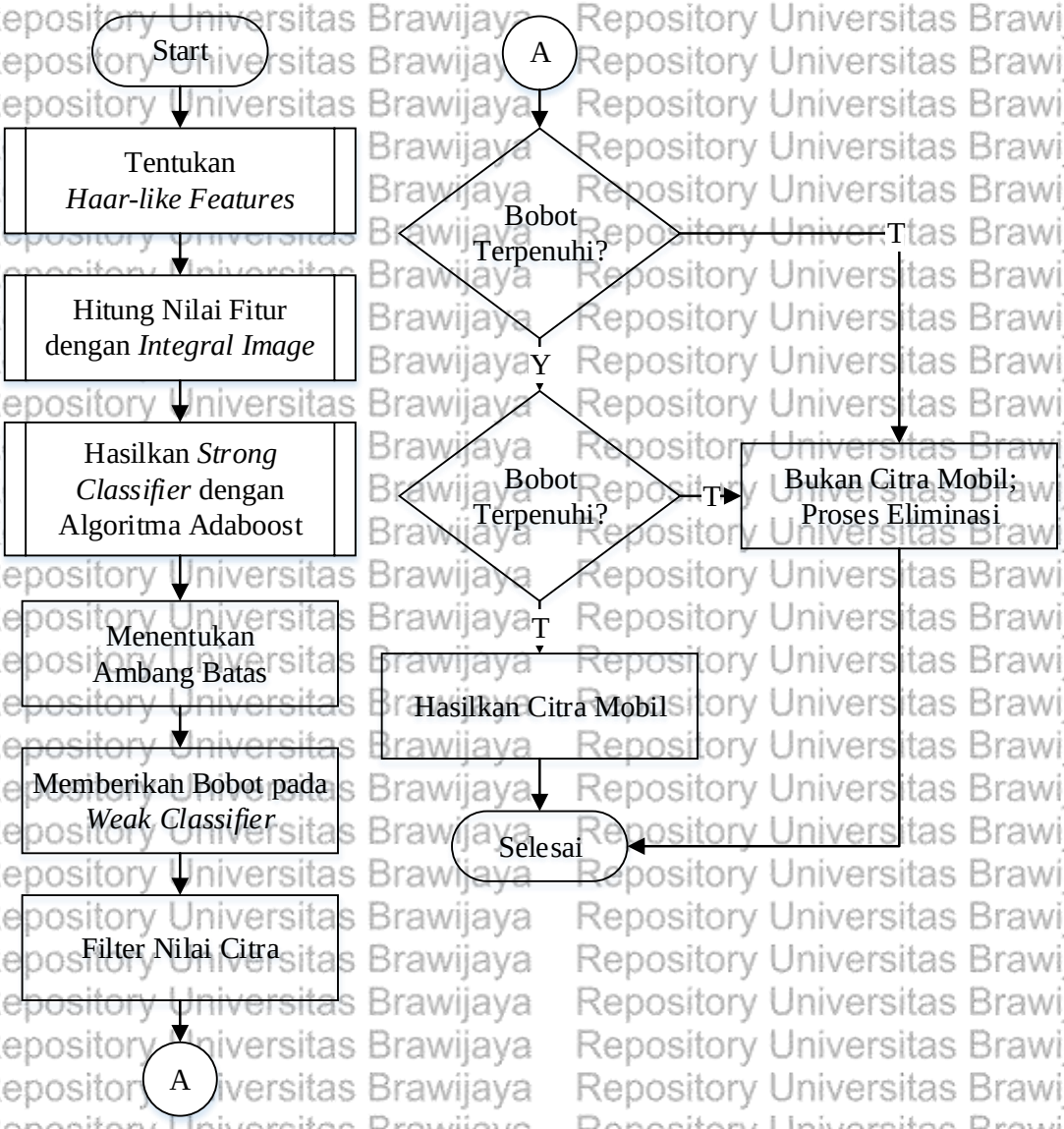
$$grayscale = 0.299\mathrm{R} + 0.587\mathrm{G} + 0.114\mathrm{B}$$

$$grayscale = 0.299 * 122 + 0.587 * 120 + 0.114 * 117$$

$$grayscale = 120{,}256$$

5.3.3. Proses *Haar Cascade Classifier*

Setelah proses *grayscale*, sistem akan mengidentifkasi setiap *frame* pada *ROI* (*ROI In* dan *ROI Out*) untuk menentukan dan mendeteksi keberadaan objek mobil. Tahap ini merupakan bagian utama dari metode *haar cascade classifier*, dimana terdapat beberapa tahapan-tahapan yang memiliki proses dan algortima tertentu yang saling berhubungan dalam proses pendeteksian objek mobil, seperti yang dapat dilihat pada Gambar 5.13 berikut ini.



Gambar 5.13 Flowchart pendeteksian *Haar Cascade Classifier*

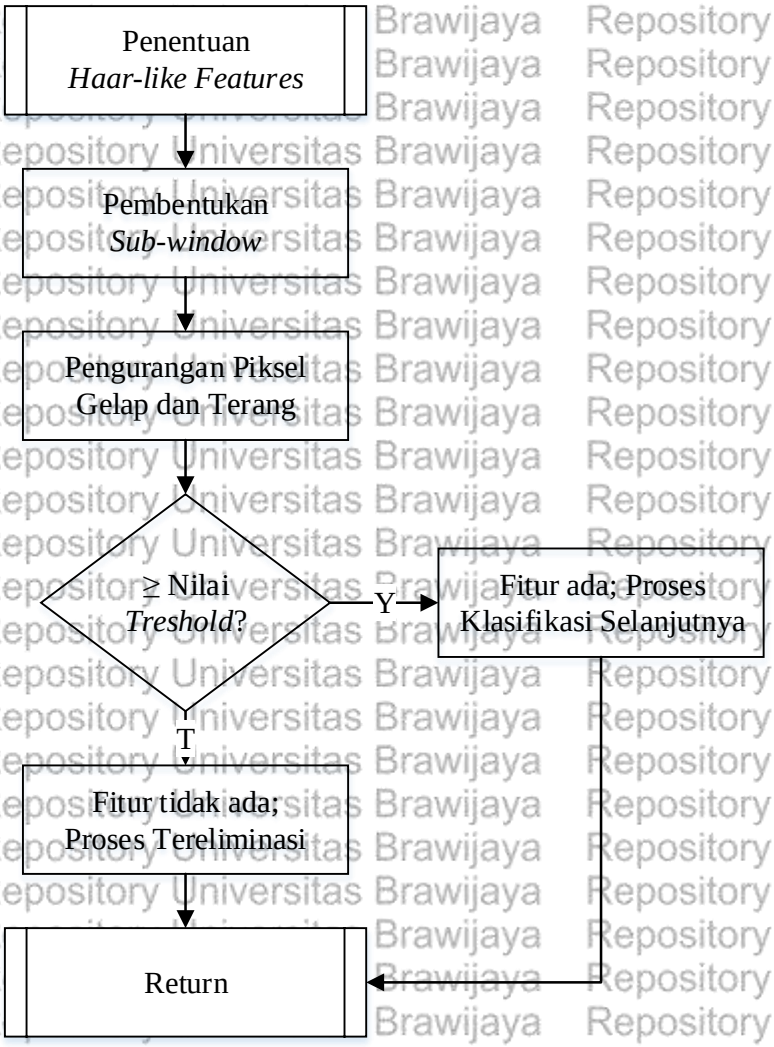
Sebagai contoh pengujian, sebuah *frame* pada *ROI* seperti pada Gambar 5.14 akan didientifikasi keberadaan objek mobil. Sehingga berdasarkan Gambar 5.13, dapat dilihat bahwa terdapat 4 proses utama dalam pendeteksian *Haar Cascade Classifier*, yaitu: menentukan *Haar-like Features*, menghitung nilai fitur dengan *integral image*, analisa algoritma *AdaBoost* untuk menentukan apakah fitur berisi objek mobil dengan serangkaian filter classifier dan analisa *Cascaded Classifier* untuk melakukan filter nilai citra, dengan memberikan nilai bobot untuk melewati setiap filter sehingga dapat diketahui citra mengandung objek mobil atau tidak.

Gambar 5.14 Identifkasi objek pada sebuah *frame ROI*

Berikut ini merupakan penjelasan dari tahapan-tahapan yang ada dalam proses pendeteksian mobil menggunakan *Haar Cascade Classifier* tersebut:

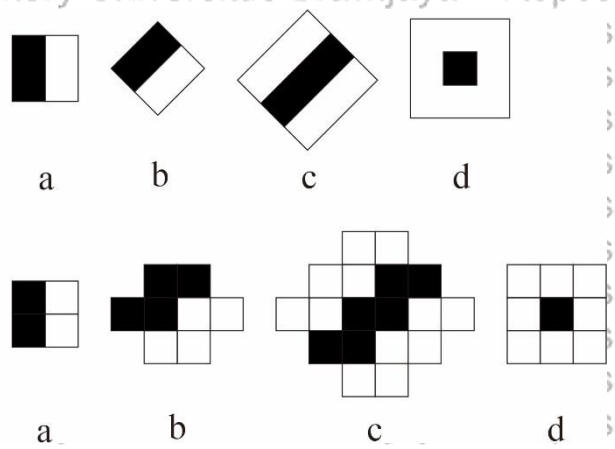
5.3.3.1.Proses *Haar-like Feature* dan *Integral Image*

Identifkasi setiap *frame* dilakukan dengan proses pemilihan fitur, dengan cara melakukan pemindaian pada citra *grayscale* menggunakan *detector* yang disebut dengan *Haar-Like features*. Teknik yang dilakukan yaitu memindai bagian citra dengan *Haar-Like feature* setiap daerah pada *image* dari mulai ujung kiri atas sampai kanan bawah. Untuk menjelaskan proses penentuan nilai fitur, dapat dilihat pada Gambar 5.15.



Gambar 5.15 Flowchart penentuan nilai fitur

Haar-like features merupakan fitur citra sederhana yang digunakan untuk mengenali objek berdasarkan pola unik dari perbedaan intensitas pada citra, yang awalnya digunakan dalam pengenalan wajah oleh Viola-Jones, dimana algoritma dapat menggunakan fitur mata, hidung, atau bibir untuk membentuk *Haar-like features* unik pada wajah. Sehingga dalam pengenalan atau pendeteksian mobil, dalam deteksinya digunakan jenis fitur yang diusulkan oleh Viola Jones dan fitur yang dimodifikasi khusus untuk pengenalan objek mobil yang dikembangkan oleh Lienhart, seperti pada Gambar 5.16.



Gambar 5.16 Haar-like features dengan, 1. bentuk fitur secara umum, dan 2. bentuk fitur yang diterapkan pada piksel.

Tiap fitur terbagi atas dua bagian, yaitu bagian berwarna putih dan bagian berwarna hitam. Untuk menentukan keberadaan sebuah fitur mobil ditentukan dengan mengurangi nilai piksel bagian hitam dengan nilai piksel bagian putih. Jika nilai dari hasil perbedaanya di atas dari ambang batas maka fitur tersebut dapat dikatakan ada. Penggunaan fitur dilakukan karena dalam pemrosesan fitur yang berlangsung lebih cepat dibandingkan pemrosesan citra perpiksel, sehingga sesuai diterapkan dalam proses pendeteksian objek secara *real time*. Pada penelitian ini untuk menghitung nilai fitur *Haar* pada sebuah citra dan pada skala yang berbeda secara cepat menggunakan satu teknik yang disebut *integral image*.

Secara umum, *integral image* digunakan untuk menambahkan bobot yang merupakan nilai-nilai piksel yang ditambahkan ke dalam citra aslinya. Nilai *integral image* dari setiap piksel merupakan hasil penjumlahan dari semua piksel bagian atasnya dan bagian kirinya. Sebagai contoh perhitungan *integral image* seperti pada Gambar 5.17.

2	4	7
1	5	1
5	6	9

(a)

→

2	6	13
3	12	20
8	23	40

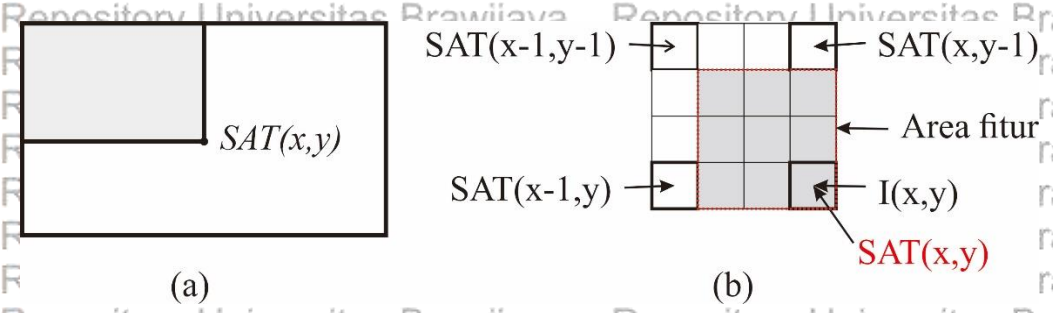
(b)

Gambar 5.17 (a) Citra masukan, (b) *integral image*

Untuk mendapatkan nilai *integral image* seperti Gambar 5.17, dapat dihitung dengan langkah sebagai berikut:

- Dengan nilai awal pada pojok kiri atas adalah 2, dilanjutkan dengan penjumlahan ke kanan (2+4=6), (2+4+7=13), dan dengan hal yang sama penjumlahan ke bawah (2+1=3), (2+1+5=8).
- Untuk menghitung kotak lainnya yaitu mencari nilai dari tengah piksel (2+4+1+5=12).
- Menghitung tiga kotak tersisa dengan cukup menjumlahkan setiap angka, yaitu (2+4+7+1+5+1=20), (2+4+1+5+5+6=23), dan (2+4+7+1+5+1+5+6+9=40).

Setelah perhitungan *integral image* dilakukan, selanjutnya yaitu menghitung nilai pada sebuah fitur, dengan tujuan untuk menentukan terdapatnya area yang mencirikan objek mobil yang dilakukan dengan mengurangi nilai piksel pada area hitam dengan piksel pada area putih. Untuk menghitung nilai suatu area fitur hitam atau putih, dapat dilakukan perhitungan nilai piksel dari sudut kiri atas di (0,0) ke sudut kanan bawah di (x, y) (lihat Gambar 5.18), dengan menggunakan persamaan 5-2.



Gambar 5.18 (a) Persegi panjang tegak dengan *Summed Area Table* (SAT), (b) Skema perhitungan jumlah piksel persegi panjang tegak

$$SAT(x,y) = SAT(x,y-1) + SAT(x-1,y) + I(x,y) - SAT(x-1,y-1) \quad \text{5-1}$$

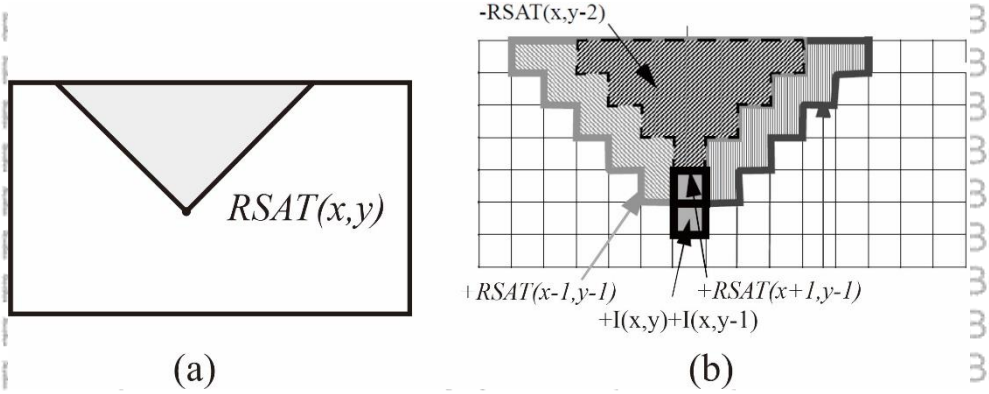
dengan;

I(x,y) = nilai awal piksel

SAT(x,y) = titik area dari nilai yang dicari

Untuk persegi panjang yang dirotasi 45°, gambar tambahan adalah *Rotation Summed Area Table* (RSAT). Yang dijelaskan sebagai jumlah piksel persegi panjang yang dirotasi

dengan sudut paling bawah (x, y) dan memanjang ke atas hingga batas gambar, yang dapat dihitung juga dalam satu lintasan dari kiri ke kanan dan atas ke bawah ke semua piksel dengan persamaan 5-2, dan penjelasan posisi piksel *x* dan *y* dapat dilihat pada Gambar 5.19.



Gambar 5.19 (a) *Rotation Summed Area Table* (RSAT), (b) Skema perhitungan untuk *Summed Area Table* (RSAT) pada piksel.

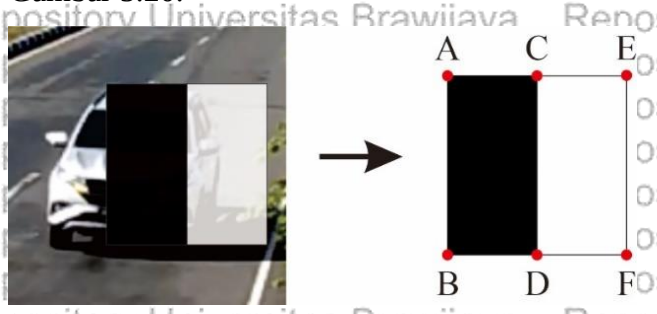
$$RSAT(x,y) = RSAT(x-1,y-1) + RSAT(x+1,y-1) - RSAT(x,y-2) + I(x,y) + I(x,y-1) \quad \text{5-2}$$

dengan;

$I(x,y) + I(x,y-1)$ = nilai awal piksel

$RSAT(x,y)$ = titik area dari nilai yang dicari

Jika proses perhitungan nilai piksel pada suatu area, selanjutnya dapat dihitung nilai fitur untuk mengetahui keberadaan objek mobil pada sebuah citra, yang ditunjukkan pada Gambar 5.20.



Gambar 5.20 Perhitungan nilai fitur *haar*

$$F(Haar) = \sum F\ Black - \sum F\ White \quad \text{(5-3)}$$
$$= (D - B - C + A) - (F - D - E + C)$$

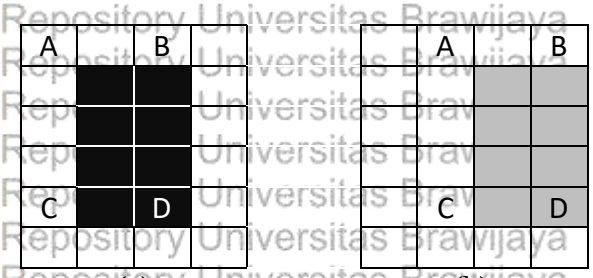
dengan,

F(Haar)= Nilai fitur total

$\sum F\ White$ = Nilai fitur pada daerah putih

$\sum F\ Black$ = Nilai fitur pada daerah hitam

Perhitungan nilai fitur *haar* dapat dicontohkan pada Gambar 5.21.



(a) (b)

Gambar 5.21 (a) Fitur hitam, (b) fitur putih

Maka, nilai fitur *haar* adalah D-C-B+A, sehingga didapatkan nilai fitur *haar* seperti pada Gambar 5.22.

2	6	13	18	26
3	12	20	32	47
8	23	40	57	78
16	40	67	90	118
26	62	97	123	157

Hitam = 97-26-13+2 = 60

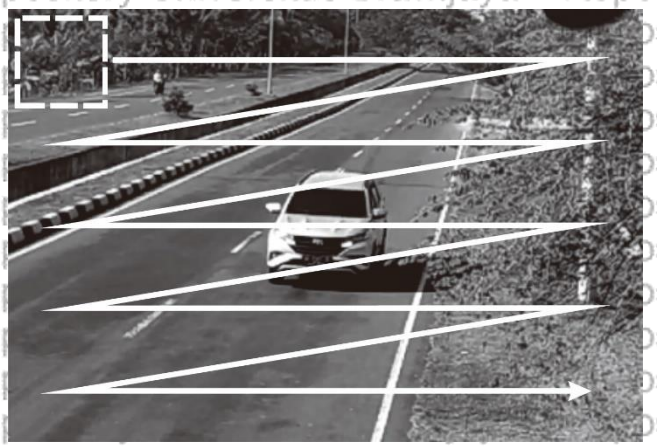
Putih = 157-97-26+13 = 47

Fitur haar = putih – hitam = 13

Gambar 5.22 Nilai piksel dari sebuah fitur

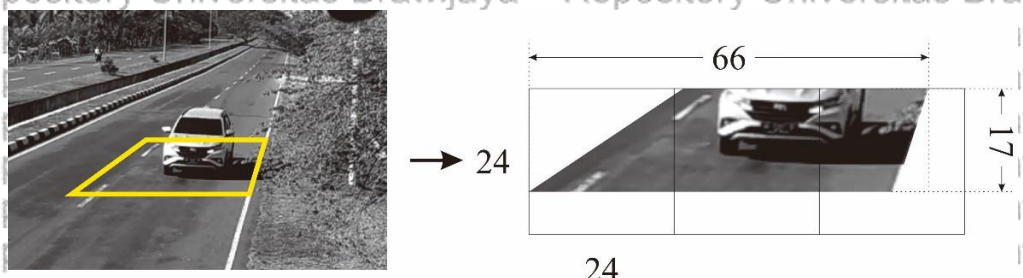
Sehingga, didapatkan nilai fitur *haar* adalah 13. Jika nilai *threshold* dimisalkan adalah 10, maka untuk nilai fitur *haar* >10, maka fitur *haar* menunjukkan adanya objek mobil. Jika nilai fitur *haar* <10, maka fitur *haar* menunjukkan tidak adanya objek mobil.

Secara umumnya, proses deteksi dilakukan secara bergeser di seluruh citra pada setiap skala untuk mendeteksi mobil, seperti yang ditunjukkan pada Gambar 5.23. Dalam implementasinya, pendeteksian dilakukan pada setiap sub-citra yang bergeser setiap piksel. Sub-citra memiliki resolusi 24x24 piksel, dimana resolusi tersebut merupakan resolusi dasar deteksi. Setiap kali sub-citra bergeser, akan melalui *cascade classifier* yang akan dijelaskan selanjutnya.



Gambar 5.23 Haar-like features

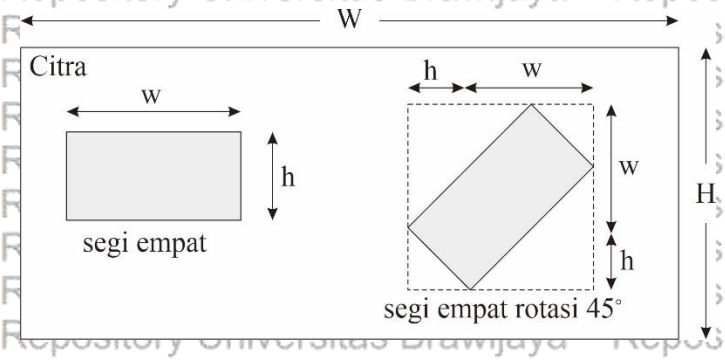
Pada penelitian ini, dikarenakan area deteksi objek dibatasi pada 1 jalur jalan raya, sehingga untuk *ROI* juga dibatasi dengan tujuan pendeteksian lebih akurat. Hal tersebut menjadikan pendeteksian setiap sub-citra hanya pada *ROI* seperti pada Gambar 5.24.



Gambar 5.24 Pendeteksian *sub-window*

Gambar 5.24 menunjukkan pendeteksian sub-citra yang dibatasi hanya pada *ROI*. Diketahui resolusi *ROI* adalah 66x17 piksel dan sub-citra berresolusi 24x24, sehingga resolusi deteksi sub-citra menjadi 24x17 piksel.

Dalam pendeteksian setiap sub-citra, digunakan 4 (empat) tipe *haar-like feature* seperti pada Gambar 5.16. Sehingga, perlu diketahui jumlah *feature* yang digunakan dalam setiap sub-citra. Resolusi minimal *Haar-like feature* dengan tipe (a) adalah 2x1 piksel, tipe (b) 1x3 piksel, tipe (c) 4x4 piksel, dan tipe (d) 3x3 piksel. Jumlah fitur yang berasal dari masing-masing tipe cukup besar dan berbeda dari tipe ke tipe lainnya dan dapat dihitung sebagai berikut.



Gambar 5.25 Contoh fitur segi empat tegak dan rotasi 45◦

Dari Gambar 5.25, diberikan $X=W/w$ dan $Y=H/h$, yang menjadi faktor penskalaan maksimum dalam arah *x* dan *y*. Fitur tegak dengan ukuran *wxh* kemudian menghasilkan fitur untuk citra ukuran *WxH*, dapat dihitung jumlah fiturnya menggunakan persamaan 5-1, sedangkan fitur yang diputar rotasi 45◦ menggunakan persamaan 5-2 (Lienhart, 2002).

$$XY.\left(W+1-w\frac{X+1}{2}\right).\left(H+1-h\frac{Y+1}{2}\right) \quad \text{(5-3)}$$

$$XY.\left(W+1-z\frac{X+1}{2}\right).\left(H+1-z\frac{Y+1}{2}\right) \text{ dengan } z=w+h \quad \text{(5-4)}$$

Dari persamaan 5-3 dan 5-4, dapat dicari jumlah fitur pada sebuah sub-citra 24x17 piksel. Dengan mengambil contoh fitur tipe (a) 2x1 piksel, didapatkan perhitungan sebagai berikut:

$XY.\left(W+1-w\frac{X+1}{2}\right).\left(H+1-h\frac{Y+1}{2}\right)$, dengan diketahui: $W=24; H=17; w=2; h=1; X=\frac{W}{w}=\frac{24}{2}=12; Y=\frac{H}{h}=\frac{17}{1}=17$

sehingga $(24x17).\left(24+1-2\frac{12+1}{2}\right).\left(17+1-1\frac{17+1}{2}\right)$ = 22032 fitur.

Sehinnga, dari 4 tipe fitur yang digunakan, dapat ditentukan jumlah penggunaan setiap fitur pada sub-citra 24x17 piksel yang dapat dilihat pada Tabel 5.1.

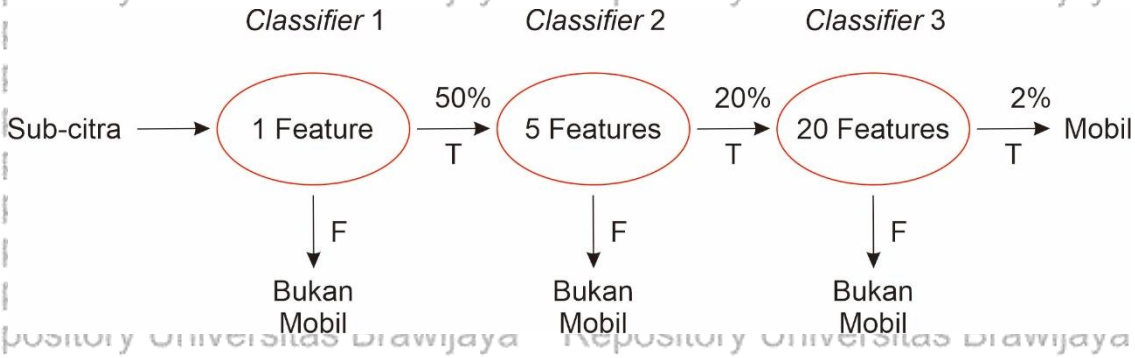
Tabel 5.1 *Jumlah fitur pada sub-citra 24x17 pada fitur tipe (a,b,c dan d)*

Tipe fitur	w/h	X/Y	Jumlah fitur
Tipe a	2/1	12/17	22032
Tipe b	3/3	8/5	4140
Tipe c	4/4	6/4	2112
Tipe d	3/3	8/5	4140
Total			32424

5.3.3.2.Proses *Adaboost (Adaptive Boosting)* dan *Cascade Classifier*

Selanjutnya, setelah nilai fitur didapatkan, proses selanjutnya menggunakan algoritma *AdaBoost machine-learning* untuk memilih fitur *Haar* yang spesifik, apakah merepresentasikan ada tidaknya mobil dalam suatu citra masukan. Hal ini dilakukan dengan cara mengevaluasi setiap fitur terhadap data latih dengan menggunakan nilai dari fitur tersebut. *AdaBoost* merupakan algortima menggabungkan banyak *classifier* lemah (citra yang kurang tajam) untuk membangun *classifier* kuat. *Classifier* adalah suatu ciri yang menandakan adanya objek mobil dalam suatu citra. Sedangkan *classifier* lemah adalah suatu jawaban benar namun memiliki tingkat kebenaran yang kurang akurat, jika digabung maka *classifier* lemah tersebut akan menghasilkan suatu *classifier* kuat.

Adaboost merupakan representasi dari proses *cascade classifier*, dimana setiap *stage classifier*-nya dibangun dengan menggunakan algoritma *adaboost.* Alur proses pendeteksian pada *cascade classifier* dapat dilihat pada Gambar 5.26.

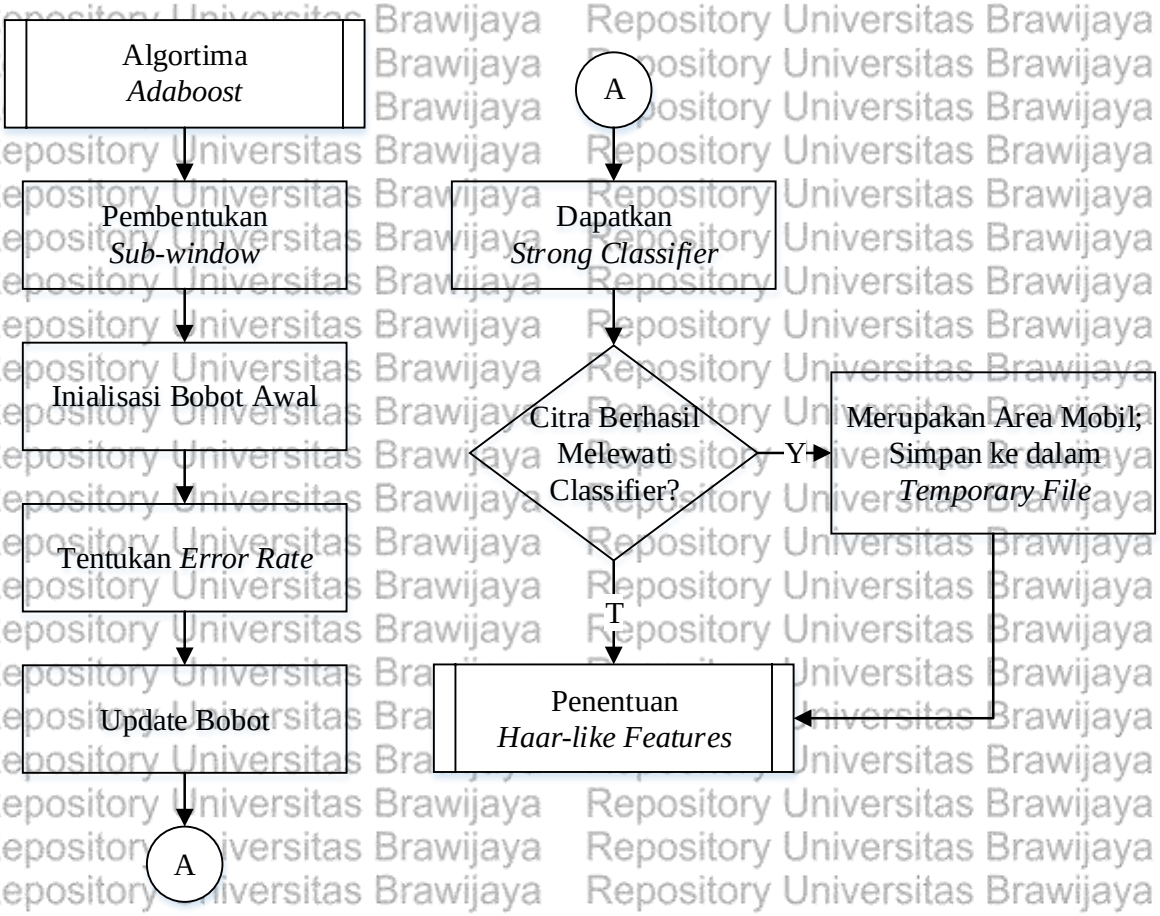


Gambar 5.26 Alur proses *cascade classifier*

Pada klasifikasi tingkat awal (*classifier* 1), tiap sub-citra akan diklasifikasi menggunakan 1 *feature*. Hasil klasifikasi pertama berupa T (*True*) untuk citra yang memenuhi fitur *Haar* tertentu dan F (*False*) bila tidak memenuhi. Klasifikasi memungkinkan akan menyisakan 50% sub-citra, dan diklasifikasikan pada klasifikasi selanjutnya (*classifier* 2). Semakin bertambahnya tingkatan klasifikasi, maka diperlukan syarat yang lebih spesifik

sehingga fitur yang digunakan menjadi lebih banyak. Jumlah sub-citra yang lolos klasifikasi akan berkurang hingga mendekati citra yang diharapkan atau mendekati citra yang telah di *training* dengan klasifikasi mencapai jumlah sekitar 2%.

Proses tahapan algoritma *adaboost* akan dijelaskan secara lebih terperinci pada Gambar 5.27.



Gambar 5.27 Alur proses Algoritma *Adaboost*

Dalam proses algoritma *adaboost*, dapat diambil sebuah contoh proses pengklasifikasian dalam sebuah citra pada Gambar 5.24. Jika dalam proses terdapat *classifier* lemah atau sebuah citra yang masih dianggap sebagai citra negatif tetapi masih memungkinkan menjadi *classifier* lemah atau citra positif, maka dilakukan proses *adaboost*. Langkah awal yaitu pengambilan beberapa sub-citra dan ditentukan nilai piksel dan *integral image*, seperti pada Gambar 5.28.

Gambar 5.28 Pengambilan *sub-window* pada citra

Dari Gambar 5.28, akan dilakukan proses perhitungan nilai fitur untuk mencari *classifier* kuat dan *classifier* lemah. Jika terdapat terdapat beberapa *classifier* lemah, maka algoritma *adaboost* akan melakukan kombinasi dengan *classifier* kuat untuk menentukan apakah setelah kombinasi, *classifier* lemah tersebut dapat menjadi *classifier* kuat atau tereliminasi atau dinyatakan tidak fitur yang mencirikan fitur objek mobil. Langkah-langkah algoritma *adaboost* dijelaskan sebagai berikut:

Bobot awal $w_{1,y_i} = \frac{1}{2m}, w_{1,y_i} = \frac{1}{2l}$ .....(5-5)

dengan;

m = jumlah citra negatif dengan $y_i = 0$ untuk citra negatif

l = jumlah citra positif dengan $y_i = 1$ untuk citra positif

dimisalkan $l = 4$, dan $m = 2$, maka;

$$w_{1,0} = \frac{1}{2x4} = 0{,}125, w_{1,1} = \frac{1}{2x2} = 0{,}25$$

Untuk mendapatkan nilai *error rate* dari setiap *classifier* lemah, maka untuk setiap fitur dilakukan proses sebagai berikut:

Untuk citra positif: $\in_t = \left(\sum_t^T w_{t,i}\right)|h_t(x) - y_i|$ .....(5-6)

Untuk citra negatif: $\in_j = \left(\sum_j^T w_{j,i}\right)|h_j(x) - y_i|$ .....(5-7)

dengan;

t = 1,2,3…T, dimana t adalah iterasi ke t untuk citra positif.

j = 1,2,3…J, dimana j adalah iterasi ke j untuk citra negatif.

$h_t(x)$ merupakan nilai fitur citra positif.

$h_j(x)$ merupakan nilai fitur citra negatif.

Dari sub-citra dengan resolusi 24x17 piksel pada Gambar 5.29, didapatkan dengan nilai piksel dan *integral image* sebagai berikut:

254	181	99	74	97	94	78	99	110	137	232	254	245	223	221	255	217	200	211	205	207	219	221	143
148	76	66	99	77	95	147	112	116	220	227	250	247	253	252	255	248	254	243	250	235	243	206	206
63	93	98	66	107	154	119	106	143	243	253	254	204	185	94	92	123	144	183	163	153	164	153	141
85	81	81	147	154	97	103	118	153	255	254	236	229	218	54	2	0	19	103	58	14	1	1	8
103	119	194	170	113	135	138	119	146	255	233	254	254	123	17	20	30	16	2	7	10	18	1	51
86	149	144	126	109	90	103	99	170	140	101	152	231	219	149	37	1	1	16	20	7	5	14	68
112	93	108	106	97	101	95	102	117	122	200	243	229	193	157	106	53	23	49	25	20	49	127	129
116	111	104	99	87	90	99	86	139	254	233	174	130	47	30	25	27	38	35	45	12	6	13	25
108	100	93	91	91	92	95	100	98	60	44	96	78	97	89	94	64	62	59	57	52	45	41	40
124	111	101	104	110	108	101	94	82	6	0	12	50	38	24	0	3	2	1	2	2	2	2	4
122	114	110	114	116	111	101	94	79	7	0	2	51	38	43	9	1	1	1	1	3	5	8	9
105	108	112	113	105	97	98	103	100	64	40	41	25	14	1	0	3	4	3	0	0	0	0	0
99	100	103	101	94	90	96	105	107	102	87	109	83	83	52	45	30	35	37	34	33	33	29	23
103	94	87	87	90	91	92	94	92	93	91	89	78	64	48	41	76	85	91	92	95	98	96	89
103	94	88	90	95	96	94	93	99	102	102	81	82	70	78	81	76	85	92	91	95	101	99	92
101	101	104	107	104	101	103	108	103	104	93	84	90	94	88	86	88	96	99	94	94	99	97	89
100	99	98	98	98	99	100	101	100	95	91	91	91	89	89	91	91	99	100	99	102	98	95	102

Gambar 5.29 Nilai piksel sub-citra dari Gambar 5.28.

254	435	534	608	705	799	877	976	1086	1223	1455	1709	1954	2177	2398	2653	2870	3070	3281	3486	3693	3912	4133	4276
402	659	824	997	1171	1360	1585	1796	2022	2379	2838	3342	3834	4310	4783	5293	5758	6212	6666	7121	7563	8025	8452	8801
465	815	1078	1317	1598	1941	2285	2602	2971	3571	4283	5041	5737	6398	6965	7567	8155	8753	9390	10008	10603	11229	11809	12299
550	981	1325	1711	2146	2586	3033	3468	3990	4845	5811	6805	7730	8609	9230	9834	10422	11039	11779	12455	13064	13691	14272	14770
653	1203	1741	2297	2845	3420	4005	4559	5227	6337	7536	8784	9963	10965	11603	12227	12845	13478	14220	14903	15522	16167	16749	17298
739	1438	2120	2802	3459	4124	4812	5465	6303	7553	8853	10253	11663	12884	13671	14332	14951	15585	16343	17046	17672	18322	18918	19535
851	1643	2433	3221	3975	4741	5524	6279	7234	8606	10106	11749	13388	14802	15746	16513	17185	17842	18649	19377	20023	20722	21445	22191
967	1870	2764	3651	4492	5348	6230	7071	8165	9791	11524	13341	15110	16571	17545	18337	19036	19731	20573	21346	22004	22709	23445	24216
1075	2078	3065	4043	4975	5923	6900	7841	9033	10719	12496	14409	16256	17814	18877	19763	20526	21283	22184	23014	23724	24474	25251	26062
1199	2313	3401	4483	5525	6581	7659	8694	9968	11660	13437	15362	17259	18855	19942	20828	21594	22353	23255	24087	24799	25551	26330	27145
1321	2549	3747	4943	6101	7268	8447	9576	10929	12628	14405	16332	18280	19914	21044	21939	22706	23466	24369	25202	25917	26674	27461	28285
1426	2762	4072	5381	6644	7908	9185	10417	11870	13633	15450	17418	19391	21039	22170	23065	23835	24599	25505	26338	27053	27810	28597	29421
1525	2961	4374	5784	7141	8495	9868	11205	12765	14630	16534	18611	20667	22398	23581	24521	25321	26120	27063	27930	28678	29468	30284	31131
1628	3158	4658	6155	7602	9047	10512	11943	13595	15553	17548	19714	21848	23643	24874	25855	26731	27615	28649	29608	30451	31339	32251	33187
1731	3355	4943	6530	8072	9613	11172	12696	14447	16507	18604	20851	23067	24932	26241	27303	28255	29224	30350	31400	32338	33327	34338	35366
1832	3557	5249	6943	8589	10231	11893	13525	15379	17543	19733	22064	24370	26329	27726	28874	29914	30979	32204	33348	34380	35468	36576	37693
1932	3756	5546	7338	9082	10823	12585	14318	16272	18531	20812	23234	25631	27679	29165	30404	31535	32699	34024	35267	36401	37587	38790	40009

Gambar 5.30 Nilai *integral image* dari Gambar 5.29.

Sub-citra yang sudah diketahui nilai *integral image* akan dilakukan proses *adaboost* dengan menggunakan 4 bentuk *haar feature* yang digunakan pada penelitian ini.

1. Citra positif ke-1

14951	15585	16343	17046	17672	18322	18918	19535
17185	17842	18649	19377	20023	20722	21445	22191
19036	19731	20573	21346	22004	22709	23445	24216
20526	21283	22184	23014	23724	24474	25251	26062
21594	22353	23255	24087	24799	25551	26330	27145
22706	23466	24369	25202	25917	26674	27461	28285
23835	24599	25505	26338	27053	27810	28597	29421

Gambar 5.31 Citra positif ke-1

Nilai positif pada citra tersebut dihitung nilai fiturnya, dengan;

Nilai fitur = |(total piksel hitam)-(total piksel putih)|

$h_t(x) = |(26674 - 25202 - 20722 + 19377) - (25202 - 23466 - 19377 + 17842)$

$h_t(x) = 74$

Maka nilai *error rate* yang didapatkan sebagai berikut:

$\epsilon_t = (0{,}125)|74 - 1| = 9{,}125$

2. Citra positif ke-2

13671	14332	14951	15585	16343	17046	17672	18322
15746	16513	17185	17842	18649	19377	20023	20722
17545	18337	19036	19731	20573	21346	22004	22709
18877	19763	20526	21283	22184	23014	23724	24474
19942	20828	21594	22353	23255	24087	24799	25551
21044	21939	22706	23466	24369	25202	25917	26674
22170	23065	23835	24599	25505	26338	27053	27810

Gambar 5.32 Citra positif ke-2

Nilai positif pada citra tersebut dihitung nilai fiturnya, dengan;

Nilai fitur = | (total piksel hitam) – (total piksel putih) |

$h_t(x) = |(22353 - 18877 - 19377 + 14951) - (26338 - 22353 - 24474 + 20573)$

$h_t(x) = -1034$

Maka nilai *error rate* yang didapatkan sebagai berikut:

$$\epsilon_t = (0{,}125)|-1034 - 1| = 129{,}375$$

3. Citra positif ke-3

13671	14332	14951	15585	16343	17046	17672	18322	18918	19535
15746	16513	17185	17842	18649	19377	20023	20722	21445	22191
17545	18337	19036	19731	20573	21346	22004	22709	23445	24216
18877	19763	20526	21283	22184	23014	23724	24474	25251	26062
19942	20828	21594	22353	23255	24087	24799	25551	26330	27145
21044	21939	22706	23466	24369	25202	25917	26674	27461	28285
22170	23065	23835	24599	25505	26338	27053	27810	28597	29421
23581	24521	25321	26120	27063	27930	28678	29468	30284	31131
24874	25855	26731	27615	28649	29608	30451	31339	32251	33187

Gambar 5.33 Citra positif ke-3

Nilai positif pada citra tersebut dihitung nilai fiturnya, dengan;

Nilai fitur = | (total piksel hitam) – (total piksel putih) |

$$h_t(x) = |(22353 - 18877 - 19377 + 14951) + (31339 - 26338 - 28285 + 24799) - (26338 - 24474 - 22353 + 20573)$$

$$h_t(x) = 481$$

Maka nilai *error rate* yang didapatkan sebagai berikut:

$$\epsilon_t = (0{,}125)|481 - 1| = 60$$

4. Citra positif ke-4

13478	14220	14903	15522	16167	16749	17298
15585	16343	17046	17672	18322	18918	19535
17842	18649	19377	20023	20722	21445	22191
19731	20573	21346	22004	22709	23445	24216
21283	22184	23014	23724	24474	25251	26062
22353	23255	24087	24799	25551	26330	27145
23466	24369	25202	25917	26674	27461	28285

Gambar 5.34 Citra positif ke-4

Nilai positif pada citra tersebut dihitung nilai fiturnya, dengan;

Nilai fitur = | (total piksel hitam) – (total piksel putih) |

$h_t(x) = |(28285 - 23466 - 17298 - 13478) - (24474 - 23014 - 20722 + 19277) - (24474 - 23014 - 20722 + 19277)|$

$h_t(x) = 769$

Maka nilai *error rate* yang didapatkan sebagai berikut:

$\epsilon_t = (0{,}125)|769 - 1| = 96{,}125$

Sehingga, *error rate* masing-masing citra adalah 9,125; 129,375; 60 dan 96,125. Sehingga untuk mempermudah pemilihan fitur terbaik, digunakan klasifikasi data menggunakan *error rate* terkecil yang telah didapatkan yaitu 9,125 dimana;

$$\beta_t = \frac{\epsilon_t}{1 - \epsilon_t} = \frac{9{,}125}{1 - 9{,}125} = -1{,}1231$$

Update bobot: $w_{t+1,i} = w_{t+i} . \beta_t$

Maka, bobot pada iterasi 1: $w_{2,1}$ = 0,1 25x -1,1231 = -0,1404

Sehingga, hasil akhir klasifikasi yang diharapkan pada citra positif sebagai berikut:

$$h_x = \begin{cases} 1 & \sum_{j=1}^{T} \alpha_j h_j(x) \geq \sum_{t}^{T} \alpha_t \\ 0 & Otherwise \end{cases} \quad \text{.....(5-6)}$$

dengan,

$$\alpha_j = log \frac{1}{\beta_j} \;\; \& \;\; \alpha_t = log \frac{1}{\beta_t} \quad \text{.....(5-7)}$$

Jika posisi h(x) = 1, maka citra tersebut merupakan objek

Jika posisi h(x) = 0, maka citra tersebut merupakan objek

α_j = tingkat pembelajaran citra negatif

α_t = tingkat pembelajaran citra positif

β_j = nilai bobot setelah *error rate* pada citra negatif

β_t = nilai bobot setelah *error rate* pada citra positif

h_j = *classifier* lemah citra negatif

h_t = *classifier* lemah citra positif

Untuk sub-citra dengan citra negatif sebagai berikut:

1. Citra negatif ke-1

815	1078	1317	1598	1941
981	1325	1711	2146	2586
1203	1741	2297	2845	3420
1438	2120	2802	3459	4124
1643	2433	3221	3975	4741

Gambar 5.35 Citra negatif ke-1

Nilai negatif pada citra tersebut dihitung nilai fiturnya, dengan;

Nilai fitur = | (total piksel hitam) – (total piksel putih) |

$$h_j(x) = |(3221 - 1643 - 1317 + 815) - (4741 - 3221 - 1941 + 1317)$$

$$h_j(x) = 180$$

Maka nilai *error rate* yang didapatkan sebagai berikut:

$$\epsilon_j = (0{,}25)|180 - 0| = 45$$

$$\beta_t = \frac{\epsilon_j}{1 - \epsilon_j} = \frac{45}{1 - 45} = -1{,}0227$$

Update bobot: $w_{t+1,i} = w_{t+i}.\beta_j$

Maka, bobot pada iterasi 1: $w_{2,1}$= 0,25 x (-0,9782) = -0,22556

Jika bobot setelah iterasi ke-n adalah < 0, maka iterasi berhenti. Sehingga β_j yang digunakan untuk mencari h(x) menggunakan ϵ_t pada citra negatif yaitu $\beta_j < 0$.

Maka, $h(x) = log\frac{1}{45}x180 \geq \frac{1}{2}log\frac{1}{-1{,}1231}$

$$h(x) = -297{,}578 \geq 0{,}0252$$

Karena $-297{,}578 \geq 0{,}0252$ itu bernilai salah jadi citra bukan merupakan objek

2.Citra negatif ke-2

2313	3401	4483	5525	6581
2549	3747	4943	6101	7268
2762	4072	5381	6644	7908
2961	4374	5784	7141	8495
3158	4658	6155	7602	9047

Gambar 5.36 Citra negatif ke-2

Nilai negatif pada citra tersebut dihitung nilai fiturnya, dengan;

Nilai fitur = | (total piksel hitam) – (total piksel putih) |

$$h_j(x) = |(6155 - 3158 - 4483 + 2313) - (9047 - 6155 - 6581 + 4483)$$

$$h_j(x) = 33$$

Maka nilai *error rate* yang didapatkan sebagai berikut:

$$\epsilon_j = (0{,}25)|33 - 0| = 8{,}3$$

$$\beta_j = \frac{\epsilon_j}{1 - \epsilon_j} = \frac{8{,}3}{1 - 8{,}3} = -1{,}1$$

Update bobot: $w_{t+1,i} = w_{t+i}.\beta_j$

Maka, bobot pada iterasi 1: $w_{2,1}$= 0,25 x -1,1= -0,3

Jika bobot setelah iterasi ke-n adalah < 0, maka iterasi berhenti. Sehingga β_j yang digunakan untuk mencari h(x) menggunakan ϵ_t pada citra negatif yaitu $\beta_j < 0$.

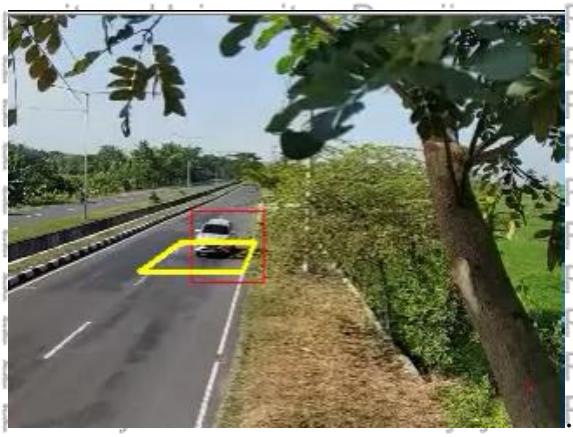
Maka, $h(x) = log\frac{1}{8{,}3}x33 \geq \frac{1}{2}log\frac{1}{-1{,}1231}$

$$h(x) = -30{,}3295 \geq 0{,}0252$$

Karena $-30{,}3295 \geq 0{,}0252$ itu bernilai salah jadi citra bukan merupakan objek

Sehingga, hasil tersebut menunjukkan jika *classifier* lemah atau citra negatif tersebut tidak memiliki ciri atau *feature* objek mobil. Jadi semakin banyak interasi yang dilakukan oleh sebuah citra positif dan negatif maka, maka semakin besar kemungkinan sebuah objek ditemukan.

Setelah dilakukan proses seperti pemilihan fitur dan klasifikasi bertingkat, maka jika suatu citra diberikan akan didapatkan sebuah hasil pendeteksian yang bisa berupa mobil atau objek lain. Jika dalam proses tersebut terdeteksi adanya mobil, maka mobil akan ditandai dengan sebuah kotak merah, seperti yang ditunjukkan pada Gambar 5.37.

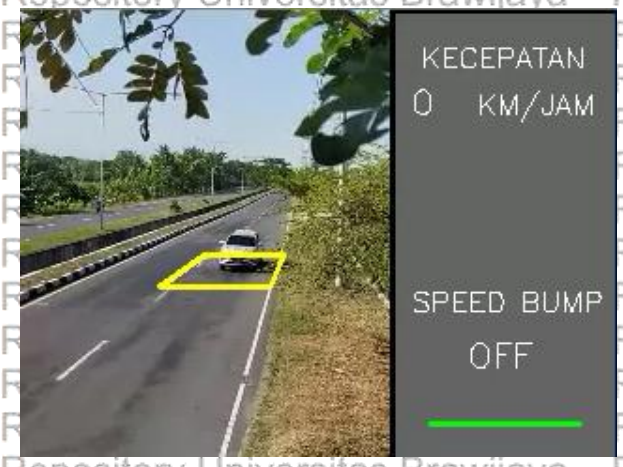


Gambar 5.37 Tampilan hasil deteksi mobil pada *ROI*

5.4. Proses Perhitungan Kecepatan

Hasil deteksi yang telah dilakukan menentukan hasil perhitungan kecepatan. Sehingga hanya objek yang terdeteksi dalam *ROI* yang akan dihitung kecepatannya. Dengan menggunakan persamaan yang telah ditentukan, parameter pertama yang diperlukan dalam perhitungan adalah jarak dari *ROI* yang diukur panjangnya pada keadaan sesungguhnya, dimana jarak *ROI* In adalah 4 meter. Parameter kedua adalah jumlah *frame* yang dihitung sejak mobil terdeteksi pada *frame* awal *ROI* sampai dengan saat mobil terdeteksi pada *frame* akhir *ROI*. Dengan kedua parameter tersebut, kecepatan mobil akan diketahui dan menjadi parameter dalam proses selanjutnya untuk mengaktifkan *speed bump*.

Gambar 5.38 menunjukkan proses perhitungan kecepatan pada kondisi mobil memasuki *frame* awal *ROI* Sedangkan Gambar 5.39 menunjukkan proses perhitungan kecepatan pada kondisi mobil akan keuar *ROI* atau memasuki *frame* akhir *ROI,* dan sisem selanjutnya akan menampilkan hasil perhitungan kecepatan mobil.



Gambar 5.38 Tampilan saat mobil memasuki *frame* awal *ROI*

Gambar 5.39 Tampilan saat mobil memasuki *frame* akhir *ROI*

5.5. Analisa Efektivitas Kerja Sistem

5.5.1. Pendeteksian Objek

Dari pengujian yang telah dilakukan, didapatkan dua bentuk hasil deteksi pada penelitian ini, yaitu hasil deteksi pada *frame* awal dan *frame* akhir *ROI*. Pengujian deteksi objek dilakukan pada beberapa jenis kendaraan yang berbeda dan memiliki intensitas warna yang berbeda yang ditunjukkan pada Tabel 5.2 dan Tabel 5.3.

Tabel 5.2 *Hasil Deteksi pada ROI In dan Out*

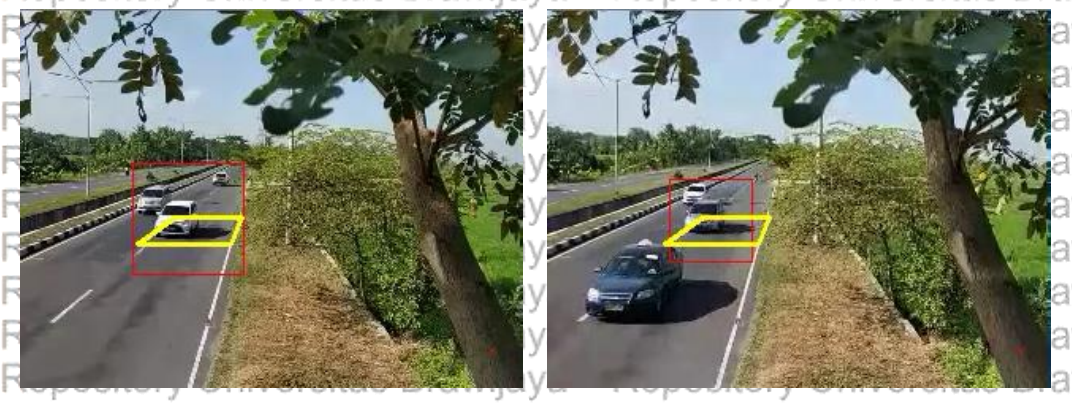
No	Warna dari mobil	Deteksi pada *ROI*	
		Frame awal	*Frame* akhir
1	Biru	terdeteksi	terdeteksi
2	Silver	terdeteksi	terdeteksi
3	Silver	terdeteksi	terdeteksi
4	Putih	terdeteksi	terdeteksi
5	Silver	terdeteksi	terdeteksi
6	Hitam	terdeteksi	terdeteksi
7	Silver	terdeteksi	terdeteksi
8	Silver	terdeteksi	terdeteksi
9	Hitam	terdeteksi	terdeteksi
10	Putih	terdeteksi	terdeteksi
11	Silver	terdeteksi	terdeteksi
12	Silver	Terdeteksi	terdeteksi
Akurasi		100%	

Hasil perhitungan akurasi rata-rata deteksi pada *frame* awal dan *frame* akhir *ROI* didapatkan sebagai berikut:

$$Akurasi = \frac{12}{12} \text{x } 100\%$$

$$Akurasi = 100\%$$

Berdasarkan tingkat akurasi, didapatkan hasil akurasi pada *frame* awal dan *frame* akhir *ROI* yaitu 100%. Hasil akurasi tersebut menunjukkan sistem deteksi menggunakan *haar cascade classifier* yang sangat baik. Hasil tersebut juga didukung oleh pendeteksian objek sangat baik pada kasus objek mobil yang tidak sepenuhnya memasuki *ROI*, seperti yang ditunjukkan pada Gambar 5.40.



(a) (b)

Gambar 5.40 Citra hasil deteksi

Selain pada kasus objek mobil yang tidak sepenuhnya memasuki area *ROI*, faktor lain yang biasanya mempengaruhi hasil deteksi yaitu faktor kemiripan warna objek yang terdeteksi dengan *background*. Diketahui dalam *preprocessing*, adanya proses *grayscale*, sehingga jika terdapat objek mobil dengan warna hitam dapat memungkina tingkat nilai piksel objek tersebut dengan *background* akan memiliki kemiripan. Namun dalam sistem ini, faktor warna tidak mempengaruhi proses pendeteksian, dan menunjukkan sistem bekerja dengan baik, seperti yang ditunjukkan pada Gambar 5.41.

(a) (b)

Gambar 5.41 Citra hasil deteksi

5.5.2. Perhitungan Kecepatan Mobil

Setelah objek mobil terdeteksi, diperlukan pengujian lebih lanjut untuk membuktikan keakuratan perhitungan kecepatan yang dihasilkan pada sistem dengan perhitungan kecepatan dari video. Namun sebelum hal tersebut dilakukan, diperlukan verifikasi hasil dari kecepatan secara *real time* dengan kecepatan pada video. Hal tersebut dilakukan dengan cara

melihat kecepatan yang ditampilkan pada speedometer pada saat mobil melintasi *ROI*. Kecepatan yang dihasilkan menjadi sumber utama kecepatan pada video dan dilakukan perbandingan kecepatan pada sistem. Pengujian dilakukan dengan menggunakan lima sampel mobil yang dihitung akurasi pendeteksiannya pada pembahasan sebelumnya dan hasil perhitungan kecepatan tersebut ditunjukkan pada Tabel 5.3.

Tabel 5.3 *Hasil perhitungan kecepatan*

Mobil	Kec. program F_t	Kec. video A_t	*Error* $A_t - F_t$	*Square of Error* (At -Ft)^2	Total	MSE
1	21	20	1	1	3	0,6
2	30	30	0	0		
3	40	40	0	0		
4	51	50	1	1		
5	59	60	1	1		

Berdasarkan hasil pengujian perhitungan kecepatan, nilai rata-rata MSE yang didapatkan yaitu 0,6. Hasil tersebut masih jauh dari kata sempurna dalam pehitungan kecepatan, dimana nilai yang sempurna yaitu nilai error yang sangat kecil yaitu mendekati 0 (nol). Hasil yang belum sempurna, disebabkan karena beberapa faktor. Jika dilihat pada Tabel 5.3, terdapat adanya perbedaan perhitungan kecepatan pada sistem dengan kecepatan pada video. Masalah umum yang terjadi yaitu terdapat proses komputasi yang berat dan mengakibatkan sistem berjalan lambat. Hal tersebut bisa terjadi karena pengujian sistem dilakukan menggunakan *device* (laptop) dengan spesifikasi *processor* Intel(R) Core i3 dan RAM 6,00 GB DDR3. Dimungkinkan sistem akan berjalan lebih baik jika diterapkan pada device dengan spesifikasi yang lebih tinggi, sehingga dapat menghasilkan perhitungan kecepatan yang sesuai dengan video secara *real time*.

BAB VI
KESIMPULAN DAN SARAN

6.1. Kesimpulan

Dari pengujian yang telah dilakukan pada penelitian ini, maka dapat disimpulkan sebagai berikut:

a. Penerapan metode *haar cascade classifier* dalam deteksi menggunakan fitur *Haar* sebagai deskriptor kemudian menggabungkan *Integral Image* dan *AdaBoost* untuk mencari dan melakukan seleksi nilai fitur dan membentuk *Cascade Classifier* yang digunakan untuk mendeteksi objek kendaraan pada citra.
b. Sistem penghitung kecepatan kendaraan dilakukan dengan menentukan *Region of Interest* (*ROI*) dari area deteksi (jalan raya) dengan menentukan titik-titik koordinat (x,y) dalam bentuk persegi panjang, sehingga mengurangi resiko kesalahan deteksi dan proses deteksi menjadi lebih fokus dan akurat.
c. Hasil penelitian didapatkan sistem bekerja secara optimal dengan tingkat akurasi deteksi yaitu 100% pada siang hari (kondisi instensitas cahaya tinggi).
d. Nilai MSE perhitungan kecepatan pada sistem yang dibandingkan dengan video didapatkan nilai MSE yang masih belum sempurna atau jauh mendekati nol yaitu 0,6, hal tersebut terjadi karena proses komputasi, sehingga dibutuhkan sistem yang berjalan pada device dengan spesifikasi yang lebih tinggi.

6.2. Saran

Penelitian yang dilakukan tentunya tidak terlepas dari kekurangan dan kelemahan. Oleh karena itu, untuk penelitian lebih lanjut, peneliti perlu memberikan saran sebagai berikut :

a. Perlunya sistem yang dapat dilakukan pada malam hari atau dengan kondisi instensitas cahaya rendah, sehingga penambahan lampu penerangan sebagai cahaya tambahan di sekitar area deteksi diperlukan untuk mendapatkan hasil deteksi yang baik.
b. Sistem deteksi tidak terbatas hanya pada kendaraan roda empat (mobil), tetapi juga untuk sepeda motor, dan perlunya pengujian menggunakan metode deteksi lainnya.

DAFTAR PUSTAKA

Bhargava, K. & Drdinesh, G. (2014). A Video Surveillance System for Speed Detection of Vehicles and Law Enforcement using Automatic Number Plate Recognition. *International Journal of Digital Application & Contemporary research on. IJDACR*:1-7.

Bradski, G. & Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*. California: O'Reilly Media, Inc.

Cheng-Yu, Ho., Huei-Yung, Lin., & Lu-Ting, Wu. (2016). Intelligent Speed Bump System With Dynamic License Plate Recognition. *International Conference on Industrial Technology (ICIT)*:1-6.

Direktorat Jenderal Bina Marga. (1997). *Manual Kapasitas Jalan Indonesia (MKJI)*. Jakarta: Departemen Pekerjaan Umum Direktorat Jenderal Bina Marga.

Hobbs, F. D. (1979). *Traffic Planning and Engineering.* Second Edition Edisi Indonesia, terjemahan Suprapto T.M. & Waldijono (1995). *Perencanaan dan Teknik Lalu Lintas*. Edisi kedua. Gadjah Mada University Press: Yogyakarta

Howse, J. (2013). *OpenCV Computer Vision with Python*. Brimingham: PACKT Publishing

Jianping, W., Zhaobin, L., Jinxiang, L., Caidong, G., Maoxin, S. & Fangyong, T. (2009). An Algorithm for Automatic Vehicle Speed Detection using Video Camera. *International Conference on Computer Science & Education*:1-4.

Keputusan Menteri Republik Indonesia. (1994). *Keputusan Menteri Perhubungan No 3 Tahun 1994 tentang Alat Pengendali dan Pengaman Pemakai Jalan*. Sekretariat Kabinet RI. Jakarta.

Lazaro, A., Joko, L, B., & Bilqis, A. (2017). Deteksi Jenis Kendaraan di Jalan Menggunakan OpenCV. *Jurnal Teknik ITS*. 6 (2):2337-3520.

Makridakis, S., & Wheelwright, S, C. (1999). Metode dan Aplikasi Peramalan. Jakarta : Binarupa Aksara

Mimbela, L, E, Y., & Lawrence A, K. (2000). *A summary of Vehicle Detection and Surveillance Technologies used in Intelligent Transportation Technologies*. The Vehicle Detector Clearinghouse, New Mexico State University. https://www.fhwa.dot.gov/ohim/tvtw/vdstits.pdf. (diakses 10 Maret 2018).

Mulkan, M, S. (2017). *Haar Like Feature Pada Viola Jones*. https://www.slideshare.net/aktis/ha feature-in-viola-jones.html (diakses 15 Maret 2018).

Munir, R. (2004). *Pengolahan Citra Digital dengan Pendekatan Algoritmik*. Bandu Informatika

Novianto, H., Ahmad, M. & Ary, S. (2014). Pembuatan Prototipe Sensor Beban Bergerak Berbasis Serat Optik Berbentuk Koil dengan Interaksi Arduino Uno dan Labview. *Prosiding Pertemuan Ilmiah XXVIII HFI Jateng & DIY*: 287-290. Yogyakarta, 26 April 2014.

Nugraha, I, S. & Muljono. (2015). Aplikasi nd*ROI*d Deteksi Mata Menggunakan Metode Viola-Jones. *Universitas Dian Nuswantoro Semarang*:1-18

Peraturan Pemerintah Republik Indonesia. (1993). *Peraturan Pemerintah Nomor 43 tahun 1993 pasal 93 tentang Prasarana dan Lalu Lintas Jalan*. Sekretariat Kabinet RI. Jakarta.

Peraturan Pemerintah Republik Indonesia. (2013). *Peraturan Pemerintah Republik Indonesia Nomor 79 Tahun 2013 Tentang Lalu Lintas dan Angkutan Jalan*. Sekretariat Kabinet RI. Jakarta.

Pignataro, L. J. (1973). *Traffic Engineering Theory and Practice*. USA: PrenticeHall, inc.

Putra, D. (2010). *Pengolahan Citra Digital*. Yogyakarta: Penerbit Andi.

Putro, M, D., Teguh, B, A. & Bondhan, W. (2012). Sistem Deteksi Wajah dengan Menggunakan Metode Viola-Jones. *Seminar Nasional "Science, Engineering and Technology"*:71-75

Sajati, Haruno. (2015). *Deteksi Obyek Menggunakan Haar Cascade Classifier*. http://jati.stta.ac.id/2015/09/deteksi-obyek-menggunakan-haar-cascade.html (diakses 15 Maret 2018).

Saputra, T. (2016). Implementasi Metode Viola Jones Dalam Sistem Absensi Siswa Berbasis Pendeteksi Wajah Studi Kasus Di SMPN 3 Rangkasbitung. *Skripsi Teknik Informatika Universitas Komputer Indonesia*. Bandung

Sudirman, D, E, J., & Nur, A, H. (2016). Traffic Monitoring : Sistem Penghitung Jumlah dan Pengukur Laju Kecepatan Kendaraan Bermotor pada Jalan Tiga Lajur Berbasis Optical Flow. *Jurnal Ilmiah Pendidikan Teknik Eelektro*: 61-66.

Sukirman, S. (1994). *Dasar-Dasar Perencanaan Geometrik Jalan Raya*. Bandung: Nova.

Sutoyo, T., Edy, M., Vincent, S., Oky, D, N. & Wijanarto. (2009). *Teori Pengolahan Citra Digital*. Yogyakarta: Penerbit Andi.

T. Ball. (2017). *Learn how to train your own OpenCV Haar classifier*. https://github.com/mrnugget/opencv-haar-classifier-training.html (diakses 20 Maret 2018).

Thadagoppula, P, K., & Vikas, U. (2016). Speed Detection using Image Processing. *International Conference on Computer, Control, Informatics and its Applications (IC3INA)*:1-6.

Undang-Undang Republik Indonesia. (2009). *Undang-undang No.22 Tahun 2009 tentang Lalu Lintas dan Angkutan Jalan*. Lembaran Negara RI Tahun 2009, No. 96. Sekretariat Negara. Jakarta

Viola, P. & Michael, J, Jones. (2004). Robust Real-Time Face Detection. *International Journal of Computer Vision*:137-154

Xiaobin, Z., Wenxiong, K., Qiuxia, Wu. (2016). Real-time vehicle detection with foregroundbased cascade classifier. *The Institution of Engineering and Technology (IET)*:1-8

Yuwono, B., Simon, P, N. & Heriyanto. (2015). Pengembangan Model Public Monitoring System menggunakan Raspberry Pi. *Jurnal Telematika*. Vol. 12 No. 02: 123-133