ISBN 978-602-8692-34-2

# Sistem Penghitung Jumlah Kendaraan Di Lintasan Dua Arah Menggunakan *Library CVBlob*

Sri Purwiyanti
*Jurusan Teknik Elektro*
*Universitas Lampung*
Bandar Lampung, Indonesia
sri.purwiyanti@eng.unila.ac.id

F.X. Arinto Setyawan
*Jurusan Teknik Elektro*
*Universitas Lampung*
Bandar Lampung, Indonesia
fx.arinto@eng.unila.ac.id

M. Hafizulahudin
*Jurusan Teknik Elektro*
*Universitas Lampung*
Bandar Lampung, Indonesia
hapizulahudin@gmail.com

Herlinawati
*Jurusan Teknik Elektro*
*Universitas Lampung*
Bandar Lampung, Indonesia
herlinawati@eng.unila.ac.id

Umi Murdika
*Jurusan Teknik Elektro*
*Universitas Lampung*
Bandar Lampung, Indonesia
umi.murdika@eng.unila.ac.id

***Abstrak*—Meningkatnya jumlah kendaraan yang beroperasi mengakibatkan masalah kemacetan di jalan raya, terutama di kota besar. Oleh karena itu, diperlukan pemantauan kepadatan lalulintas yang dapat mengetahui jumlah kendaraan yang melintas pada jalan raya, agar pihak yang berkepentingan dapat membuat kebijakan untuk mengatasi permasalahan transportasi tersebut. Penelitian ini bertujuan untuk merancang sistem pemantau dan penghitung jumlah kendaraan yang melintas di dua jalur jalan raya. Metode yang digunakan adalah berdasarkan pengolahan citra menggunakan microsoft visual studio 2013 dengan bahasa c++ dan dilengkapi dengan sebuah library dari open cv. Hasil pengujian yang diperoleh memperlihatkan bahwa sistem ini berhasil melakukan pendeteksian dan penghitungan jumlah kendaraan yang melintas di jalan dua jalur dengan tingkat error sekitar 10 %. Namun sistem ini belum berhasil diterapkan pada jalan yang padat sehingga masih diperlukan perbaikan lebih lanjut.***

***Kata Kunci*— CvBlob, Open cv ,Visual Studio, Pengolahan citra***

## I. PENDAHULUAN

Untuk menentukan kepadatan rata-rata lalu lintas diperlukan adanya survey penghitungan jumlah kendaraan yang melintas di jalan raya. Pelaksanaan survey tersebut biasanya dilakukan oleh seorang pengamat sehingga dimungkinkan terjadinya *human error* dalam proses penghitungannya. Selain itu, penghitungan yang dilakukan oleh manusia memerlukan biaya tersendiri untuk setiap pelaksanaannya sehingga kurang efisien [1].

Berdasarkan pada permasalahan yang ada maka telah dilakukan penelitian untuk membuat sistem penghitung jumlah kendaraan di jalan raya. Pada penelitian sebelumnya dilakukan penghitungan hanya untuk jalan satu jalur [2]. Untuk itu maka penelitian ini bertujuan untuk membuat suatu sistem penghitung jumlah kendaraan di lintasan jalan raya dua jalur dengan menggunakan bahasa pemrograman C++ dengan *library* OpenCV. OpenCV digunakan karena memiliki banyak subprogram atau *library* yang dapat dikombinasikan sehingga memiliki berbagai fungsi dalam pemrograman yang berkaitan dengan pengolahan citra digital [3].

.

## II. DETEKSI OBYEK

Pada pengolah citra, proses pendeteksian obyek dapat dilakukan dengan memisahkan citra *foreground* dengan *background*-nya. Terdapat metode-metode dalam pendeteksian obyek bergerak, beberapa diantaranya adalah *background substraction* dan *haar-like feature* [4]. Disimpulkan bahwa metode *background substraction* memiliki keunggulan, karena lebih peka terhadap perubahan suatu obyek ataupun perubahan lingkungan di sekitar obyek. Selain itu metode *background substraction* dapat bekerja secara optimal dalam berbagai kondisi, seperti perubahan warna obyek, perubahan *background*, kecepatan obyek, serta perubahan intensitas cahaya di lingkungan sekitar obyek.

Prinsip kerja dari metode *background substraction* ini adalah dengan cara substraksi, *frame* saat ini dengan *frame* sebelumnya, kemudian hasil substraksi ini akan dianalisa untuk menemukan pergerakan obyek pada citra video. Kelemahan dalam metode ini, banyaknya derau yang ditimbulkan akibat terlalu sensitifnya sistem, sehingga obyek menjadi sulit untuk dikenali. Oleh karena itu biasanya *background substraction* ditambah dengan metode *Morphological Image Processing*.

### A. Background Substraction

*Background Subtraction* adalah proses untuk menemukan obyek pada gambar dengan cara membandingkan gambar yang ada dengan sebuah model latar belakang. Prosedur *Background Subtraction* terdiri dari 3 tahap, yaitu *pre-processing*, *background modeling*, dan *foreground detection* [5,6].

Pada tahap *pre-processing* data mentah dari kamera (atau input lainnya) diproses menjadi bentuk yang dapat dimengerti oleh bagian program lain. Pada tahap ini akan dilakukan eliminasi obyek kecil seperti peningkatan kualitas pada citra (kecerahan, kontras, dsb). Perbaikan citra juga dilakukan sesuai kebutuhan seperti *resize* (mengubah resolusi atau ukuran suatu citra) dan *cropping* (penghapusan bagian sudut dari citra.

Tahap *background modeling* bertujuan untuk membentuk model *background* yang konsisten, namun

tetap dapat beradaptasi dengan perubahan lingkungan yang ada. Algoritma *background modelling* sendiri sangat banyak, namun pada penelitian ini dipakai *Approximated Filter* yaitu hanya pendekatan batas atas dan batas bawah berdasarkan piksel yang didapat, karena proses komputasinya cepat dan hasilnya cukup memuaskan.

Pada tahap *foreground detection*, dilakukan proses ekstraksi *foreground* dari *background*. Secara sederhana hal ini dilakukan dengan persamaan berikut [5]:

$$R(r,c) = I(r,c) - B(r,c) \quad (1)$$

dengan R = hasil foreground
I = citra 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). Nilai *threshold* adalah suatu nilai untuk mentolerasi *error* yang mungkin terjadi dan threshold memang biasanya dipakai untuk mengurangi *error* yang disebabkan oleh suatu derau.

### B. Morphological Image Processing(MIP)

MIP merupakan teknik pengolahan citra dengan menggunakan bentuk (*shape*) sebagai bentuk acuan. Operasi morphologi tergantung pada suatu urutan kemunculan dari piksel tanpa memperhatikan nilai numerik dari piksel tersebut. Dengan demikian teknik ini sangat baik apabila digunakan untuk melakukan pengolahan citra biner dan citra dalam bentuk *grayscale*. Operasi morphologi standar yang dilakukan adalah proses erosi dan dilatasi [7].

Proses Dilatasi merupakan suatu proses atau teknik untuk memperbesar segmen suatu obyek dalam bentuk citra biner dengan menambahkan suatu lapisan sekeliling obyek tersebut dengan menggunkan persamaan sebagai berikut, dengan A adalah sebuah citra awal dan B adalah *structuring* atau suatu lapisan dari sekeliling obyek citra tersebut.

$$A + B = C \quad (2)$$

Operasi erosi yaitu suatu proses pemindahan atau pengurangan titik piksel pada batas suatu obyek citra digital. Operasi erosi dilakukan menggunakan persamaan

$$A - B = C \quad (3)$$

Proses dilatasi dan operasi erosi merupakan suatu teknik penambahan dan pengurang pada titik piksel. Jumlah piksel yang ditambahkan atau yang dikurangkan dari batas obyek pada citra digital masukan tergantung pada ukuran dan bentuk dari *structuring element* yang digunakan.

### C. CvBlob

*CvBlob* merupakan sebuah *library* tambahan pada *OpenCv* yang berisi algoritma dan dapat digunakan untuk menentukan suatu grup dari piksel yang saling terhubung. Metode ini sangat berguna untuk mengidentifikasi obyek yang terpisah-pisah pada suatu citra, atau dapat digunakan sebagi penghitung jumlah dari suatu obyek pada citra.

Metodenya disebut *blob detection*, yaitu suatu algoritma yang mendeteksi kumpulan titik-titik piksel yang memiliki warna berbeda (terang atau gelap) dari latar belakang dan menyatukannya dalam suatu region. Dalam deteksi *blob*, algoritma yang di pakai adalah algoritma *growing ragions*. Algoritma ini digunakan untuk menemukan *blob* di dalam gambar dan bisa diaplikasikan di *squence image*. Konsep dari algoritma ini adalah menampilkan sebuah citra sebagai matriks piksel dan nilai garis yang sudah pasti [8].

### D. Garis Deteksi

Garis deteksi merupakan sebuah garis hitung yang dilalui oleh obyek yang melintas di jalan raya. Garis deteksi pada penelitian ini terdiri dari dua garis yaitu garis kiri dan garis kanan. Prinsip kerja dari garis deteksi yaitu apabila sebuah obyek yang ter-*boundingbox* bergerak melalui garis deteksi, maka garis deteksi tersebut akan berganti warna yaitu menjadi berwarna putih untuk garis deteksi sebelah kiri dan hijau untuk sebelah kanan dengan notasi sebagai berikut [8,9]:

```
const cv::Scalar WARNA_PUTIH = cv::Scalar(255.0, 255.0, 255.0);
const cv::Scalar WARNA_KUNING = cv::Scalar(0.0, 255.0, 255.0);
const cv::Scalar WARNA_HIJAU = cv::Scalar(0.0, 200.0, 0.0);
const cv::Scalar WARNA_MERAH = cv::Scalar(0.0, 0.0, 255.0);
```

setelah dilakukan persamaan maka garis deteksi akan ditentukan penempatannya dengan notasi:

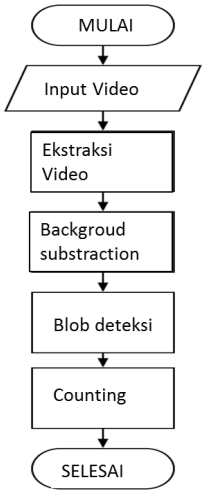
```
int posisiGarisHorizontal =
(int)std::round((double)frameImage_ke1.rows * 0.35);
posisiGarisHorizontal = posisiGarisHorizontal * 1.40;
PosisiGarisVertikal =
```

Agar obyek terdeteksi dan garis dapat berubah warna atau garis deteksi dapat berfungsi menjadi *line counting* maka digunakan notasi:

```
line kanan
lewatiGaris[0].x = 400;
lewatiGaris[0].y = posisiGarisHorizontal;
        lewatiGaris[1].x = frameImage_ke1.cols - 1;
        lewatiGaris[1].y = posisiGarisHorizontal;
line kiri
        lewatiGarisKiri[0].x = 0;
        lewatiGarisKiri[0].y = posisiGarisHorizontal;
lewatiGarisKiri[1].x = 300; lewatiGarisKiri[1].y =
posisiGarisHorizontal;   lewatiGarisKiri[1].y =
posisiGarisHorizontal;
```

## III. METODE

Gambar 1 memperlihatkan diagram alir dari metode yang digunakan. Proses yang dilakukan terbagi menjadi empat bagian, yaitu inisialisasi masukan video, *Background Subtraction*, deteksi *blob*, dan algoritma penghitung jumlah kendaraan yang melintas di dua jalur jalan raya. Dalam perancangan ini data untuk input berupa video dari hasil perekaman kamera yaitu berisi kumpulan informasi citra biner yang memuat intensitas piksel yang terdapat di dalamnya. Sedangkan keluaran dari hasil *Background Subtraction* berupa gambar *Foreground* yang akan diproses dalam suatu *blobtracking*.




Gambar 1. Diagram alir metode yang digunakan

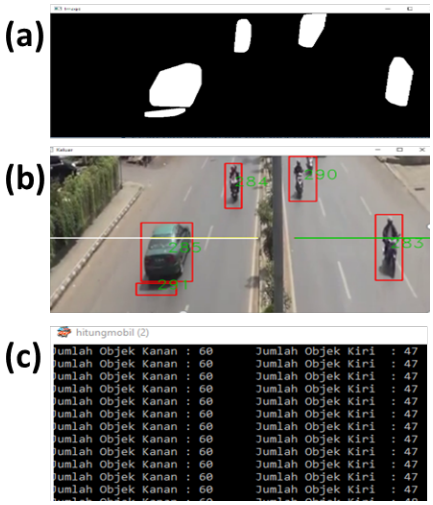
Pada perancangan ini pustaka *CvBlob* digunakan untuk mempermudah pembuat program dalam memisahkan tiap-tiap gumpalan yang terdapat dalam suatu gambar dan kemudian melabelinya sehingga antara gumpalan yang satu dengan gumpalan yang lainnya dapat dibedakan dengan identitas yang berbeda. Proses penghitungan jumlah kendaraan yang melintas di jalan raya terbagi menjadi empat bagian yaitu, pengaturan metode penghitungan, pengaturan garis hitung, proses penghitungan dan proses penandaan jalur (*track*). Pengaturan garis hitung dilakukan untuk mendapatkan garis hitung yang paling sesuai dengan kondisi jalan yang akan diamati.

## IV. HASIL DAN PEMBAHASAN

Setelah pembuatan program selesai dilakukan, contoh hasil eksekusi sistem dapat dilihat pada Gambar 2. Bagian atas adalah hasil dari *blob detection*, dimana obyek terdeteksi dan dipisahkan dari *background*-nya. Setelah itu dilakukan *blob tracking*, yaitu memberikan tanda kotak pada obyek hasil *blob detection*. Selanjutnya dihitung berapa banyak kotak yang melintasi garis deteksi, dan hasil perhitungan lalu ditampilkan di layar komputer.

Pengujian lalu dilakukan untuk mengetahui performa dari sistem yang dihasilkan. Penghitungan dilakukan secara terpisah antara jalur kiri dan kanan, kemudian hasilnya dibandingkan dengan hasil penghitungan secara manual. Tingkat kesalahan (error) lalu dihitung menggunakan rumus:

$$\%\ error = \frac{penghitungan\ sistem - penghitungan\ manual}{penghitungan\ manual}\ x\ 100\%$$



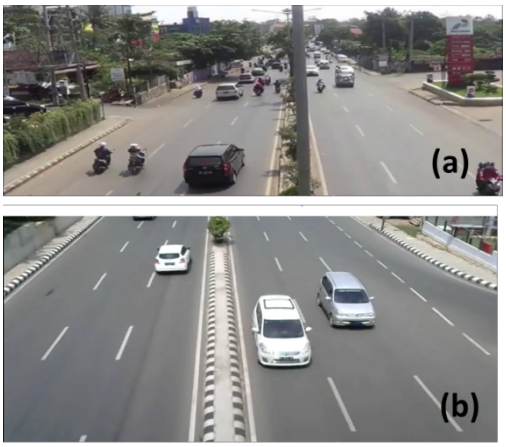

Gambar 2. Hasil dari proses: (a) *blob detection*, (b) *line counting*, dan (c) tampilan jumlah obyek

TABLE I. HASIL PENGHITUNGAN JUMLAH KENDARAAN PADA JALUR KANAN

| Frame | Jumlah Kendaraan | | error (%) |
|---|---|---|---|
| | *Sistem* | *Manual* | |
| 50 | 2 | 2 | 0 |
| 100 | 3 | 3 | 0 |
| 150 | 4 | 4 | 0 |
| 200 | 5 | 5 | 0 |
| 250 | 6 | 6 | 0 |
| 300 | 6 | 6 | 0 |
| 350 | 7 | 7 | 0 |
| 400 | 8 | 8 | 0 |
| 450 | 12 | 8 | 50 |
| 500 | 12 | 9 | 33 |
| 550 | 12 | 10 | 20 |
| 600 | 12 | 11 | 9 |
| 650 | 13 | 13 | 0 |
| 700 | 15 | 17 | 12 |
| 750 | 16 | 19 | 16 |
| 800 | 17 | 20 | 15 |
| 850 | 18 | 21 | 14 |
| 900 | 19 | 23 | 17 |
| Rata-rata error | | | 10,3 |

Tabel I memperlihatkan hasil pengujian untuk area jalur kanan, dengan jumlah *frame* sebanyak 900 *frame*. Dapat terlihat bahwa persentase *error* sebesar 0 % untuk *frame* 50 sampai 400 atau dengan kata lain sistem telah melakukan pendeteksian dengan benar. Namun untuk *frame* ke 450 jumlah *error* mulai berubah. Hal ini mungkin disebabkan faktor penghitungan obyek yang berkelipatan. Artinya satu obyek dapat terhitung lebih daripada satu saat nilai *framerate* tinggi. Fenomena ini sesuai dengan teori yang

menyatakan bahwa semakin detail sebuah video memberikan *frame* gambarnya maka semakin mudah sistem pengolahan citra mengolah data koordinatnya [7].




Gambar 3. Citra awal lokasi pengujian di : (a) Jl. Zainal Abidin dan (b). Jl. Pramuka, Bandarlampung

Selanjutnya pengujian dilakukan untuk membandingkan kinerja sistem pada dua jalan yang memiliki kepadatan berbeda. Pengujian dilakukan di Jln. Zainal Abidin, Bandarlampung dan Jl. Pramuka, Bandarlampung. Citra awal kedua gambar diperlihatkan pada Gambar.3.

TABLE II. PERBANDINGAN HASIL PENGHITUNGAN PADA JALAN YANG BERBEDA HANYA PADA JALUR KANAN

| Frame | Jumlah Kendaraan | | | | error (%) | |
|---|---|---|---|---|---|---|
| | *Sistem* | | *Manual* | | | |
| | *Jalan 1* | *Jalan 2* | *Jalan 1* | *Jalan 2* | *Jalan 1* | *Jalan 2* |
| 50 | 2 | 3 | 2 | 3 | 0 | 0 |
| 100 | 3 | 3 | 3 | 3 | 0 | 0 |
| 150 | 4 | 3 | 4 | 3 | 0 | 0 |
| 200 | 5 | 5 | 5 | 3 | 0 | 67 |
| 250 | 6 | 5 | 6 | 4 | 0 | 25 |
| 300 | 6 | 5 | 6 | 5 | 0 | 0 |
| 350 | 7 | 6 | 7 | 5 | 0 | 20 |
| 400 | 8 | 6 | 8 | 5 | 0 | 20 |
| 450 | 12 | 6 | 8 | 6 | 50 | 0 |
| 500 | 12 | 6 | 9 | 6 | 33 | 0 |
| 550 | 12 | 7 | 10 | 7 | 20 | 0 |
| 600 | 12 | 7 | 11 | 7 | 9 | 0 |
| 650 | 13 | 7 | 13 | 7 | 0 | 0 |
| 700 | 15 | 8 | 17 | 8 | 12 | 0 |
| 750 | 16 | 8 | 19 | 9 | 16 | 11 |
| 800 | 17 | 9 | 20 | 9 | 15 | 0 |
| 850 | 18 | 9 | 21 | 9 | 14 | 0 |
| 900 | 19 | 10 | 23 | 10 | 17 | 0 |

Hasil pengujian sampai frame ke-900, beserta prosentase *error* diperlihatkan pada Tabel II. Bila dilakukan pembandingan pada *framerate* yang tinggi pada kedua jalan, dapat terihat bahwa pada jalan yang lengang tingkat *error* ternyata lebih rendah daripada jalan yang padat. Hal ini terjadi mungkin karena terjadi kegagalan pada proses pengenalan citra pada obyek yang berhimpit di jalan yang padat. Solusi dari permasalahan ini adalah dengan mengembangkan metode *blob tracking* yang lebih baik agar dapat menghitung jumlah obyek secara *real time*.

## KESIMPULAN

Sistem yang dibuat telah berhasil melakukan menghitungan jumlah kendaraan pada dua jalur di jalan taya yang padat dan lenggang dengan rata-rata error sebesar 10,3 %. Hasil yang didapat menyatakan bahwa tingkat error yang besar biasanya terjadi pada framerate yang tinggi dengan kondisi jalan yang padat. Untuk mengatasi permasalahan tersebut untuk pengembangan selanjutnya diperlukan metode blob tracking yang lebih baik. Walaupun demikian, hasil yang didapat merupakan studi awal yang baik bagi mengembangan teknologi di bidang transportasi.

## UCAPAN TERIMA KASIH

Penulis mengucapkan terimakasih kepada Universitas Lampung atas support pembiayaan bagi riset ini melalui jalur DIPA FT UNILA.

## REFFERENSI

[1] WidiyardiniS.T.,"Implementasi Deteksi Gerakan untuk sistem pemantauan ruangan menggunakan webcam", Malang, 2010

[2] Cahaya, Fajar Mit. 2014. Perancangan Program Penghitung Jumlah kendaraan di Lintasan Satu Arah Menggunakan Bahasa Pemrograman C++ dengan Pustaka OpenCv. Skripsi. Universitas brawijaya.

[3] Stackoverflow, http://stackoverflow.com/review/suggested-edits/8575035 (diakses 22April 2017)

[4] Febriyanto, A. 2013. Analisis Kinerja Background Substraction dan Haar-Like Feature Untuk Monitoring Pejalan Kaki Menggunakan Kamera Webcam, Laporan Tugas Akhir. Fakultas Sains dan Teknologi. Universitas Islam Negeri Sunan Kalijaga

[5] Amaluddin, F., dkk. 2015. Klasifikasi Kendaraan Menggunakan Gaussian Mixture Model (GMM) dan Fuzzy Cluster Means (FCM), Jurnal EECCIS Volume 9 No.1, pp. 19-24. Program Studi Magister Teknik Elektro. Universitas Brawijaya

[6] Solichin, Achmad. Harjoko, Agus. 2013. Metode Background Substraction untuk Deteksi Obyek Pejalan Kaki pada Lingkungan Statis. Fakultas Teknologi Informasi. Universitas Budi Luhur.

[7] Kadir, Abdul. Susanto, Adhi. 2012. Pengolahan Citra Teori dan Aplikasi. Yogyakarta

[8] Fadliasyah, S.si (2007). Computer vision dan pengolahancitra Yogyakarta: Andi

[9] Barandiaran, J., Cortez, A., Nieto, M., Otaegui, O., Sanchez, P., Unzueta, L., 2011. Adaptive Multi-Cue Background Subtraction for Robust Vehicle Counting and Classification. IEEE Transactions on Intelligent Transportation System Journal.