PENERAPAN METODE *GAUSSIAN BLUR DAN ABSOLUTE DIFFERENCE* PADA JUMLAH DAN KECEPATAN KENDARAAN

Alfian Firdaus[1], Imelda[2]
[1,2]*Program Studi Anda, Fakultas Teknologi Informasi, Universitas Budi Luhur*
Jl. Raya Ciledug, Petukangan Utara, Kebayoran Lama, Jakarta Selatan 12260
Telp. (021) 5853753, Fax. (021) 5866369
E-mail : alfianfirdaus007@gmail.com[1], imelda@budiluhur.ac.id[2]

ABSTRAK

Perkembangan teknologi pada bidang transportasi darat di lalu lintas semakin pesat. UP SPLL Dinas Perhubungan DKI Jakarta sedang melakukan beberapa pengembangan teknologi yang diharapkan dapat menurunkan tingkat kecelakaan lalu lintas. Salah satu faktor penyebab kecelakaan lalu lintas adalah pengguna kendaraan yang membawa kendaraan melebihi kecepatan yang telah ditentukan. Untuk menyelesaikan masalah, dibuat sebuah aplikasi yang dapat menghitung jumlah dan kecepatan kendaraan melalui rekaman video CCTV. Proses deteksi kendaraan melalui rekaman video diawali dari proses Preprocessing dimana pada proses ini dilakukan pengaturan untuk mengatur Rest Of Interest (ROI) untuk garis awal dan garis akhir yang menjadi batas wilayah perhitungan. Tahap selanjutnya adalah proses segmentasi dengan menggunakan metode Gaussian Blur, Absolute Difference, Morfologi Erode dan Dilate , dan metode Blob Detection. Pada tahap selanjutnya dilakukan ekstraksi ciri. Setelah itu dilakukan perhitungan jumlah kendaraan untuk mendapat urutan kendaraan. Tahap terakhir adalah menghitung kecepatan kendaraan. Dari proses perhitungan ditampilkan setiap informasi objek Blob yang terdeteksi berupa jarak, waktu, kecepatan pada setiap kendaraan yang terdeteksi. Kontribusi pada penelitian ini adalah perhitungan jumlah kendaraan dan kecepatan kendaraan dengan menggunakan metode Gaussian Blur, Absolute Difference, dan Blob Detection. Pengujian aplikasi menggunakan contoh rekaman CCTV dan rekaman HandPhone. Hasil pengujian menunjukkan bahwa akurasi tertinggi untuk perhitungan jumlah kendaraan adalah 100%. Selisih terkecil antara kecepatan sebenarnya dan kecepatan uji coba adalah 5,11%. Dengan menggunakan metode Blob Detection diharapkan aplikasi ini dapat membantu Dinas Perhubungan DKI Jakarta melakukan perhitungan jumlah dan kecepatan kendaraan melalui rekaman CCTV.

Kata kunci : CCTV, *Gaussian Blur*, *Absolute Difference*, *Blob Detection*, Jumlah Kendaraan, Kecepatan kendaraan.

1. PENDAHULUAN

Perkembangan teknologi saat ini sudah mempengaruhi berbagai aspek kehidupan. Indonesia sebagai negara berkembang sedang mempersiapkan diri untuk menatap dunia teknlogi yang terus berkembang. Salah satu contoh, Indonesia sedang mencoba menerapkan teknologi pada lalu lintas. Unit Pengelola Sistem Pengaturan Lalu Lintas (UP SPLL) DKI Jakarta sedang berusaha mengembangkan teknologi dengan ingin membuat sistem yang dapat digunakan untuk mengurangi pelanggaran lalu lintas di Ibukota DKI Jakarta.

Harapan pihak TMC dan UP SPLL adalah CCTV lalu lintas yang ada saat ini dapat menunjang teknologi yang ingin di terapkan dan ada beberapa aplikasi yang dibuat untuk menunjang petugas memantau atau mengatur lalu lintas. Masalah yang ada saat ini adalah masalah kecelakaan lalu lintas yang diakibatkan oleh pengguna jalan yang menggunakan kendaraan dengan kecepatan diatas aturan yang ada.

Pada penelitian ini, dikembangkan suatu cara untuk mengatasi masalah tersebut dengan membuat sebuah program atau aplikasi yang dapat menghitung jumlah kendaraan dan kecepatan kendaraan melalui CCTV lalu lintas.

Pada penelitian Danang Wahyu Wicaksono dan Budi Setiyono, 2017 [1] membahas tentang cara Mengetahui estimasi kecepatan kendaraan dengan menggunakan pengolahan citra dari data video dan metode jarak *Euclidean* dengan berbagai sudut kamera. Sistem ini mampu mengestimasi kecepatan kendaraan yang bergerak dengan tingkat akurasi terendah yaitu 87,01% dan keakuratan tertinggi adalah 99,38%. Persyaratan Indeks-Pengolahan citra digital, kendaraan bergerak, estimasi kecepatan. Yang belum diselesaikan penelitian ini adalah menghitung jumlah kendaraan yang terekam kamera.

Pada penelitian Muhammad Rizki Muliawan dkk., 2015 [2] membahas tentang pembuatan aplikasi

absensi dengan pengenalan wajah menggunakan algoritma *Eigenface* yang terdapat pada *OpenCv*. Penelitian ini menjelaskan kerja *OpenCv* dapat mendeteksi objek dalam gambar. Hasil yang didapat berbeda-beda antara wajah satu dengan wajah yang lainnya, pada saat database berisi 10 data wajah, hasil rata-rata persentase kecocokan mencapai 88%, sedangkan pada saat database berjumlah 20 data wajah, hasil rata-rata persentase kecocokan mencapai 52%. Penyebab dari perbedaan hasil tersebut adalah karena faktor pencahayaan, jarak, bentuk wajah, serta jumlah data yang tersedia. Pada penelitian ini belum menggunakan metode olah citra untuk mendukung akurasi deteksi gambar.

Pada penelitian Ferhat Bozkurt dkk., 2015 [3] membahas tentang Penerapan *Gaussian Blur* untuk melakukan teknik pengolahan gambar. Penelitian ini ini menunjukkan bahwa pesanan dengan kecepatan tinggi dapat dicapai dengan menggunakan metode ini pada arsitektur CUDA dengan bantuan paralelisme yang tinggi. Penelitian ini belum memadukan metode *Gaussian Blur* dengan metode lain untuk menghasilkan objek yang kurang baik untuk dideteksi.

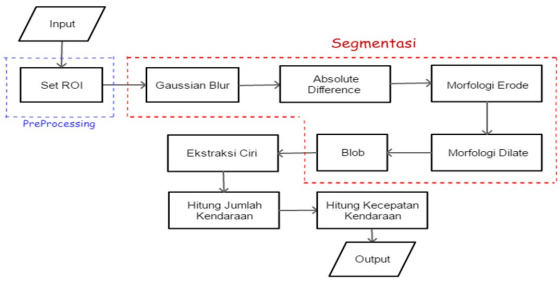
Pada penelitian Jyotsna Tripathi dkk., 2015 [4] membahas tentang Penggunaan Berbagai teknik pengolahan citra seperti diferensiasi *frame*, segmentasi, penyaringan, operasi morfologi, deteksi gumpalan, dll digunakan dengan menggunakan perangkat lunak MATLAB. Penelitian ini dapat menghitung dan mengklasifikasi kendaraan yang terekam kamera. Aplikasi ini belum mampu menghitung kecepatan kendaraan yang terekam kamera.

Pada penelitian Hardy Santosa Sundoro dan Agus Harjoko , 2015 [5] membahas tentang Metode penguraian latar belakang. Metode deteksi *Blob* memberikan koordinat dalam bentuk *Centroid* sehingga akan menunjukkan pergerakan kendaraan. Keakuratan penghitungan kendaraan yang didapat pada kondisi pagi hari adalah 75,69%. Keakuratan penghitungan kendaraan yang dilakukan pada sore hari adalah 90,50%. Akurasi menghitung kendaraan diperoleh 85,31% di malam hari. Nilai ketidakpastian sekitar kondisi tersebut kira-kira ± 3 km / jam. Pada penelitian ini belum menggabungkan metode *Gaussian Blur* dan *Absolute Difference* untuk mendapatkan akurasi perhitungan jumlah kendaraan yang sempurna.

Namun, pada penelitian ini aplikasi yang dibuat jika dibandingkan dengan beberapa jurnal sebelumnya adalah aplikasi ini menggunakan metode *Gaussian Blur, Absolute Difference,* dan *Blob Detection*. Pada penelitian sebelumnya tidak menemukan penelitian yang menggunakan *Gaussian Blur* dan *Absolute Difference* secara bersamaan dalam *Blob Detection* untuk mendeteksi objek pada gambar yang bergerak. Penggunaan *Gaussian Blur* dan *Absolute Difference* secara bersamaan akan menghasilkan gambar deteksi yang lebih baik sehingga *Blob Detection* dapat dengan mudah menjadikan gambar deteksi menjadi sebuah objek *blob*. Hasil dari penelitian ini menghasilkan perhitungan yang cepat dan akurasi yang tinggi tanpa mengeluarkan banyak biaya untuk pemrosesan aplikasi ini.

2. PERHITUNGAN JUMLAH DAN KECEPATAN KENDARAAN

Metode yang digunakan dalam proses perhitungan jumlah dan kecepatan kendaraan adalah metode *Gaussian Blur, Absolute Difference,* Morfologi *Erode* dan *Dilate*, dan *Blob Detection.* Gambar 1 adalah urutan proses dan metode yang digunakan.



Gambar 1. Model Perhitungan Jumlah dan Kecepatan Kendaraan

2.1. Input

Input aplikasi adalah berkas rekaman sebuah CCTV lalu lintas yang berekstensi .Mp4. Terdapat video yang diambil dari data CCTV Dinas Perhubungan DKI Jakarta dan rekaman *HandPhone*. Video Input adalah video rekaman kendaraan yang melintas dari arah belakang kamera ke arah depan.

2.2. Proses

Pada tahap ini, dijelaskan bertahap langkah-langkah yang dilakukan sistem, dimulai dari tahap *Preprocessing,* segmentasi, ekstraksi ciri, perhitungan jumlah kendaraan, perhitungan kecepatan kendaraan, dan *Output.*

2.2.1. Preprocessing

Melakukan Normalisasi *Frame Rate* sesuai kebutuhan Untuk Perhitungan dalam aplikasi hasil rekaman CCTV tersebut akan diubah menjadi Ekstensi file : .mp4, *Frame Width* = 1280, *Frame Height* = 720, Data *Rate* =1730 kbps, Total *BitRate* = 1730 kbps,

dan *Frame Rate* = 10 dan 15 *Fps.* Setelah Normalisasi *frame Raet.*

Pada tahap ini juga dilakukan pengaturan terhadap posisi garis awal (garis berwarna merah), garis akhir (garis berwarna biru), dan perkiraan jarak sesungguhnya antara garis awal dan garis akhir dalam gambar yang telah ditentukan.

Gambar 2. Penentuan Jarak Sebenarnya

Gambar 2 adalah contoh penentuan perkiraan jarak antara garis awal dan garis akhir menggunakan garis marka jalan yang ada dalam gambar berdasarkan peraturan pemerintah [6] yang menyebutkan bahwa panjang masing-masing garis pada garis putus-putus harus sama, berdasarkan kecepatan rencana, jika kurang dari 60 km per jam, panjang garis putus-putus 3,0 meter sedangkan jika 60 km per jam atau lebih, panjang garis putus-putus 5,0 meter. Panjang celah diantara garis putus-putus harus sama, berdasarkan kecepatan rencana, jika kurang dari 60 km perjam, panjang celah garis putus-putus 5,0 meter sedangkan jika 60 km perjam atau lebih, panjang celah garis putus-putus 8,0 meter.

2.2.2. Segmentasi

Fungsi segmentasi adalah untuk memisahkan antara *background* dan *foreground,* segmentasi juga berguna untuk memisahkan objek yang satu dengan objek yang lain. Metode yang digunakan adalah *Gaussian Blur* dan *Absolute Difference*. Alasan pemilihan metode ini adalah untuk meningkatkan akurasi deteksi agar mendapat nilai akurasi perhitungan yang tinggi [3][7] . Segmentasi juga melakukan Morfologi *Erode* dan *dilate*. Alasan menggunakan Morfologi adalah untuk memperbaiki hasil dari segmentasi pada proses sebelumnya. Morfologi Erode berguna untuk menipiskan piksel yang dianggap minimum agar piksel tersebut dihilanngkan [8]. Morfologi Dilate berguna untuk menebalkan piksel yang dianggap maksimum untuk memperjelas objek yang sudah ditipiskan pada proses erode [8]. Segmentasi juga melakukan *Blob Detection*. Alasan menggunakan *Blob Detection* adalah untuk memperjelas objek hasil segmentasi. *Blob Detection* berguna untuk menggabungkan piksel-piksel yang berdekatan untuk dijadikan sebuah objek [5]. Masukan segmentasi adalah frame dan keluaran segmentasi adalah segmen-segmen.

2.2.2.1. Gaussian Blur

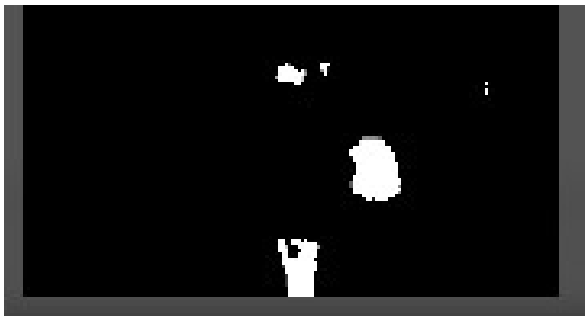
Gambar 3. Gaussian Blur

Metode *Gaussian Blur* berfungsi untuk mereduksi noise dalam sebuah *Image* dengan menggunakan fungsi *Gaussian* yang terdapat dalam *OpenCv*. Untuk menggunakan *Gaussian Blur* perlu dilakukan proses konvolusi. Konvolusi adalah proses penjumlahan seluruh hasil perkalian antara matriks *filter* dengan matriks tetangga dari titik (x,y) pada sebuah citra [3]. Gambar 3 adalah gambar setelah dilakukan metode *Gaussian Blur. Filter Gaussian* yang digunakan adalah *filter* 2 dimensi dengan persamaan :

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2+y^2}{2\sigma^2}} \qquad (1)$$

Persamaan (1) adalah persamaan untuk menghitung nilai *Gaussian.* G(x,y) adalah elemen matriks *Gauss* pada posisi (x,y), σ adalah *filter* radius atau standar deviasi dan (x,y) adalah ukuran matriks *Gauss* yang jangkauannya dari –x sampai +x dengan titik tengah x=0 dan y=0

2.2.2.2. Absolute Difference



Gambar 4. Absolute Difference

Gambar 4 adalah gambar setelah dilakukan metode *Absolute Difference.* Metode *Absolute Difference* adalah metode yang berfungsi untuk menghitung perbedaan mutlak antara dua gambar dengan mengurangkan satu dari yang lain dan mengambil nilai mutlak tersebut. Gambar yang ingin diproses harus memiliki tipe data atau ukuran yang sama. *Absolute Difference* dalam olah citra gambar dijelaskan sebagai berikut [7] :

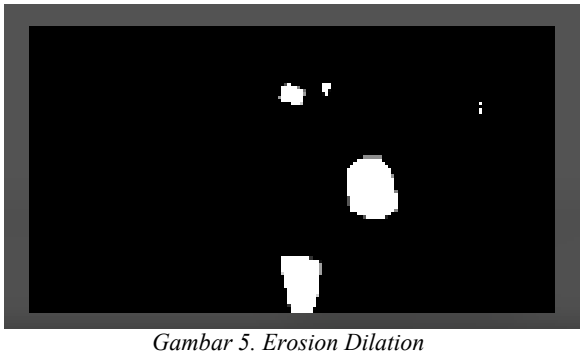
$$I_{d(k,k+1)} = |I_{k+1} - I_k| \quad (2)$$

Persamaan (2) adalah persamaan untuk mencari nilai *Absolute Difference.* Id adalah *Image Difference*, k adalah *Frame Image,* dan k + 1 adalah untuk *frame* selanjutnya.

$$d_{ij}(x,y) = \begin{cases} 1 = if\ , & I_d \geq T \\ 0 = if\ , & I_d \leq T \end{cases} \quad (3)$$

Persamaan (3) adalah persamaan untuk mengetahui nilai *Different Image*. Variabel T adalah Threshold dan d_{ij} (x,y) adalah nilai koodinat *Different Image.* Nilai Koordinat *Different Image* bernilai 1 jika selisih mutlak dua *frame* yang dibandingkan lebih besar dari *Threshold*. Jika tidak lebih besar dari *Threshold* akan bernilai 0. Hasil dari segmentasi objek ini adalah jika koordinat *different Image* yang dihasilkan bernilai 1.

2.2.2.3. Erosion Dilation



Gambar 5. Erosion Dilation

Gambar 5 adalah gambar setelah dilakukan metode *Erosion* dan *Dilation*. *Erosion* atau erosi adalah metode menipiskan piksel yang dianggap *minimum* yang berarti sebuah piksel pada gambar asli (1 atau 0) akan dianggap 1 hanya jika semua piksel dibawah kernel adalah 1, jika tidak maka akan ditipiskan(dibuat menjadi 0). Dibutuhkan dua *Input* yaitu gambar yang akan ditipiskan dan koordinat yang disebut struktur elemen yang digunakan untuk erosi [8]. Berikut ini adalah langkah-langkah matematika pada proses erosi:

Misalkan x adalah himpunan koordinat yang sesuai biner masukan gambar dan K menjadi himpunan yang akan menjadi elemen penataan. Maka, K(x) memiliki arti asal pada daerah x. Kemudian penipisan x oleh k hanyalah himpunan semua titik x sehingga K(x) adalah *subset* dari x.

$$A \ominus B = \{z|(\hat{B})_z \cap A^c \neq \emptyset\} \quad (4)$$

Persamaan (4) adalah persamaan untuk mencari nilai *Erosion.* Variabel A adalah Gambar awal sebelum dilakukan erosi, Variabel B adalah Elemen penataan B, $A \ominus B$ menyatakan bahwa A adalah erosi dari B dan A^c menyatakan bahwa *Komplement* dari A.

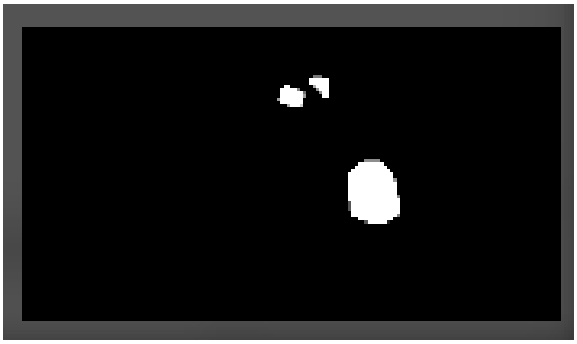
Dilation atau dilasi adalah metode yang merupakan pendamping metode erosi, jika erosi mengambil nilai *minimum* piksel, maka dilasi mengambil nilai *maximum* piksel. Dibutuhkan dilasi karena pada proses erosi akan banyak piksel yang terkikis dan menyebabkan penyusutan pada objek. Dilasi akan meningkatkan daerah yang tidak ditipiskan pada gambar sehingga terjadi dilasi (menebalkan) pada objek yang tidak ditipiskan. Operasi Dilasi adalah operasi penataan elemen B pada gambar A dan memindahkannya ke gambar dengan cara seperti konvolusi (proses memperoleh suatu piksel didasarkan pada nilai piksel itu sendiri dan tetangganya). Dibutuhkan dua *Input* yaitu gambar asal dan koordinat yang disebut struktur elemen yang digunakan untuk dilasi [8].

Berikut ini adalah langkah-langkah matematika pada proses dilasi: Misalkan x adalah himpunan koordinat yang sesuai biner masukan gambar dan K menjadi himpunan yang akan menjadi elemen penataan. Maka, K(x) memiliki arti asal pada daerah x. Kemudian pelebaran x oleh k hanyalah himpunan semua titik x sedemikian rupa sehingga persimpangan K(x) dengan x tidak kosong.

$$A \oplus B = \{z|(\hat{B})_z \cap A \neq \emptyset\} \quad (5)$$

Persamaan (5) adalah persamaan untuk mencari nilai *Dilation*. Variabel A adalah Gambar awal sebelum dilakukan dilasi, variabel B adalah Elemen penataan B, $A \oplus B$ menyatakan bahwa A adalah dilasi dari B, dan $\hat{B}$ menyatakan bahwa Refleksi dari penataan B.

2.2.2.4. Blob Detection



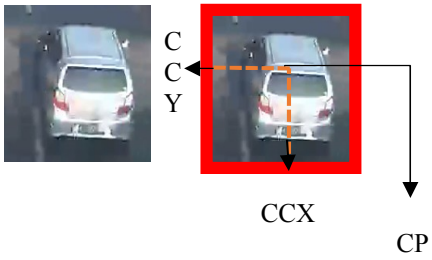
Gambar 6. Blob Detection

Gambar 6 adalah gambar setelah dilakukan metode *Blob Detection*. *Blob Detection* (deteksi *Blob*) adalah metode mendeteksi kumpulan piksel yang memiliki warna berbeda (lebih terang atau lebih gelap) dari latar belakang dan menyatukannya dalam suatu region. *Blob* adalah daerah yang merupakan gabungan dari piksel-piksel yang saling terhubung. *Blob* dapat memisahkan objek statis (*bakground*) dan objek dinamis (*foreground*). Metode *Blob* lebih efisien, kebutuhan memori dapat ditekan bila citra yang ditangani mempunyai tingkat abu-abu yang rendah, dengan arti hanya mengandung informasi hitam dan putih pada piksel penyusunnya. Oleh karena itu, selain lebih efisien metode ini lebih tepat dan akurat. *Blob Detection* digunakan untuk mendeteksi bentuk kendaraan. Deteksi *Blob* akan mendeteksi dan menghitung jumlah kendaraan yang melewati garis perhitungan [5].

2.2.3. Ekstraksi Ciri

Tahap ekstraksi ciri adalah tahap untuk mengurangi dimensi hiperspektral untuk menjaga variabel-variabel yang digunakan dalam proses perhitungan . Ciri-ciri objek meliputi *Bounding Box Area* pada Persamaan (9), *Current Aspect Ratio*(CAR) pada Persamaan (11), *Width of Bounding Box* (WBB) adalah ukuran *horizontal* x dari objek yang terdeteksi, *Height of Bounding Box* (HBB) adalah ukuran *vertikal* y dari objek yang terdeteksi, *Current Diagonal Size* (CDS) pada Persamaan (11), *Centroid* atau *Center Point* (CP) pada persamaan (8), *Real Distance* adalah jarak sesungguhnya yang di *input* saat awal proses, *Line Distance* adalah jarak antara garis awal dan garis akhir yang telah ditentukan dalam piksel, *Car Count* adalah variabel yang digunakan untuk menampung perhitungan jumlah kendaraan, *Speed Count* adalah variabel yang digunakan untuk menampung perhitungan kecepatan kendaraan, Total Waktu adalah perubahan waktu pada objek saat kendaraan menyentuh garis awal dan garis akhir, Posisi Awal Kendaraan adalah posisi kendaraan saat terdeteksi melewati garis awal, Posisi Akhir Kendaraan adalah posisi akhir kendaraan terdeteksi melewati garis akhir, Waktu Awal adalah waktu saat kendaraan terdeteksi melewati garis awal, Waktu Akhir adalah waktu saat kendaraan terdeteksi melewati garis akhir. Setelah itu, proses selanjutnya adalah menghitung jumlah kendaraan dan kecepatan kendaraan.

Pembuatan *Bounding Boxes* pada objek yang terdeteksi merupakan hasil dari ekstraksi ciri. *Bounding Boxes* adalah metode untuk menentukan titik tengah sebuah objek sehingga dapat menghasilkan *Rectangle Area* objek tersebut. Misal x(i) adalah titik penanda objek dalam *Line horizontal* objek O(i). Nilai tengah *horizontal* x (*Current Center X* / CCX) dapat ditemukan pada Persamaan (6) melalui *Width Of Bounding Box* (WBB). Misal y(i) adalah titik penanda objek dalam *Line vertikal* objek O(i). Nilai tengah *vertikal* y (*Current Center Y* / CCY) dapat ditemukan pada Persamaan (7) melalui *Height Of Bounding Box* (HBB). Mencari nilai CCX dan CCY dapat membantu untuk menemukan titik tengah objek (*Center Point* / CP) yang ditunjukkan pada Persamaan (8) dan menentukan *area Bounding Boxes*. *Center Point* disebut sebagai *Centroid* dalam *Bounding Boxes*[1]. Gambar 7 adalah contoh objek mobil yang telah dibuat *Bounding Box.*



(a) (b)

Gambar 7. Pencarian Titik Tengah Objek dan pembentukan Bounding Boxes
(a) Sebelum dan (b) Sesudah Bounding Boxes

$$CCX = X + \frac{1}{2} WBB \quad (6)$$

$$CCY = Y + \frac{1}{2} HBB \quad (7)$$

$$CP = \{CCX, CCY\} \quad (8)$$

$$Bounding\ Box\ Area = WBB\ x\ HBB \quad (9)$$

$$CDS = \sqrt{WBB^2 + HBB^2} \quad (10)$$

$$CAR = WBB : HBB \quad (11)$$

Persamaan (6) digunakan untuk menjadi titik tengah garis horizontal x (CCX). Persamaaan (7) digunakan untuk menjadi titik tengah garis vertikal y (CCY). Persamaan (8) untuk mendapatkan koordinat *Center Point* (CP). Persamaan (9) untuk menghitung ukuran *Bounding Box* yang sesuai objek. Persamaan (10) untuk menghitung diagonal *Bounding Box* (CDS). Persamaan (11) untuk mendapatkan nilai *Current Aspect Ratio* (CAR).

CAR adalah perbandingan tinggi dan lebar pada sebuah gambar. Dalam aplikasi ini perbandingan antara WBB dan HBB. CDS pada Persamaan (10) adalah ukuran diagonal pada *Rectangle Area* yang berfungsi untuk mengurangi terjadinya *double detection*.

Objek akan terdeteksi selama memenuhi syarat yang telah ditentukan sebagai sebuah objek, adalah jika *Current Rectangle Area* > 400, *Current Aspect Ratio* bernilai 0,2 – 4,0, *Width of Bounding Box* > 30 , H*eight of Bounding Box* > 30, dan *Current Diagonal Size* > 60,0

2.2.4. Menghitung Jumlah Kendaraan

Menghitung jumlah kendaraan, dapat dengan menghitung jumlah *Centroid* dari semua *Frame Blob* yang tersedia, jumlah *Centroid* kendaraan yang bergerak akan memberi informasi tentang jumlah kendaraan yang ada. Untuk menghitung kendaraan yang melalui suatu jalan melalui video diperlukan batas perhitungan berupa dua garis pembatas dalam video. Dengan masukan berupa video MP4 video akan diproses dalam program untuk menghitung kendaraan. *Count* (hitung) kendaraan dalam program dibuat *default CarCount = 0*. Sehingga jika sebuah kendaraan terdeteksi melintas pada garis akhir (garis biru) yang ditentukan maka akan dimulai perhitungan kendaraan. Gambar 8 adalah gambar saat nilai *CarCount = 0*. Gambar 9 adalah saat kendaraan terdeteksi garis akhir dan *CarCount* menjadi *Carcount + 1*.

Gambar 8. Sebelum Perhitungan Kendaraan

Gambar 9. Sesudah Perhitungan Kendaraan

2.2.5. Menghitung Kecepatan kendaraan.

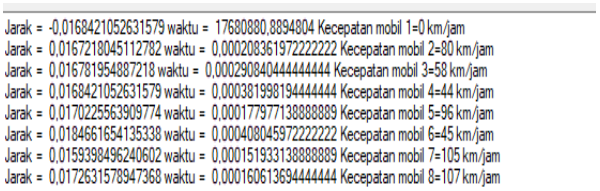
Untuk mencari kecepatan kendaraan pada umumnya adalah membagi jarak perpindahan benda dengan waktu perpindahan benda tersebut seperti pada persamaan (12) dan (13). Formula jarak tempuh kendaraan (14) didapat dari penelitian sebelumnya

$$v = \frac{\Delta s}{\Delta t} \quad (12)$$

$$\Delta t = tf\ x\ \left(t(n) - t(o)\right) \quad (13)$$

$$\Delta s = sf\ x\ \left(\frac{s}{sx}\right) x\ (P_n - P_0) \quad (14)$$

Persamaan (12) adalah persamaan untuk menghitung kecepatan kendaraan. Persamaan (13) untuk menghitung waktu tempuh sebenarnya. Persamaan (14) digunakan untuk menghitung jarak sebenarnya. Variabel v adalah kecepatan kendaraan, Δs adalah Jarak perpindahan benda (perkiraan), sxadalah Jarak antar dua garis deteksi (awal dan akhir) dalam piksel , s adalah Jarak sebenarnya, sf adalah Konversi jarak ke kilometer, P_n adalah Posisi ujung kanan kendaraan pada t_n, P_0adalah Posisi ujung kanan kendaraan pada t_0, t_0 adalah Waktu awal t_n adalah Waktu akhir, dan tf adalah Konversi waktu ke jam.



Gambar 10. Hasil Perhitungan

Gambar 10 adalah gambar yang berisi informasi jarak yang didapat dari Persamaan (12) dalam satuan km, waktu yang didapat dari Persamaan (11) dalam satuan jam dan kecepatan yang didapat dari Persamaan (10) dalam satuan km/jam.

3. HASIL DAN PEMBAHASAN

3.1. Lingkungan Penerapan

Dalam Penerapan diperlukan beberapa software pendukung seperti *library OpenCv* dan IDE *Visual Studio 2017* dengan Sistem Operasi *Windows 10* agar aplikasi dapat berjalan dengan baik. Rekaman CCTV lalu lintas dan rekaman yang diambil dari *Handphone* dijadikan sampel untuk implementasi aplikasi ini.

3.2. Pengujian

3.2.1. Skenario Pengujian

Pengujian menggunakan dua skenario dengan masing-masing memiliki dua kondisi fps yang berbeda namun memiliki durasi rekaman yang sama. Tabel 1 adalah gambaran kondisi skenario pengujian.

Tabel 1. Skenario Pengujian

Skenario		Durasi Video	Jarak *Line* sesungguhnya	Jumlah Kendaraan
Video *Input*	fps			
CCTV	10	8 detik	16 meter	10
	15	8 detik	16 meter	10
Hand Phone (HP)	10	8 detik	5 meter	8
	15	8 detik	5 meter	8

3.2.2. Metode Pengujian

3.2.2.1. Metode Evaluasi Berbasis Data dan Prediksi.

Metode ini digunakan dengan dasar data yang diperoleh dalam sistem seperti jumlah seluruh kendaraan yang terhitung sistem, jumlah kendaraan yang terhitung satu atau dua dalam sistem, dan jumlah kendaraan yang tidak terhitung dalam sistem. Tabel 2 menjelaskan jumlah kedaraan yang terhitung dalam sistem. Alasan menggunakan metode ini adalah dapat menghasilkan tingkat akurasi yang efektif karena melibatkan data yang didapat sistem untuk dijadikan pehitungan akurasi jumlah kendaraan [5].

Tabel 2. Tabel Pengujian

No	Kondisi	Jumlah	Frekuensi kendaraan	Total
1	Jumlah Kendaraan Pada Kondisi Nyata.	10	1	10
2	Jumlah Kendaraan Terhitung Sistem			10
3	Terhitung satu	10	1	10
	Terhitung Dua	0	0	0
	Distraction	0	0	0
	Tidak Terdeteksi	0	0	0

True Positive (TP) = Kemungkinan kendaraan pada kondisi nyata dan dinyatakan kendaraan dalam sistem. *True Negative* (TN) = Kemungkinan bukan kendaraan pada kondisi nyata dan dinyatakan kendaraan dalam sistem. *False Positive* (FP) = Kemungkinan bukan kendaraan pada kondisi nyata dan tidak dinyatakan kendaraan dalam sistem. *False Negative* (FN) = Kemungkinan kendaraan pada kondisi nyata dan tidak dinyatakan kendaraan dalam sistem.

$$Accuracy = \frac{(TN+TP)}{(TN+TP+FN+FP)} x\ 100\%$$

$$Accuracy = \frac{(0+10)}{(0+10+0+0)} x\ 100\%$$

$$Accuracy = \frac{10}{10} x\ 100\%$$

$$Accuracy = \ 100\%$$

3.2.2.2.Akurasi Pengujian Fungsionalitas

$$Selisih = \left(\frac{ks-ku}{ks}\right) x\ 100\% \quad (15)$$

Persamaan 15 adalah persamaan untuk menghitung selisih kecepatan nyata dan kecepatan ujicoba dalam bentuk persen. Variabel ks adalah kecepatan sesungguhnya dan variabel ku adalah kecepatan uji coba.

Tabel 3. Tabel Pengujian Nyata

Car_id	Jarak Sesungguhnya (kilometer)	Waktu tempuh (jam)	Kecepatan ssesungguhnya (km/jam)
1	0,016	0	-
2	0,016	0,0002061489722	81
3	0,016	0,00029795622	56
4	0,016	0,0003878164722	43
5	0,016	0,0002086534722	82
6	0,016	0,0004629308056	40
7	0,016	0,000160802611	99
8	0,016	0,0002195021944	79
9	0,016	0,0002804289167	60
10	0,016	0,0004056214167	42

Tabel 3 adalah tabel pengujian dalam perhitungan kondisi nyata dalam rekaman yang diuji. Kecepatan yang dihasilkan dalam perhitungan nyata akan digunakan dan dibandingkan dengan kecepatan yang terhitung dalam sistem untuk ditemukan selisih perbedaan kecepatan nyata dan kecepatan yang terdeteksi dalam sistem.

Tabel 4. Percobaan estimasi kecepatan

Car_id	Percobaan					Rata-Rata	Selisih
	1	2	3	4	5		
1	0	0	0	0	0	00,0	0%
2	73	74	62	82	70	72,2	10,86%
3	52	48	44	54	45	48,6	13,21%
4	40	38	35	43	33	37,8	12,09%
5	90	83	96	68	93	86	4,87%
6	41	35	42	33	35	37,2	7,00%
7	100	83	108	84	79	90,8	8,28%
8	70	76	83	79	71	75,8	4,05%
9	55	58	61	56	50	56	6,67%
10	39	36	38	35	36	36,8	12,38%
Rata-Rata							**7,94%**

Tabel 4 adalah tabel pengujian hasil perhitungan kecepatan pada 5 percobaan yang dilakukan oleh sistem. Setelah dilakukan 5 percobaan diperoleh kecepatan rata-rata uji coba yang akan dibadingkan dengan kecepatan sesungguhnya yang ada pada tabel 3. untuk diperoleh selisih antara kecepatan sesungguhnya dan kecepatan rata-rata uji coba yang diperoleh sistem.

3.3. Hasil Pengujian

Tabel 5 . Hasil Pengujian

	Akurasi Jumlah Kendaraan		Selisih perhitungan kecepatan	
	10 fps	15 fps	10 fps	15 fps
Cctv	100%	100%	7,94%	5,11%
HP	100%	69,23%	11,83%	11,05%

Tabel 5 adalah hasil Pengujian yang diperoleh pada percobaan menggunakan rekaman CCTV akurasi jumlah kendaraan mencapai 100% pada 10 fps dan 15 fps sedangkan untuk selisih perhitungan kecapatan terendah adalah 5,11% pada rekaman 15 fps. Pada percobaan rekaman *HandPhone* akurasi jumlah kendaraan pada 10 fps adalah 100% dan pada 15 fps bernilai 69,23% sedangkan untuk selisih perhitungan kecepatan pada 10 fps dan 15 fps mencapai 11,83% dan 11,05%. Pengujian ini membuktikan bahwa sistem yang dibuat sesuai digunakan untuk mendeteksi jumlah dan kecepatan kendaraan melalui rekaman CCTV.

4. KESIMPULAN

Dari hasil pengamatan selama proses pengamatan, implementasi, dan pengujian sistem, dapat disimpulkan bahwa:

a. Pengujian dengan menggunakan metode *Gaussian Blur* dan *Absolute Difference* menghasilkan akurasi perhitungan jumlah kendaraan tertinggi adalah 100%. Akurasi perhitungan jumlah kendaraan terendah adalah 69,23%. Selisih terkecil yang didapat antara kecepatan nyata dan kecepatan uji coba adalah sebesar 5,11%.

b. Faktor yang memengaruhi akurasi perhitungan dan selisih kecepatan pada aplikasi adalah pengurangan fps, peletakkan garis, dan penentuan perkiraan jarak sesungguhnya.

DAFTAR PUSTAKA

[1] Wicaksono, D. W. dan Setiyono, B. (2017) "Speed Estimation On Moving Vehicle Based On Digital Image Processing," *International Journal of Computing Science and Applied Mathematics,* 3(1), hal. 21–26.

[2] Muliawan, M. R., Irawan, Beni dan Brianorman, Yulrion. (2015) "Metode Eigenface Pada Sistem Absensi," *Jurnal Coding, Sistem Komputer Untan*, 3(1), hal. 41–50.

[3] Bozkurt, F., Yağanoğlu, M. dan Günay, F. B. (2015) "Effective Gaussian Blurring Process on Graphics Processing Unit with CUDA," *International Journal of Machine Learning and Computing*, 5(1), hal. 57–61. doi: 10.7763/IJMLC.2015.V5.483.

[4] Tripathi, J. *et al.* (2015) "Automatic Vehicle Counting and Classification," *International Journal of Innovative and Emerging Research in Engineering,* 2(4), hal. 32–36.

[5] Sundoro, H. S. dan Harjoko, A. (2016) "Vehicle counting and vehicle speed measurement based on video processing," *Journal of Theoretical and Applied Information Technology*, 84(2), hal. 233–241.

[6] Republik Indonesia. 1993. Keputusan Menteri Nomor KM 60 Tahun 1993 tentang Marka Jalan. Dinas Perhubungan. Jakarta.

7] Singla, N. (2014) "Motion Detection Based on Frame Difference Method," International Journal of Information & Computation Technology, 4(15), hal. 1559–1565. Tersedia pada: http://www.ripublication.com/irph/ijict_spl/ijictv4n15spl_10.pdf.

[8] Raid, A. M. et al. (2014) "Image Restoration Based on Morphological Operations," International Journal of Computer Science, Engineering and Information Technology (IJCSEIT), 4(3), hal. 9–21. doi: 10.5121/ijcseit.2014.4302.