OPEN ACCESS
Indo-JC Indonesian Journal of Computing
ISSN 2460-9056
socj.telkomuniversity.ac.id/indojc

Ind. Journal on Computing
Vol. 2, Issue. 2, Sept 2017. pp. 23-36
doi:10.21108/indojc.2017.22.174

Deteksi Kemacetan Lalu Lintas dengan Menggunakan Algoritma Monte Carlo

Bagus Gigih Adisalam [#1], P. H. Gunawan [*2], Mahmud Imrona [#3]

 # School of Computing, Telkom University
Jl. Telekomunikasi No. 1, Ters. Buah Batu Bandung 40257 Indonesia

[1] bagusgigihadisalam@gmail.com
[2] phgunawan@telkomuniversity.ac.id
[3] mahmudimrona@telkomuniversity.ac.id

Abstrak

Sensor dan kamera telah digunakan sekian lama dalam montoring lalu lintas untuk mendeteksi kemacetan lalu lintas. Kemacetan lalu lintas disebabkan karena perkembangan infrastruktur yang lambat tidak sebanding dengan peningkatan jumlah kendaraan. Penelitian ini bertujuan untuk mendeteksi status lalu lintas berdasarkan citra lalu lintas dengan sudut pandang yang berbeda. Citra lalu lintas diolah dengan menggunakan pengolahan citra menjadi 3 kondisi lalu lintas yaitu lancar, ramai, dan padat. Pengolahan citra merupakan metode untuk mengolah citra. Pengolahan citra dilakukan untuk mengubah citra RGB menjadi citra biner sehingga Monte Carlo dapat diterapkan dengan cara menghitung luas dari area piksel putih. Luas area putih pada citra biner digunakan untuk menentukan status lalu lintas. Hasil *testing 1* menunjukkan skenario 1 menghasilkan performa terbaik dengan precision ramai bernilai 44%, recall ramai bernilai 77%, precision lancar bernilai 92%, recall lancar bernilai 73%, dan akurasi bernilai 73%. Hasil testing 2 menunjukkan skenario 2 menghasilkan performa terbaik dengan precision padat bernilai 100%, recall padat bernilai 99%, dan akurasi bernilai 99%.

Kata Kunci: *grayscale*, pengolahan citra, segmen gambar, algoritma Monte Carlo, kemacetan lalu lintas, status lalu lintas

Abstract

Sensors and cameras have been used for so long in montoring traffic to detect traffic congestion. Traffic congestion is caused due to the slow development of infrastructure is not proportional to the increase in the number of vehicles. This study aims to detect traffic status based on traffic images with different viewpoints. Traffic image is processed by using image processing into 3 traffic conditions that is smooth, crowded, and solid. Image processing is a method for image processing. Image processing is done to convert RGB image into binary image so that Monte Carlo can be applied by calculating the area of the white pixel area. The white area of the binary image is used to determine the status of traffic. The result of testing 1 shows scenario 1 gives the best performance with 44% crowded precision, 77% crowded recall, 92% smooth precision, 73% smooth recall, and the accuracy is 73%. Test results 2 show scenario 2 yields the best performance with 100% solid precision, 99% solid recall, and 99% accuracy.

Keywords: grayscale, image processing, image segment, Monte Carlo algorithm, traffic congestion, traffic status

Received on Aug 2017. Accepted on Sept 2017

I. PENDAHULUAN

SENSOR dan kamera sering digunakan untuk mengontrol kemacetan lalu lintas di kota-kota besar [1]. Sistem deteksi kemacetan lalu lintas dapat menyediakan informasi kemacetan dan membantu dalam meningkatkan kebijakan lalu lintas [2]. Citra RGB adalah gambar yang memiliki warna primer yaitu merah, hijau, dan biru serta hasil kombinasi dari ketiga warna primer untuk menghasilkan warna-warna sekunder seperti coklat, ungu, dan lain-lain. Citra RGB mengandung 3 layer dimensi yaitu layer merah, layer hijau, dan layer biru. Ketiga layer dimensi memiliki nilai piksel yang berbeda-beda bergantung dari warna objek-objek yang ada pada citra RGB. Jika pada citra RGB memiliki objek dengan kadar warna merah yang tinggi maka objek tersebut memiliki intensitas piksel yang tinggi pada layer merah, begitu pun sebaliknya jika kadar warna merah semakin sedikit maka objek tersebut memiliki intensitas piksel yang rendah pada layer merah. Hal ini juga berlaku untuk layer hijau dan layer biru.

Suatu penelitian membandingkan 13 metode berbeda dalam konversi dari citra berwarna menjadi *grayscale*. Dengan mengevaluasi semua metode-metode yang telah digunakan secara luas, dan juga beberapa metode populer. Pembandingan dilakukan dengan *Naive bayes nearest neighbor framework* pengakuan gambar dan empat tipe berbeda dalam deskripsi gambar [3].

Pada suatu penelitian, pengolahan citra dilakukan dengan memisahkan nilai-nilai piksel menjadi 2 yaitu warna putih sebagai background dan warna hitam sebagai foreground. Dengan binerisasi gambar maka diperlukan peranan dari *Otsu's thresholding*. Metode *Otsu's thresholding* merupakan metode umum yang paling sering digunakan dalam global threshold [4].

Pada makalah ini, peneliti menggunakan algoritma Monte Carlo untuk menghitung luas daerah gambar biner untuk menentukan data status lalu lintas. Algoritma Monte Carlo digunakan untuk menghitung luas daerah foreground pada masing-masing *frame* untuk mentukan status lalu lintas. Hasil *accuracy*, *precision*, dan *recall* bergantung pada status lalu lintas yang dihasilkan oleh algoritma Monte Carlo.

II. PENGOLAHAN CITRA

A. Citra Grayscale

Citra *grayscale* adalah gambar yang memiliki nilai piksel pada sampel yang mengandung informasi mengenai intensitas. Gambar yang dimaksud adalah hitam dan putih berasal dari komposisi bayangan abu-abu. Warna *gray scale* bervriasi mulai dari warna dengan intensitas level terendah (hitam) hingga intensitas level tertinggi (putih) [5].

Gambar 1 *Grayscale*

Gambar 1 merupakan contoh citra grayscale lalu lintas dalam kondisi lancar. Ketika mengubah dari RGB menjadi *grayscale*, bobot spesifik pada saluran R, G, dan B seharusnya dapat diterapkan [6]. Semua algoritma *grayscale* memanfaatkan tiga tahapan proses yang sama :

1. Mendapatkan nilai piksel *red*, *green*, dan *blue*.
2. Menggunakan perhitungan matematis untuk mengubah nilai-nilai tersebut menjadi nilai keabuan tunggal.
3. Mengganti nilai asli *red*, *green*, dan *blue* dengan nilai keabuan yang baru.

$$gray = \frac{(R + G + B)}{3} \quad (1)$$

Gray adalah citra *grayscale*, R adalah layer dimensi merah, G adalah layer dimensi hijau, dan B adalah layer dimensi biru.

B. Deteksi Foreground

Deteksi foreground membandingkan *frame* input video dengan model background, dan mengidentifikasi kandidat piksel-piksel foreground dari *frame* input. Pendekatan yang paling umum untuk deteksi foreground yaitu memeriksa apakah piksel input berbeda secara signifikan dengan estimasi background yang sesuai.

$$\mathbf{D_k}(x, y) = \|\mathbf{F_k}(x, y) - \mathbf{B}(x, y)\| \quad (2)$$

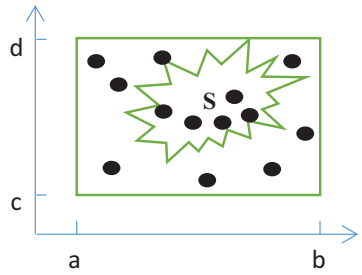
$\mathbf{D_k}$ adalah gambar substraksi pada saat k, $\mathbf{F_k}$ adalah *frame* pada saat k, dan model background adalah $\mathbf{B}$. Transformasi gambar foreground ke dalam bentuk *binary mask* oleh threshold T.

$$M(i, j) = 1, \text{ if } D(i, j) > T$$
$$M(i, j) = 0, \text{ if } D(i, j) <= T \quad (3)$$

M adalah citra biner yang memiliki nilai 0 atau 1, dan T adalah threshold yang digunakan untuk membatasi nilai piksel dari gambar substraksi.

C. Algoritma Monte Carlo

Monte Carlo adalah salah satu metode numerik yang dideskripsikan sebagai metode simulasi statistik. Monte Carlo telah diaplikasikan pada proses yang melibatkan perilaku acak dan untuk mengukur parameter-parameter fisik yang sulit bahkan tidak mungkin untuk kalkulasi dengan pengukuran eksperimental [9]. Algoritma Monte Carlo diimplementasikan pada penentuan area. Area S yang berada pada matrix persegi [a, b] x [c, d]. Gambar 2 menunjukkan ilustrasi dari algoritma Monte Carlo dalam menentukan luas daerah S. Algoritma dalam menentukan area S adalah sebagai berikut [10] :



Gambar 2 Ilustrasi Algoritma Monte Carlo

1) Generate titik random pada wilayah persegi. Dua variabel random yaitu x dan y yang berada pada range [a, b] dan [c, d] masing – masing digunakan sebagai koordinat titik pada wilayah persegi. Semakin besar jumlah titik maka semakin akurat area yang didapatkan.
2) Hitung jumlah titik yang masuk ke dalam S dan indikasikan sebagai N_0
3) Aproksimasi area S dengan menggunakan rumus :

$$L_s = \frac{(b-a)(d-c)N_0}{N} \quad (4)$$

L_s adalah luas daerah *S*, *a* adalah batas bawah kolom gambar, *b* adalah batas atas kolom gambar, *c* adalah batas bawah baris gambar, *d* adalah batas atas baris gambar, N_0 adalah jumlah koordinat yang jatuh pada area *S*, dan *N* adalah total koordinat pada gambar

D. Accuracy, Precision, dan Recall

Precision dan *recall* sering digunakan pada literatur untuk mengukur seberapa baik objek yang dideteksi terhadap objek referensi. *Recall* dapat diinterpretasikan sebagai jumlah objek positif benar dideteksi oleh algoritma, sedangkan *precision* cenderung mengevaluasi algoritma sebagai positif salah [11].

$$Accuracy = \frac{\text{total status deteksi yang benar}}{\text{total status yang dideteksi}} \quad (5)$$

$$Precision = \frac{\text{jumlah status deteksi yang benar}}{\text{total status deteksi pada data hasil}} \quad (6)$$

$$Recall = \frac{\text{jumlah status deteksi yang benar}}{\text{total status deteksi pada data asli}} \quad (7)$$

Accuracy digunakan untuk mengukur tingkat kesesuaian semua status lalu lintas yang benar dengan total status lalu lintas yang dideteksi. *Precision* digunakan untuk mengukur tingkat kepresisian status lalu lintas yang terdeteksi oleh algoritma terhadap status lalu lintas yang benar. *Recall* digunakan untuk mengukur tingkat kesesuaian status lalu lintas yang benar dengan status lalu lintas yang sebenarnya

III. METODOLOGI DAN ALGORITMA

A. Konversi Grayscale

Grayscale adalah range dari warna abu-abu yang merupakan transisi warna dari putih menuju hitam. Warna yang paling terang adalah warna putih, sedangkan warna yang paling gelap adalah warna hitam. Jadi citra *grayscale* merupakan konversi dari citra RGB menjadi suatu gambar dengan warna dari interval putih ke hitam. *Frame* gambar bersifat dinamis karena jumlahnya disesuikan dengan jumlah gambar lalu lintas. Sama seperti background, *frame* citra RGB juga dibagi menjadi 3 layer dimensi.

1) Layer Merah *Frame*

$$A_{k(m\,x\,n)}^{red} = \begin{bmatrix} a_{k}{}_{11}^{red} & \cdots & a_{k}{}_{1n}^{red} \\ \vdots & \ddots & \vdots \\ a_{k}{}_{m1}^{red} & \cdots & a_{k}{}_{mn}^{red} \end{bmatrix},$$

$A_{k(m\,x\,n)}^{red}$ adalah layer merah dari *frame* ke-k yang berukuran m x n, dan $a_k{}^{red}$ adalah nilai-nilai piksel dari layer merah *frame* ke-k, k ∈ {1, 2, 3, ..., N}, dan N adalah total gambar.

2) *Layer* Hijau *Frame*

$$A_{k(m\,x\,n)}^{green} = \begin{bmatrix} a_{k}{}_{11}^{green} & \cdots & a_{k}{}_{1n}^{green} \\ \vdots & \ddots & \vdots \\ a_{k}{}_{m1}^{green} & \cdots & a_{k}{}_{mn}^{green} \end{bmatrix},$$

$A_{k(m\,x\,n)}^{green}$ adalah layer hijau dari *frame* ke-k yang berukuran m x n, dan $a_k{}^{green}$ adalah nilai-nilai piksel dari layer hijau *frame* ke-k, k ∈ {1, 2, 3, ..., N}, dan N adalah total gambar.

3) *Layer* Biru *Frame*

$$A_{k(m\,x\,n)}^{blue} = \begin{bmatrix} a_{k}{}_{11}^{blue} & \cdots & a_{k}{}_{1n}^{blue} \\ \vdots & \ddots & \vdots \\ a_{k}{}_{m1}^{blue} & \cdots & a_{k}{}_{mn}^{blue} \end{bmatrix},$$

$A_{k(m\,x\,n)}^{blue}$ adalah layer biru dari *frame* ke-k yang berukuran m x n, dan $a_k{}^{blue}$ adalah nilai-nilai piksel dari layer biru *frame* ke-k, k ∈ {1, 2, 3, ..., N}, dan N adalah total gambar. Setelah membagi *frame* menjadi 3 layer dimensi, maka selanjutnya adalah menghitung *grayscale* dengan menggunakan rumus berikut:

$$\alpha_k(x, y) = \frac{\rho_{k(x,y)}^{red} + \rho_{k(x,y)}^{green} + \rho_{k(x,y)}^{blue}}{3} \quad (8)$$

$\alpha_k(x, y)$ adalah nilai-nilai piksel *grayscale* dari *frame* ke-*k*, $\rho_{k(x,y)}^{red}$ adalah setiap nilai piksel dari layer merah *frame* ke-*k*, $\rho_{k(x,y)}^{\text{hijau}}$ adalah setiap nilai piksel dari layer hijau *frame* ke-k, $\rho_{k(x,y)}^{red}$ adalah setiap nilai piksel dari layer biru *frame* ke-*k*, dan untuk setiap (x, y) ∈ {1, 2, 3, ..., m} x {1, 2, 3, ..., n}.

B. Deteksi Foreground

Deteksi foreground dilakukan dengan cara mengurangi *grayscale* pada setiap *frame* dengan *grayscale* background untuk mendapatkan kandidat-kandidat foreground dari citra *grayscale*. *Grayscale* background bersifat statik, karena untuk mendapatkan foreground dari setiap *frame* maka background dari setiap *frame* juga harus dihilangkan atau dijadikan hitam. *Grayscale* dari setiap *frame* memiliki background yang sama dengan *grayscale* background sehingga pada saat dikurangkan, piksel background akan bernilai nol dan sisanya adalah foreground yang masih memiliki nilai piksel. Kandidat foreground dapat diperoleh dengan menggunakan rumus berikut ini :

$$D_k(x, y) = \left|\left|F_k(x, y) - B(x, y)\right|\right| >= 0 \quad (9)$$

$D_k(x, y)$ adalah gambar substraksi ke-*k*, $F_k(x, y)$ adalah *grayscale frame* ke-k, dan $B(x, y)$ adalah *grayscale* dari background. Gambar substraksi mengandung kandidat-kandidat foreground sebelum diubah ke dalam bentuk biner.

Gambar 3 Substraksi dari Suatu *Frame*

Gambar 8 menunjukan gambar subtraksi pada *frame* yang dihasilkan dengan menggunakan persamaan (9). Mobil pada gambar substraksi merupakan kandidat foreground pada citra biner. Gambar substraksi masih berupa grayscale sehingga harus dilakukan proses untuk mengubah grayscale menjadi citra biner.

C. Algoritma Deteksi Kemacetan

Peneliti menggunakan algoritma deteksi kemacetan lalu lintas berdasarkan algoritma Monte Carlo untuk menghitung luas foreground dari citra biner. Hasil dari algoritma ini berupa data status lalu lintas yang ditentukan berdasarkan range nilai luas foreground pada citra biner yang dibandingkan dengan nilai luas pada citra biner. Algoritma deteksi lalu lintas dapat dilihat pada Algoritma 1.

Algoritma 1 digunakan untuk menghitung jumlah kendaraan yang dideteksi oleh algoritma Monte Carlo dengan membatasi jumlah kendaraan pada jumlah tertentu dalam menentukan status lalu lintas. Status lalu lintas terdeteksi lancar apabila jumlah kendaraan yang terdeteksi kurang dari 6 mobil. Status lalu lintas terdeteksi ramai apabila jumlah kendaraan yang terdeteksi lebih dari 6 mobil dan kurang dari atau sama dengan 15 mobil. Status lalu lintas terdeteksi padat apabila jumlah kendaraan yang terdeteksi lebih dari 15 mobil.

Algoritma 1: Deteksi Kemacetan
input : k : *frame* ke-k citra biner, N : total gambar **output** : status lalu lintas (lancar/ramai/padat) **1. for** *frame* k=1 to N do **2.** J_m = jumlah_mobil; **3.** threshold1=6; **4.** threshold2=15; **5.** **if** J_m > threshold2 **then** output padat; **6.** else **if** thresold1 < J_m <= threshold2 **then** output ramai; **7.** else **if** J_m < threshold1 **then** output lancar; **8. end**

IV. PENGUJIAN DAN ANALISIS

A. Skenario Pengujian

Citra RGB lalu lintas dibagi menjadi 5 segmen untuk meningkatkan presisi dalam mendeteksi status lalu lintas. Pembagian segmen pada citra RGB lalu lintas dapat dilihat pada Gambar 4. Masing-masing segmen memiliki ukuran baris dan kolom yang berbeda-beda. Segmen 1 memiliki ukuran baris dan kolom paling besar karena berada paling dekat dengan kamera peengamatan.

Gambar 4 Pembagian Segmen RGB Lalu Lintas

Segmen 1 paling dekat dengan kamera sehingga memiliki tinggi paling besar dibandingkan dengan segmen-segmen yang lainnya. Semakin jauh dengan kamera maka tinggi segmen akan semakin kecil. Segmen 5 paling jauh dari kamera sehingga memiliki tinggi paling kecil. Berikut ini adalah spesifikasi ukuran dari setiap segmen :

1) Segmen 1

Baris pada segmen 1 dimulai dari 155 sampai dengan 240 dan kolom dimulai dari 1 sampai dengan 320.

2) Segmen 2

Baris pada segmen 2 dimulai dari 105 sampai dengan 155 dan kolom dimulai dari 216 sampai dengan 265.

3) Segmen 3

Baris pada segmen 3 dimulai dari 65 sampai dengan 105 dan kolom dimulai dari 156 sampai dengan 235.

4) Segmen 4

Baris pada segmen 4 dimulai dari 32 sampai dengan 65 dan kolom dimulai dari 112 sampai dengan 211.

5) Segmen 5

Baris pada segmen 5 dimulai dari 1 sampai dengan 32 dan kolom dimulai dari 72 sampai dengan 191.

Data yang digunakan dalam penelititan ini memiliki 2 sudut pandang kamera yang berbeda. Masing-masing sudut pandang menggambarkan kondisi lalu lintas serta arah gerak mobil yang berlawanan. Data yang diambil dengan menggunakan sudut pandang 1 memiliki kondisi lancar dan ramai. Sudut pandang 1 dan sudut pandang 2 dapat dilihat pada Gambar 5.

(a) (b)

Gambar 5 Lalu Lintas dengan Sudut Pandang 1 (a), Sudut Pandang 2 (b)

Gambar 5 (a) menunjukkan gambar lalu lintas dengan arah gerak mobil mendekati kamera pengamatan. Data yang diambil dengan menggunakan sudut pandang 2 mengandung status padat. Gambar 5 (b) menunjukkan gambar lalu lintas dengan arah gerak mobil menjauhi kamera pengamatan yang memiliki kondisi padat karena posisi mobil yang saling berdekatan.

(a) (b) (c)

Gambar 6 Lalu Lintas Sepi (a), ramai (b), padat (c)

Gambar 6 (a) menunjukkan lalu lintas dengan kondisi sepi. Kondisi sepi dipenuhi jika jumlah mobil adalah kurang dari atau sama dengan 6. Gambar 6 (b) menunjukkan lalu lintas dengan kondisi ramai. Kondisi ramai dipenuhi jika jumlah mobil lebih dari 6 dan kurang dari atau sama dengan 15. Gambar 6 (c) menunjukkan lalu lintas dengan kondisi padat. Kondisi padat dipenuhi jika jumlah mobil lebih dari 15. Pada penelitian ini mengggunakan tiga skenario yang digunakan sebagai pembanding :

L_s = luas segmen
L_p = luas putih
J_m= jumlah mobil

1) Skenario 1 dengan *Rule* berikut :

Tabel 1 *Rule* Segmen 1 pada Skenario 1

No	*Rule*	J_m
1	4% L_s < L_p <= 14% L_s	1
2	14% L_s < L_p <= 24% L_s	2
3	24% L_s < L_p <= 34% L_s	3
4	34% L_s < L_p <= 44% L_s	4
5	L_p > 44% L_s	5

Tabel 2 *Rule* Segmen 2 pada Skenario 1

No	*Rule*	J_m
1	4% L_s < L_p <= 16% L_s	1
2	16% L_s < L_p <= 28% L_s	2
3	28% L_s < L_p <= 40% L_s	3
4	40% L_s < L_p <= 52% L_s	4
5	L_p > 52% L_s	5

Tabel 3 *Rule* Segmen 3 pada Skenario 1

No	*Rule*	J_m
1	4% L_s < L_p <= 15% L_s	1
2	15% L_s < L_p <= 26% L_s	2
3	26% L_s < L_p <= 37% L_s	3
4	37% L_s < L_p <= 48% L_s	4
5	L_p > 48% L_s	5

Tabel 4 *Rule* Segmen 4 pada Skenario 1

No	*Rule*	J_m
1	4% L_s < L_p <= 12% L_s	1
2	12% L_s < L_p <= 20% L_s	2
3	20% L_s < L_p <= 28% L_s	3
4	28% L_s < L_p <= 36% L_s	4
5	L_p > 36% L_s	5

Tabel 5 *Rule* Segmen 5 pada Skenario 1

No	*Rule*	J_m
1	4% L_s < L_p <= 7% L_s	1
2	7% L_s < L_p <= 10% L_s	2
3	10% L_s < L_p <= 13% L_s	3
4	13% L_s < L_p <= 16% L_s	4
5	16% L_s < L_p <= 19% L_s	5
6	19% L_s < L_p <= 22% L_s	6
7	22% L_s < L_p <= 25% L_s	7

No	*Rule*	J_m
8	25% L_s < L_p <= 28% L_s	8
9	28% L_s < L_p <= 31% L_s	9
10	31% L_s < L_p <= 34% L_s	10
11	34% L_s < L_p <= 37% L_s	11
12	37% L_s < L_p <= 40% L_s	12
13	40% L_s < L_p <= 43% L_s	13
14	43% L_s < L_p <= 47% L_s	14
15	L_p > 47% L_s	15

2) Skenario 2 dengan *Rule* :

Tabel 6 *Rule* Segmen 1 pada Skenario 2

No	*Rule*	J_m
1	4% L_s < L_p <= 13% L_s	1
2	13% L_s < L_p <= 22% L_s	2
3	22% L_s < L_p <= 31% L_s	3
4	31% L_s < L_p <= 40% L_s	4
5	L_p > 40% L_s	5

Tabel 7 *Rule* Segmen 2 pada Skenario 2

No	*Rule*	J_m
1	4% L_s < L_p <= 15% L_s	1
2	15% L_s < L_p <= 26% L_s	2
3	26% L_s < L_p <= 37% L_s	3
4	37% L_s < L_p <= 48% L_s	4
5	L_p > 48% L_s	5

Tabel 8 *Rule* Segmen 3 pada Skenario 2

No	*Rule*	J_m
1	4% L_s < L_p <= 14% L_s	1
2	14% L_s < L_p <= 24% L_s	2
3	24% L_s < L_p <= 34% L_s	3
4	34% L_s < L_p <= 44% L_s	4
5	L_p > 44% L_s	5

Tabel 9 *Rule* Segmen 4 pada Skenario 2

No	*Rule*	J_m
1	4% L_s < L_p <= 11% L_s	1
2	11% L_s < L_p <= 18% L_s	2
3	18% L_s < L_p <= 25% L_s	3
4	25% L_s < L_p <= 32% L_s	4
5	L_p > 32% L_s	5

Tabel 10 *Rule* Segmen 5 pada Skenario 2

No	*Rule*	J_m
1	4% L_s < L_p <= 6% L_s	1
2	6% L_s < L_p <= 8% L_s	2
3	8% L_s < L_p <= 10% L_s	3
4	10% L_s < L_p <= 12% L_s	4
5	12% L_s < L_p <= 14% L_s	5
6	14% L_s < L_p <= 16% L_s	6
7	16% L_s < L_p <= 18% L_s	7

No	*Rule*	J_m
8	18% L_s < L_p <= 20% L_s	8
9	20% L_s < L_p <= 22% L_s	9
10	22% L_s < L_p <= 24% L_s	10
11	24% L_s < L_p <= 26% L_s	11
12	26% L_s < L_p <= 28% L_s	12
13	28% L_s < L_p <= 30% L_s	13
14	30% L_s < L_p <= 32% L_s	14
15	L_p > 32% L_s	15

3) Skenario 3 dengan *Rule* :

Tabel 11 *Rule* Segmen 1 pada Skenario 3

No	*Rule*	J_m
1	4% L_s < L_p <= 15% L_s	1
2	15% L_s < L_p <= 26% L_s	2
3	26% L_s < L_p <= 37% L_s	3
4	37% L_s < L_p <= 48% L_s	4
5	L_p > 48% L_s	5

Tabel 12 *Rule* Segmen 2 pada Skenario 3

No	*Rule*	J_m
1	4% L_s < L_p <= 17% L_s	1
2	17% L_s < L_p <= 30% L_s	2
3	30% L_s < L_p <= 43% L_s	3
4	43% L_s < L_p <= 56% L_s	4
5	L_p > 56% L_s	5

Tabel 13 *Rule* Segmen 3 pada Skenario 3

No	*Rule*	J_m
1	4% L_s < L_p <= 16% L_s	1
2	16% L_s < L_p <= 28% L_s	2
3	28% L_s < L_p <= 40% L_s	3
4	40% L_s < L_p <= 52% L_s	4
5	L_p > 52% L_s	5

Tabel 14 *Rule* Segmen 4 pada Skenario 3

No	*Rule*	J_m
1	4% L_s < L_p <= 13% L_s	1
2	13% L_s < L_p <= 22% L_s	2
3	22% L_s < L_p <= 31% L_s	3
4	31% L_s < L_p <= 40% L_s	4
5	L_p > 40% L_s	5

Tabel 15 *Rule* Segmen 5 pada Skenario 3

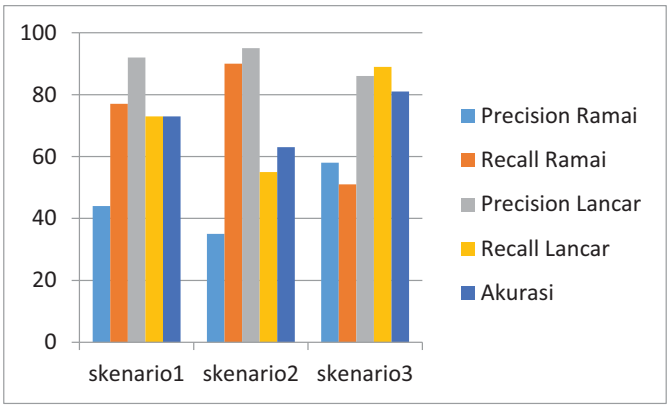
No	*Rule*	J_m
1	4% L_s < L_p <= 8% L_s	1
2	8% L_s < L_p <= 12% L_s	2
3	12% L_s < L_p <= 16% L_s	3
4	16% L_s < L_p <= 20% L_s	4
5	20% L_s < L_p <= 24% L_s	5
6	24% L_s < L_p <= 28% L_s	6
7	28% L_s < L_p <= 32% L_s	7

No	*Rule*	J_m
8	32% L_s < L_p <= 36% L_s	8
9	36% L_s < L_p <= 40% L_s	9
10	40% L_s < L_p <= 44% L_s	10
11	44% L_s < L_p <= 48% L_s	11
12	48% L_s < L_p <= 52% L_s	12
13	52% L_s < L_p <= 56% L_s	13
14	56% L_s < L_p <= 60% L_s	14
15	L_p > 60% L_s	15

Parameter yang dibedakan antar skenario adalah *Rule* yang dibuat untuk menentukan jumlah mobil tiap *frame*. Tiap *frame* dibagi menjadi lima segmen karena ukuran mobil tiap segmen berbeda. Jika posisi mobil semakin dekat dengan kamera atau di dalam segmen satu, maka ukutan mobil semakin besar. Jika posisi mobil semakin menjauhi kamera atau maksimal berada di segmen lima maka ukuran mobil semakin kecil. Hal ini juga sangat berpengaruh terhadap *Rule* yang dibuat tiap segmen berdasarkan perhitungan luas dari algoritma Monte Carlo.

B. Hasil Testing 1

Data *testing 1* ada sebanyak 300 data yang digunakan pada ketiga skenario. Data *testing 1* mengandung status lancar dan ramai. *Accuracy* digunakan untuk mengukur tingkat kesesuaian semua status yang benar terhadap total status yang dideteksi. *Precision* digunakan untuk mengukur tingkat presisi algoritma dalam mendeteksi status lalu lintas yang benar, sedangkan *recall* digunakan untuk mengukur keakuratan status yang terdeteksi benar terhadap status lalu lintas yang sebenarnya.



Gambar 7 Grafik Rata-Rata *Accuracy*, *Precision*, dan *Recall* Data *Testing 1*

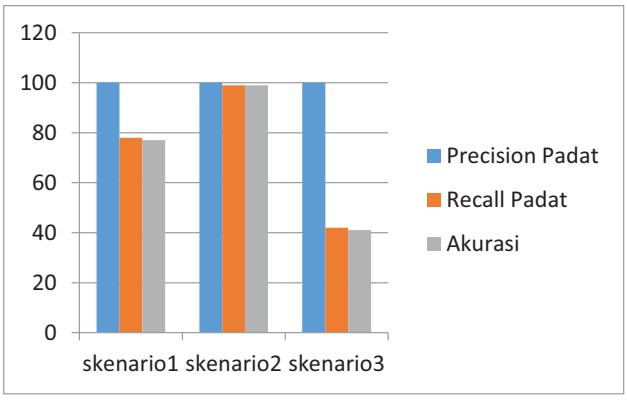
Berdasarkan Gambar 4 diketahui bahwa skenario 1 menunjukkan rata-rata nilai *accuracy* sebesar 73%, *precision* ramai sebesar 44%, *recall* ramai sebesar 77%, *precision* lancar sebesar 92%, dan *recall* lancar sebesar 73%. Status lalu lintas yang benar-benar ramai sudah sesuai dengan status ramai yang sebenarnya, tetapi masih belum sesuai dengan status ramai yang dideteksi oleh algoritma. Status lalu lintas yang benar-benar lancar sudah sesuai dengan status lancar yang dideteksi oleh algoritma, dan cukup sesuai dengan status lancar yang sebenarnya. Status lalu lintas yang benar-benar ramai dan benar-benar lancar cukup sesuai dengan total status lalu lintas yang dideteksi.

Skenario 2 menunjukkan rata-rata nilai *accuracy* sebesar 63%, *precision* ramai sebesar 35%, *recall* ramai sebesar 90%, *precision* lancar sebesar 95%, dan *recall* lancar sebesar 55%. Status lalu lintas yang benar-benar ramai sudah sesuai dengan status ramai yang sebenarnya, tetapi belum sesuai dengan status ramai yang dideteksi oleh algoritma. Status lalu lintas yang benar-benar lancar sudah sesuai dengan status lancar yang dideteksi oleh algoritma, tetapi belum sesuai dengan status lancar yang sebenarnya. Status lalu lintas yang benar-benar ramai dan benar-benar lancar masih belum sesuai dengan total status lalu lintas yang dideteksi.

Skenario 3 menunjukkan nilai rata-rata *accuracy* sebesar 81%, *precision* ramai sebesar 58%, *recall* ramai sebesar 51%, *precision* lancar 86%, dan *recall* lancar sebesar 89%. Status lalu lintas yang benar-benar ramai masih belum sesuai dengan status ramai yang dideteksi oleh algoritma dan status ramai yang sebenarnya. Status lalu lintas yang benar-benar lancar sudah sesuai dengan status lancar yang dideteksi oleh algoritma dan status lancar yang sebenarnya. Status lalu lintas yang benar-benar ramai dan benar-benar lancar sudah sesuai dengan total status lalu lintas yang dideteksi.

C. Hasil Testing 2

Data testing 2 ada sebanyak 140 data yang digunakan pada ketiga skenario. Data testing 2 hanya mengandung status padat. Hasil *accuracy*, *precision* dan *recall* dari data testing 2 adalah sebagai berikut:



Gambar 8 Grafik Rata-Rata *Accuracy*, *Precision*, dan *Recall* Data Testing 2

Berdasarkan gambar 5 dapat diketahui bahwa skenario 1 menunjukan rata-rata nilai *accuracy* sebesar 77%, *precision* padat sebesar 100%, dan *recall* padat sebesar 78%. Status lalu lintas yang benar-benar padat sudah sesuai dengan status padat yang dideteksi oleh algoritma, status padat yang sebenarnya, dan total status lalu lintas yang dideteksi.

Skenario 2 menunjukkan rata-rata nilai *accuracy* sebesar 99%, *precision* padat sebesar 100%, dan *recall* padat sebesar 99%. Status lalu lintas yang benar-benar padat sudah sesuai dengan status padat yang dideteksi oleh algoritma, status padat yang sebenarmya, dan total status lalu lintas yang dideteksi.

Skenario 3 menunjukkan rata-rata nilai *accuracy* sebesar 41%, *precision* padat sebesar 100%, *recall* padat sebesar 42%. Status lalu lintas yang benar-benar padat sudah sesuai dengan status padat yang dideteksi oleh algoritma, tetapi masih belum sesuai dengan status padat yang sebenarnya dan total status lalu lintas yang dideteksi.

V. KESIMPULAN

Status lalu lintas diperoleh dengan menghitung jumlah mobil yang terdeteksi. Hasil dari *testing 1* menunjukkan bahwa skenario 1 yang lebih sesuai karena menghasilkan recall ramai, precision lancar, recall lancar, dan akurasi yang cukup tinggi. Recall ramai bernilai 77%, precision lancar bernilai 92%, recall lancar bernilai 73%, dan akurasi bernilai 73%, sedangkan precision ramai bernilai 44%. Precision ramai selalu bernilai rendah karena data *testing 1* didominasi oleh kondisi lancar, sedangkan kondisi ramai jarang terjadi. Hal ini membuat algoritma Monte Carlo kurang sesuai digunakan untuk mendeteksi status ramai. Algoritma Monte Carlo sudah cukup sesuai digunakan untuk status lancar dan ramai dengan menggunakan *Rule-Rule* pada skenario 1. Hasil dari testing 2 menunjukkan bahwa skenario 2 yang lebih sesuai karena menghasilkan precision padat, recall padat, dan akurasi tertinggi. Precision padat bernilai 100%, recall padat bernilai 99%, dan akurasi 99% bernilai. Algoritma Monte Carlo sudah sesuai digunakan untuk mendeteksi status padat dengan menggunakan *Rule-Rule* pada skenario 2.

DAFTAR PUSTAKA

[1] R. Danescu, S. Nedevschi, M. M. Mienecke, and T. Graf, "Stereovision Based Vehicle Tracking in Urban Traffic Environments," *Intelligent Transportation Systems Conference*, pp. 400-404, September-October 2007.

[2] Anh Vu and Matthew Barth, "Catadiopetric Omnidirectional Vision Sensor Integration for Vehicle-Based Sensing," *International IEEE Conference on Intelligent Transportation Systems*, vol. 12, pp. 120-126, October 2009.

[3] Christopher Kanan and Garrison W. Cottrell, "Color-to-*Grayscale*: Does the Method Matter in Image Recognition?," vol. 7, no. 1, pp. 1-7, January 2012.

[4] T. Romen Singh, Sudipta Roy, O. Imocha Singh, Tejmani Sinam, and Kh. Manglem Singh, "A New Local Adaptive *Thresholding* Technique in Binarization," *International Journal of Computer Science Issues*, vol. 8, no. 6, pp. 271-277, November 2011.

[5] Neeraj Bhargava, Anchal Kumawat, and Ritu Bhargava, "Threshold and Binarization for Document Image Analysis Using *Otsu's* Algoritm," *International Journal of Computer Trends and Technology* , vol. 17, no. 5, pp. 272-275, November 2014.

[6] G. Jyothi, CH. Sushma, and D. S.S. Veeresh, "Luminance Based Conversion of *Gray scale* Image to RGB Image," *International Journal of Computer Science and Information Technology Research*, vol. 3, no. 3, pp. 279-283, July-September 2015.

[7] Siddhartha A. Meshram and Vishal B. Raskar, "Vehicle Detection and Tracking Techniques Used in Moving Vehicles," *International Journal of Innovative Science, Engineering & Technology*, vol. 2, no. 7, pp. 48-59, July 2015.

[8] Joanna Sekulska-Nalewajko and Jaroslaw Goclawski, "An Image Analysis Method for the Automatic Measurement of Selected Morphological Features of Wheat Shoots," pp. 243-257, 2011.

[9] P. F. Liaparinos, "Monte Carlo Simulations in Medical Imaging," *e-Journal of Science & Technology*, pp. 7-13, April 2010.

[10] Joko Siswantoro, Anton Satria Prabuwono, and Azizi Abdullah, "Volume Measurement Algoriithm for Food Product with Irregular Shape using Computer Vision based on Monte Carlo Method," *Journal of Information and Communication Technology*, vol. 8, no. 1, pp. 1-17, February 2014.

[11] Bahadir Ozdemir, Selim Aksoy, Sandra Eckert, Martino Pesaresi, and Daniele Ehrlich, "Performance Measures for Object Detection Evaluation," pp. 1-11, July 2009.