Counters Speed of Vehicle System Based on Camera Using

Optical Flow Method For Detecting Traffic Violations

Misbakhul Munir, Nana Ramadijanti, Fadilah Fahrul Hardiansyah

Program Studi Lanjut Jenjang D4 Teknik Informatika

Departemen Informatika dan Komputer

Politeknik Elektronika Negeri Surabaya

Kampus PENS, Jalan Raya ITS Sukolilo, Surabaya 60111

*Tel: (031) 594 7280; Fax: (031) 594 6114*

*Email:* misbakhulmunir@it.student.pens.ac.id, nana@pens.ac.id, fahrul@pens.ac.id

Abstract

*Speed restriction is a provision to limit the speed of vehicular traffic in order to reduce the number of traffic accidents. Moreover, for on the highway, where the road is a freeway that will make the driver will drive the vehicle faster. During this time, monitoring is done using only the eye without the aid of tools. However, over the increasing number of vehicles, of course in this way could not be used anymore. So they invented a system of vehicle speed counters using camera-based optical flow method to detect traffic violations, especially on the highway. In addition to further facilitate supervision, the use of these applications also tend to be cheaper than having to use a speed measuring device that already exists. This application can estimate the speed of passing vehicles, then when the vehicle is exceeding the speed limit that is specified then this application will take a picture of the vehicle. In addition, this application can also display the report in the form of the number of vehicles that exceed the speed limit and the average speed of offenders. With this application, is expected to use the authorities to impose penalties or warning to the offenders. This application was developed using Delphi 7.*

**Keywords:** *Optical Flow, Speed Detection, Delphi 7, Traffic*

1. Pendahuluan

Mengidentifikasi gerakan merupakan salah satu kegiatan yang bisa dilakukan dari proses analisis video. Dari identifikas ini bisa diketahui objek yang bergerak, jumlah objek yang bergerak hingga kecepatan pergerakannya. Ada berbagai macam cara yang bisa dikembangkan untuk melakukan analisa kecepatan gerakan, tetapi kebanyakan biaya yang dibutuhkan tidak murah. Untuk itu, dalam tugas akhir ini akan digunakan kamera sebagai alat pengambil gambar situasi lalu-lintas di jalan dan untuk menganalisis kecepatan gerakan kendaraannya digunakan metode *optical flow.*

Hasil perhitungan kecepatan pada sebuah gambar yang ditangkap oleh kamera akan digunakan untuk mengawasi kecepatan kendaraan yang melintas di jalan. Hal ini sebagai alat bantu untuk memberikan hukuman bagi para pengguna kendaraan yang melanggar batas kecepatan maksimal.

Dibutuhkan sebuah sistem yang dapat berjalan sendiri untuk membantu polisi lalu lintas dalam melakukan pengawasan kecepatan kendaraan di lalu lintas. Karena tidak mungkin jika polisi lalu lintas melakukan pengawasan siang-malam selama 24 jam tanpa berhenti. Oleh karena itu, muncul ide untuk membuat “Sistem Penghitung Kecepatan Kendaraan Berbasis Kamera Menggunakan Metode Optical Flow Untuk Mendeteksi Pelanggaran Lalu Lintas”.

1. **Desain Sistem**

Proyek akhir ini bertujuan untuk membuat sebuah sistem yang bisa membantu memantau kecepatan kendaraan menggunakan sensor kamera. Dalam proyek akhir ini, lokasi pemantauannya adalah di Tol Surabaya-Gresik. Tidak menutup kemungkinan sistem ini juga bisa digunakan di lokasi lain selain jalan tol.

* 1. **Batas Kecepatan**

Secara umum, pembatasan kecepatan kendaraan diatur dalam pasal 80 Peraturan Pemerintah no. 43 tahun 1993 tentang Prasarana dan Lalu Lintas Jalan. Dilihat dari kelas jalannya, batas kecepatan bisa dilihat pada tabel dibawah ini [3]:

**Tabel 1.** Batas Kecepatan Kendaraan

|  |  |  |  |
| --- | --- | --- | --- |
| **Kelas**  **Jalan** | **Fungsi** | **Jenis**  **Kendaraan** | **Batas Kecepatan**  **(Km/Jam)** |
| Kelas I, II dan IIIA | Primer | Mobil Pribadi, Bus, Truk | 100 |
| Kelas I, II dan IIIA | Primer | Gandengan Dan Tempelan | 80 |
| Kelas IIIB | Primer | Mobil Pribadi, Bus, Truk | 80 |
| Kelas IIIC | Primer | Mobil Pribadi, Bus, Truk | 60 |
| Kelas II, IIIA | Sekunder | Mobil Pribadi, Bus, Truk | 70 |
| Kelas II, IIIA | Sekunder | Gandengan, Tempelan | 60 |
| Kelas IIIB | Sekunder | Mobil Pribadi, Bus, Truk | 50 |
| Kelas IIIC | Sekunder | Mobil Pribadi, Bus, Truk | 40 |

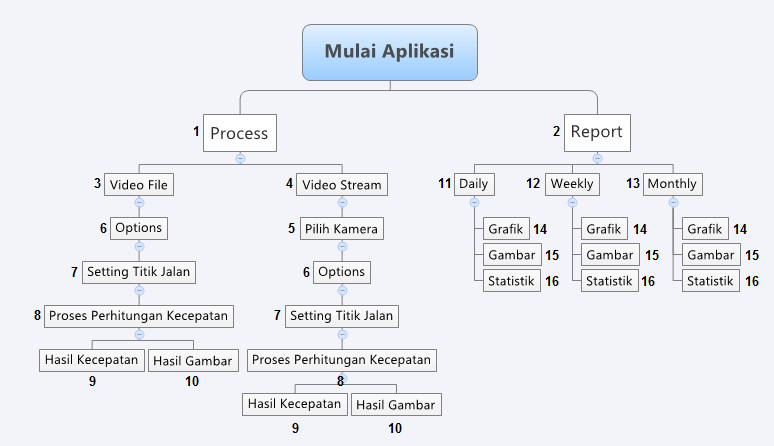
Selain berdasarkan kelas, pembatasan kecepatan juga dilakukan berdasarkan desain jalan tersebut atau bisa disebut dengan kecepatan rencana. Kecepatan rencana ditetapkan berdasarkan klasifikasi jalan dan medan yang dilalui, di medan jalan yang datar akan lebih tinggi dari perbukitan ataupun pegunungan. Besaran kecepatan rencana di Indonesia bisa dilihat pada tabel dibawah ini [4] :

**Tabel 2.** Batas Kecepatan Rencana

|  |  |  |  |
| --- | --- | --- | --- |
| **Kelas** | **Datar**  **(Km/Jam)** | **Bukit**  **(Km/Jam)** | **Gunung**  **(Km/Jam)** |
| Jalan Tol  (Arteri & kolektor) | 100 – 120 | 80 – 100 | 80 |
| Jalan Raya  (Arteri & Kolektor) | 80 – 100 | 60 – 80 | 40 – 60 |
| Jalan Sedang  (Kolektor) | 60 – 80 | 40 – 60 | 40 |
| Jalan Kecil  (Lokal & Lingkungan) | 20 – 40 | 20 – 40 | 20 – 40 |

* 1. **Rancangan Aplikasi**

Aplikasi ini terdiri dari beberapa bagian yang saling berhubungan satu sama lain sesuai alur atau rancangan yang telah ditentukan. Rancangan tersebut dapat dijelaskan melalui diagram blok seperti gambar di bawah ini.



**Gambar 1.** Rancangan Aplikasi

* 1. **Tools Yang Digunakan**

Aplikasi ini dibuat menggunakan program Delphi 7, tanpa menggunakan library pengolahan citra seperti OpenCV, Matlab ataupun library lainnya.

* 1. **Pengambilan Video**

Proses pengambilan video dilakukan pada sebuah jembatan penyeberangan yang berada diatas jalan tol. Karena proyek akhir ini bertujuan untuk mengetahui para pelanggar batas kecepatan kendaraan, maka dibutuhkan video yang merekam kendaraan yang melaju dengan kecepatan tinggi. Dibawah ini adalah detail hasil video dan kamera yang digunakan untuk merekam lalu-lintas pada jalan tol Surabaya-Gresik dan sebaliknya.

**Tabel 3.** Detail Kamera dan Hasil Video

|  |  |
| --- | --- |
| **Nama** | Lenovo S820 |
| **Pixel** | 12 Megapixel |
| ***Frame* *Width*** | 1280 |
| ***Frame* *Height*** | 720 |
| ***Frame Rate*** | 30 *Frame/Second* |
| **Tanggal Pengambilan** | 6 & 8 Desember 2014 |
| **Lokasi Pengambilan** | Tol Surabaya - Gresik  Tol Gresik - Surabaya |



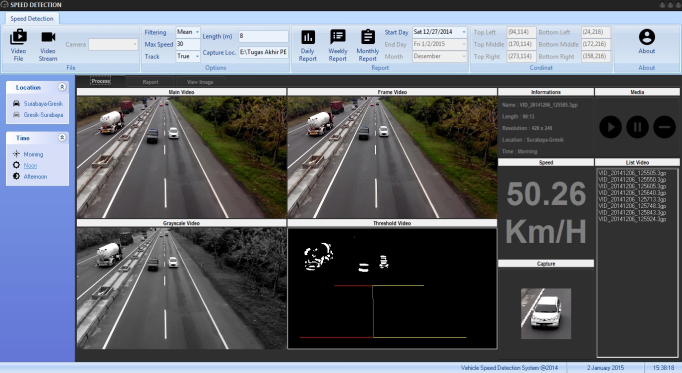
**Gambar 2.** Cara Pengambilan Video

* 1. **Pembuatan Aplikasi**

Beberapa hal penting dalam pembuatan proyek akhir ini adalah bagaimana mengimplementasikan metode *optical flow*, *threshold*, *mean filtering*, *adaptive thresholding* dan *BLOB analysis* untuk mendeteksi obyek bergerak dalam video dimana dalam proyek akhir ini obyeknya adalah kendaraan roda 4 atau lebih. Selain itu, yang juga penting adalah bagaimana bisa menghitung kecepatan kendaraan dari hasil deteksi obyek yang dilakukan.

* 1. **Desain Interface**

Sesuai dengan rancangan aplikasi pada gambar 1, maka harus kita buat tampilan yang sesuai dengan kebutuhan yang ada pada rancangan aplikasi. Bentuk tampilan dibuat agar mudah digunakan nantinya oleh calon pengguna.



**Gambar 3.** Desain *Interface* Aplikasi

* 1. **Implementasi Grayscale Image**

Sebelum mengimplementasikan metode *optical flow* terhadap video rekaman, terlebih dahulu dilakukan perbuhan gambar asli menjadi abu-abu (*Grayscale*). Hal ini bertujuan untuk mengurangi nois yang mungkin akan terjadi. Caranya adalah dengan menggunakan rata-rata dari warna dalam sebuah pixel. Implementasi dan hasil *grayscale* bisa dilihat dari gambar dibawah ini.



procedure Grayscale(x,y);

begin

R = Nilai Red Pixel (x,y)

G = Nilai Green Pixel (x,y)

B = Nilai Blue Pixel (x,y)

Pixel(x,y) = (R + G + B) div 3

end;

**Gambar 4.** Prosedur Dan Hasil Grayscale

* 1. **Implementasi Program Optical Flow**

*Optical Flow* adalah perkiraan gerakan suatu bagian dari sebuah citra berdasarkan turunan intensitas cahayanya pada sebuah sekuen citra. Pada ruang 2D hal ini berarti seberapa jauh suatu piksel citra berpindah diantra dua frame citra yang berurutan. Sedangkan pada ruang 3D hal ini berarti seberapa jauh suatu volume piksel (voxel) berpindah pada dua volume yang berurutan. Perhitungan turunan dilakukan berdasarkan perubahan intensitas cahaya pada kedua frame citra maupun volume. Perubahan intensitas cahaya pada suatu bagian citra dapat disebabkan oleh gerakan yang dilakukan oleh obyek, gerakan sumber cahaya, ataupun perubahan sudut pandang.[1]

Setelah melakukan *grayscale* dari gambar asli, tahap selanjutnya adalah mengimplementasikan metode *optical flow* untuk mengetahui obyek yang bergerak atau kendaraan. Prosesnya adalah dengan membandingkan setiap piksel pada sebuah sekuen gambar. Adanya perbedaan nilai pada sebuah piksel antara *frame* sekarang dengan *frame* sebelumnya menandakan adanya obyek bergerak. *Pseudocode* bisa dilihat dari gambar dibawah ini.

procedure OpticalFlow;

begin

frameSekarang = Nilai Pixel (frame)

frameSebelum = Nilai Pixel (frame-1)

hasilOpticalFlow = frameSekarang - frameSebelum

end;

**Gambar 5.** Prosedur Implementasi *Optical Flow*

* 1. **Implementasi Program Threshold**

Agar bentuk obyek yang bergerak bisa diketahui dengan baik, hasil dari *optical flow* akan ditampilkan dalam bentuk gambar hitam putih. Yang berwarna hitam adalah *background* sedangkan putih adalah obyek yang bergerak atau kendaraan. *Pseudocode* bisa dilihat dari gambar dibawah ini. Pada *pseudocode* tersebut, hasil merupakan nilai hasil perhitungan *optical flow*. Sedangkan angka 60 adalah batas *threshold* yang sudah ditentukan. Pada tahap ini hasilnya berupa gambar yang menunjukkan ada atau tidaknya obyek bergerak dalam sekuen gambar.



procedure Threshold;

begin

if hasilOpticalFlow < 60 then pixel(x,y) = 0 else

pixel(x,y) = 255;

end;

**Gambar 6.** Prosedur Implementasi dan Hasil *Threshold*

* 1. **Implementasi Mean Filtering**

Mean filtering yang digunakan untuk efek smoothing ini merupakan jenis *spatial filtering*, yang dalam prosesnya mengikutsertakan piksel-piksel disekitarnya. Piksel yang akan diproses dimasukkan dalam sebuah matrik yang berdimensi N x N. Ukuran N ini tergantung pada kebutuhan, tetapi nilai N haruslah ganjil sehingga piksel yang diproses dapat diletakkan tepat ditengah matrik. Sebagai contoh matrik berdimensi 3 x 3 seperti gambar 7 di bawah ini.

Hasil dari *thresholding* masih menghasilkan banyak noise. Dimana yang dianggap obyek bergerak bukan hanya kendaraan, melainkan juga ranting-ranting pohon yang bergoyang karena terkena angin. Diperlukan metode tambahan untuk bisa membedakan antara ranting pohon yang bergerak dengan mobil. Untuk itu digunakan *mean filtering*. *Pseudocode* implementasi dari metode ini bisa dilihat pada gambar dibawah ini.

procedure MeanFiltering(x,y);

begin

nilai\_1 = hasilOpticalFlow(x,y)

nilai\_2 = hasilOpticalFlow(x+1,y)

nilai\_3 = hasilOpticalFlow(x+1,y+1)

nilai\_4 = hasilOpticalFlow(x,y+1)

nilai\_5 = hasilOpticalFlow(x-1,+1y)

nilai\_6 = hasilOpticalFlow(x-1,y)

nilai\_7 = hasilOpticalFlow(x-1,y-1)

nilai\_8 = hasilOpticalFlow(x,y-1)

nilai\_9 = hasilOpticalFlow(x+1,y-1)

hasilMeanFiltering(x,y) = (nilai\_1+ nilai\_2+ nilai\_3+ nilai\_4+ nilai\_5+ nilai\_6+ nilai\_7+ nilai\_8+ nilai\_9) div 3;

end;



**Gambar 8.** Matriks *Mean Filtering* dan Prosedur Implementasi *Mean Filtering*

* 1. **Implementasi Adaptive Thresholding**

*Adaptive thresholding* atau disebut juga *thresholding* local berfungi untuk menentukan nilai *threshold* setiap piksel berdasarkan nilai grayscale sendiri dan nilai grayscale tetangga. Oleh karena itu, pendekatan ini disebut algoritma *thresholding* adaptif. Beberapa algoritma *thresholding* adaptif telah dikembangkan, diantaranya adalah metode Niblack pada tahun 1986. Namun, metode Niblack tidak efektif ketika background berisi tekstur berupa cahaya. Pada tahun 2000, Sauvola menyusun metode yang memodifikasi metode Niblack dengan melakukan hipotesis pada nilai keabuan piksel teks dan piksel background. Pada tahun 2004, Sezgin dan Sankur melakukan perbandingan terhadap 40 metode. Hal itu menunjukkan bahwa metode thresholding lokal Sauvola dan metode dari Trier melakukan yang terbaik untuk proses thresholding dokumen.[8]

****

procedure AdaptiveThresholding(x,y);

begin

hasilAdaptive(x,y) = hasilMeanFiltering(x,y) \*

( 1 + (0.34\*((deviasi(x,y)/128)-1)) )

end;

**Gambar 9.** Prosedur Implementasi *Adaptive Thresholding*

* 1. **Implementasi BLOB Analysis**

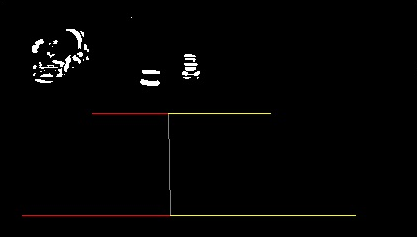
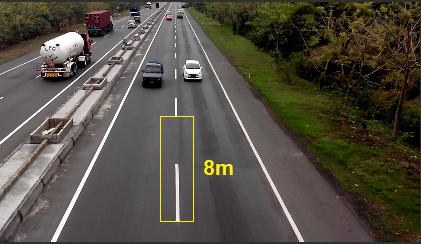
Algoritma ini digunakan untuk mendeteksi titik atau area pada sebuah citra yang berbeda baik warna ataupun kecerahannya jika dibandingkan dengan sekitarnya. Setelah mendapatkan hasil *thresholding* yang sudah di hilangkan noisenya, dari hasil tersebut bisa diterapkan metode BLOB *analysis* untuk menandai obyek yang sudah memasuki daerah perhitungan yang sudah ditentukan. Hasil dari tahap ini bisa dilhat pada gambar 10.



**Gambar 10.** Hasil BLOB Analysis

* 1. **Penghitungan Jarak Sebenarnya**

Penentuan jarak sebenarnya dalam video ditentukan dengan menggunakan panjang marka jalan sebagai acuan. Sebelum aplikasi bisa melakukan proses perhitungan kecepatan kendaraan dalam video, *user* harus menentukan dimana titik yang menjadi acuan panjang jarak sebenarnya. Jarak antara marka adalah 5m dan panjang marka tersebut adalah 3m. Jarak yang digunakan dalam proses ini adalah gabungan panjang keduanya, sehingga jarak yang bisa diketahui dalam video adalah sepanjang 8m. Setelah itu kita menentukan dimana titik kordinat 0m dan 8m. Hal ini berguna untuk mendapatkan panjang sebenarnya dalam satuan pixel.



**Gambar 11.** Penentuan Jarak Sebenarnya dan Titik Kordinat Jarak

*Pseudocode* dan rumus penghitungan jarak sebenarnya dalam video bisa dilihat pada gambar dibawah ini

procedure JarakSebenarnya(x1,y1,x2,y2);

begin

jarak := Sqrt(Sqr(Abs(x1-x2))+Sqr(Abs(y1-y2)));

end;

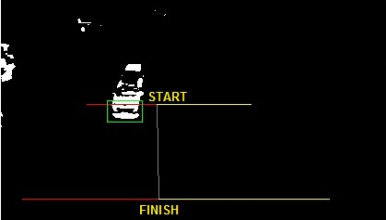
**Gambar 12.** Rumus dan Prosedu Perhitungan Jarak Sebenarnya

* 1. **Penghitungan Kecepatan**

Setelah mengetahui jarak sebenarnya dalam video, maka kita bisa melakukan perhitungan kecepatan dari kendaraan yang ada di video. Ada beberapa tahapan yang dilakukan dalam proses penghitungan kecepatan.

* 1. **Mencari Titik Obyek Yang Paling Mendekati Garis Finish Sebagai Titik Awal Kendaraan**

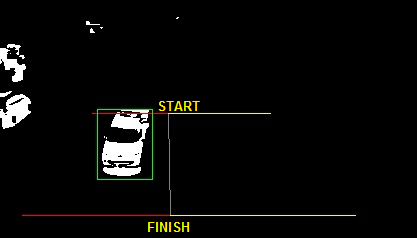
Sebelum bisa mengetahui kecepatannya, terlebih dahulu harus diketahui besarnya jarak perpindahan oleh kendaraan. Ketika obyek sudah memasuki daerah perhitungan yang sudah ditentukan, titik hasil *thresholding* obyek yang paling mendekati garis *finish* menjadi lokasi titik awal obyek.



**Gambar 13.** Penentuan Titik Awal Obyek

* 1. **Mencari Titik Obyek Yang Paling Mendekati Garis Finish Setelah 200ms Sebagai Titik Akhir Kendaraan**

Hampir sama dengan proses pada tahap 1, pencarian titik akhir dilakukan selang 200ms setelah titik awal ditemukan. Hal ini bertujuan untuk mengetahui besarnya perpindahan obyek selama 200ms.



**Gambar 14.** Penentuan Titik Akhir Obyek

* 1. **Menghitung Jarak Antara Titik Awal Dan Titik Akhir**

Pada tahap ini sudah diketahui dimana titik awal dan titik akhir perpindahan obyek. Dengan menggunakan rumus jarak, kita bisa melakukan perhitungan jarak perpindahan obyek setelah 200ms.

****

**Gambar 15.** Ilustrasi Perhitungan Jarak

* 1. **Konversi Jarak Dari Satuan Pixel Ke Satuan Meter**

Hasil perpindahan obyek masih dalam satuan pixel. Agar bisa dihitung kecepatannya, harus dilakukan konversi satuan dari pixel ke meter dengan menggunakan rumus dibawah ini.

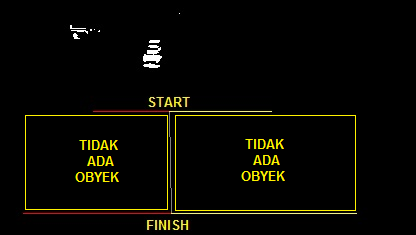
Perpindahan (meter) =

* 1. **Hitung Kecepatan Kendaraan**

Setelah mengetahui besarnya perpindahan obyek selama 1000ms atau 1 detik dalam satuan meter, kita tinggal konversi satuan tersebut menjadi kilometer per jam (Km/h) dengan acuan 1 meter/s = 3.6 kilometer/jam.

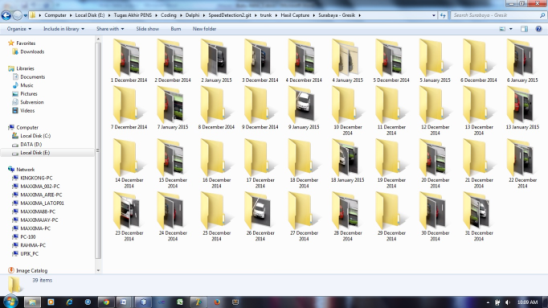
* 1. **Pengulangan Proses**

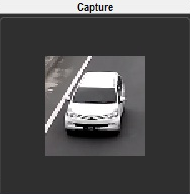
Pengulangan proses perhitungan mulai dari tahap (1) akan dilakukan setelah tidak ada kendaran atau obyek yang terdeteksi didalam daerah perhitungan yang sudah ditentukan.



**Gambar 16.** Ilustrasi Perulangan Proses

* 1. **Capture Kendaraan**

Tahap ini digunakan untuk mengambil gambar dari pelanggar batas kecepatan yang sudah ditentukan. Penentuan titik pengambilan gambar dilakukan berdasarkan hasil pencarian obyek menggunakan BLOB *analysis*. Semua hasil *capture* disimpan dalam masing-masing folder sesuai tanggal pengambilan gambar. Hal ini bertujuan untuk memudahkan saat menampilkan *report*.



**Gambar 17.** Hasil *Capture* dan Penyimpanan Gambar Kendaraan

* 1. **Report**

*Report* bisa ditampilkan harian, mingguan atau bulanan. Yang ditampilkan dalam *report* ini adalah jumlah pelanggar dan rata-rata kecepatan pelanggar. Jumlah pelanggar bisa di dapat dari jumlah hasil capture yang ada pada folder sesuai tanggal yang ingin ditampilkan reportnya. Sedangkan rata-rata kecepatan pelanggar bisa diperoleh dari nama file hasil capture.

1. **Eksperimen**

Pada eksperimen ini dilakukan uji coba program yang sudah dibuat dan analisa hasil percobaan apakah sudah sesuai dengan yang diharapkan. Pengujian dilakukan dengan menggunakan mobil *remote control.* Membandingkan antara hasil perhitungan kecepatan secara manual dengan hasil perhirungan program.

* 1. **Eksperimen Berdasarkan Waktu Pengambilan**

**Tabel 4.** Analisa Hasil Pehitungan Berdasarkan Waktu Pengambilan

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Ke** | **Manual (Km/h)** | **Pagi** | | | **Siang** | | | **Sore** | | |
| **Prorgam (Km/h)** | **Perbedaan (Km/h)** | **Akurasi (%)** | **Prorgam (Km/h)** | **Perbedaan (Km/h)** | **Akurasi (%)** | **Prorgam (Km/h)** | **Perbedaan (Km/h)** | **Akurasi (%)** |
| 1 | 5.79 | 5.5 | 0.29 | 95 | 5.73 | 0.06 | 99 | 5.37 | 0.42 | 92 |
| 2 | 5.79 | 5.85 | 0.06 | 99 | 5.34 | 0.45 | 92 | 5.47 | 0.32 | 94 |
| 3 | 5.79 | 5.32 | 0.47 | 91 | 5.22 | 0.57 | 90 | 5.52 | 0.27 | 95 |
| 4 | 5.79 | 5.45 | 0.34 | 94 |  |  |  | 5.87 | 0.08 | 98 |
|  | | **Rata – rata Akurasi** | | **94.75** | **Rata – rata Akurasi** | | **93.66** | **Rata – rata Akurasi** | | **94.75** |

Dari hasil diatas dapat diketahui bahwa tingkat akurasi paling rendah diperoleh ketika pengambilan dilakukan waktu siang hari. Sedangkan untuk pagi dan soe hari memiliki tingkat akurasi yang sama.

* 1. **Eksperimen Berdasarkan Sudut Pengambilan**

**Tabel 4.** Analisa Hasil Pehitungan Berdasarkan Waktu Pengambilan

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Ke** | **Manual (Km/h)** | **0 Derajat** | | | **45 Derajat** | | |
| **Prorgam (Km/h)** | **Perbedaan (Km/h)** | **Akurasi (%)** | **Prorgam (Km/h)** | **Perbedaan (Km/h)** | **Akurasi (%)** |
| 1 | 5.79 | 5.5 | 0.29 | 95 | 5.32 | 0.47 | 91 |
| 2 | 5.79 | 5.85 | 0.06 | 99 | 5.45 | 0.34 | 94 |
| 3 | 5.79 | 5.73 | 0.06 | 99 | 5.22 | 0.57 | 90 |
| 4 | 5.79 | 5.34 | 0.45 | 92 | 5.52 | 0.27 | 95 |
| 5 |  | 5.37 | 0.42 | 92 | 5.87 | 0.08 | 98 |
| 6 |  | 5.47 | 0.32 | 94 |  |  |  |
|  | | **Rata – rata Akurasi** | | **95.16** | **Rata – rata Akurasi** | | **93.6** |

Dilihat dari hasil analisa berdasarkan sudut pengambilan video, tingkat akurasi paling tinggi adalah ketika sudut pengambilan 0 derajat. Sedangkan yang rendah adalah ketika sudut pengambilan 45 derajat.

1. **Kesimpulan**

Dari hasil pengujian dan analisa dapat disimpulkan bahwa hasil BLOB *analysis* dapat membantu proses *capture* dan pendeteksian kordinat obyek kendaraan, pemilihan penggunaan marka jalan sebagai acuan jarak sebenarnya lebih mudah penerapannya daripada menggunakan acuan panjang jalan atau panjang kendaraan, hasil akurasi perhitungan kecepatan tertinggi berdasarkan waktu pengambilannya adalah saat pengambilan video dilakukan pada waktu pagi hari dan sore hari dan jika berdasarkan sudut pengambilannya, akurasi tertinggi diperoleh ketika sudut pengambilannya adalah 0°.

1. **Daftar Pustaka**

[1] Ubaidillah Umar, Reni Soelistijorini, Haryadi Amran Darwito. 2011, “Tracking Arah Gerakan Telunjuk Jari Berbasis Webcam Menggunakan Metode Optical Flow”. Surabaya

[2] Dimas Arioputra. 2012, “Analisa Perbandingan Akurasi Metode Optical Flow Dan Gaussian Mixture Model untuk Sistem Pemantau Lalu Lintas Berbasis Computer Vision”. Depok

[3] Pembatasan kecepatan-http://id.wikipedia.org/wiki/Pembatasan\_ kecepatan (Diakses 12 Oktober 2014)

[4] Kecepatan rencana-http://id.wikipedia.org/wiki/Kecepatan\_rencana (Diakses 12 Oktober 2014)

[5] Thresholding citra - http://kuliahinformatika.wordpress.com/ 2010/02/13/buku-ta- thresholding-citra/ (Diakses 14 Oketober 2014)

[6] Indu .S, Manjari Gupta and Prof. Asok Bhattacharyya.2011.” Vehicle Tracking and Speed Estimation using Optical Flow Method”. ECE Department, Delhi Technological University India

[7] Yuwono, Bambang.2010.“Image Smoothing Menggunakan Mean Filtering, Median Filtering, Modus Filtering Dan Gaussian Filtering”. Teknik Informatika UPN “Veteran” Yogyakarta

[8] Arifin, Agus Zainal, Arya Yudhi Wijaya, Laili Cahyani.” Algoritma Thresholding Adaptif Berdasarkan Deteksi Blok Terhadap Citra Dokumen Terdegradasi”. Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS), Surabaya

[9] Singh, T.Romen , Sudipta Roy, O.Imocha Singh, Tejmani Sinam , Kh.Manglem Singh .2011.” A New Local Adaptive Thresholding Technique in Binarization”. IJCSI International Journal of Computer Science Issues