APLIKASI PENGENALAN RAMBU LALU LINTAS MENGGUNAKAN METODE *SCALE INVARIANT FEATURE TRANSFORM* (SIFT) DAN *SUPPORT VECTOR MACHINE* (SVM) BERBASIS ANDROID

Traffic Sign Recognition Application using Scale Invariant Feature Transform (SIFT) Method and Support Vector Machine (SVM) Based on Android

Ranti Ratnasari[1], Budhi Irawan, S.Si., M.T.[2], Casi Setianingsih, S.T., M.T.[3]

[1,2,3]Prodi S1 Sistem Komputer, Fakultas Teknik Elektro, Universitas Telkom
[1]ranti.ratnasari13@gmail.com, [2]budhiirawan@telkomuniversity.ac.id, [3]setiacasie@telkomuniversity.ac.id

Abstrak

Meski terdapat banyak sekali rambu lalu lintas yang terpasang, namun seringkali masyarakat mengabaikannya dengan berbagai alasan, seperti tidak melihat keberadaan rambu tersebut dan tidak mengetahui makna rambu yang ada. Hal itu berdampak pada ketertiban lalu lintas. Dengan adanya masalah tersebut, dalam tugas akhir ini akan dirancang sebuah aplikasi *mobile* berbasis android yang dapat mengenalkan rambu lalu lintas dengan cara yang berbeda. Aplikasi yang dirancang menggunakan metode *Scale Invariant Feature Transform* dan *Support Vector Machine* ini dapat memberikan informasi kepada pengguna mengenai nama rambu dan penjelasan singkat tentang rambu yang sudah di *input*kan oleh pengguna dalam Bahasa Indonesia dan Bahasa Inggris. Dari hasil pengujian yang telah dilakukan, aplikasi ini memiliki tingkat akurasi tertinggi sebesar 88%. Hal tersebut menunjukan bahwa aplikasi ini dapat mendeteksi 22 rambu dari 25 rambu yang diujikan. Nilai tersebut didapat ketika pengambilan citra dilakukan pada 10896 lux – 4105 lux dalam keadaan citra berada tepat pada *Region of Interest* (ROI) yang ada pada kamera.

Kata Kunci : Rambu Lalu Lintas, Pengenalan, Aplikasi *Mobile*, Android, *Scale Invariant Feature Transform* (SIFT), *Support Vector Machine* (SVM)

Abstract

Although there are a lot of traffic signs that exist, but oftentimes people disregard it for various reasons, such as not seeing the existence of these signs and do not know the meaning of signs that exist. It has an impact on traffic order. With this problem, in this final assignment will be designed a mobile application based on android that can introduce traffic signs in different ways. Applications designed using the Scale Invariant Feature Transform (SIFT) and Support Vector Machine (SVM) methods that can provide users with information on the name of the signs and a brief explanation of the sign in Bahasa and English. Based on the results, this application has the highest accuracy rate of 88%. This shows that application can detect 22 of 25 signs. The value obtained when the image taken on 10896 lux - 4105 lux with the position of the image on the Range of Interest (ROI) that contained in the camera.

***Keywords** : Traffic Signs, Recognition, Mobile Application, Android, Scale Invariant Feature Transform (SIFT), Support Vector Machine (SVM)*

1. Pendahuluan

1.1 Latar Belakang

Rambu lalu lintas adalah bagian perlengkapan jalan yang merupakan lambang, huruf, angka, kalimat, dan/atau perpaduan yang berfungsi sebagai peringatan, larangan, perintah atau petunjuk bagi pengguna jalan[1]. Namun, seringkali masyarakat mengabaikannya dengan berbagai alasan, seperti tidak melihat keberadaan rambu tersebut dan tidak mengetahui makna rambu yang ada. Hal itu berdampak pada ketertiban lalu lintas.

Dengan adanya masalah tersebut, dalam tugas akhir ini akan dirancang sebuah aplikasi *mobile* berbasis android dengan menggunakan metode *Scale Invariant Feature Transform* (SIFT) dan *Support Vector Machine* (SVM). SIFT adalah sebuah metode ekstraksi ciri yang akan mengubah suatu citra menjadi vector lokal, kemudian vektor lokal tersebut akan digunakan sebagai pendekatan dalam mendeteksi objek yang dimaksud[3]. Sedangkan SVM adalah sebuah metode klasifikasi dimana akan dicari sebuah *hyperplane* antara citra latih dan citra masukan[6].

Aplikasi ini dapat memberikan informasi kepada pengguna berupa nama rambu dan penjelasan singkat dalam Bahasa Indonesia dan Bahasa Inggris dengan cara memasukan data berupa foto yang dapat diambil memalui kamera *smartphone* maupun galeri.

1.2 Perumusan Masalah

Berdasarkan latar belakang yang sudah dipaparkan, maka rumusan masalah yang akan dibahas dalam tugas akhir ini adalah sebagai berikut :

(1) Bagaimana perancangan dan pembuatan aplikasi sehingga aplikasi tersebut dapat memberikan informasi berupa teks mengenai rambu lalu lintas yang di*input*kan ?
(2) Bagaimana pengimplementasian metode SIFT kedalam sebuah aplikasi *mobile* berbasis android ?
(3) Bagaimana pengimplementasian metode SVM kedalam sebuah aplikasi *mobile* berbasis android ?

1.3 Tujuan

Tujuan dalam pengerjaan tugas akhir ini adalah sebagai berikut :

(1) Merancang dan membuat aplikasi yang dapat memberikan informasi berupa teks mengenai rambu lalu lintas yang di*input*kan.
(2) Menguji dan menganalisis metode SIFT pada aplikasi untuk melakukan pengenalan rambu lalu lintas.

1.4 Batasan Masalah

Agar tidak terjadi perluasan pembahasan dalam tugas akhir ini, penulis membatasi beberapa masalah sebagai berikut :

(1) Keluaran sistem berupa teks informasi dalam Bahasa Indonesia dan Bahasa Inggris.
(2) Citra yang diolah dalam format *Portable Network Graphics* (PNG) dan *Joint Photographic Experts Group* (JPEG).
(3) Aplikasi dapat dijalankan tanpa koneksi internet (*offline*).

2. Dasar Teori

2.1 Rambu Lalu Lintas

Rambu lalu lintas adalah bagian perlengkapan jalan berupa lambang, huruf, angka, kalimat, dan/atau perpaduan yang berfungsi sebagai peringatan, larangan, perintah, atau petunjuk bagi pengguna jalan[1]. Berdasarkan jenis pesan yang disampaikan, rambu lalu lintas di bagi kedalam beberapa kelompok sebagai berikut[1]:

(1) Rambu Peringatan
(2) Rambu Petunjuk
(3) Rambu Larangan
(4) Rambu Perintah
(5) Rambu Lokasi Utilitas Umum
(6) Rambu Sementara

2.2 Citra

Citra merupakan representasi dari sebuah objek yang memiliki informasi berbentuk visual. Citra terbagi menjadi dua, yaitu citra analog dan citra digital. Citra analog merupakan citra yang bersifat kontinu seperti gambar pada televisi. Sedangkan citra digital merupakan citra yang dapat diolah oleh komputer dalam bentuk matriks yang terdiri dari M baris dan N kolom.[9]

2.3 *Scale Invariant Feature Transform*(SIFT)

Scale Invariant Feature Transform (SIFT) adalah sebuah algorima untuk melakukan ekstraksi ciri yang dikenalkan oleh seorang peneliti dari University of British Columbia pada tahun 1999, David .G Lowe. Hasil akhir dari algoritma ini adalah fitur-fitur lokal yang akan dijadikan sebagai *descriptor* dari sebuah citra. Untuk mendapatkan fitur-fitur lokal tersebut citra harus melewati empat tahap yang akan dijelaskan pada poin-poin berikut ini.[3]

2.4.1 Pencarian Nilai Ekstrim pada Skala Ruang

Pada tahap pencarian nilai ekstrim pada skala ruang akan digunakan fungsi *difference of gaussian.* Tahap ini akan menghasilkan kandidat *keypoint* yang akan diproses pada tahap selanjutnya.[3]

2.4.2 Lokalisasi *Keypoint*

Tahap ini akan mengeliminasi kandidat *keypoint*. Kandidat *keypoint* yang dieliminasi adalah kandidat *keypoint* yang memiliki nilai kontras, $|D(\hat{x})|$, kurang dari 0.03 dan berada pada garis tepi, $r > 10$.[3]

2.4.3 Pemberian Orientasi

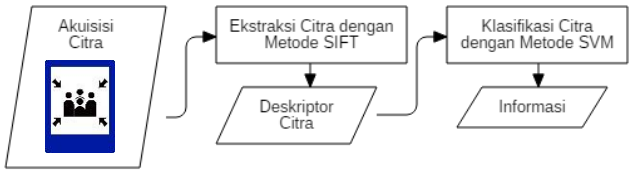
Keypoint yang tidak dieliminasi akan diberikan orientasi agar titik *keypoint* tidak berubah meski citra mengalami perubahan. [3]

2.4.4 Deskriptor *Keypoint*

Deskriptor sebuah *keypoint* dibutuhkan untuk menetapkan titik *keypoint* yang sudah teridentifikasi meski terjadi perubahan, baik pada sudut pandang maupun pencahayaan. Deskriptor akan berbentuk vektor yang didapat dari hasil keseluruhan orientasi histogram. Orientasi histogram tersebut didapat dari perhitungan gradien magnitude dan orientasi pada daerah sekitar *keypoint* (sisi kiri pada gambar). Dari perhitungan gradien magnitude dan orientasi tersebut akan didapatkan sebuah array berukuran 4x4 dengan 8 orientasi. Maka dari itu, akan digunakan 4x4x8 = 128 buah vektor pada setiap *keypoint*. [3]

3. Perancangan dan Analisis Sistem

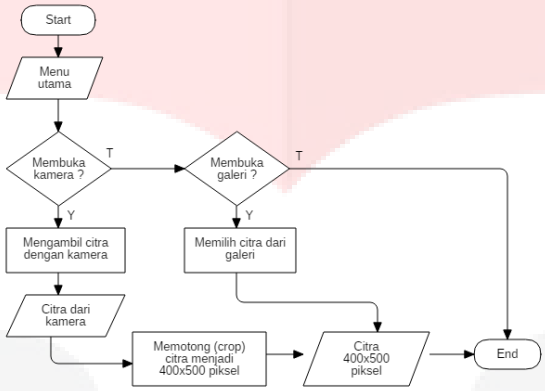
3.1 Blok Diagram Sistem



Gambar 3.1 Blok Diagram Sistem

Gambar 3.1 menjelaskan tentang gambaran umum sistem yang digunakan pada aplikasi ini. Tahap pertama yang akan dilakukan oleh sistem adalah akuisisi citra, dimana pada tahap ini pengguna akan meng*input*kan citra berupa rambu lalu lintas melalui kamera atau galeri. setelah itu, citra akan melewati proses ekstraksi menggunakan metode SIFT dan diklasifikasi dengan metode SVM. Keluaran yang dihasilkan oleh aplikasi ini adalah sebuah informasi berupa teks tentang rambu lalu lintas dalam Bahasa Indonesia dan Bahasa Inggris.

3.2 Akuisisi Citra

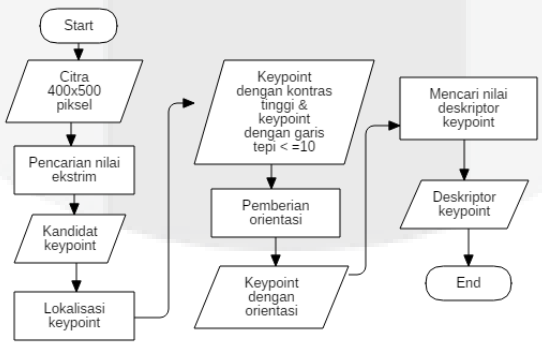


Gambar 3.2 Diagram Akuisisi Citra

Gambar 3.2 menunjukan proses akuisisi citra yang akan dilakukan oleh sistem. Ketika pengguna meng*input*kan citra melalui kamera, maka citra tersebut akan melewati tahap pemotongan. Pemotongan ini akan merubah ukuran citra menjadi 400x500 piksel. Setelah itu, citra berukuran 400x500 piksel akan masuk pada tahap selanjutnya. Saat pengguna meng*input*kan citra melalui galeri, pengguna dapat memilih citra yang akan di proses dengan ukuran 400x500 piksel untuk memasuki tahap selanjutnya.

3.3 Ekstraksi Ciri

Setelah melewati tahap akuisisi citra, selanjutnya citra akan melewati tahap ekstraksi ciri dengan metode SIFT. Citra akan melewati empat proses, yaitu pencarian nilai ekstrim, lokalisasi *keypoint*, pemberian orientasi dan *descriptor keypoint*. Keluaran dari tahap ini adalah sebuah vektor citra yang akan menjadi *input*an pada tahap selanjutnya.



Gambar 3.3 Diagram Proses *Ekstraksi* Ciri dengan Metode SIFT

Gambar 3.3 menjelaskan tentang proses ekstraksi ciri pada citra menggunakan metode SIFT. Citra dari proses sebelumnya akan menjadi citra *input*an. Citra yang sudah masuk kedalam sistem akan dicari nilai ekstrimnya dan akan menghasilkan kandidat *keypoint*. Setelah kandidat *keypoint* ditemukan maka langkah selanjutnya adalah lokalisasi keypoint. Pada tahap ini, keypoint yang memiliki nilai kontras rendah dan berada

pada garis tepi lebih dari 10 akan dihilangkan. Setelah itu, *keypoint* akan diberikan orientasi agar nilai keypoint tetap meski citra mengalami perubahan. Tahap terakhir adalah mencari nilai deskriptor pada setiap titik keypoint.

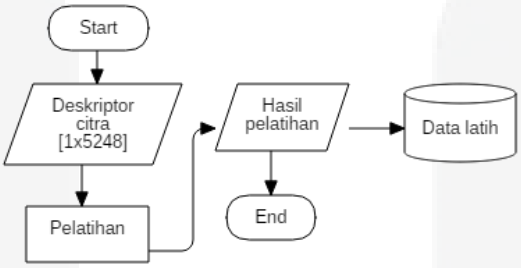
3.4 Klasifikasi Citra

Setelah didapat deskriptor *keypoint* maka tahap selanjutnya adalah mengubah matriks deskriptor *keypoint* menjadi matriks satu dimensi.

Gambar 3. 4 Proses Mendapatkan Matriks Satu Dimensi

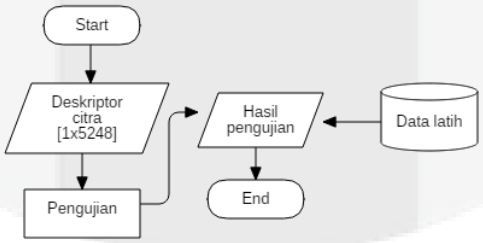
Gambar 3.4 menjelaskan tentang proses perubahan matriks deskriptor citra menjadi matriks satu dimensi. Hal ini dilakukan karena data *input* yang digunakan oleh metode SVM berupa matriks satu dimensi dengan ukuran yang sama. Pada proses ini, deskriptor citra berupa matriks [128xjumlah *keypoint*] akan masuk kedalam tahap pencarian nilai *keypoint* minimun. Nilai *keypoint* minimum inilah yang akan menjadi acuan untuk merubah matriks deskriptor citra. Setelah nilai *keypoint* minimum didapatkan, yaitu 41 maka langkah selanjutnya adalah merubah ukuran baris matriks menjadi [1x(128xjumlah *keypoint*)]. Setelah itu, jumlah *keypoint* akan dipangkas mengikuti nilai *keypoint* minimum sehingga didapatkan matriks [1x(128x41)]. Hasil akhir dari proses ini adalah deskriptor citra berupa matriks satu dimensi dengan ukuran [1x5248]

Setelah mendapatkan matriks satu dimensi dengan ukuran yang sama, maka tahap selanjutnya adalah klasifikasi citra menggunakan metode SVM. Dengan metode ini, klasifikasi citra dilakukan dengan menghitung nilai *hyperplane*.



Gambar 3. 5 Diagram Proses Pelatihan Citra dengan SVM

Gambar 3.5 menjelaskan tentang alur proses pelatihan citra dengan menggunakan metode SVM. Deskriptor citra yang sudah di dapatkan dari proses sebelumnya akan dilatih agar sistem yang dibuat dapat mengenali rambu yang akan di*input*kan. Hasil pelatihan tersebut akan disimpan dalam bentuk *.xml



Gambar 3. 6 Diagram Proses Pengujian Citra dengan Metode SVM

Gambar 3.6 menjelaskan tentang alur pengujian citra menggunakan metode SVM. Deskriptor citra yang merupakan inputan pada sistem didapatkan dari proses akuisisi citra. Setelah mendapatkan deskriptor citra, sistem akan melakukan pengujian dengan cara mencari nilai *hyperplane* terbaik yang didapat dari deskriptor citra dan data hasil pelatihan.

4. Implementasi dan Pengujian Sistem

4.1 Pengujian Resolusi Citra

Pengujian resolusi citra dilakukan untuk mengetahui resolusi yang tepat dengan melihat nilai akurasi yang dihasilkan. Terdapat 4 resolusi yang akan diujikan, yaitu 48x60, 96x120, 192x140 dan 400x500 piksel dengan

menggunakan parameter *keypoint* minimum, waktu, ukuran file dan akurasi yang didapat. Pengujian ini dilakukan menggunakan aplikasi Netbeans dengan menguji 25 rambu yang belum melewati tahap *preprocessing*. Dengan kata lain, rambu tersebut merupakan citra asli dan belum mengalami perubahan, seperti *grayscale* dan biner.

Tabel 4. 1 Hasil Pengujian Resolusi Citra

No	Resolusi (piksel)	*Keypoint* Minimum	Waktu (detik)	Ukuran File (KB)	Tingkat Akurasi (%)
1	48x60	10	28	6.919	60
2	96x120	18	33	12.396	56
3	192x240	33	56	22.639	64
4	400x500	41	93	28.146	84

Tabel 4.1 memperlihatkan hasil dari pengujian resolusi citra. Dari tabel tersebut dapat disimpulkan bahwa resolusi citra sangat mempengaruhi jumlah *keypoint* minimum. Ketika jumlah *keypoint* minimum rendah, tingkat akurasi yang dihasilkan pun akan rendah. Namun, ketika jumlah *keypoint* minimum tinggi, tingkat akurasi yang dihasilkan pun akan tinggi. Hasil pengujian ini pun menunjukan resolusi dengan tingkat akurasi terbaik, yaitu resolusi 400x500 piksel.

4.2 Pengaruh P*reprocessing*

Dalam pengujian ini, akan dilakukan analisis pengaruh *preprocessing* terhadap tingkat akurasi. *Preprocessing* yang di lakukan berupa perubahan citra menjdai *grayscale* dan biner dengan parameter nilai *keypoint* minimum, waktu, ukuran file dan tingkat akurasi yang didapat. *Grayscale* merupakan perubahan citra menjadi abu-abu dengan menghitung nilai rata-rata dari komponen warna RGB (*Red, Green, Blue*) sehingga citra tersebut memiliki tingkat derajat keabuan. Sedangkan citra biner merupakan perubahan komponen warna menjadi BW (*Black White*) dengan menggunakan nilai *threshold* sebagai acuan. Pengujian ini dilakukan menggunakan aplikasi Netbeans dengan menguji 25 rambu.

Tabel 4. 2 Hasil Pengujian Citra *Grayscale*

No	Resolusi (piksel)	*Keypoint* Minimum	Waktu (detik)	Ukuran File (KB)	Tingkat Akurasi (%)
1	48x60	12	27	8.291	68
2	96x120	18	38	12.398	60
3	192x240	31	51	21.269	60
4	400x500	42	90	28.831	80

Table 4.2 merupakan hasil pengujian citra *grayscale*. Dari tabel tersebut dapat disimpulkan bahwa tingkat akurasi terbaik dimiliki oleh citra dengan resolusi 400x500 piksel. Namun, tingkat akurasi tersebut lebih rendah 4% dari hasil tingkat akurasi citra yang belum melewati tahap *grayscale.*

Tabel 4. 3 Hasil Pengujian Citra Biner

No	Resolusi (piksel)	*Keypoint* Minimum	Waktu (detik)	Ukuran File (KB)	Tingkat Akurasi (%)
1	48x60	6	26	4.175	52
2	96x120	21	33	14.461	72
3	192x240	30	45	20.507	68
4	400x500	40	80	26.997	60

Tabel 4.3 memperlihatkan hasil pengujian citra biner. Berbeda dari pengujian sebelumnya, pada pengujian ini tingkat akurasi terbaik dimiliki oleh citra dengan ukuran 96x120 piksel dimana tingkat akurasi yang didapat sebesar 72%. Namun, hasil tersebut lebih rendah 8% dari hasil pengujian citra *grayscale*.

4.3 Penguian Jumlah Citra Pada Data Latih

Pada tahap ini akan dilakukan pengujian terhadap pengaruh banyaknya citra pada data latih dengan menggunakan data terbaik yang didapat dari hasil pengujian sebelumnya, yaitu resolusi 400x500 piksel dengan menggunakan citra asli. Parameter yang digunakan adalah akurasi dan ukuran file latih yang dihasilkan. Pengujian ini dilakukan dengan menggunakan aplikasi Netbeans terhadap 25 rambu.

Tabel 4. 4 Hasil Pengujian Jumlah Data Latih

No	Jumlah Citra	Tingkat Akurasi (%)	Ukuran File (KB)
1	75	12	4.114
2	150	4	14.320
3	225	84	28.146

Dari tabel 4.4 dapat disimpulklan bahwa semakin banyak jumlah citra yang terdapat pada data latih akan membuat tingkat akurasi semakin tinggi. Namun, ukuran file yang dihasilkan akan lebih besar.

4.4 Pengujian Sudut Pengambilan Citra

Tujuan dari pengujian ini adalah untuk mengetahui sudut pengambilan yang tepat untuk mendeteksi citra. Terdapat beberapa kondisi yang akan diujikan, yaitu pengambilan dilakukan dari arah kanan, kiri dan bawah citra dengan sudut 15°, 30° dan 45° serta pengambilan dilakukan tepat pada bagian tengah citra dengan sudut 0°**.** Pengujian ini dilakukan kepada 25 rambu menggunakan kamera utama *smartphone* dengan parameter nilai akurasi.

Tabel 4. 5 Hasil Pengujian Pengambilan Citra Berdasarkan Sudut

No.	Sudut (°)	Tingkat Akurasi (%)			
		Kanan	Kiri	Bawah	Tengah
1	0	-	-	-	88
2	15	68	68	72	-
3	30	56	48	56	-
4	45	28	28	36	-

Tabel 4.5 memperlihatkan hasil pengujian pengambilan citra berdasarkan sudut. Dari tabel tersebut dapat disimpulkan bahwa sudut terbaik untuk mengambil citra adalah 0°**.**

4.5 Pengujian Jarak Pengambilan Citra

Pengujian berdasarkan jarak pengambilan citra ini dilakukan untuk mengetahui jarak yang baik dalam pengambilan citra. Pengujian ini dilakukan kepada 25 rambu dengan 3 jarak yang berbeda, yaitu 78 cm, 90 cm dan 148 cm.

Tabel 4. 6 Hasil Pengujian Pengambilan Citra Berdasarkan Jarak

No	Jarak (cm)	Tingkat Akurasi (%)
1	78	36
2	90	88
3	148	24

Tabel 4.6 merupakan hasil pengujian pengambilan citra berdasarkan jarak. Data tersebut menunjukan bahwa jarak terbaik untuk melakukan pengambilan citra adalah 90 cm. Dimana pada jarak tersebut posisi citra berada tepat pada batas hijau yang berada pada kamera.

4.6 Pengujian Pengaruh Cahaya

Pengujian cahaya dilakukan untuk mengetahui keadaan seperti apa yang tepat untuk melakukan deteksi rambu. Terdapat dua jenis pengujian yang akan dilakukan, saat cahaya terang dan cahaya redup. Untuk cahaya terang, dilakukan pengujian pada pukul 9.30 WIB – 12.30 WIB dengan cahaya 10896 lux – 4105 lux. Sedangkan pengujian untuk cahaya redup dilakukan pada pukul 16.00 WIB – 18.00 WIB dengan cahaya 2124 lux – 191 lux. Nilai cahaya didapat dengan menggunakan aplikasi *Light* Meter. Pengambilan citra dilakukan dalam keadaan lurus dan dengan jarak tepat pada batas yang terdapat pada kamera aplikasi.

Tabel 4. 7 Hasil Pengujian Pengaruh Cahaya

No.	Cahaya (lux)	Tingkat Akurasi (%)
1	10896-4105	88
2	2124-191	68

Pada tabel 4.7 memperlihatkan hasil tingkat akurasi yang didapat pada saat pengujian pengaruh cahaya. Dari data tersebut dapat disimpulkan bahwa tingkat cahaya yang dibutuhkan untuk menghasilkan tingkat akurasi terbaik adalah 10896-4105 lux.

4.7 Pengujian Waktu Respon Sistem

Pengujian waktu respon sistem ini bertujuan untuk mengetaui waktu yang dibutuhkan saat melakukan pendeteksian dengan perangkat *smartphone* yang berbeda-beda. Parameter yang digunakan berupa spesifikasi perangkat, seperti *processor,* versi android, ukuran RAM (*Random Access Memory*).

Tabel 4. 8 Hasil Pengujian Waktu Respon Sistem

No.	Perangkat	Versi Android	Jenis *Processor* (GHz)	Ukuran RAM (GB)	Waktu (detik)
1	I	5.1	1,3 Quad-Core	2	5,99
2	II	6.1	2,5 Quad-Core	2	1,17
3	III	4.4	1,2 Quad-Core	2	7,56

Tabel 4.8 memperlihatkan bahwa pada perangkat I dengan spesifikasi *processor* 1,3 GHz Quad-Core, RAM 2 GB dan versi android 5.1, waktu yang dibutuhkan aplikasi untuk mendeteksi citra adalah 5,99 detik. Pada perangkat II dengan spesifikasi *processor* 2,5 GHz Quad-Core, RAM 2 GB dan versi android 6.1 dibutuhkan waktu 1,17 detik. Pada perangkat III dengan spesifikasi I *processor* 1,2 GHz Quad-Core, RAM 2 GB dan versi android 4.4, dibutuhkan waktu 7,56 detik untuk mendeteksi.

5. Kesimpulan

5.1 Kesimpulan

(1) Metode *Scale Invariant Feature Transform* (SIFT) dan Support Vector Machine dapat diimplementasilan pada aplikasi *mobile* berbasis android untuk mendeteksi rambu lalu lintas.
(2) Untuk hasil yang maksimal, citra yang digunakan untuk metode SIFT dan SVM adalah citra yang tidak melewati tahap *grayscale* dan biner dengan resolusi 400x500.
(3) Jumlah data *training* yang digunakan akan berbanding lurus dengan hasil deteksi dan ukuran file *training*. Semakin banyak data *training* yang digunakan maka semakin baik deteksi yang dihasilkan. Namun, file *training* akan semakin besar.

5.2 Saran

(1) Menggunakan metode *matching* untuk ROI pada tahap akuisisi citra.
(2) Mengimplementasikan metode SIFT dengan metode klasifikasi lain.
(3) Mengimplementasikan metode SIFT dengan menggunakan objek lain.

Daftar Pustaka

[1] Republik Indonesia. 2014. Peraturan Menteri Perhubungan Republik Indonesia No. 13 Tahun 2014 tentang Rambu Lalu Lintas.
[2] Satyaputra, Alfa dan Eva Maulina Aritonang. 2016. "***Let's Build Your Android Apps with Android Studio***". Jakarta : PT. Elex Media Komputindo.
[3] Lowe, David G. (2014). "***Distinctive Image Features from Scale-Invariant Keypoints***". Computer Science Departement. University of British Columbia. Canada.
[4] Satriyo Nugroho, Anto, Arief Budi Wirarto dan Dwi Handoko. "***Support Vector Machine – Teori dan Aplikasinya dalam Bioinformatika*".** Proceeding of Indonesian Scientific Meeting in Central Japan, Desember 2003.
[5] Kus, Merve Can, Muhittin Gokmen, Sima Etaner – Uyar. 2008. "**Traffic Sign Recognition using Scale Invariant Feature Transform and Color Classification**". *IEEE. IEEE.*
[6] Gunn, Steve. R. 1998. "***Support Vector Machine for Classification and* Regression".** School of Electronics and Computer Science. University of Southampton.
[7] Sanabila. 2015. "***Emergency Assembly Point***". (http://www.sanabila.com/2015/10/emergency-assembly-point.html?m=1, diakses tanggal 6 Agustus 2017).

[8] Republik Indonesia. 1995. Peraturan Pemerintah Republik Indonesia No. 19 Tahun 1995 tentang Pemeliharaan dan Pemanfaatan Benda Cagar Budaya di Museum.
[9] Putri, Fardilla Zardi. "**Perancangan dan Implementasi *Directional Feature Extraction* dan *Support Vector Machines* untuk Menerjemah Kata dengan Pengenalan Huruf Hiragana dalam Bahasa Jepang ke Bahasa Indonesia Berbasis Android**", Fakultas Teknik Elektro Universitas Telkom. Juni 2016
[10] Pratama, Lingga Eka. 2014. "**Implementasi Alhoritma SIFT untuk Melaukan Klasifikasi Bahan Bakar Kendaraan Roda Empat pada SPBU**", Fakultas Teknik Informatika, Universitas Komputer Indonesia.