Deteksi Jenis Kendaraan di Jalan Menggunakan OpenCV

Alvin Lazaro, Joko Lianto Buliali, Bilqis Amaliah.
Departemen Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: lazaro.alvin13@mhs.if.its.ac.id, joko@if.its.ac.id, bilqis@its-sby.edu

***Abstrak*—Saat ini, jalan raya merupakan komponen penting dalam lalu lintas. Dalam setiap kota, tidak semua jalan dapat dilalui oleh semua jenis kendaraan. Untuk membantu memantau kondisi jalan, diperlukan suatu program yang bisa mendeteksi jenis kendaraan yang melalui jalan tersebut. Sebelumnya, sudah ada program-program serupa yang mampu mendeteksi kendaraan di jalan raya. Namun, program-program tersebut hanya melakukan deteksi kendaraan saja. Program-program tersebut tidak mampu melakukan deteksi terhadap jenis kendaraan di jalan raya. Untuk itu, Studi ini adalah pengembangan lebih lanjut dari deteksi kendaraan yaitu melakukan deteksi jenis kendaraan serta menghitung kendaraan yang terdeteksi berdasarkan jenisnya. Tujuan Studi ini yaitu untuk membuat program yang mampu mengidentifikasi jenis kendaraan pada suatu input video dan menghitung jumlah kendaraan yang terdeteksi berdasarkan jenisnya. Pengguna cukup memasukkan video rekaman lalu lintas ke program CarDetection. Nantinya, program akan memproses video yang dimasukkan dan menghasilkan file .txt sebagai keluaran. File ini berisi jenis kendaraan yang terdeteksi beserta 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.**

***Kata Kunci*—Cascade, Classifier, Jenis kendaraan, Haar-like feature, OpenCV.**

I. PENDAHULUAN

DEWASA ini, kondisi lalu lintas di jalan sudah mulai dipantau menggunakan CCTV. CCTV ini nantinya akan menghasilkan rekaman kondisi lalu lintas. Rekaman ini bisa dimanfaatkan untuk berbagai keperluan. Salah satunya adalah untuk mendeteksi jenis kendaraan yang melalui jalan tersebut.

Program akan mengenali objek kendaraan yang terdapat pada video rekaman. Objek kendaraan yang dikenali dapat digolongkan menjadi beberapa jenis. Misalkan, objek kendaraan yang terdeteksi digolongkan menjadi tiga jenis yaitu mobil kecil, mobil sedang, dan mobil besar. Nantinya, bisa dilakukan penghitungan terhadap setiap jenis kendaraan yang melewati jalan tersebut. Sehingga bisa diketahui, jumlah mobil berdasarkan jenisnya yang melalui jalan tersebut.

Pada kali ini, penulis akan menjawab implementasi kebutuhan mengenai deteksi jenis kendaraan di jalan. Input yang diperlukan adalah video. Nantinya, video ini diproses dengan bantuan library OpenCV. Proses yang dilakukan adalah menangkap frame video untuk mengenali kendaraan yang ada. Nantinya, kendaraan yang dikenali akan digolongkan menjadi tiga golongan yaitu mobil kecil, mobil sedang, dan mobil besar. Program ini dikembangkan menggunakan bahasa pemrograman Python. Dan juga, menggunakan library OpenCV untuk memproses inputan.

Tujuan Studi ini adalah membuat sebuah program yang mampu melakukan deteksi jenis kendaraan serta menghitung kendaraan yang terdeteksi berdasarkan jenisnya. Terdapat beberapa batasan masalah dalam Studi ini. Batasan masalah ini meliputi penggolongan kendaraan, resolusi kamera yang digunakan, sudut pandang objek kendaraan terhadap kamera, format file video, kondisi cuaca, serta nilai fps video yang digunakan. Kendaraan digolongkan menjadi 3 golongan yaitu kendaraan kecil, sedang, dan besar. Resolusi kamera yang digunakan minimal 3 MP. File video yang digunakan berformat MP4. Sudut pandang objek kendaraan terhadap kamera adalah tampak depan dari tengah jalan. Kondisi cuaca cerah dan siang hari. Nilai fps video yang digunakan adalah 10-15 fps.

Diskusi pada jurnal ini dibagi dalam struktur sebagai berikut: Bab II membahas teori pendukung yang digunakan dalam penelitian. Bab III membahas analisis dan perancangan sistem. Bab IV membahas hasil dan diskusi dari pengujian yang dilakukan. Terakhir, Bab V membahas mengenai kesimpulan dari penelitian serta saran untuk pengembangan di masa depan.

II. TEORI PENDUKUNG

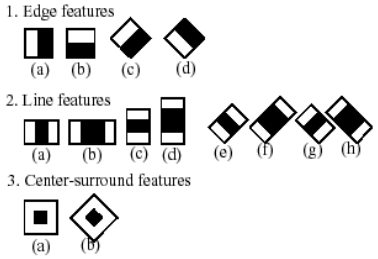
A. OpenCV

OpenCV (Open Source Computer Vision) adalah library dari fungsi pemrograman untuk realtime visi komputer [1]. OpenCV menggunakan lisensi BSD dan bersifat gratis baik untuk penggunaan akademis maupun komersial. OpenCV dapat digunakan dalam bahasa pemrograman C, C++, Python, Java, dan sebagainya [2]. OpenCV dapat digunakan pada sistem operasi Windows, Linux, Android, iOS dan Mac OS. OpenCV memiliki lebih dari 2500 algoritma yang telah dioptimalkan. Dalam penelitian ini, penulis menggunakan

OpenCV 2.4.13.

B. Haar-like Feature

Haar-like feature pertama kali diusulkan oleh Paul Viola dan Michael Jones [3] untuk keperluan mendeteksi wajah manusia [4]. Metode ini kemudian diperbaharui kembali oleh Rainer Lienhart dan Jochen Maydt [5]. Ide dari Haar-like feature adalah sebuah *classifier* yang di-training dengan sejumlah sampel citra dari suatu objek. *Classifier* di-*training* menggunakan algoritma Adaboost. Dalam kasus ini, sampel citra yang digunakan adalah citra dari kendaraan, berupa tampak samping (kiri dan kanan), depan, dan belakang. Ukuran citra yang digunakan untuk training harus sama (misalkan 20x20), di mana nantinya ini akan menjadi sampel positif. Sampel negatif adalah citra dari objek yang berbeda-beda namun tetap memililki ukuran yang sama. Kumpulan citra ini akan menghasilkan kumpulan fitur objek yang disebut sebagai *cascade*. *Classifier* akan menghasilkan nilai "1" jika pada citra yang dimasukkan terdapat objek yang dikenali dan nilai "0" jika tidak ada objek yang dikenali.



Gambar 1. Berbagai bentuk Haar-like feature

Hasil penerapan masing-masing fitur pada suatu wilayah gambar tertentu dihasilkan melalui jumlah piksel yang terletak di dalam segi empat hitam dari fitur yang dikurangkan oleh jumlah piksel yang overlap dengan segi empat putih. Segi empat ini didefinisikan lewat koordinat kiri atas x, y, lebar w, dan tinggi h. Total piksel yang berada dalam area segi empat ri direpresentasikan oleh RecSum(ri).

$$feature_1 = \sum_{i=1}^{N} W_i \times RecSum(r_i) = \sum_{i=1}^{N} W_i \times RecSum(x, y, w, h) \quad (1)$$

Di mana dalam persamaan ini, nilai N, W_i, dan r_i dipilih secara acak, tergantung objek yang diidentifikasi.

Dalam mengenali suatu objek, *cascade* akan mengabaikan area citra yang tidak memiliki objek yang memnuhi kriteria. Ini sangat membantu dalam meningkatkan performa *classifier* tersebut. Efeknya, kemungkinan kesalahan deteksi objek berkurang dan akurasi deteksi menjadi meningkat.

C. Frame Rate

Frame rate adalah frekuensi di mana sebuah gambar ditampilkan dalam suatu video atau film. Satuan yang digunakan adalah frame per second (fps). Semakin tinggi nilai fps dari sebuah video maka semakin halus pergerakan gambarnya.

D. Mean Squared Error (MSE)

Mean squared error digunakan untuk mengetahui perbedaan antara estimator dengan hasil estimasi [6]. MSE digunakan sebagai tolok ukur suatu estimator. Hasil yang diperoleh selalu berupa angka positif. Semakin mendekati nol maka semakin baik kinerja estimator tersebut.

$$MSE = \frac{\sum_{i=0}^{n}(X_i - F_i)^2}{n} \quad (2)$$

Dalam penelitian ini, MSE digunakan untuk mengecek apakah sebuah citra merupakan sebuah objek kendaraan atau bukan. Ini dilakukan dengan membandingkan kesimetrisan secara horizontal dan vertikal dari sebuah citra. Sebuah citra kendaraan bersifat simetris secara horizontal maupun vertikal. Kesimetrisan horizontal digunakan untuk mengetahui tinggi objek kendaraan sedangkan kesimetrisan vertikal digunakan untuk mengetahui apakah objek yang terdeteksi sisi kiri dan kanannya simetris atau tidak. Ini dapat mempermudah dalam menandai objek kendaraan yang terdeteksi. Jika perbedaan kesimetrisan terlalu besar, citra yang terdeteksi bukan merupakan objek kendaraan.

E. Classifier Cascade

Penelitian ini menggunakan fitur objek sebagai dasar dalam deteksi. Alasannya, konsep ini lebih cepat dibandingkan dengan menggunakan piksel. *Classifier* dikembangkan dengan menggunakan fitur-fitur objek yang akan dideteksi. Fitur-fitur ini merupakan turunan dari fungsi Haar seperti yang digunakan oleh Papageorgiu et al [7].

Dalam penelitian ini, penulis menggunakan konsep Two Rectangle Feature (TRF) [8]. Nilai TRF diperoleh melalui selisih antara jumlah piksel dalam dua wilayah segi empat pada suatu *Region of Interest* (ROI). ROI merupakan wilayah di mana objek yang akan dideteksi berada. Untuk menggunakan TRF, kedua wilayah segi empat harus berukuran sama dan bersifat simetris secara horizontal maupun vertikal.

F. Kendaraan

Menurut Kamus Besar Bahasa Indonesia (KBBI) kendaraan merupakan sesuatu yang digunakan untuk dikendarai atau dinaiki [9]. Dalam penelitian ini, penulis melakukan deteksi terhadap kendaraan mobil. Kemudian, penulis melakukan identifikasi terhadap jenis mobil yang terdeteksi. Jenis mobil yang terdeteksi akan dikelompokkan menjadi 3 golongan yaitu kendaraan kecil, kendaraan sedang, dan kendaraan besar. Hasil yang diperoleh dapat dimanfaatkan untuk mengetahui jumlah kendaraan berdasarkan jenisnya yang melewati jalan tersebut dalam suatu interval waktu tertentu. Program yang dikembangkan oleh penulis mampu mendeteksi jenis kendaraan serta menghitung jenis kendaraan yang terdeteksi berdasarkan jenisnya dalam suatu interval waktu tertentu, tergantung pada panjangnya video yang digunakan sebagai inputan.

III. ANALISIS DAN PERANCANGAN SISTEM

A. Penelitian Terkait

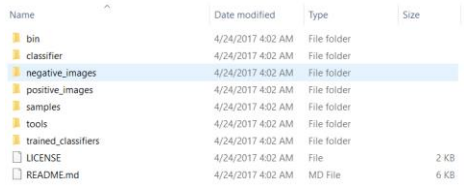
Sebenarnya sudah ada penelitian-penelitian terkait mengenai deteksi kendaraan di jalan raya. Contoh pertama adalah deteksi kendaraan secara otomatis di jalan raya oleh M. Oliveira dan

V. Santos [10] pada tahun 2008. Contoh kedua adalah deteksi kendaraan di jalan raya secara *real time* menggunakan fitur Haar oleh Han et al [11] pada tahun 2009. Kedua penelitian tersebut mampu mendeteksi objek kendaraan dengan baik. Pada tahun 2010, Sayaman Sivaraman dan Mohan Manubhai Trivedi mengembangkan suatu *framework* [12] untuk mengenali dan melakukan *tracking* objek kendaraan. Penelitian ini menghasilkan sebuah *vehicle recognizer* yang mampu mengenali objek kendaraan. Hasil yang diperoleh dari penelitian ini menunjukkan bahwa *framework* yang dibangun mampu melakukan deteksi dan *tracking* kendaraan dengan akurasi yang baik.

Semua penelitian tersebut hanya melakukan deteksi terhadap objek kendaraan. Penelitian-penelitian tersebut belum melakukan klasifikasi atas jenis kendaraan yang terdeteksi. Sehingga terobosan dari penelitian yang dilakukan oleh penulis adalah klasifikasi jenis kendaraan di jalan raya.

B. Perancangan Template

Template kendaraan (berupa file XML) digunakan untuk mendeteksi objek kendaraan pada video. Tahap pertama adalah mempersiapkan direktori dengan susunan sebagai berikut. Susunan direktori ini menggunakan contoh dari *repository* milik github mrnugget [13] (https://github.com/mrnugget/opencv-haar-classifier-training).

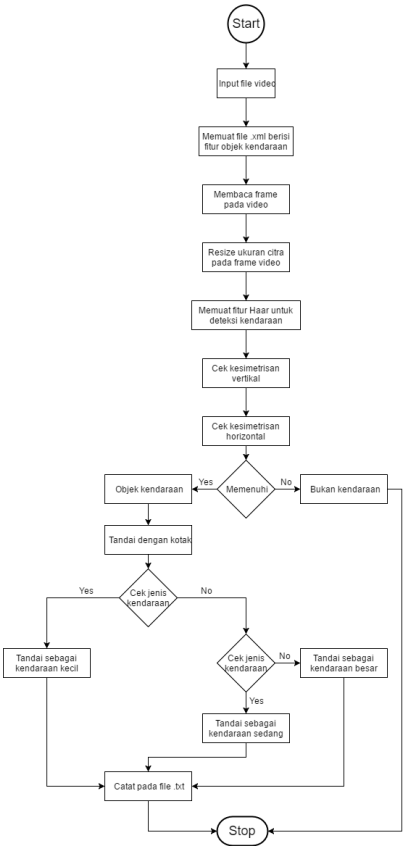


Gambar 2. Susunan direktori

Tahap berikutnya adalah menghasilkan sampel positif dan negatif dari objek kendaraan. Tahapan ini menggunakan *tools* OpenCV (opencv_createsamples) untuk menghasilkan sampel dari gambar positif yang dimasukkan. Tahap terakhir adalah melakukan *training classifier*. Proses ini memakan waktu yang cukup lama (sekitar 2 hari), tergantung spesifikasi perangkat yang digunakan. Namun, proses ini bisa dihentikan sementara atau di-restart. Hasil akhir yang diperoleh adalah sebuah file bernama classifier.xml. File inilah yang digunakan sebagai *template* untuk medeteksi objek kendaraan

C. Deskripsi Umum Sistem

Program yang dibuat pada Studi ini adalah program untuk komputer dengan nama CarDetection. Program dikembangkan dengan bahasa pemrograman Python dan OpenCV versi 2.4.13. Alur jalan sistem digambarkan pada Gambar 3.



Gambar 3. Alur jalan system

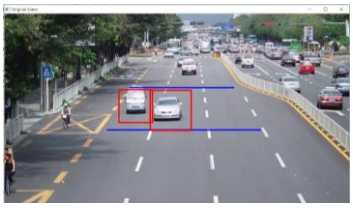
Untuk menggunakan program CarDetection, langkah pertama adalah memasukkan file video ke program. Proses ini dilakukan dengan cara menuliskan nama file video beserta ekstensinya (misal center.mp4). Kemudian program akan memuat file XML yang berisi fitur-fitur objek kendaraan. File XML ini merupakan *template* yang digunakan untuk menetapkan apakah sebuah objek merupakan kendaraan atau bukan. Lalu, program akan membaca video secara *frame-by-frame*. Pembacaan ini dilakukan pada wilayah deteksi yang telah ditetapkan yaitu di antara dua garis biru.

Gambar 4. Batas area deteksi (antara dua garis biru)

Hal ini dilakukan untuk mengurangi risiko kesalahan deteksi. Dengan demikian, pengenalan objek kendaraan akan menjadi lebih fokus. Sehingga pengenalan jenis kendaraan yang dideteksi akan lebih akurat.

Kemudian dilanjutkan dengan mendeteksi objek kendaraan menggunakan *classifier*. *Classifier* akan mencocokkan objek kendaraan file XML yang telah dimuat sebelumnya. Jika memenuhi syarat, selanjutnya akan dicek kesimetrisan objek kendaraan yang terdeteksi. Objek kendaraan yang terdeteksi akan dicek apakah simetris secara vertikal dan horizontal atau tidak. Jika simetris secara vertikal dan horizontal, objek

tersebut dianggap sebagai kendaraan. Aplikasi akan memberikan tanda segi empat pada objek yang dianggap sebagai kendaraan.



Gambar 5. Penandaan objek yang terdeteksi sebagai kendaraan

Langkah berikutnya adalah pengenalan jenis kendaraan. Pengenalan jenis kendaraan dilakukan dengan mengukur luas segi empat yang digunakan untuk menandai objek kendaraan. Tabel 1 memberikan detil ukuran luas segi empat untuk tiga jenis kendaraan yaitu kecil, sedang, dan besar. Setelah dilakukan pengenalan jenis kendaraan, aplikasi akan mencatat ID kendaraan serta jenisnya pada file .txt. Ini untuk memudahkan pengguna dalam mengetahui jenis kendaraan yang telah terdeteksi dari video yang dimasukkan. Terakhir, program menghasilkan sebuah file .txt yang berisi jenis kendaraan yang terdeteksi dari video inputan. File ini memudahkan pengguna untuk mengetahui jenis kendaraan yang terdeteksi dari video inputan beserta jumlahnya.

Tabel 1.
Luas minimal penanda tiap jenis kendaraan

Jenis Kendaraan	Luas Segi Empat (piksel)
Kendaraan kecil	≤132
Kendaraan sedang	133 – 148
Kendaraan besar	≥149

IV. HASIL DAN DISKUSI

A. *Skenario Uji Coba*

Pada bagian ini akan dijelaskan tentang skenario pengujian yang dilakukan. Uji coba dilakukan untuk mengenali jenis kendaraan melalui input video. Skenario uji coba yang digunakan adalah kondisi jalan sepi, normal, dan padat. Pada setiap skenario, penulis menggunakan 4 nilai batas waktu maksimal objek kendaraan untuk melewati wilayah deteksi yaitu 25, 30, 35, dan 40. Akurasi pengenalan jenis kendaraan dihitung menggunakan persamaan 3 yang merupakan perhitungan dengan cara sebagai berikut:

$$Acc = \frac{V_{detect}}{V_{real}} * 100\% \quad (3)$$

Di mana V_{detect} adalah banyaknya suatu jenis kendaraan yang terdeteksi dan V_{real} adalah jumlah sebenarnya dari suatu jenis kendaraan. V_{real} diperoleh dengan cara menghitung secara manual setiap jenis kendaraan.

B. *Pengujian Kondisi Jalan Sepi*

Dari pengujian yang dilakukan, maka dilakukan penghitungan akurasi dengan persamaan 3. Jumlah asli kendaraan kecil, sedang, dan besar secara berurutan adalah 1, 3, dan 1. Hasil pengujian bisa dilihat secara berturut-turut pada Tabel 2, Tabel 3, Tabel 4, dan Tabel 5. Dalam pengujian ini, terdapat kesalahan deteksi yaitu kendaraan sedang terdeteksi sebagai kendaraan besar. Hal ini disebabkan karena sudut pandang objek kendaraan terhadap kamera yang tidak pas. Sehingga objek terlihat lebih besar dari ukuran sebenarnya. Untuk kondisi jalan sepi, akurasi rata-rata adalah 77.8% dan nilai batas waktu maksimal yang ideal adalah 35 dan 40.

Tabel 2.
Hasil pengujian kondisi jalan sepi (batas waktu maksimal = 25)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraan Kecil	1	0	0	100%
Kendaraan Sedang	0	1	2	33%
Kendaraan Besar	0	0	1	100%

Tabel 3.
Hasil pengujian kondisi jalan sepi (batas waktu maksimal = 30)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraan Kecil	1	0	0	100%
Kendaraan Sedang	0	1	2	33%
Kendaraan Besar	0	0	1	100%

Tabel 4.
Hasil pengujian kondisi jalan sepi (batas waktu maksimal = 35)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraan Kecil	1	0	0	100%
Kendaraan Sedang	0	1	2	33%
Kendaraan Besar	0	0	1	100%

Tabel 5.
Hasil pengujian kondisi jalan sepi (batas waktu maksimal = 40)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraan Kecil	1	0	0	100%
Kendaraan Sedang	0	1	2	33%
Kendaraan Besar	0	0	1	100%

C. *Pengujian Kondisi Jalan Normal*

Dari pengujian yang dilakukan, maka dilakukan penghitungan akurasi dengan persamaan 3. Jumlah asli kendaraan kecil, sedang, dan besar secara berurutan adalah 10, 8, dan 4. Hasil pengujian bisa dilihat secara berturut-turut pada Tabel 6, Tabel 7, Tabel 8, dan Tabel 9. Akurasi deteksi untuk kendaraan kecil, sedang, dan besar pada batas waktu maksimal 25 dan 30 adalah 30%, 25%, dan 75%. Akurasi rata-rata untuk

batas waktu maksimal 25 dan 30 adalah 43.3%. Terdapat kesalahan deteksi pada batas waktu maksimal 25 dan 30 yaitu kendaraan sedang terdeteksi sebagai kendaraan besar. Ini disebabkan karena objek kendaraan bergerak lebih lambat dari nilai batas waktu maksimal yang ditetapkan. Dan juga, jarak antar objek kendaraan turut mempengaruhi akurasi deteksi. Pada pengujian ini, jarak antar kendaraan cenderung lebih rapat dibandingkan dengan pengujian kondisi jalan sepi. Untuk batas waktu maksimal 35 dan 40, hasil yang diperoleh untuk kendaraan kecil, sedang, dan besar secara berurutan yaitu 25%, 37.5%, dan 75%. Tidak ditemukan kesalahan kesalahan penggolongan jenis kendaraan pada batas waktu maksimal 35 dan 40. Untuk kondisi jalan normal, nilai batas waktu maksimal yang ideal adalah 35 dan 40 dengan akurasi rata-rata 47.5%.

Tabel 6.
Hasil pengujian kondisi jalan normal (batas waktu maksimal = 25)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraan Kecil	3	0	0	30%
Kendaraan Sedang	0	2	1	25%
Kendaraan Besar	0	0	3	75%

Tabel 7.
Hasil pengujian kondisi jalan normal (batas waktu maksimal = 30)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraan Kecil	3	0	0	30%
Kendaraan Sedang	0	2	1	25%
Kendaraan Besar	0	0	3	75%

Tabel 8.
Hasil pengujian kondisi jalan normal (batas waktu maksimal = 35)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraan Kecil	3	0	0	30%
Kendaraan Sedang	0	3	0	37.5%
Kendaraan Besar	0	0	3	75%

Tabel 9.
Hasil pengujian kondisi jalan normal (batas waktu maksimal = 40)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraan Kecil	3	0	0	30%
Kendaraan Sedang	0	3	0	37.5%
Kendaraan Besar	0	0	3	75%

D. Pengujian Kondisi Jalan Padat

Dari pengujian yang dilakukan, maka dilakukan penghitungan akurasi dengan persamaan 3. Hasil pengujian bisa dilihat secara berturut-turut pada Tabel 10, Tabel 11, Tabel 12, dan Tabel 13. Jumlah asli kendaraan kecil, sedang, dan besar secara berurutan adalah 9, 8, dan 8. Akurasi deteksi untuk kendaraan kecil, sedang, dan besar pada batas waktu maksimal 25 dan 30 adalah 11.1%, 25%, dan 37.5%. Akurasi rata-rata untuk batas waktu maksimal 25 dan 30 adalah 24.5%. Untuk batas waktu maksimal 35, akurasi deteksi kendaraan kecil, sedang, dan besar secara berurutan adalah 22.2%, 25%, dan 37.5%. Akurasi rata-rata untuk nilai batas waktu maksimal 35 adalah 28.2%. Untuk batas waktu maksimal 40, akurasi deteksi kendaraan kecil, sedang, dan besar secara berurutan adalah 22.2%, 12.5%, dan 37.5%. Akurasi rata-rata untuk nilai batas waktu maksimal 40 adalah 24.1%.

Dalam pengujian ini, terdapat kesalahan penggolongan jenis kendaraan yaitu kendaraan sedang terdeteksi sebagai kendaraan besar. Ada juga kendaraan kecil yang terdeteksi sebagai kendaraan besar. Dalam pengujian ini, kedua kejadian ini dipengaruhi oleh dua faktor. Faktor pertama adalah jarak antar objek kendaraan. Pada pengujian ini, jarak antar kendaraan cenderung lebih rapat dibandingkan dengan pengujian kondisi jalan sepi. Kemudian, faktor kedua adalah sudut pandang kamera. Dalam pengujian ini, terdapat mobil kecil yang terlihat seperti mobil besar. Ini disebabkan karena objek tersebut tersorot kamera dari sisi samping (tidak tepat dari tengah). Akibatnya, objek terlihat lebih besar dari ukuran sebenarnya. Untuk kondisi jalan padat, nilai batas waktu maksimal yang ideal adalah 35 dengan akurasi rata-rata 28.2%.

Tabel 10.
Hasil pengujian kondisi jalan padat (batas waktu maksimal = 25)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraan Kecil	1	0	1	11.1%
Kendaraan Sedang	0	2	1	25%
Kendaraan Besar	0	0	3	37.5%

Tabel 11.
Hasil pengujian kondisi jalan padat (batas waktu maksimal = 30)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraan Kecil	1	0	1	11.1%
Kendaraan Sedang	0	2	1	25%
Kendaraan Besar	0	0	3	37.5%

Tabel 12.
Hasil pengujian kondisi jalan padat (batas waktu maksimal = 35)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraan Kecil	1	0	1	11.1%
Kendaraan Sedang	0	2	1	25%
Kendaraan Besar	0	0	3	37.5%

Tabel 13.
Hasil pengujian kondisi jalan padat (batas waktu maksimal = 40)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraan Kecil	1	0	1	11.1%
Kendaraan Sedang	0	2	1	25%
Kendaraan Besar	0	0	3	37.5%

E. Pengujian dalam Skenario Lain

Untuk skenario lainnya, penulis mencoba memperkecil besaran wilayah deteksi yang digunakan. Wilayah deteksi diperkecil menjadi seukuran 76400 piksel, dari ukuran semula sebesar 119350 piksel. Penulis menggunakan nilai batas waktu maksimal 35 dalam skenario ini. Hasil pengujian bisa dilihat secara berturut-turut pada Tabel 14, Tabel 15, dan Tabel 16. Untuk kondisi jalan sepi, akurasi rata-rata yang diperoleh adalah 66.7%. Untuk kondisi jalan normal, akurasi rata-rata yang diperoleh adalah 15.8%. Sedangkan untuk kondisi jalan padat, akurasi rata-rata yang diperoleh adalah 12.5%. Dalam pengujian pada skenario ini, terdapat lebih banyak kesalahan deteksi dibandingkan dengan pengujian dalam skenario-skenario sebelumnya. Nilai akurasi rata-rata mengalami penurunan, jika dibandingkan dengan nilai akurasi rata-rata dengan batas waktu yang sama namun dengan besaran wilayah deteksi yang lebih besar. Dapat dilihat bahwa akurasi yang diperoleh relatif kurang baik jika dibandingkan dengan besaran wilayah deteksi senilai 76400 piksel. Sehingga, penulis memutuskan untuk menggunakan besaran wilayah deteksi senilai 119350 piksel dalam pengembangan program CarDetection ini.

Tabel 14.
Kondisi jalan sepi

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraan Kecil	1	0	0	100%
Kendaraan Sedang	0	0	1	0%
Kendaraan Besar	0	0	1	100%

Tabel 15.
Kondisi jalan normal

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraan Kecil	1	1	1	10%
Kendaraan Sedang	1	1	1	12.5%
Kendaraan Besar	0	2	1	25%

Tabel 16.
Kondisi jalan padat

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraan Kecil	0	0	0	0%
Kendaraan Sedang	0	1	2	12.5%
Kendaraan Besar	0	0	2	25%

V. KESIMPULAN

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut. Pertama, Haar-like feature dapat digunakan untuk melakukan deteksi objek kendaraan. Kedua, jenis kendaraan dapat digolongkan berdasarkan luas segi empat penanda objek kendaraan. Ketiga, batas waktu maksimal yang optimal untuk tiap objek kendaraan dalam melewati wilayah deteksi adalah 35 milidetik. Sehingga mengurangi risiko objek kendaraan yang sama terdeteksi lebih dari sekali. Keempat, tingkat akurasi rata-rata untuk tiga kondisi jalan yang berbeda (sepi, normal, padat) adalah 77.8%, 47.5%, dan 28.2%.

Saran untuk pengembangan di masa depan yakni pertama, memperkaya fitur jenis kendaraan lain pada file XML yang digunakan sebagai template untuk mendeteksi objek kendaraan. Sehingga program tetap bisa mengenali objek kendaraan jenis baru di masa depan. Kedua, penambahan kamera dari sudut pandang lain. Sehingga akurasi deteksi dapat meningkat. Ketiga, menambahkan fitur untuk menyambungkan program dengan kamera. Sehingga program mampu melakukan deteksi jenis kendaraan secara real-time. Keempat, penambahan fitur untuk menuliskan hasil deteksi ke dalam file dalam format lain (misalkan PDF atau spreadsheet).

DAFTAR PUSTAKA

[1] I. Corporation, "OpenCV," *Itseez*, 2000. [Online]. Available: http://opencv.org.

[2] I. Corporation, "http://opencv.org/platforms/," 2000. [Online]. Available: http://opencv.org/platforms/.

[3] P. V. and M. Jones, "Rapid Object Detection using A Boosted Cascade of Simple Features," in *Conference on Computer Vision and Pattern Recognition CVPR 2001*, 2001.

[4] P. V. and M. J. Jones, "Robust Real-time Face Detection," *Int. J.*

Comput. Vis., vol. 57, no. 2, pp. 137–154, 2004.

[5] R. L. and J. Maydt, "An Extended Set of Haar-like Features for Rapid Object Detection," in *International Conference on Image Processing*, 2002.

[6] and T. E. U. of I. Office for Mathematics, Science, *Mean Square Error*. 2004.

[7] C. P. and T. Poggio, "A Trainable System for Object Detection," *Int. J. Comput. Vis.*, vol. 38, no. 1, pp. 15–33, 2000.

[8] V. K. Narayanan, *Vision Based Robust Vehicle Detection and Tracking VIA Active Learning*. Gainesville: University of Florida, 2013.

[9] B. P. dan P. Bahasa, "KBBI Daring," 2016. [Online]. Available: https://kbbi.kemdikbud.go.id/entri/kendaraan.

[10] M. O. and V. Santos, "Automatic Detection of Cars in Real Roads using Haar-like Features," vol. 3810, 2008.

[11] Y. H. and H. H. S. Han, "Vehicle Detection Method using Haar-like Feature on Real Time System," *World Acad. Sci. Eng. Technol.*, vol. 455–459, 2009.

[12] S. S. and M. M. Trivedi, "A General Active-Learning Framework for on-road Vehicle Recognition and Tracking," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 2, pp. 267–276, 2010.

[13] T. Ball, "Coding Robin," 2013. [Online]. Available: http://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html.