APLIKASI E-TOUR GUIDE DENGAN FITUR PENGENALAN IMAGE MENGGUNAKAN METODE HAAR CLASSIFIER

Derwin Suhartono; William Surya Permana; Antoni Wiguna; Ferlan Gisman Putra

Computer Science Department, School of Computer Science, Binus University
Jl. K.H. Syahdan No. 9, Palmerah, Jakarta Barat 11480
dsuhartono@binus.edu

ABSTRACT

Smartphone has became an important instrument in modern society as it is used for entertainment and information searching except for communication. Concerning to this condition, it is needed to develop an application in order to improve smart phone functionality. The objective of this research is to create an application named E-Tour Guide as a tool for helping to plan and manage tourism activity equipped with image recognition feature. Image recognition method used is the Haar Classifier method. The feature is used to recognize historical objects. From the testing result done to 20 images sample, 85% accuracy is achieved for the image recognition feature.

Keywords: *smartphone, e-tour guide, Haar Classifier, image recognition*

ABSTRAK

Smartphone telah menjadi instrumen penting dalam kehidupan masyarakat modern untuk hiburan dan mencari informasi selain untuk berkomunikasi. Melihat kondisi tersebut, dinilai perlu untuk mengembangkan aplikasi dengan tujuan untuk meningkatkan fungsionalitas smartphone. Tujuan penelitian ini adalah menghasilkan aplikasi E-Tour Guide sebagai alat bantu dalam merencanakan dan mengatur kegiatan wisata yang dilengkapi fitur image recognition. Metode image recognition yang digunakan adalah Haar Classifier. Fitur ini digunakan untuk mengenali objek bersejarah. Dari hasil pengujian yang dilakukan pada aplikasi ini khususnya pada fitur image recognition, diperoleh akurasi sebesar 85% dari jumlah image yang diuji sebanyak 20 buah.

Kata kunci: *smartphone, e-tour guide, Haar Classifier, image recognition*

PENDAHULUAN

Masyarakat modern sangat sering bepergian atau berwisata sebagai salah satu aktivitas untuk menghilangkan kejenuhan. Hal ini dapat dilihat dari jumlah orang yang mengunjungi suatu daerah perharinya. Sebagian besar dari mereka akan merencanakan kegiatan wisatanya dan mencari informasi terlebih dahulu mengenai tempat wisata yang akan mereka kunjungi.

Smartphone memberikan fasilitas dasar yang dimiliki oleh sebuah *mobile phone* pada umumnya, dan didukung dengan beberapa fasilitas dari sebuah *personal computer* (PC). Sebagian besar masyarakat modern yang mengikuti perkembangan teknologi telah memiliki *device* tersebut dan selalu membawanya ke mana saja.

Melihat kondisi tersebut, *smartphone* dapat digunakan sebagai pemandu wisata. Dengan rancangan aplikasi *E-Tour Guide*, *smartphone* dapat digunakan untuk mengatur kegiatan liburan pengguna saat berkunjung ke tempat wisata. Cara penggunaan aplikasi ini akan dibuat semudah mungkin sehingga dapat membuat pengguna semakin tertarik menggunakannya.

Aplikasi yang diberi nama *E-Tour Guide* ini akan memiliki fungsi pemilihan tempat wisata, penjadwalan kegiatan wisata, dan memberikan pengguna hak untuk memberikan *rating* dan komentar untuk suatu tempat wisata. Aplikasi ini dilengkapi pula dengan fitur *image recognition* untuk mengenali objek bersejarah yang ada di suatu tempat wisata agar pengguna lebih tahu mengenai benda tersebut. Metode image recognition yang digunakan adalah Haar Classifier.

Menurut Seo (2012), *Haar classifier* adalah metode *object detection* yang telah disediakan oleh *OpenCV* dalam membangun sebuah *boosted rejection cascade* yang akan membuang data *training* negatif sehingga didapat suatu keputusan untuk menentukan data positif. *Haar classifier* merupakan metode *supervised learning*, yaitu membutuhkan data *training* untuk dapat mendeteksi objek-objek tertentu. Untuk itu, *Haar classifier* membutuhkan data positif (objek yang akan dideteksi) dan data negatif (bukan objek yang akan dideteksi).

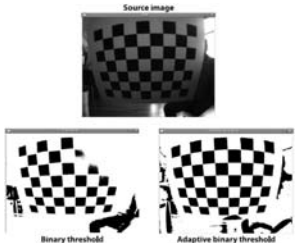
Untuk mengenali gambar, perlu dilakukan konversi warna, *median filter*, dan *edge detection* (Suri, et al., 2010). Konversi warna gambar sendiri dapat dipecah menjadi tiga *channel,* yaitu merah, hijau, dan biru; atau diubah menjadi satu *channel,* yaitu *grayscale*. *Grayscale* lebih umum digunakan karena lebih sederhana dalam perhitungannya dan lebih mendekati hasil dalam penerapannya. *Median filter* digunakan untuk menghilangkan *noise*. Suri et al. menggunakan 5x5 *smoothing* lalu dilanjutkan dengan 5x5 *median filter*. *Edge detection* digunakan untuk mendapatkan magnitudo gradien di setiap titik dari sampel gambar. Ada beberapa teknik *edge detection* seperti Roberts, Sobel, dan Prewitt. Namun mereka menggunakan teknik Sobel. Teknik Sobel akan melakukan pengukuran gradien *spatial* 2D pada suatu gambar, sehingga akan menekankan area dengan frekuensi *spatial* yang tinggi, yang biasanya dimiliki oleh *edge*. Teknik Sobel sendiri menggunakan matriks $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ untuk Gx-nya, dan matriks $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$ untuk Gy-nya. Mereka menyimpulkan bahwa dengan menggunakan teknik Sobel, mereka dapat melakukan pendeteksian plat kendaraan. Namun dalam beberapa kasus, hasilnya kurang maksimal karena adanya segmen cahaya yang kurang tepat atau adanya efek-efek pencahayaan yang beragam.

Image processing adalah penggunaan operator dengan tingkat yang lebih tinggi dibanding dengan yang terdefinisi pada struktur gambar untuk menyelesaikan tugas-tugas yang maknanya secara

alami dalam konten grafis (Bradski & Kaehler, 2008). Beberapa contoh *image processing* adalah: (1) *smoothing/blurring* (Gambar 1), untuk mengurangi *noise* atau *camera artifacts*. *Smoothing* juga penting ketika kita ingin mengurangi resolusi suatu gambar; (2) *resize* (Gambar 2), yaitu mengubah ukuran suatu gambar menjadi lebih besar atau kecil; (3) *threshold* (Gambar 3), untuk menghilangkan piksel yang berada di atas atau di bawah nilai tertentu dan menampilkan sisanya.

Gambar 1 Contoh *smoothing/blurring* (Bradski & Kaehler, 2008)

Gambar 2 Contoh *resize* (Bradski & Kaehler, 2008)



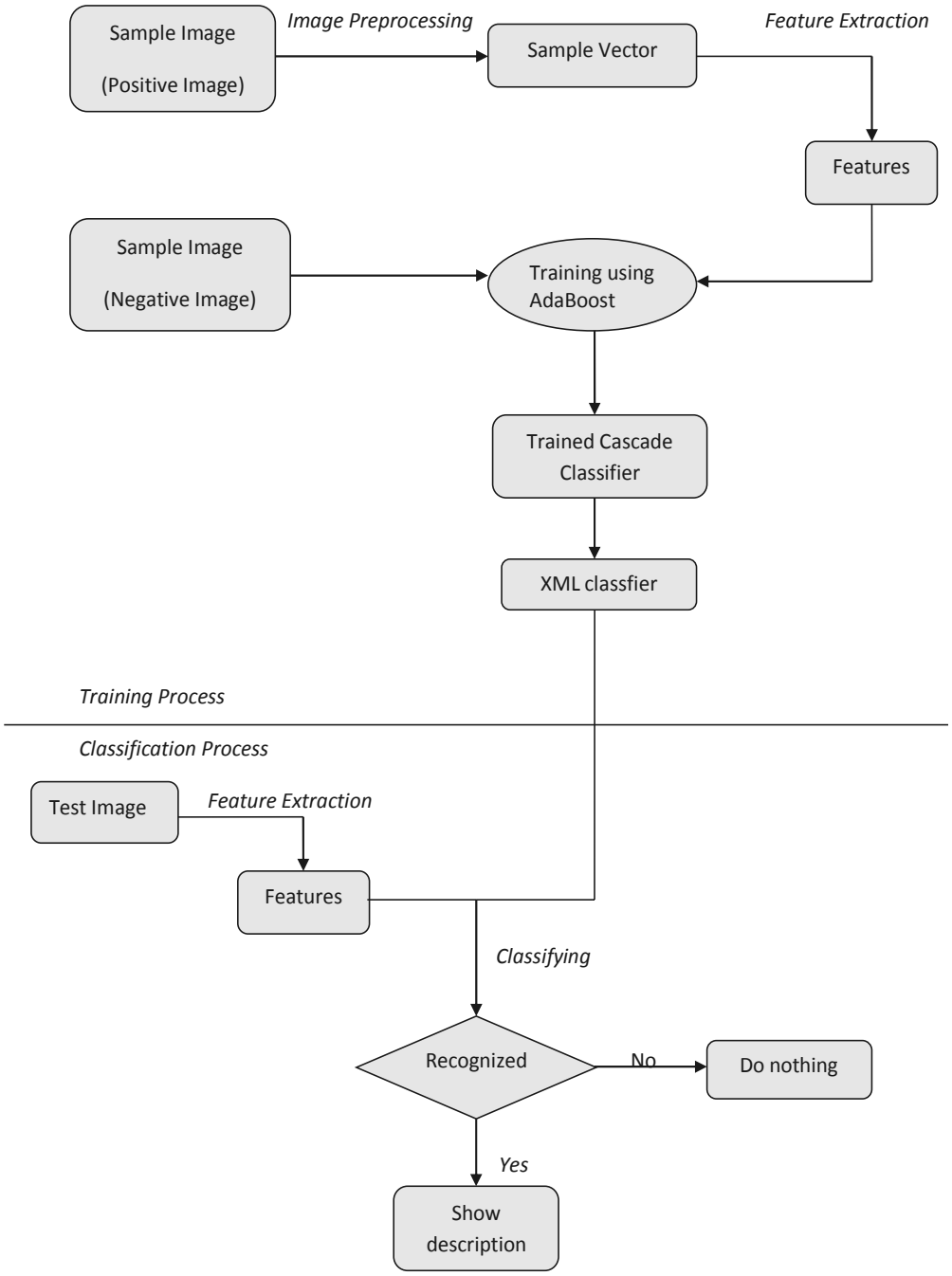
Gambar 3 Contoh *threshold* (Bradski & Kaehler, 2008)

Singh & Kathane (2011) menyatakan bahwa dalam pengolahan gambar (*image processing*) perlu dilakukan *denoising* terlebih dahulu sebelum *edge detection*. *Denoising* adalah proses untuk menghilangkan *noise.* Mereka melakukan *mean filtering* terlebih dahulu, lalu diikuti dengan *gaussian filtering* dan *strong gaussian*. *Edge detection* digunakan untuk mempermudah segmentasi dan identifikasi. Ada beberapa teknik yang digunakan dalam *edge detection,* seperti *binary morphology*, *sobel*, *canny*, *log* dan *differential operator*. Mereka telah melakukan pengujian terhadap teknik-teknik di atas dan menyimpulkan bahwa teknik *binary morphology* maupun *sobel operator* lebih baik dibandingkan teknik-teknik yang lain.

Penelitian selanjutnya dilakukan oleh Kaur, e al. (2011) yang menghasilkan sebuah Ant Colony Optimization, sebuah teknik untuk Image Processing. *Ant colony optimization* (ACO) adalah teknik probabilistik untuk memecahkan banyak masalah yang dapat dikurangi untuk menemukan jalur yang baik melalui grafik. *Ant* tersebut bisa mencari jalur terpendek untuk memuat sumber makanan ke sarang mereka dengan cairan substansi yang disebut *pheromone*, yang akan mereka keluarkan di rute transit. Jalur terpendek dari *ant* memiliki lebih *pheromone* dibanding jalur terpanjang. Jadi informasi *pheromone* memperbarui yang diperlukan dalam ACO. Langkah- langkah dari algoritma *ant colony optimization* (ACO) adalah *problem graph representation*, *ants allocation initalizing, ants possibility distribution rule, update global trail* dan *stopping procedure.*

METODE

Berikut adalah kerangka kerja yang telah dibuat (Gambar 4).



Gambar 4 Kerangka kerja metode

Training Process

Pada proses ini mula-mula dilakukan pengumpulan gambar objek yang ingin dikenali. Setelah itu, *image* diubah menjadi *sample vector*. Untuk memasuki tahap *training*, dibutuhkan juga pengumpulan gambar *negative* yang tidak mengandung objek yang ingin dikenali. *Training* dilakukan dengan menggunakan metode AdaBoost yang bertujuan utama untuk memilih beberapa set *feature* yang menggambarkan objek sebaik mungkin dan membuat *classifier* kuat yang merupakan kombinasi dari *best feature.* Secara matematis, *weak classifier* dirumuskan dengan

$$h(x, f, p, \theta) = \begin{cases} 1 & \text{if } pf(x) > p\theta \\ 0 & \text{otherwise} \end{cases}$$

di mana *x* adalah sebuah 24 x 24 *pixel sub-window*, *f* adalah *feature* yang digunakan, *p* adalah *polarity*, dan θ adalah *threshold* di mana *x* diklasifikasikan sebagai *positive* atau *negative image*.

Sedangkan s*trong classifier* dirumuskan dengan

$$C(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$ di mana $\alpha_t = \log \frac{1}{\beta_t}$

Hasil *training* yang berupa *cascade classifier* perlu diubah menjadi *XML file* agar bisa digunakan pada aplikasi.

Classification Process

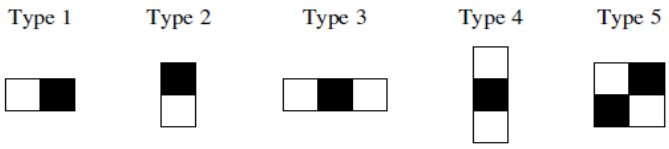
Tahap ini akan menggunakan *XML file* yang didapat dari proses sebelumnya. Dengan menggunakan *classifier* tersebut dilakukan pengecekan pada gambar, apakah gambar mengandung objek yang dikenali apa tidak. Jika terdapat objek yang dikenali, aplikasi akan menampilkan deskripsi dari objek tersebut.

Image Preprocessing

Proses ini pada *sample image* dilakukan untuk beberapa tujuan berikut ini: (1) mengurangi ukuran gambar agar tidak membebani CPU; (2) membuang detail gambar yang tidak diperlukan. Tahapan *preprocessing* ini terbagi menjadi dua, yaitu: (1) *crop image,* untuk mendapatkan bagian gambar yang diinginkan. *Cropping* ini penting untuk menentukan fitur yang akan didapat nantinya; (2) *mark the object* – pemberian tanda pada gambar dilakukan untuk digunakan pada pembuatan sampel vektor yang akan digunakan saat *training*.

Feature Extraction

Aplikasi ini menggunakan *feature* yang terdiri dari dua atau lebih persegi untuk menganalisis suatu gambar (Gambar 5).



Gambar 5 *Feature* (Jensen, 2008)

Fitur-fitur ini mungkin tampak terlalu sederhana untuk melakukan tugas sulit seperti sebagai deteksi gambar, tapi fitur dengan kompleksitas yang rendah pasti memiliki efisiensi komputasi yang tinggi.

Prosedur Training

Dalam melakukan *training* untuk membuat *haar trained .xml file,* dibutuhkan banyak gambar dengan dan tanpa objek yang ingin dikenali, yang biasa disebut *positive* dan *negative images*. Sekitar 1000-2000 *positives* dan 3000 *negatives* dibutuhkan untuk melakukan *training*.

Pertama-tama objek yang ada pada *positives* harus didata dengan format "[path] [jumlah objek] [posisi x] [posisi y] [lebar] [tinggi]" (Gambar 6). Selanjutnya, *negatives* juga didata (Gambar 7).

```
rawdata/B-train001.bmp 1 7 15 43 44
rawdata/B-train002.bmp 1 10 19 48 48
rawdata/B-train003.bmp 1 14 15 35 47
rawdata/B-train004.bmp 1 11 17 35 44
rawdata/B-train005.bmp 1 15 14 37 48
rawdata/B-train006.bmp 1 15 16 35 47
rawdata/B-train007.bmp 1 14 16 41 46
rawdata/B-train008.bmp 1 17 19 36 44
rawdata/B-train009.bmp 1 14 16 38 52
```

Gambar 6 Contoh teks berisi data *positives*

```
0.bmp
01novel.1.600.bmp
124020.bmp
1256290319-191853525.bmp
1256290431-1354793674.bmp
1256290756-869792524.bmp
```

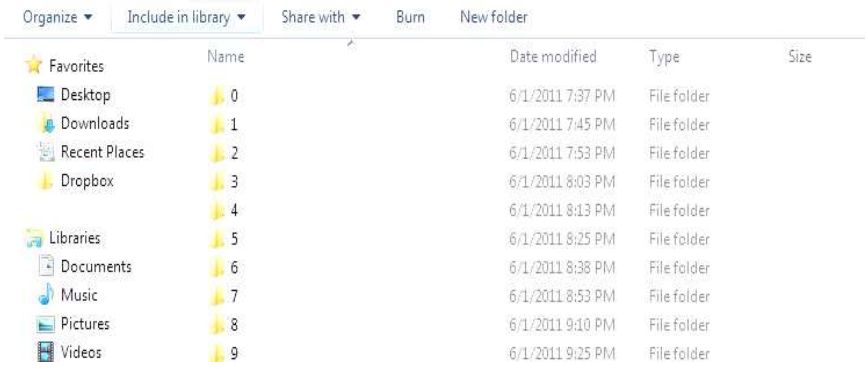
Gambar 7 Contoh teks berisi data *negatives*

Lalu, buatlah vektor dari *positives* menggunakan perintah "`createsamples.exe -info positive/info.txt -vec data/vector.vec -num 527 -w 24 -h 24`". Untuk melihat hasil dari vektor bisa menggunakan perintah "`createsamples.exe -vec data/vector.vec -w 24 -h 24`".

Setelah itu, lakukan *training* dengan menggunakan perintah "`haartraining.exe -data data/cascade -vec data/vector.vec -bg negative/infofile.txt -npos 527 -nneg 1142 -nstages 30 -mem 1000 -mode ALL -w 24 -h 24 -nonsym`". Proses *training* (Gambar 8) ini biasanya memakan waktu yang sangat lama. Walaupun proses *training* belum selesai, hasil *training* (Gambar 9) yang sudah dilakukan bisa digunakan.

```
+---+----+---+---------+---------+---------+
| 2|100%|-|-0.962359| 1.000000| 1.000000| 0.317829|
+---+----+---+---------+---------+---------+
| 3| 96%|-|-1.083836| 1.000000| 1.000000| 0.343023|
+---+----+---+---------+---------+---------+
| 4| 89%|-|-0.630941| 0.995910| 0.966006| 0.347545|
+---+----+---+---------+---------+---------+
| 5| 89%|-|-1.144969| 1.000000| 0.977337| 0.315891|
+---+----+---+---------+---------+---------+
| 6| 88%|-|-0.812312| 0.995910| 0.901794| 0.247416|
+---+----+---+---------+---------+---------+
| 7| 85%|-|-0.887186| 0.995910| 0.898961| 0.218346|
+---+----+---+---------+---------+---------+
| 8| 86%|-|-1.136884| 0.995910| 0.898961| 0.252584|
+---+----+---+---------+---------+---------+
| 9| 90%|-|-1.022194| 0.995910| 0.839471| 0.217054|
+---+----+---+---------+---------+---------+
|10| 83%|-|-0.935341| 0.995910| 0.779981| 0.193152|
+---+----+---+---------+---------+---------+
```

Gambar 8 Contoh tampilan saat *training* dilakukan



Gambar 9 Contoh hasil *training*

Untuk mengubah hasil *training* menjadi *xml file,* gunakan perintah "`convert_cascade.exe --size="20x20" haarcascade haarcascade.xml`". Untuk mengecek tingkat keakuratan *cascade file* dalam mengenali objek bisa menggunakan perintah "`performance.exe -data haarcascade.xml -info tests.txt -ni`"

HASIL DAN PEMBAHASAN

Metode Haar Classifier yang digunakan untuk pendeteksian *image* diletakkan pada salah satu dari berbagai macam fitur yang dibangun dalam aplikasi. Pada saat "Scan Object" dipilih dari menu utama, akan tampil gambar yang di-*capture* dari kamera (Gambar 10). Apabila objek terdeteksi, akan tampil juga keterangan mengenai gambar tersebut.

Gambar 10 Tampilan layar *scan object*

Untuk mengevaluasi ketepatan fitur *object recognition* dalam aplikasi *E-Tour Guide*, dilakukan *testing* yang menggunakan sampel berupa 20 *positive image* (Tabel 3).

Tabel 3 Hasil Testing terhadap 20 *Sampel Positive Image*

File	*Hit*	*Miss*	*False*
1	1	0	1
2	0	1	0
3	1	0	0
4	1	0	0
5	1	0	0
6	1	0	0
7	1	0	0
8	1	0	0
9	1	0	0
10	1	0	0
11	1	0	0

12	1	0	0
13	0	1	0
14	1	0	0
15	1	0	0
16	1	0	0
17	1	0	0
18	0	1	0
19	1	0	0
20	1	0	0

Dari data *testing* di atas didapat persentase keberhasilan untuk mendeteksi objek sebagai berikut:
Persentase keberhasilan pendeteksian objek = 17/20 x 100% = 85%
Persentase kegagalan pendeteksian objek = 3/20 x 100% = 15%
Persentase kesalahan pendeteksian objek = 1/20 x 100% = 5%

Dari keempat kegagalan klasifikasi dan satu kesalahan klasifikasi yang terjadi, diperoleh hasil analisisnya sebagai berikut: (1) proses *image preprocessing* yang dilakukan hanya menggunakan *cropped image*, sehingga hasilnya kurang menggambarkan objek secara keseluruhan; (2) objek berupa 3 dimensi dan ukurannya yang besar sehingga perbedaan posisi saat pengambilan gambar sangat mempengaruhi hasil klasifikasi; (3) *stage training* yang dilakukan masih kurang sehingga masih kurangnya jumlah *weak classifier* yang perlu dikumpulkan agar menghasilkan suatu *strong classifier*.

PENUTUP

Dari uji coba yang dilakukan, dapat disimpulkan bahwa; (1) aplikasi *E-Tour Guide* dapat memudahkan pengguna mengatur kegiatan wisatanya dan menambah fungsionalitas *smartphone* berbasis *Android*; (2) persentase keberhasilan aplikasi ini dalam mendeteksi *image* sangat baik, yaitu 85% dari 20 sampel *image;* (3) pengambilan *image* harus mempertimbangkan jarak dan kemiringan dari kamera ke *image*, sehingga presisi yang lebih tepat bisa diperoleh; (4) keterbatasan penelitian ini adalah pada saat dilakukan pengujian. Pengujian yang dilakukan masih terbatas dengan menggunakan satu objek dengan 20 sample *image*. Metode pengenalan *image* belum diujicobakan dengan menggunakan objek-objek lain.

DAFTAR PUSTAKA

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV.* California: O'Reilly.

Jensen, O. H. (2008). *Implementing the Viola-Jones Face Detection Algorithm.* Skripsi tidak diterbitkan. Technical University of Denmark, Lyngby, Denmark.

Kaur S., Agarwal P., Rana, R.S. (2011). Ant colony optimization: a technique used for image processing. *International Journal of Computer Science and Technology, 2* (2). ISSN: 2229-4333 / 0976-8491.

Seo, N. (2012). *Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features.* Diakses dari http://note.sonots.com/SciSoftware/haartraining.html.

Singh, S. K., dan Kathane, A. (2011). Various methods for edge detection in digital image processing, *International Journal of Computer Science and Technology, 2* (2). ISSN: 2229-4333 / 0976-8491.

Suri, P.K., Walia E., Verma, E.A. (2010). Vehicle number plate detection using sobel edge detection technique. *International Journal of Computer Science and Technology,* 1(2). ISSN: 2229-4333 / 0976-8491.