JNATIA Volume 1, Nomor 1, November 2022
Jurnal Nasional Teknologi Informasi dan Aplikasinya

Comparative Analysis of Background Subtraction, Haar Cascade and SSD Methods in Detecting Cars

Tristan Nugraha Tanaya[a1], I Gusti Ngurah Anom Cahyadi Putra, ST., M.Cs[a2]

[12] Program Studi Informatika, Fakultas Matematika dan Ilmu Pengetahuan Alam
,Universitas Udayana
Jl. Raya Kampus UNUD, Bukit Jimbaran, Kuta Selatan, Badung, Bali
,Indonesia
[1]tristan.tanaya@gmail.com
[2]anom.cp@unud.ac.id

Abstract

This research was conducted to detect and count cars that are passing on the highway quickly. In this study, 3 methods will be applied, namely the Background Subtraction, Haar Cascade, and Single Shot Detector (SSD) methods. This study will compare the best methods for calculating cars on the highway. Based on the results of research using the Haar Cascade method, the results are quite good compared to other methods. But using these three methods the author has not gotten satisfactory results. Due to many factors that cause this comparison is not satisfactory. Starting with simple coding, poor video quality is also one of the factors that this research is unsatisfactory.

Keywords: *Background Subtraction, Haar Cascade, Single Shot Detector (SSD), Car, Comparison*

1. Introduction

Seiring meningkatnya pertumbuhan jumlah penduduk setiap tahunnnya, maka jumlah kendaraan akan semakin banyak. Di kota-kota besar sangat sering sekali terjadi kemacetan karena kendaraan yang ada sangat tidak sebanding dengan fasilitas jalan raya yang ada.pertumbuhan jumlah kendaraan akan menyebabkan kemacetan. Untuk menentukan kepadatan lalu-lintas diperlukan adanya survey untuk perhitungan jumlah kendaraan yang melintas. Survey ini dilakukan oleh seorang pengamat atau peneliti. Namun, seorang peneliti bisa salah dalam melakukan proses penghitungan dan bakal terjadi *human error.* Sehingga pelaksanaan survey kurang efisien. Berdasarkan permasalahan tersebut, maka perlu dlakukan penelitian untuk menghitung jumlah mobil yang menggunakan fasilitas jalan raya. Dalam penelitian ini penulis akan melakukan perbandingan antara metode *background subtraction, haar cascade*, dan *SSD.* Metode *background subtraction* atau juga dikenal sebagai deteksi tepi adalah suatu proses yang menghasilkan tepi-tepi dari objek-objek untuk proses segmentasi dan identifikasi objek, tujuannya adalah untuk melacak titik-titik yang dianggap sebagai tepi dari suatu objek yang mebatasi suatu wilayah objek satu dengan lainnya. Metode *haar cascade* adalah mengenali objek berdasarkan nilai sederhana dari fitur tetapi bukan merupakan nilai piksel dari image objek tersebut. Metode ini merupakan metode yang menggunakan statistical model (*classifier*) pendekatan untuk mendeteksi objek dalam gambar menggabungka tiap kunci utam yaitu *Haar like Feature, Internal Image, Adaboost learning, dan Cascade classifier*. Metode *Single Shot Detector* (SSD) yang terintegrasi dengan dataset dari MobleNet-SSD. Integrasi tersebut mampu untuk mendeteksi objek dari data input yang berasal dari hasil rekaman kamera. *Single Shot Detector* (SSD) adalah algoritma yang dapat mendeteksi objek dalam sebuah gambar atau video dan meiliki akurasi yang lebih tinggi. Penulis akan membandingkan dari ketiga metode tersebut untuk mencari yang paling efektif dan akurat untuk menghitung mobil yang berada di jalan raya. Penelitian ini dibantu dengan menggunakan aplikasi jupyter notebook dan menghasilkan proses penghitungan mobil.

2. Reseach Methods

2.1. OpenCV

OpenCV (*Open Source Computer Vision Library*) adalah library dari fungsi pemrograman untuk realtime visi computer [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]. Di OpenCV, kata Open dipahami sebagai open source gratis, tidak berbayar, dan siapa pun dapat mengunduhnya sedangkan CV sendiri merupakan singkatan dari Computer Vision yang artinya computer digunakan untuk mengolah gambar yang diambil oleh alat perekam. Tujuan OpenCV ini diantaranya untuk memperbaiki kualitas gambar atau untuk mengindentifikasi gambar.[3]

Figure 1. OpenCV

2.2. Python

Phyton adalah Bahasa pemrograman yang popular di seluruh dunia yang digunakan untuk mengembangkan situs web dan algoritma. Bahasa pemrograman python membuat program apapun lebih sederhana daripada Bahasa pemrograman yang lainnya.

2.3. Algoritma Background Substraction

Langkah-langkah proses dalam *algoritma background subtraction* untuk mendeteksi objek

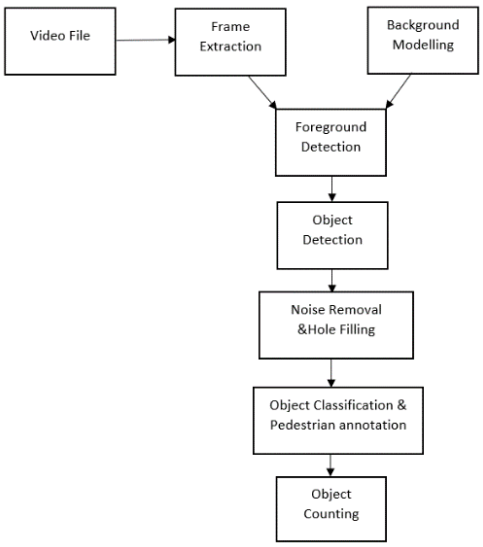

Figure 2. Algoritma Background Substraction

2.3.1. Pre-Processing

Dalam tahap ini dilakukan proses pengubahan data mentah yang diambil dari kamera menjadi bentuk yang dapat dimengerti oleh komputer. Dalam tahap ini juga dilakukan noise removal dan eliminasi objek kecil pada gambar [4].

2.3.2. Background Modelling

Latar Belakang yang digunakan dalam proses ini merupakan latar belakang yang terdapat di video. Citra latar belakang yang diambil adalah latar belakang yang tidak ada objek dan dilakukan secara manual. latar belakang yang diambil ini bersifat statis yaitu tidak berpindah tempat maupun bergerak. Penerapan teknik statis ini lebih efisien dibandingkan dengan penerapan teknik adaptif.[5]

2.3.3. Foreground Detection

Proses ekstraksi foreground dari background dilakukan dalam tahap ini. proses ini memiliki persamaan matematis yaitu:

$$R_{r,c} = I_{r,c} - B_{r,c} \quad (1)$$

Ket
R = hasil
I = gambar saat ini
B = background model
r = baris
c = kolom

Nilai R lalu dibandingkan dengan nilai *threshold* yang telah ditentukan, jika lebih besar dari nilai *threshold* maka piksel di I(r,c) dapat dianggap berbeda dengan piksel di B(r,c).[4]

2.3.4. Object Detection

Tahap ini yaitu mendeteksi dan menemukan adanya pergerakan dari objek di setiap frame. Tahap ini memakai metode *Kalman filter* dan algoritma *blob analysis*. *Kalman filter* (KF) adalah suatu metode estimasi keadaan yang dapat diimplementasikan pada model dinamik linear saja.[6] *Blob analysis* adalah pendeteksi pixel-pixel yang memiliki warna sama dibandingkan latar belakangnya. Pendeteksi ini bertujuan untuk mendeteksi *low-level* di suatu objek dua diemnsi maupun tiga dimensi. [7]

2.3.5. Noise removal dan Hole Filling

Dilakukan proses *noise removal* dan *hole filling* pada citra. Citra hasil perlu dilakukan *noise removal* agar lebih baik. Selanjutnya hasil citra juga dilakukan proses *filling* karena masih terdapat banyak lubang *hole. Filling* dilakukan dengan menutupi lubang-lubang kecil pada citra agar citra menjadi lebih halus. Proses *noise removal* dan *hole filling* ini menggunakan Operasi Morfologi.[4]

2.3.6. Object Classification dan Pedestrian Annotation

Tahap ini merupakan tahap untuk membedakan dimana objek dan dimana yang bukan objek. Pada tahap ini dilakukan seleksi pada objek dengan melihat ukuran setiap objek. Ukuran tersebut didapat dari mendeteksi *boundary* dari setiap objek yang terdeteksi. *Boundary* pada objek bertujuan untuk mendeteksi 1 objek serta beberapa objek yang berdekatan dengan lebih akurat. Setelah melalui tahap klasifikasi objek, objek akan dibingkai dengan kotak berwarna hijau. Kotak tersebut didapat dari nilai garis batas terluar dari objek. [4]

2.3.7. Object Counting

Pada tahap akhir yaitu menghitung berapa jumlah objek yang terdeteksi. Jumlah tersebut didapat dari objek yang melewati garis.

2.4. Contour

Contour adalah sebuah list yang berisi point yang dapat dikatakan mewakili dalam suatu curva dari sebuah gambar. *Contour* digambarkan dalam *OpenCV* sebagai urutan (*sequence*) informasi yang dikodekan tentang lokasi dari point berikutnya dalam kurva. Fungsi yang ada pada *OpenCV*, menghitung *contour* dari gambar biner. Gambar biner dapat dihasilkan dari suatu *threshold* yang memiliki sudut yang implisit sebagai batas antara area yang positif dan negatif. [8]

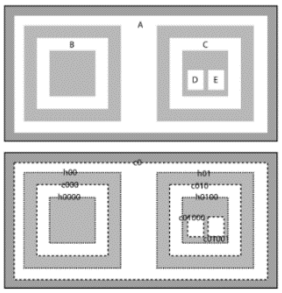

Figure 3. Contour pada OpenCV

Beberapa tahapan yang harus dilalui apabila ingin mendapatkan contour dari gambar RGB. Tahapannya adalah :

1. Membalikkan warna citra (*negative color*)
2. Membuat gambar menjadi citra keabuan (*gray scale*)
3. *Thresholding* dengan *threshold binary*. Operasi *thresholding* dapat dihitung dengan

$$dst(x,y) = \begin{cases} maxVal, & if\ src(x,y) > thres \\ 0, & otherwise \end{cases} \quad (2)$$

4. Menentukan *contour* dari gambar

2.5. Haar Cascade Classifier

Haar like feature atau yang dikenal sebagai *Haar Cascade Classifier* merupakan *rectangular feature*, yang memberikan indikasi secara spesifik pada sebuah gambar atau image. Metode ini berasal dari gagasan Paul Viola dan Michael Jhon, karena itu dinamakan metode Viola & Jhon. Ide dari Haar like feature adalah mengenali obyek berdasarkan nilai sederhana dari fitur tetapi bukan merupakan nilai piksel dari image obyek tersebut. Metode ini memiliki kelebihan yaitu komputasi yang sangat cepat, karena hanya tergantung pada jumlah piksel dalam persegi bukan setiap nilai piksel dari sebuah image. Metode ini merupakan metode yang menggunakan statistikal model (*classifier*). [9]

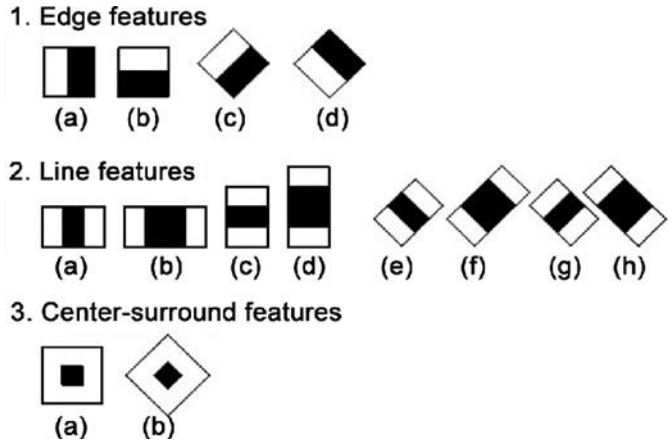

Figure 4. Haar Like Features

2.6. Single Shot Detector

Single Shot Detector (SSD) adalah sebuah metode untuk mengenali atau mendeteksi sebuah objek pada suatu gambar dengan menggunakan *single deep neural network* dan salah satu algoritma deteksi objek yang paling populer karena kemudahan implementasi, serta akurasi yang baik relatif terhadap komputasi yang dibutuhkan [10]. Metode *Single Shot Detector* (SSD) ini termasuk kedalam deteksi object secara *real time*. Arstitektur SSD termasuk kedalam jenis *Convolutation Neural Network* (CNN) [11], yang merupakan salah satu jenis *Neural Network* yang biasa digunakan pada data image. Arsitektur dari CNN dibagi menjadi 2 bagian besar, *Feature Extraction Layer* dan *Convolutional Layer*. Dimana pada bagian *Feature Extraction Layer* ini adalah melakukan *encoding* dari sebuah image menjadi *features* yang merepresentasikan gambar tersebut. Sedangkan bagian *Convolutional Layer* terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi (*pixels*).
Dataset pada penelitian menggunakan model pendeteksian objek *SSDLite-MobileNet-v2* terkuantisasi yang dilatih dari *set data Common Object in Context Detection Challenge* (COCO) [11] dan dikonversi yang telah disimpan pada ruang penyimpan *Raspberry Pi*. Dan dataset tersebut digunakan untuk klasifikasi objek. Dataset ini telah meliputi 90 jenis objek, mulai dari manusia, hewan (kucing, anjing, kuda, dsb.), serta benda (mobil, kursi, meja, dsb.)
Secara umum metode *Single Shot Detector* (SSD) mempunyai sebuah rumus sederhana dalam menentukan *default boxes* dan *scale default boxes*.
Untuk menentukan *default boxes*

$$L(x,c,l,g) = \frac{1}{N}\left(L_{conf}(x,c) + \alpha L_{loc}(x.l,g)\right) \quad (3)$$

Untuk menentukan *scale default boxes*

$$S_k = S_{min} + \frac{S_{max}-S_{min}}{m-1}(k-1), k \in [1,m] \quad (4)$$

3. Result and Discussion

3.1. Algoritma Background Substraction

Algoritma Background Substraction mendeteksi objek mobil dengan cara membandingkan gambar yang memiliki objek dengan gambar latar belakang yang tidak memiliki objek. Video rekaman dimasukkan ke syntax dan diproses untuk mendeteksi mobil. Dalam algoritma ini mendeteksi semua gerakan yang ada di dalam rekaman tersebut menjadi contour dapat dilihat di gambar no. 5. Namun, tidak semua gerakan terhitung sebagai mobil karena hanya objek yang melewati garis biru yang akan terdeteksi sebagai mobil dan akan terhitung pada

system. Objek yang bergerak tersebut akan dikelilingi dengan box berwarna hijau dan ditengah box hijau terdapat titik merah dapat dilihat di gambar no. 6. System akan menghitung objek itu sebagai mobil jika titik merah yang berada di tengah-tengah box berwarna hijau melewati garis biru. Hasil yang didapatkan adalah 325 mobil.

Figure 5. Background Subtraction

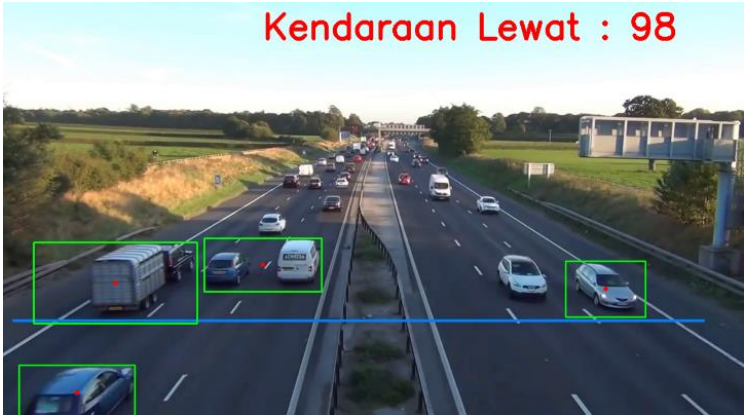

Figure 6. Video Background Subtraction

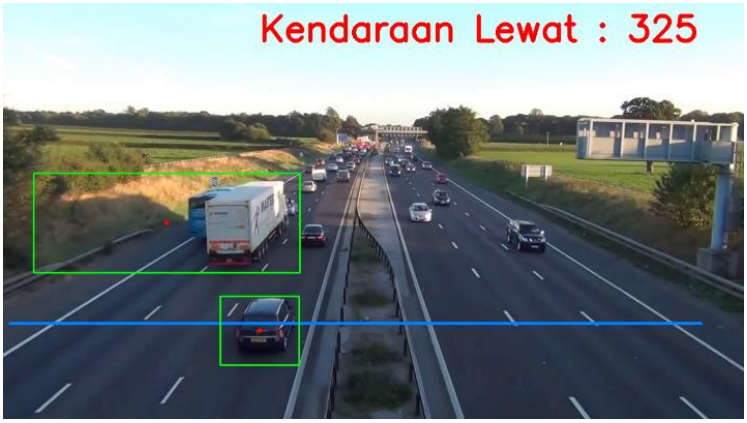

Figure 7. Hasil Background Subtraction

3.2. Haar Cascade Classification

Pada tahapan penelitian ini dimulai dengan mamasukkan file cars.xml kedalam syntax. Tahap selanjutnya memasukkan video original ke dalam syntax. Pada proses pendeteksian objek mobil dengan menggunakan metode Haar Cascade Classifier, ada beberapa proses yang dilakukan sebelum akhirnya akan menghasilkan sebuah output objek mobil yang

terdeteksi pada sebuah video. Sama seperti metode sebelumnya dalam menghitung mobil yang berada dalam video tersebut. Mobil akan dikelilingi box berwarna hijau dan ditengah-tengah box tersebut terdapat titik merah yang menandakan bahwa yang dikelilingi box hijau tersebut adalah objek mobil dapat dilihat di gambar no. 8. Penghitungan objek mobil juga sama seperti metode sebelumnya jika titik merah yang ada didalam box berwarna hijau itu melewati garis biru maka mobil akan terhitung ke dalam system. Namun, penulis melihat kekurangan dari metode ini bisa dilihat di gambar 8. Objek yang bukan mobil tetap terdeteksi sebagai mobil. Dikarenakan kualitas video yang kurang bagus. Kemungkinan system dapat mengira bahwa objek tersebut adalah mobil yang terparkir atau objek yang menyerupai mobil. Hasil yang didapatkan adalah 171 mobil.

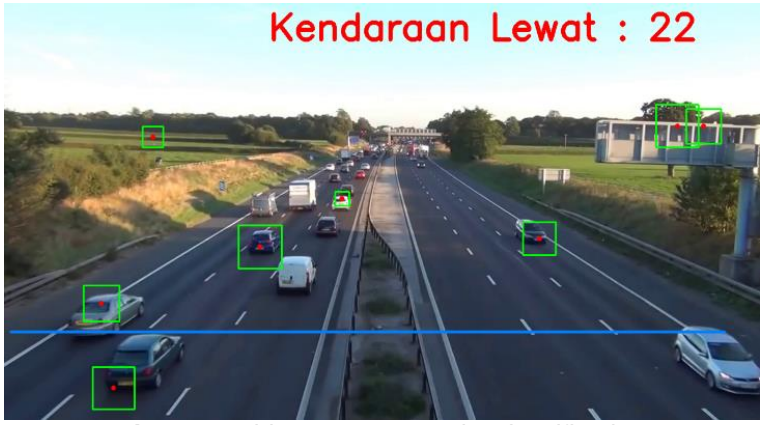

Figure 8. Video Haar Cascade Classification

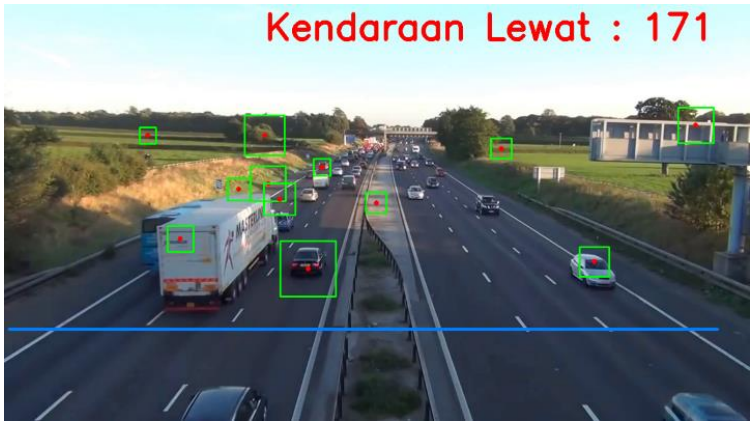

Figure 9. Hasil Haar Cascade Classification

3.3. Single Shot Detector

Pada metode ini diperlukannya MobileNetSSD untuk mendeteksi beberapa objek. Penulis disini hanya menggunakan klasifikasi objek terhadap mobil. System akan mendeteksi mobil jika mobil yang ada dalam sebuah video memliki ciri-ciri yang sama didalam model MobileNetSSD. Dalam mendeteksi mobil sama seperti metode sebelumnya. Jika objek yang dideteksi adalah mobil maka mobil akan dikelilingi box berwarna hijau dan ditengah-tengah box berwarna hijau itu terdapat titik merah dapat dilihat di gambar no. 10. Dalam penghitungan objek mobil juga sama seperti penghitungan metode sebelumnya jika titik merah tersebut melewati garis biru maka mobil akan terhitung ke dalam system. Namun kekurangan dari metode ini kurangnya bisa mendeteksi jika objek berada jauh dari kamera. 1 Objek mobil juga dapat terhitung 2 kali atau lebih. Sehingga hasil yang dihasilkan lebih dari video originalnya dapat dilihat di gambar 11. Hasil yang didapatkan adalah 320 mobil.

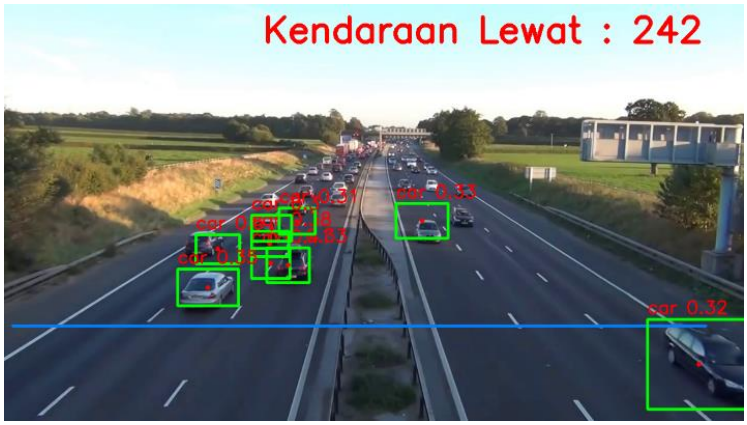

Figure 10. Video Single Shot Detetctor

Figure 11. Hasil Single Shot Detetctor

3.4. Video Original

Dalam video original penulis menghitung manual mobil yang terdapat di dalam 1 menit video tersebut. Hasil yang didapat dari menghitung mobil dengan manual adalah 158 mobil. Bisa dilihat di gambar no. 12.

Figure 12. Video Original

4. Result

Berdasarkan penelitian yang dilakukan yaitu menghitung banyak mobil yang berlalu-lintas dengan menggunakan 3 metode yaitu metode Background Substraction, metode Haar Cascade Classification, dan SSD. Pengujian menggunakan input video yang dilakukan disetiap metodenya. Namun disetiap metode menghasilkan hasil yang berbeda-beda. Metode Background Substraction mendapat nilai prediksi yang jauh lebih tinggi dari video originalnya yaitu 205,69%. Banyak objek yang terhitung 2 kali atau lebih menyebabkan hasilnya jauh melewati nilai prediksi yang diharapkan. Metode Haar Cascade Classification mendapat nilai prediksi 108,22%. Metode ini sangat banyak mendeteksi sebagai objek mobil yang seharusnya itu bukan objek mobil ini dikarenakan kualitas video yang kurang bagus. Metode SSD mendapatkan nilai prediksi yang hampir sama dengan metode Background Substraction yaitu 202,53%. Karena menggunakan model dari MobileNet-SSD, 1 mobil yang terkena cahaya saat jalan dan saat melewati garis untuk menghitung ke system tidak terkena cahaya. Mobil dapat terhitung 2 kali atau lebih. Menurut hasil yang didapatkan penulis menyimpulkan bahwa metode Haar Cascade Classification lebih efisien dan akurat dibandingkan metode Background Subtraction dan metode Single Shot Detector (SSD) dalam menghitung mobil yang berlalu-lintas.

References

[1] I. Corporation, “OpenCV”, 2000. [Online]. Available : https://opencv.org/about/

[2] Alvin. Lazaro, Joko. Lianto. Buliali and Bilqis. Amaliah, “Deteksi Jenis Kendaraan di Jalan Menggunakan OpenCV” Jurnal Teknik ITS, vol. 6, no. 2, p. A293, 2017.

[3] Srimulia, “Mengenal OpenCV Dalam Python: Pengertian, Sejarah, Dukungan pada OS, Fitur-fitur”, 31 August 2022. [Online]. Available: https://idmetafora.com/news/read/1177/Mengenal-OpenCV-Dalam-Python-Pengertian-Sejarah-Dukungan-pada-OS-Fitur-fitur.html

[4] Karina. Kaloh, Vecky. C. Poekoel, Muhamad. Dwisnanto. Putro, “Perbandingan Algoritma *Background Subtraction* dan *Optical Flow* Untuk Deteksi Manusia” E-Journal Teknik Informatika, vol. 13, no. 1, p. 2, 2018.

[5] Achmad. Solichin, Agus. Harjoko. “Metode *Background Subtraction* untuk Deteksi Obyek Pejalan Kaki pada Lingkungan Statis” in Seminar Nasional Aplikasi Teknologi Informasi (SNATI), Yogyakarta, 2013, pp. B-3.

[6] Tria. Nugrahini, “Perbandingan Metode *Kalman Filter* dan Metode *Ensemble Kalman Filter* Dalam Mendeteksi Gangguan Konduksi Panas Pada Batang Logam”, Universitas Jember, 2012.

[7] Deny. Nugroho. Triwibowo, Ema. Utami, Sukoco, “Analisis *Blob Detection* Pada Pendeteksian dan Perhitungan Kendaraan di Jalan Tol” Jurnal Teknologi Informasi dan Komunikasi, vol. 10, no. 1, p. 3, 2020.

[8] Muhammad. Syarif, Wijanarto., “Deteksi Kedipan Mata dengan Haar Cascade Classifier dan Contour untuk Password Login Sistem” *Techno COM,* vol. 14, no. 4, p. 244-246, 2015

[9] RD. Kusumanto, Wahyu. S. Pambudi, Alan. N. Tompunnu, “Aplikasi Sensor Visison untuk Deteksi MultiFace dan Menghitung Jumlah Orang” in Seminar Nasional Teknologi Informasi & Komunikasi Terapan 2012 (Semantik 2012), Universitas Internasional Batam, 2012.

[10] Liu. Wei, Dragomir. Angueloy, Dumitru. Erhan, “SSD: Single Shot Multibox Detector” in *European Conference on Computer Vision,* Springer, 2016 pp. 2-3.

[11] Yosia. Pradeska. Admaja, “ Sistem Penghitung Jumlah Pengunjung di Restoran Menggunakan Kamera Berbasis *Single Shot Detector* (SSD)”, Universitas Dinamika, 2021.