Deteksi Kecepatan Kendaraan Berjalan di Jalan Menggunakan OpenCV

Andrew, Joko Lianto Buliali, dan Arya Yudhi Wijaya
Departemen Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: joko@cs.its.ac.id

Abstrak*—Saat ini, di berbagai kota telah dipasang CCTV pada setiap ruas jalan. Dari CCTV, dapat diketahui kondisi lalu lintas, namun tidak dapat diketahui kecepatan setiap kendaraaan. Oleh karena itu, dibuat perangkat lunak yang dapat mendeteksi kecepatan kendaraan di ruas jalan dari video yang diambil oleh CCTV. Tujuan lainnya adalah untuk mengetahui perbedaan hasil deteksi kecepatan dengan berbagai nilai FPS (*Frame Per Second*). *Input* untuk aplikasi ini adalah video (.avi). Pertama, sistem mengambil *Region of Interest* (ROI). Selanjutnya, sistem melakukan *background subtraction*, membuat garis awal dan akhir, memperbarui posisi kendaraan, dan menyimpan hasil kecepatan rata-rata kendaraan ke berkas Excel (.xls). Skenario uji coba dilakukan berdasarkan nilai FPS pada video (30 FPS, 27 FPS, 25 FPS, dan 20 FPS). Setiap skenario terdapat sub-skenario berdasarkan posisi koordinat garis akhir {(296,0); (282,0); (270,0); dan (248,0)}. Pengujian dilakukan 5 kali setiap skenario, lalu dibandingkan dengan hasil sebenarnya untuk mendapatkan nilai *error* pada sistem. *Error* terkecil yang dihasilkan sistem sebesar 2,75% dengan posisi koordinat garis akhir di (282,0) pada skenario 30 FPS.

***Kata Kunci*—CCTV, OpenCV, ROI, FPS, Deteksi Kecepatan.**

I. PENDAHULUAN

DEWASA ini, di berbagai kota telah dipasang CCTV pada setiap ruas jalan, terutama pada kota Surabaya. Namun, petugas lalu lintas belum sepenuhnya memanfaatkan kegunaan CCTV. Petugas lalu lintas hanya dapat mengetahui kondisi pada suatu jalan, seperti kepadatan, kecelakaan dan lain-lain. Kecepatan kendaraan sangat berguna untuk diketahui, karena dapat ditentukan apakah kecepatan suatu kendaraan diatas hukum batas kecepatan yang berlaku [1].

Studi ini merupakan implementasi dari kebutuhan mengenai deteksi kecepatan kendaraan di jalan dari video. Penelitian yang dilakukan terkait dengan implementasi *library* OpenCV pada aplikasi yang akan dibuat.

Konsep yang digunakan adalah pengambilan *region of interest* (ROI) dari video, dilakukan *background subtraction* untuk mendapatkan latar depan, deteksi kendaraan berjalan, menggambar kotak pada kendaraan yang terdeteksi, dan memperbarui posisi kendaraan yang terdeteksi [2].

Diskusi pada jurnal ini dibagi dalam struktur sebagai berikut: Bab II membahas dasar teori yang digunakan dalam percobaan ini. Bab III membahas hasil dan diskusi percobaan yang dilakukan. Bab IV membahas kesimpulan dari hasil yang didapatkan dalam penelitian ini.

II. TINJAUAN PUSTAKA

A. *OpenCV*

OpenCV (*Open Source Computer Vision*) adalah *library* dari fungsi pemrograman untuk visi komputer. OpenCV menggunakan lisensi BSD, sehingga OpenCV gratis, baik untuk penggunaan akademis maupun komersial. OpenCV dapat digunakan pada bahasa pemrograman C, C++, Python dan Java. OpenCV mendukung Windows, Linux, Android, iOS dan Mac OS. OpenCV dibuat dalam bahasa pemrograman C / C++ dengan OpenCL, dan memiliki lebih dari 2500 algoritma yang telah dioptimalkan. *Library* ini digunakan oleh pengguna di seluruh dunia. OpenCV memiliki lebih dari 47 ribu orang dari komunitas pengguna dan perkiraan jumlah pengunduhan lebih dari 14 juta kali [3], [4].

Open Computing Language (OpenCL) adalah suatu kerangka kerja untuk membuat aplikasi yang mengeksekusi seluruh platform heterogen, seperti *Central Processing Units* (CPU), *Graphics Processing Units* (GPU), *Digital Signal Processors* (DSPs), *Field-Programmable Gate Arrays* (FPGAs) dan prosesor lainnya [5].

B. *Frame Rate*

Frame rate (dinyatakan dalam *Frame Per Second* atau FPS) adalah frekuensi dimana gambar (*frame*) berturut-turut ditampilkan. Istilah ini berlaku sama untuk film, video kamera, komputer grafis, dan sistem *motion capture* [6].

C. *Background Subtraction*

Background subtraction adalah teknik di bidang pengolahan citra untuk pengambilan latar depan gambar. *Background subtraction* merupakan teknik untuk mendeteksi benda bergerak dalam video. Namun, *background subtraction* memiliki kekurangan, yaitu pada video dengan kondisi diluar ruangan yang tidak stabil, seperti hujan, angin, dan perubahan pencahayaan [7], [8]. Algoritma yang akan digunakan berdasarkan penelitian oleh Z.Zivkovic [9], [10], yang berjudul "*Improved adaptive Gausian mixture model for background subtraction*" pada tahun 2004 dan "*Efficient Adaptive Density Estimation per Image Pixel for the Task of Background Subtraction*" pada tahun 2006. Terdapat metode berdasarkan *Gaussian Mixture Model* (GMM) yang tercantum pada penelitian tersebut. *Background subtraction* melakukan analisa pada setiap piksel, apakah suatu piksel merupakan latar

belakang atau latar depan yang sesuai dengan definisi pada (1).

$$P(x_t) > C_{thr} \quad (1)$$

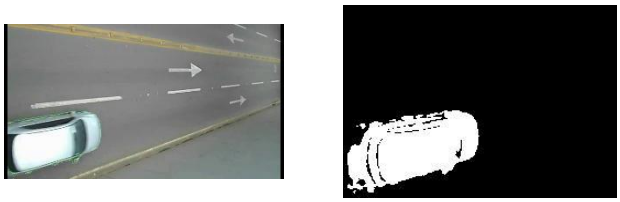
dimana $p(x_t)$ adalah probabilitas nilai piksel tertentu, x_t adalah titik atau piksel pada waktu t, dan c_{thr} adalah nilai threshold. Estimasi model pada *Gaussian mixture model* (GMM) dapat dilihat pada (2), (3), dan (4)

$$P(x_t) = \sum_{m=1}^{K} \omega_m . N\left(x_t \mid \mu_m, \sigma_m^2 I\right) \quad (2)$$

$$N\left(x_t \mid \mu_m, \sigma_m^2 I\right) = \frac{1}{(2\pi)^{D/2}} . \frac{1}{\left|\sigma_m^2 . I\right|^{1/2}} . e^{z} \quad (3)$$

$$z = -\frac{1}{2}\left(x_t - \mu_m\right)^T . \sigma_m^2 . I^{-1} . \left(x_t - \mu_m\right) \quad (4)$$

dimana K adalah distribusi K *Gaussian*, ω_m adalah estimasi bobot campuran, μ_m adalah estimasi means, σ^2_m adalah estimasi variance, I adalah matriks identitas, D adalah ukuran dimensi, e adalah bilangan *euler* [11], dan T adalah periode. Gambar sebelum dan sesudah penerapan background subtraction dapat dilihat pada Gambar 1.



(a) (b)

Gambar 1. (a) sebelum dan (b) setelah background subtraction

D. *Region of Interest*

Region of Interest (ROI) adalah sampel yang diambil dari satu set data untuk tujuan tertentu. Contoh penggunaan ROI adalah mengetahui batas-batas tumor pada gambar untuk mengukur ukurannya. Dalam sistem informasi geografis (GIS), ROI dapat dipahami sebagai pilihan poligonal dari peta 2D [12]. Untuk melihat perbedaan gambar sebelum dan sesudah pemilihan ROI dapat dilihat pada Gambar 2.

(a) (b)

Gambar 2. (a) sebelum dan (b) setelah ROI

E. *Pencarian Bentuk Mobil*

Algoritma pencarian bentuk mobil didapat dari penelitian oleh S.Suzuki [13] yang berjudul "*Topological Structural Analysis of Digitized Binary Images by Border Following*" pada tahun 1985. Peneliti mengusulkan 2 algoritma untuk mendeteksi objek pada gambar biner (hitam-putih).

Algoritma 1:

1. *Input* algoritma adalah suatu gambar biner dan lakukan analisa setiap piksel (i,j). Hentikan analisa jika terdapat suatu piksel (i,j) yang merupakan batas luar (Gambar 3a) atau batas lubang (Gambar 3b). Jika piksel (i,j) memenuhi kedua kondisi tersebut, maka piksel (i,j) dianggap sebagai titik awal dari batas luar. Tetapkan suatu angka yang unik pada perbatasan yang baru ditemukan, disebut sebagai NBD.

(a) (b)

Gambar 3. kondisi perbatasan titik awal (i,j) dari (a) batas luar dan (b) batas lubang.

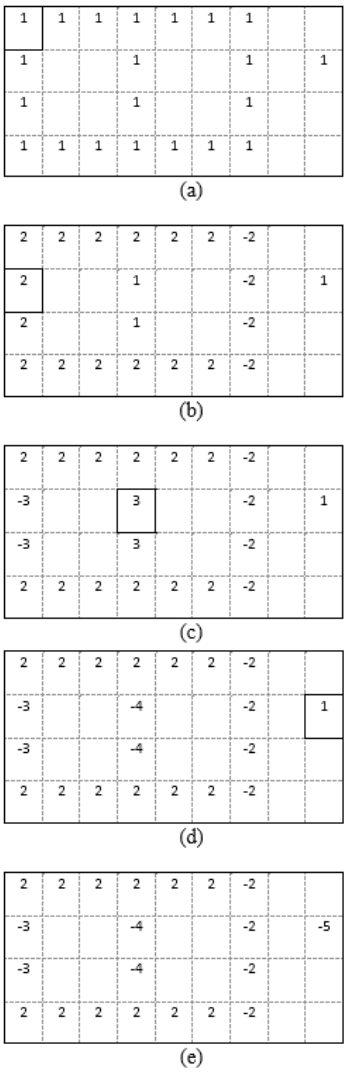
2. Tentukan *parent* perbatasan dari perbatasan yang baru ditemukan. Selama proses analisa, simpanlah suatu nomor urut perbatasan, disebut sebagai LNBD, dari perbatasan yang baru ditemukan. LNBD ini merupakan *parent* perbatasan dari perbatasan sebelumnya (NBD). Oleh karena itu, dapat ditentukan urutan nomor *parent* perbatasan (LNBD) dari perbatasan sebelumnya (NBD) dengan jenis perbatasan menurut Tabel 1.

Tabel 1.
Decision rule untuk parent perbatasan dari perbatasan B

Type of B	Type of border B′ with sequential number LNBD	
	Outer border	Hole border
Outer border	The parent border of the border B′	The border B′
Hole border	The border B′	The parent border of the border B′

3. Ikuti perbatasan yang ditemukan dari titik awal dengan menandai piksel. Ketentuan penandaan setiap piksel sebagai berikut
   a. Jika perbatasan sekarang adalah diantara 0-komponen yang berisi piksel (p,q+1) dan 1-komponen yang berisi piksel (p,q), maka ubah nilai piksel (p,q) dengan -NBD.
   b. Jika tidak, tetapkan nilai piksel (p,q) dan piksel (p,q+1) dengan NBD kecuali piksel berada pada perbatasan yang sudah diikuti.
4. Penandaan ini mencegah piksel (p,q) menjadi perbatasan titik awal. Jika tanda suatu piksel bernilai positif maka bisa disebut dengan batas terluar suatu objek. Lakukan algoritma ini hingga pengamatan piksel mencapai sudut kanan bawah gambar.

Gambar 4 merupakan ilustrasi algoritma 1 yang menunjukkan nilai piksel suatu gambar.



Gambar 4. ilustrasi proses algoritma 1. Figur ini menunjukkan nilai piksel. Piksel dalam kotak adalah titik awal proses.

Untuk algoritma 2, salah satu perbedaannya adalah pengamatan hanya pada perbatasaan dengan titik awal batas luar saja atau *outer border* (Gambar 3a), sehingga pada algoritma 2 lebih efisien karena waktu komputasi lebih cepat.

F. *Kecepatan Kendaraan pada Video*

Formula kecepatan pada umumnya adalah membagi jarak tempuh suatu kendaraan dengan waktu tempuh seperti pada (5) dan (7). Formula jarak tempuh kendaraan (6) didapat dari penelitian oleh Chan Chia Y [14].

$$Speed = \frac{Dist}{Time} \tag{5}$$

$$Dist = Df \times \left(\frac{D}{Dx}\right) \times \left(P_n - P_0\right) \tag{6}$$

$$Time = Tf \times \left(t(n) - t(0)\right) \tag{7}$$

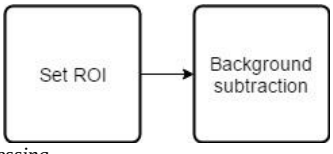
dimana D*f* adalah konversi jarak dari meter ke kilometer (0.001), D adalah jarak sesungguhnya (penulis menggunakan perkiraan), D_x adalah jarak antara dua garis area deteksi (awal dan akhir) dalam piksel, P_n adalah posisi ujung kanan kendaraan pada t_n, P_0 adalah posisi ujung kanan kendaraan pada t_0, T*f* adalah konversi waktu dari millidetik ke jam (1/1000*60*60), t(n) adalah waktu akhir, dan t(0) : waktu awal.

III. ANALISIS DAN HASIL

Aplikasi yang dibuat pada penelitian ini adalah aplikasi desktop. Untuk alur jalannya sistem digambarkan pada Gambar 7. Penjelasan keterangan pada penomoran Gambar 7 adalah sebagai berikut:

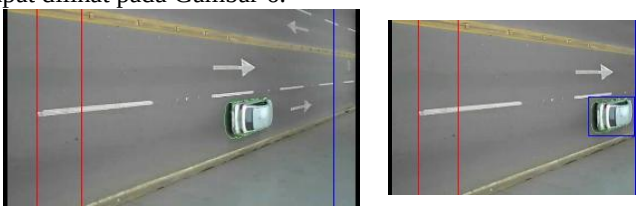
1. *Input* video.
2. Video dilakukan *preprocessing* (pengambilan ROI dan melakukan *background subtraction*).
3. Hasil *preprocessing* dilakukan *processing* (pencarian bentuk kendaraan, pemasangan garis awal dan akhir, pembaruan posisi kendaraan, penghitungan kecepatan kendaraan, dll).
4. Hasil akan disimpan dalam bentuk berkas berekstensi Excel.
5.

A.*Preprocessing*



Gambar 5. *Preprocessing*

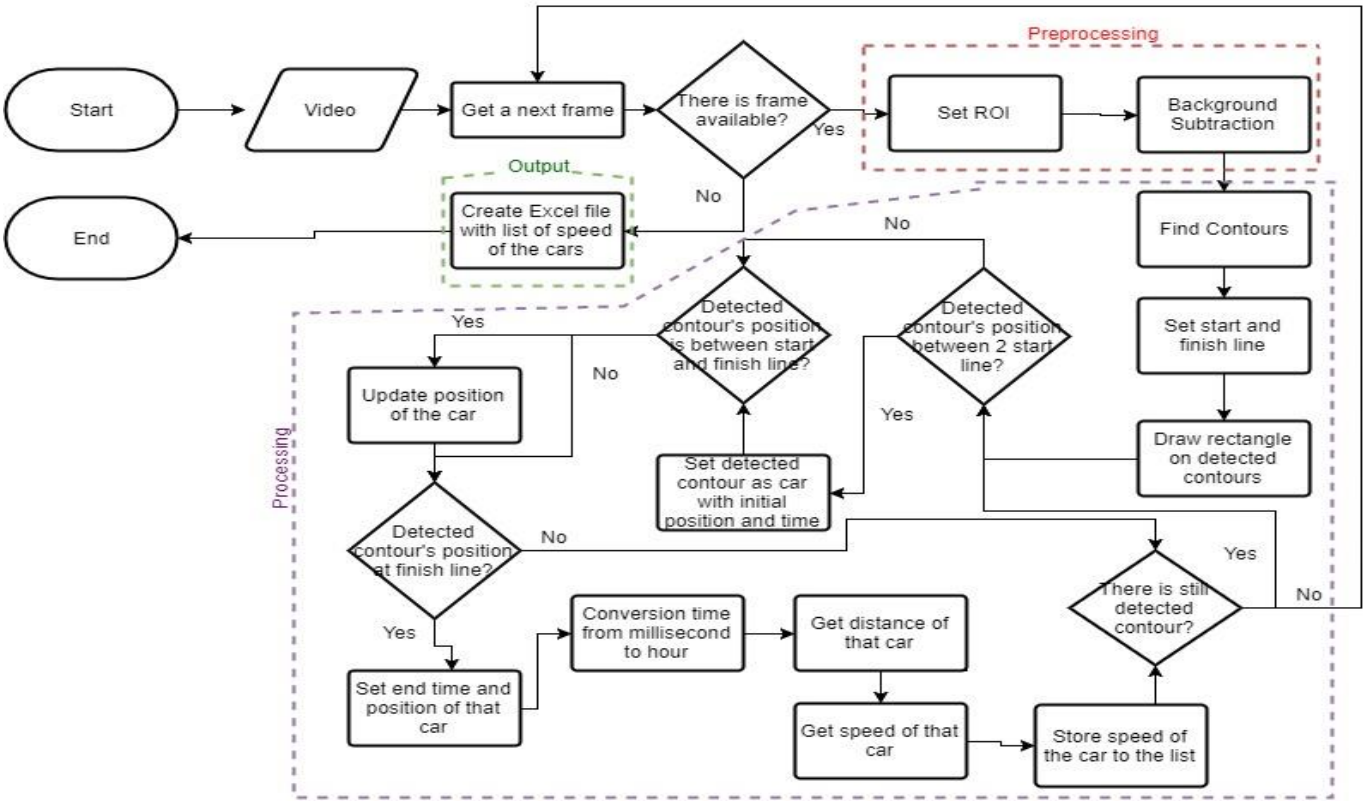
Pada Gambar 5 merupakan gambaran umum untuk *preprocessing* yang diambil dari *flowchart* sistem pada Gambar 7. Setelah video masuk dalam sistem, sistem akan mengambil ROI yang sudah ditentukan. Tahap pengambilan ROI ini bertujuan untuk lebih fokus menganalisis area, dan menghindari *error* pada sistem. *Error* terjadi pada saat mobil terpotong oleh batas resolusi video. Ilustrasi pengambilan ROI dapat dilihat pada Gambar 6.



(a) (b)

Gambar 6. Ilustrasi (a) sebelum dan (b) setelah pengambilan ROI

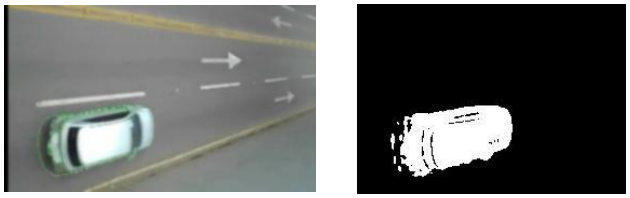
Setelah pengambilan ROI, tahap selanjutnya adalah *background subtraction* untuk mendapatkan benda bergerak pada video. Ilustrasi *background subtraction* dapat dilihat pada Gambar 8.



Gambar 7. Flowchart Sistem

B. *Processing*

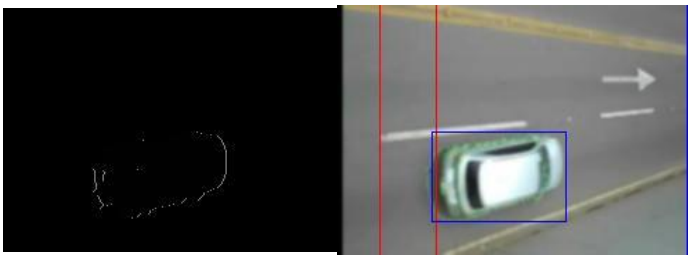
Pada Gambar 10 merupakan *flowchart* untuk *processing* yang diambil dari *flowchart* sistem pada Gambar 7. Langkah pertama adalah pencarian bentuk mobil dari gambar hasil *preprocessing* dengan menggunakan algoritma pada penelitian oleh S.Suzuki [13]. Langkah selanjutnya adalah pemasangan 3 garis (dua garis awal dan satu garis akhir), dan pembuatan kotak pada mobil yang terdeteksi. Ilustrasi pembuatan kotak pada mobil yang terdeteksi dapat dilihat pada Gambar 9. Sistem akan melakukan pengecekan apakah sisi kanan kotak (ujung mobil) berada diantara 2 garis awal. Jika iya, maka informasi mobil (*id*, posisi awal dan waktu awal mobil) tersebut akan disimpan. Ilustrasi kondisi ujung mobil pada posisi diantara 2 garis awal dapat dilihat pada Gambar 11(a). Jika iya, maka sistem akan menyimpan bentuk itu kedalam sebuah objek mobil dengan nilai id, posisi awal dan waktu awal (terdeteksi sebagai mobil).



(a) (b)

Gambar 8. Ilustrasi (a) sebelum dan (b) setelah *background subtraction*

Posisi mobil akan selalu diperbarui selama posisi mobil tersebut berada diantara garis awal dan garis akhir. Pada saat posisi mobil berada di garis akhir, waktu akhir dan posisi akhir mobil tersebut akan disimpan. Ilustrasi kondisi ujung mobil pada posisi garis akhir dapat dilihat pada Gambar 11(b). Selanjutnya, sistem akan mengkonversi waktu tempuh suatu mobil ke satuan jam dengan (7). Sistem juga akan menyimpan jarak tempuh dengan melakukan pengurangan antara posisi akhir dan posisi awal menggunakan (6). Setelah itu, dengan (5) didapatlah nilai kecepatan rata-rata suatu mobil.

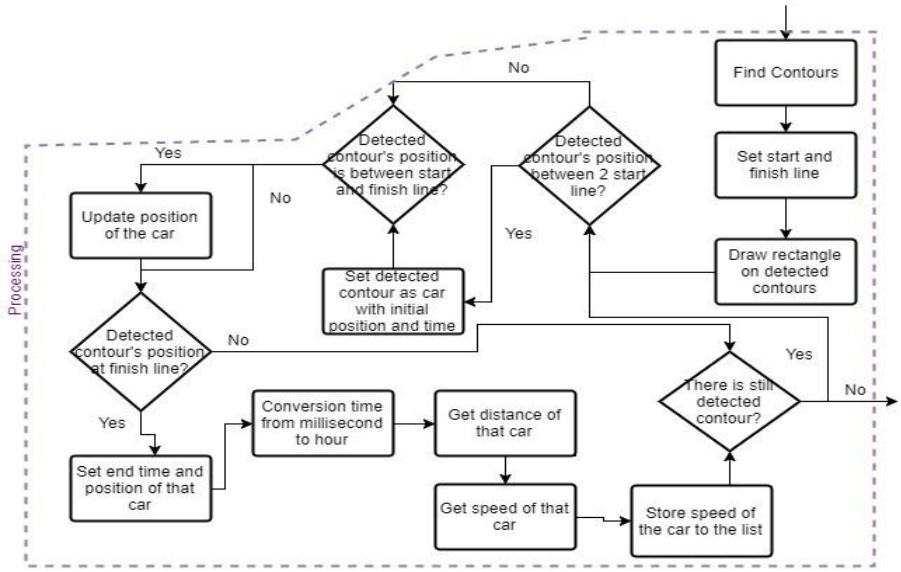


(a) (b)

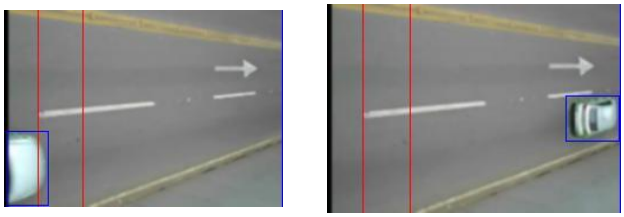
Gambar 9. Ilustrasi (a) penggunaan algoritma pencarian bentuk mobil dan (b) pembuatan kotak pada gambar setelah *background subtraction.*

C. *Hasil Percobaan*

Terdapat 4 skenario dengan perubahan nilai FPS dan 4 sub-skenario dengan perubahan garis akhir. Pemilihan skenario dan sub-skenario ini bertujuan untuk melihat dampak pada nilai *error* dari sistem. Penjelasan 4 skenario dan 4 sub-skenario dapat dilihat pada Tabel 2.



Gambar 10. *Processing*



(a) (b)

Gambar 11. (a) posisi sisi kanan kotak diantara 2 garis awal, dan (b) posisi sisi kanan kotak pada garis akhir

Ilustrasi skenario letak garis akhir dapat dilihat pada Gambar 12.

Tabel 2.
Skenario Pengujian

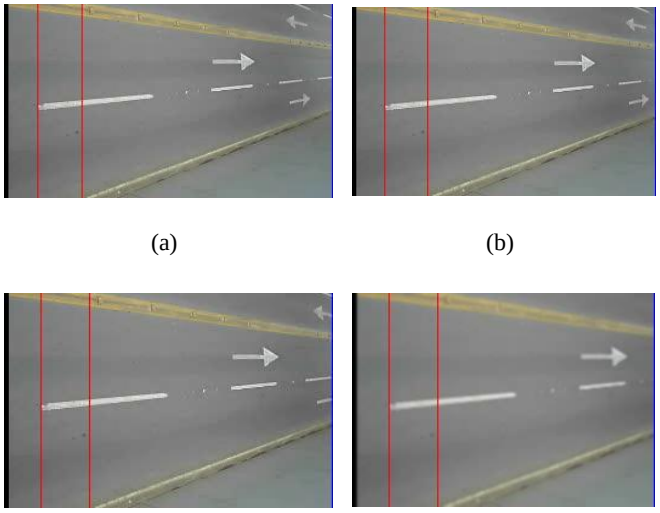
Skenario		FPS	Koordinat Garis Akhir	Jarak Sebenarnya (m)
1	a	30	(296,0) dan (296,240)	32
	b		(282,0) dan (282,240)	27
	c		(270,0) dan (270,240)	23
	d		(248,0) dan (248,240)	18
2	a	27	(296,0) dan (296,240)	32
	b		(282,0) dan (282,240)	27
	c		(270,0) dan (270,240)	23
	d		(248,0) dan (248,240)	18
3	a	25	(296,0) dan (296,240)	32
	b		(282,0) dan (282,240)	27
	c		(270,0) dan (270,240)	23
	d		(248,0) dan (248,240)	18
4	a	20	(296,0) dan (296,240)	32
	b		(282,0) dan (282,240)	27
	c		(270,0) dan (270,240)	23
	d		(248,0) dan (248,240)	18

Pada penghitungan *error* sistem, penulis akan membandingkan hasil skenario dengan hasil sebenarnya (kecepatan mobil sebenarnya).

Error sistem dihitung menggunakan (8).

$$Error = \left(\frac{KS - KU}{KS} \right) \times 100\% \quad (8)$$

dimana KS adalah kecepatan sebenarnya dan KU adalah kecepatan uji coba.



(a) (b)

(c) (d)

Gambar 12. skenario letak garis akhir (a) sub-skenario a, (b) sub-skenario b, (c) sub-skenario c, (d) sub-skenario d

Tabel 3.
Rata-rata *error* dari beberapa skenario

Skenario Start/Finish Line	Skenario FPS 1	2	3	4
a	4,19	5,23	12,15	28,01
b	2,75	6,53	14,61	29,94
c	6,37	6,36	14,37	29,24
d	3,21	8,92	16,33	31,37
Rata-rata	4,13	6,76	14,37	29,64

Dapat dilihat pada Tabel 3, bahwa skenario FPS 1 (30 FPS) memiliki *error* paling kecil. Hal ini dapat disimpulkan bahwa pengurangan FPS meningkatkan *error* pada sistem, sehingga hasil yang terbaik terdapat pada video dengan 30 FPS. Dapat dilihat pada tabel tersebut, bahwa peletakan garis awal dan akhir yang memiliki *error* paling kecil adalah skenario b pada video dengan 30 FPS. Oleh karena itu, hasil yang terbaik untuk keseluruhan terdapat pada skenario dengan video 30 FPS dan peletakan garis akhir ((282,0) dan (282,240)) dengan jarak sebenarnya 27 meter.

IV. KESIMPULAN DAN SARAN

Error terkecil yang dihasilkan oleh sistem sebesar 2,75%. Pengurangan FPS pada video dapat meningkatkan *error* pada sistem. Video dengan 30 FPS memiliki hasil yang terbaik. Peletakan garis akhir berpengaruh pada *error* yang dihasilkan oleh sistem. Peletakan garis akhir (282,0) dan (282,240) dengan jarak sebenarnya 27 meter memiliki hasil yang paling baik.

Penggunaan *enhancement* pada gambar yang mungkin berpotensi mempengaruhi hasil deteksi kecepatan. Peningkatan tampilan antarmuka pada aplikasi.

DAFTAR PUSTAKA

[1] A. G. Rad, A. Dehghani, and M. R. Karim, "Vehicle speed detection in video image sequences using CVS method," *Int. J. Phys. Sci.*, vol. 5, no. 17, pp. 2555–2563, 2010.

[2] H. S. Sundoro and A. Harjoko, "VEHICLE COUNTING AND VEHICLE SPEED MEASUREMENT BASED ON VIDEO PROCESSING," *J. Theor. Appl. Inf. Technol.*, vol. 84, no. 2, p. 233, 2016.

[3] "OpenCV," *Wikipedia*. 29-Apr-2017.

[4] "OpenCV library." [Online]. Available: http://opencv.org/. [Accessed: 07-May-2017].

[5] "OpenCL," *Wikipedia*. 25-Apr-2017.

[6] "Frame rate," *Wikipedia*. 29-Apr-2017.

[7] "Background subtraction," *Wikipedia*. 08-Feb-2017.

[8] "OpenCV: Background Subtraction." [Online]. Available: http://docs.opencv.org/trunk/db/d5c/tutorial_py_bg_subtraction.html. [Accessed: 07-May-2017].

[9] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, 2004, vol. 2, pp. 28–31.

[10] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognit. Lett.*, vol. 27, no. 7, pp. 773–780, May 2006.

[11] "e (mathematical constant)," *Wikipedia*. 06-May-2017.

[12] "Region of interest," *Wikipedia*. 12-Jan-2017.

[13] S. Suzuki and Abe, Keiichi, "Topological structural analysis of digitized binary images by border following," *Comput. Vis. Graph. Image Process.*, vol. 30, no. 1, pp. 32–46, 1985.

[14] C. Chia Yik, "Vehicle Tracking and Speed Estimation System," University Malaysia Pahang, Malaysia, Jun. 2012.