Sistem Pengenalan Plat Nomor Mobil Dengan Metode Principal Components Analysis

Resmana Lim [1], Lukman Vendy W. [2], Kartika Gunadi [2]
[1]Fakultas Teknologi Industri, Jurusan Teknik Elektro, Universitas Kristen Petra
[2]Fakultas Teknologi Industri, Jurusan Teknik Informatika, Universitas Kristen Petra
e-mail: resmana@petra.ac.id, kgunadi@petra.ac.id

Abstrak

Paper ini membahas sistem pengenalan plat nomor mobil menggunakan teknik computer vision. Image mobil diambil dari kamera, dan selanjutnya diidentifikasi secara otomatis dengan cara mencari lokasi plat nomor mobil tersebut, mensegmentasi setiap karakter yang ada dari plat tersebut dan kemudian melakukan pengenalan pada setiap karakter dengan metode feature reduction PCA. Aplikasi ini dibangun menggunakan Microsoft Visual C++ 6.0®, Microsoft® DirectShow®, Intel® Image Processing Library dan Open Source Computer Vision Library. Aplikasi ini telah diimplementasikan dan dapat mendeteksi letak plat nomor serta mengenalinya pada PC Pentium II/400 MHz. Sistem telah diujicobakan pada suatu basis data yang mewakili 26 karakter (0-9, A-Z) yang masing-masing terdiri dari beberapa variasi gambar mobil. Hasil uji coba menunjukkan tingkat keberhasilan yang cukup menggembirakan, dimana tingkat keberhasilan mencapai ±82%. Sistem cukup prospektif digunakan sebagai salah satu sistem kontrol dan sekuriti pada area parkir.

Kata kunci : deteksi plat nomor mobil, pengenalan plat nomor, PCA, pengenalan pola, OpenCV.

Abstract

The paper describes a vehicle plate recognition system based on computer vision technique. The vehicle image/video was taken from a digital camera and then the vehicle plate automatically identified, segmented and recognized by the system. The feature reduction technique of Principal Components Analysis (PCA) was used in the system. This application was built using Microsoft Visual C++ 6.0®, Microsoft® DirectShow®, Intel® Image Processing Library and Open Source Computer Vision (OpenCV) Library. The application has been implemented and was able to detect vehicle plate position and recognize it using a PC Pentium II/400 MHz. The recognition rate of ±82% was achieved based on recognition of hundreds alphanumeric images (alphanumeric A-Z, 0-9). *This sistem is prospective enough to be used for control and security system in parking area.*

***Keywords** : Vehicle Plate Detection, Car Plate Recognition, PCA, pattern recognition, OpenCV.*

Pendahuluan

Sistem deteksi dan pengenalan plat nomor kendaraan bermotor secara otomatis telah menjadi suatu aplikasi yang sangat penting dalam bidang *computer vision*. Sistem yang dibuat disini bekerja dengan 2 tahap yaitu modul deteksi mobil dan modul pengenalan plat nomor. Mobil yang lewat akan terdeteksi bila ia berada pada kisaran image window tertentu, dan hasil deteksi berupa pengambilan image digital yang selanjutnya diproses untuk dikenali nomornya.

Proses pengenalan karakter pada paper ini dilakukan dengan menggunakan motode ekstraksi feature Principal Components Analysis (PCA). Dalam fase pembelajaran (training), sejumlah image karakter sampel digunakan untuk mencari eigen object (image basis) yang selanjutnya digunakan untuk mentrasformasi setiap image karakter training/database menjadi feature yang lebih sedikit (kompak). Sistem berawal dari pencarian posisi kandidat plat nomor dari sebuah image input, yang kemudian dilanjutkan dengan melakukan segmentasi karakter pada plat nomor yang ditemukan tersebut. Selanjutnya dilakukan ekstraksi feature menggunakan PCA pada image hasil segmentasi tersebut. Proses akhir adalah pengenalan karakter dengan metode klasifikasi sederhana yaitu K-Nearest Neightbor.

Sistem yang telah berhasil diimplementasikan menggunakan platform PC Windows dengan memanfaatkan library OpenCV. Sistem telah diujicobakan pada suatu basis data yang mewakili 26 karakter (0-9, A-Z) yang masing-masing terdiri dari beberapa variasi gambar. Hasil uji coba menunjukkan tingkat keberhasilan

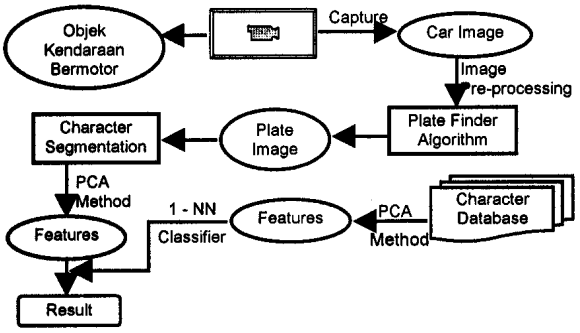
Catatan: Diskusi untuk makalah ini diterima sebelum tanggal 1 Juni 2003. Diskusi yang layak muat akan diterbitkan pada Jurnal Teknik Elektro volume 3, nomor 2, September 2003.

yang cukup menggembirakan, dimana tingkat keberhasilan mencapai ±82%.

Berikutnya paper ini disusun sebagai berikut: pada pasal 2 akan dijelaskan tentang sistem secara keseluruhan. Hasil-hasil percobaan diberikan pada pasal 3 dan ditutup dengan diskusi/kesimpulan pada pasal 4.

Deskripsi Sistem

Secara blok diagram sistem yang dibuat adalah seperti gambar 1.



Gambar 1. Blok Diagram Sistem.

Sistem ini dibagi menjadi menjadi empat bagian modul yaitu v*ideo capturing, database training, plate detection*, dan *character recognition*. Bagian modul *video capturing* terdiri atas proses input kamera atau video dan peroses untuk menangkap (*capture*) objek. Modul *plate detection* terdiri atas *image pre-processing*, *digit location*, *plate area location*, dan *final plate area location*. Modul *character recognition* meliputi *character segmentation* dan *classifier*. Dan untuk modul *database trainning* meliputi perhitungan *eigen object* dan *average object* serta perhitungan transformasi image menjadi *feature* yang selanjutnya akan dimasukkan ke dalam database

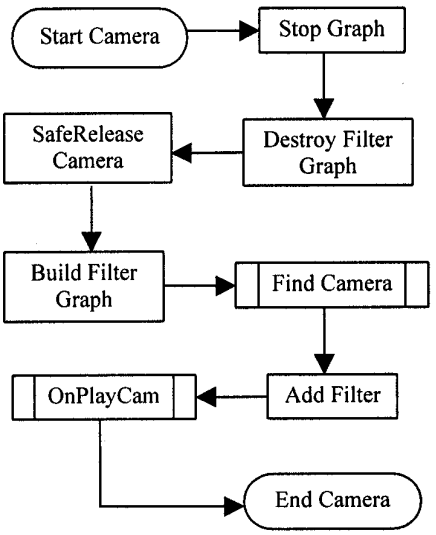
1. Video Capturing – DirectShow Process

Modul ini bertujuan untuk melakukan pengambilan gambar yang berasal dari kamera atau dari file video. Dalam proses ini terdiri dari dua sub program, yang pertama adalah proses untuk menerima *input* baik dari kamera atau dari file video dan yang kedua adalah proses untuk melakukan pengambilan gambar (*grabbing*) dari *input* tersebut. Kedua proses tersebut menggunakan fasilitas *library* yang telah disediakan oleh Microsoft® DirectShow®. Penggunaan *library* ini sangat bermanfaat karena proses *input* dapat berlangsung secara *realtime*.

Untuk proses ini, pertama kali adalah membuat beberapa *variable* dengan tipe data objek yang disediakan oleh *library* directshow, antara lain:

- `IGraphBuilder*` m_CamBuilder1
  Berfungsi sebagai *interface* dari Filter Graph Manager.
- `IMediaControl*` m_CamControl1
  Berfungsi untuk menangani kontrol terhadap *stream* yang sedang berlangsung.
- `IVideoWindow*` m_CamView1
  Berfungsi untuk membangun *interface output video window.*
- `IBaseFilter*` m_Cam1, m_CamTrans1
  Berfungsi untuk pengontrol filter dan pada aplikasi berfungsi untuk menspesifikasikan *pin* dan *query* informasi filter.
- `IFilterGraph*` m_CamGraph1
  Berfungsi untuk membangun filter. Pada aplikasi digunakan untuk menambahkan filter pada graph, menghubungkan dan memutuskan hubungan dengan filter, menghapus filter, dan melakukan beberapa operasi dasar lainnya.

Program aplikasi DirectShow yang dibuat dalam sistem ini, berdasarkan sumber *input*nya (*source filter*) secara garis besar dapat dibagi menjadi dua bagian, yaitu: *input* dari file berupa AVIFile, dan *input* dari kamera. Bagian *input* AVIFile dibangun dengan pendekatan pertama, sedangkan bagian *input* kamera dibangun dengan menggunakan pendekatan ketiga. Kedua bagian ini secara garis besar memiliki prosedur utama yang hampir sama. Berikut diagram alir dari bagian *input* :



Gambar 2. Proses Input Kamera.

2. Database Training – PCA Algorithm

Bagian ini adalah proses untuk melakukan proses *training* PCA pada kumpulan image database. Dimana database ini berupa sekumpulan gambar karakter *alphanumeric* yang terdiri dari beberapa macam variasi untuk masing-masing karakternya (0-9 dan A-Z). Database yang digunakan sebanyak ± 600 gambar. Dalam modul *training* ini dihasilkan output berupa *Eigen Object* dan *Average Object* serta bobot masing masing karakter yang terdapat dalam *database*, hasil bobot ini disimpan ke dalam suatu *list array.*

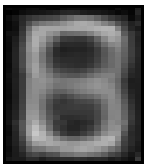
Principal Components Analysis (PCA)[1] digunakan untuk mereduksi dimensi image menjadi feature yang lebih sedikit. Ini dilakukan dengan mentransformasi ruang image kedalam basis atau sistem koordinat yang baru dengan representasi yang lebih kompak. Basis-basis image didapatkan dari sekumpulan karakter-karakter yang dilatihkan (*training alphanumeric*). Feature yang baru tersebut akan dibentuk melalui kombinasi bobot linear dalam ruang eigen (basis image). Komponen dari feature ruang eigen ini tidak saling berkorelasi dan akan memaksimalkan perbedaan yang ada di dalam ruang aslinya.

Maksud dari *Principal Components Analysis* adalah untuk menangkap variasi total di dalam kumpulan karakter yang dilatihkan, dan untuk merepresentasikan variasi ini dengan variabel yang lebih sedikit. Suatu image yang direpresentasikan dengan variabel yang sedikit akan lebih mudah untuk ditangani dan dimengerti daripada jika direpresentasikan dengan raw pixel yang banyak dari image tersebut.

Apabila didefinisikan sebuah objek u = {u_1, u_2, u_3,...,u_n} sebagai vektor pada n dimensi. Objek u dapat berupa suatu gambar dan mempunyai komponen u_1, u_2, u_3,...,u_n, dimana u_1, u_2, u_3,...,u_n adalah nilai pixel dari gambar tersebut. Dengan kondisi ini maka n dapat diartikan sebagai jumlah pixel (=panjang x lebar) yang terdapat dalam gambar. Kemudian, apabila objek tersebut ditambah dengan objek-objek yang lain hingga menjadi sekumpulan atau sekelompok objek maka : $u^i = \{u_1^i, u_2^i, \ldots, u_n^i\}$, dimana $i = 1,...m$ dan $m << n$. Nilai rata-rata atau *mean* objek: $\bar{u} = \{\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_n\}$ dapat didefinisikan sebagai berikut:

$$\bar{u}_i = \frac{1}{m}\sum_{k=1}^{m} u_i^k \qquad (1)$$

Berikut ini adalah contoh gambar average image dari proses perhitungan diatas:



Gambar 3. Average Training Image.

Covariance Matrix c = [c_{ij}] adalah sebuah matrix simetris *m* x *m*, dimana tiap elemennya mempunyai persamaan:

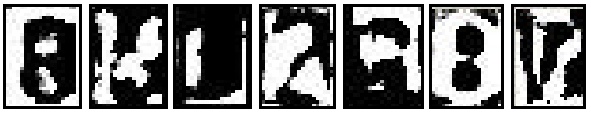
$$c_{ij} = \sum_{l=1}^{n} \left(u_l^i - \bar{u}_l\right) \cdot \left(u_l^j - \bar{u}_l\right) \qquad (2)$$

Image basis atau *Eigen Object* $e^i = \{e_1^i, e_2^i, \ldots, e_n^i\}$, $i = 1,\ldots, m_1 \le m$ (seperti gambar 4) dari sekumpulan image training dapat dihitung dengan persamaan 3.

$$e_l^i = \frac{1}{\sqrt{\lambda_i}} \sum_{k=l}^{m} v_k^i \cdot \left(u_l^k - \bar{u}_l\right) \qquad (3)$$

Dimana λ_i dan $v^i = \{v_1^i, v_2^i, \ldots, v_m^i\}$ adalah *eigen value* dan *eigen vector* dari matrik c yang didapat dengan metode Jacobi.

Gabmar 4 dibawah ini adalah beberapa contoh gambar eigen object hasil dari perhitungan persamaan 3:



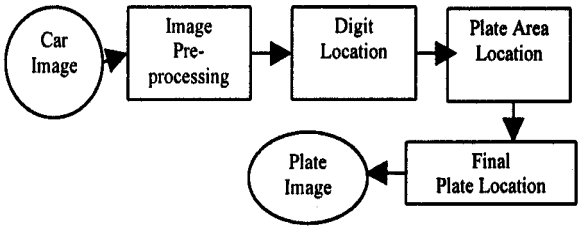
Gambar 4. Tujuh Eigen Objects pertama.

Semua *input* objek u (image training) dan image testing lainnya selanjutnya dikomposisikan di dalam ruang *eigen* m_1 menghasilkan bobot/ feature baru (w) yang dihitung dengan persamaan 4. Bobot w ini yang selanjutnya digunakan untuk pengenalan pola.

$$w_i = \sum_{l=1}^{n} e_l^i \cdot \left(u_l - \bar{u}_l\right) \qquad (4)$$

3. Plate Finder Algorithm

Berikut ini adalah proses diagram alir secara sederhana dari algoritma Plate Finder:



Gambar 5. Diagram Pendeteksian Plat Nomor

Pada bagian ini, sistem akan menggunakan gambar yang dihasilkan pada proses capture dari kamera atau gambar dari input user. Gambar ini kemudian di konversi dari gambar berwarna menjadi gambar grayscale. Selanjutnya dilakukan proses *thresholding* pada gambar *grayscale* tersebut menghasilkan gambar biner (hitam dan putih saja).

Tahap selanjutnya adalah mencari kandidat angka/digit pada gambar biner ini (Digit Location). Semua kemungkinan objek yang berupa *alphanumeric* dari gambar akan dicari pada tahap digit location ini. Algoritma ini akan mencari pada setiap *pixel* yang terdapat pada gambar dan melakukan pengecekan pada setiap iterasi dengan menggunakan sejumlah aturan, dimana apakah terdapat sebuah kandidat *digit* pada posisi pixel yang bersangkutan. Proses ini terdiri dari dua langkah, yaitu:

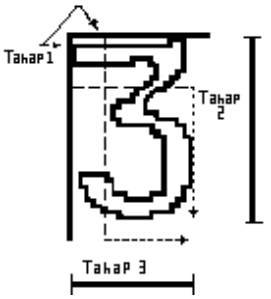
❖ *Adaptive size bounding box searching*

Proses ini dibagi lagi menjadi tiga tahap:

1. Sebuah model yang berbentuk "L" terbalik akan digerakkan pada gambar. Model "L" terbalik ini memiliki tinggi 20 *pixel* dan lebar 12 *pixel.* Nilai ini didapat dari hasil rata-rata setiap gambar kendaraan yang ada. Pada setiap posisi pixel beberapa hal berikut dilakukan:

   (a) Jika tidak terdapat *pixel* (putih) pada posisi garis vertikal atau horisontal dari model "L" yang terbalik, maka tidak ada *digit* yang terdeteksi pada posisi ini.

   (b) Jika terdapat *pixel* (putih) tepat pada posisi atas dan kiri dari model "L", maka berarti ada terdeksi *digit* pada posisi ini. Jika suatu posisi memenuhi kondisi ini maka dapat diartikan bahwa telah ditemukan posisi pojok kiri atas dari suatu *digit*. Dibawah ini adalah visualisasi cara kerja dari model "L" terbalik:



Gambar 6. Deteksi Karakter .

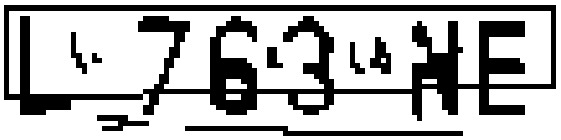
2. Garis vertikal yang kedua digerakkan, untuk tiap pixel demi pixel, ke arah kanan, dari posisi terakhir yang ditemukan pada tahap sebelumnya. Garis vertikal ini digerakkan hingga tidak ditemukan satu *pixel* putih pun sepanjang garis vertikal ini. Dalam hal ini panjang dari kandidat *digit* yang ditemukan dihitung, dan apabila panjang dari kandidat *digit* ini melebihi batas rata-rata dari suatu *digit*, maka posisi akan didiskualifikasi.
3. Pada tahap ini, garis horisontal yang kedua dengan panjang yang telah ditemukan pada tahap sebelumnya, digerakkan ke bawah hingga tidak ditemukan satu *pixel* putih pun sepanjang garis horisontal ini. Jika tinggi dari kandidat *digit* ini memenuhi kondisi tinggi rata-rata suatu *digit*, maka proses dapat melanjutkan pada tahap selanjutnya, jika tidak maka proses akan kembali pada tahap yang pertama.

❖ *Pixel Coverage Checking*

Setelah posisi suatu kandidat *digit* ditemukan maka, nilai semua *pixel* putih yang terdapat pada posisi ini akan dihitung semua. Jika nilai *pixel* memenuhi kondisi maka posisi ini akan disimpan sebagai kandidat karakter, jika tidak memenuhi akan didiskualifikasi. Tujuan dari proses ini adalah untuk menghilangkan suatu kandidat yang terlalu besar atau terlalu kecil.

Setelah semua kandidat digit ditemukan maka selanjutnya adalah mencari lokasi kandidat plat nomor kendaraan (Plate Area Location). Logika pada bagian ini adalah menggunakan posisi objek karakter yang ditemukan pada bagian sebelumnya untuk menentukan letak plat nomor. Langkah pertama adalah membangun suatu model yang berbentuk persegi panjang (*plate bounding box*). Panjang dan lebar model ini telah diatur berdasarkan rata-rata dari image kendaraan yang ada (lebar 120 *pixel* dan tinggi 33 *pixel*). Model ini akan digerakkan mulai dari

posisi kandidat *digit* pertama yang ditemukan pada tahap sebelumnya hingga pada posisi terakhir kandidat *digit* yang ditemukan. Berikut ini adalah visualisasi dari model persegi panjang yang digunakan untuk mencari lokasi plat.



Gambar 7. Lokalisasi Karakter.

Pada setiap iterasi akan dilakukan pengecekan apakah pada daerah sekitar model persegi panjang ini terdapat berapa banyak kandidat *digit*. Kandidat *digit* yang ada harus terdiri dari minimal tiga atau maksimal delapan digit. Hal ini diasumsikan bahwa plat nomor standart yang ada di negara kita memiliki kondisi yang demikian. Selain dilakukan pengecekan banyaknya kandidat *digit*, juga dilakukan pengecekan terhadap posisi dari masing masing kandidat *digit* tersebut. Hal ini dilakukan sebagai pencegahan terhadap pengambilan kandidat digit yang salah. Apabila posisi teratas antara kandidat *digit* yang satu dengan yang lain terpaut sangat jauh, maka posisi kandidat ini akan didiskualifikasi. Posisi model persegi panjang yang dapat diterima adalah posisi dimana terdapat kandidat *digit* yang lebih dari dua dan kurang dari sembilan, serta memiliki posisi yang tingginya tidak terpaut jauh antara satu dengan yang lainnya. Posisi yang benar akan disimpan dalam suatu *list array*.

Setelah semua kandidat plat nomor ditemukan, maka selanjutnya adalah mencari lokasi sebenarnya dari kandidat-kandidat plat nomor yang ditemukan pada tahap sebelumnya (Final Plate Location). Pada tahap ini akan melakukan iterasi terhadap seluruh kandidat plat nomor yang ditemukan pada tahap sebelumnya. Umumnya setelah memasuki tahap terakhir ini kandidat plat nomor yang ada sering menghasilkan satu kandidat saja. Dan apabila kandidat plat nomor yang dihasilkan ternyata lebih dari satu, maka plat nomor yang dipilih adalah kandidat yang memiliki posisi paling bawah pada image. Pada lokasi yang ditemukan tersebut kemudian dilakukan proses *cropping*. Hasil dari *cropping* tersebut disimpan ke dalam sebuah file yang digunakan lebih lanjut untuk proses pengenalan.

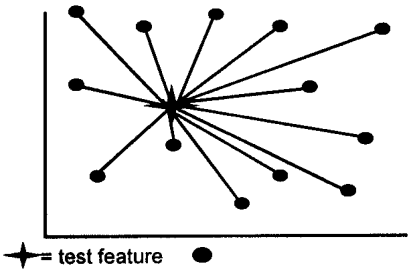
4. Character Recognition: dengan Nearest Neighbour Classifier

Pada tahap akhir ini, image plat nomor yang ditemukan akan dilakukan segmentasi karakter terlebih dahulu. Proses segmentasi karakter dilakukan untuk mengekstraksi masing masing karakter yang terdapat pada plat nomor. Pada tahap ini algoritma yang dilakukan adalah menggunakan metode seperti pada modul deteksi letak plat nomor tetapi dengan melakukan perubahan pada algoritmanya. Perubahan yang dilakukan adalah apabila model "L" terbalik telah menemukan kandidat *digit*, maka pada setiap posisi tersebut dilakukan proses *cropping* dan dilakukan proses *resize* pada kandidat digit ini. Proses *resize* ini bertujuan agar semua karakter yang akan dikenali memiliki ukuran yang sama. *Resize* dilakukan dengan ukuran 20 *pixel* untuk lebar dan 30 *pixel* untuk tingginya (disesuaikan dengan data image pada *database*). Hasil *cropping* tersebut disimpan kedalam *list*. Pada bagian ini apabila terdapat kandidat *digit* yang saling berhubungan kemungkinan akan dapat terektraksi bersama. Setelah proses *cropping* selesai maka dilanjutkan pada proses selanjutnya, yaitu pengenalan karakter menggunakan N*earest Neighbour Classifier* .

Pada N*earest Neighbour Classifier* perhitungan yang dilakukan adalah dengan mengkalkulasi semua bobot nilai yang ada pada sekumpulan image *training* $W_d = (w_d^1, \ldots . w_d^m)$ dengan menggunakan metode PCA. Perhitungan juga dilakukan pada karakter yang akan di tes $W_t = (w_t^1, \ldots . , w_t^m)$. Jarak antara bobot image *training* dengan tes kemudian dihitung dengan menggunakan persamaan *Euclidean Distance* (∂) pada m-dimensi yaitu:

$$\partial = \sqrt{\sum_{i=1}^{m} \left(w_d^i - w_t^i \right)^2} \qquad (5)$$

Dimana m adalah banyaknya jumlah feature bobot (principal components). Dari hasil ini kemudian dicari jarak paling kecil diantara set jarak tersebut. Hasil ini dicari pada *database* yang mempunyai nilai pada posisi yang bersangkutan. karakter pada posisi yang memiliki jarak minimum adalah hasil dari pengenalan tersebut.



Gambar 8. Pencarian Jarak Minimum.

5. Open Source Computer Vision (OpenCV)

Implementasi sistem yang dikerjakan disini menggunakan platform Windows dengan bahasa pemrograman Visual C++. Di sini digunakan library perangkat lunak *Open Source Computer Vision (OpenCV).* OpenCV mulai dikembangkan sejak tahun 1999 oleh *Visual Interactivity Group* didalam *Intel's Microprocessor Research Lab* [4]. Proyek ini dibuat dengan tujuan untuk mendirikan sebuah komunitas *open source vision* dan menyediakan sebuah situs dimana usaha terdistribusi dari komunitas dapat dikonsolidasikan dan *performance*-nya dapat dioptimalkan. *Library* ini ditujukan untuk digunakan oleh peneliti dan pengembang software komersial. *Open Source Computer Vision Library* dibuat berdasarkan fungsi-fungsi dasar yang terdapat pada *Intel Performance Library*. Keunggulan *library* ini adalah semua fungsi-fungsinya telah dioptimasi untuk prossesor Intel sehingga dapat berjalan jauh lebih cepat.

Open Source Computer Vision Library Committee terdiri dari beberapa orang antara lain Dr. Gary Bradski, Prof. Trevor Darrell, Prof. Irfan Essa, Prof. Jitendra Malik, Prof. Pietro Perona, Prof. Stan Sclaroff, dan Prof. Carlo Tomasi [4]. Berikut adalah beberapa area dari fungsi umum yang dapat didukung oleh *Open Source Computer Vision Library Committee* :

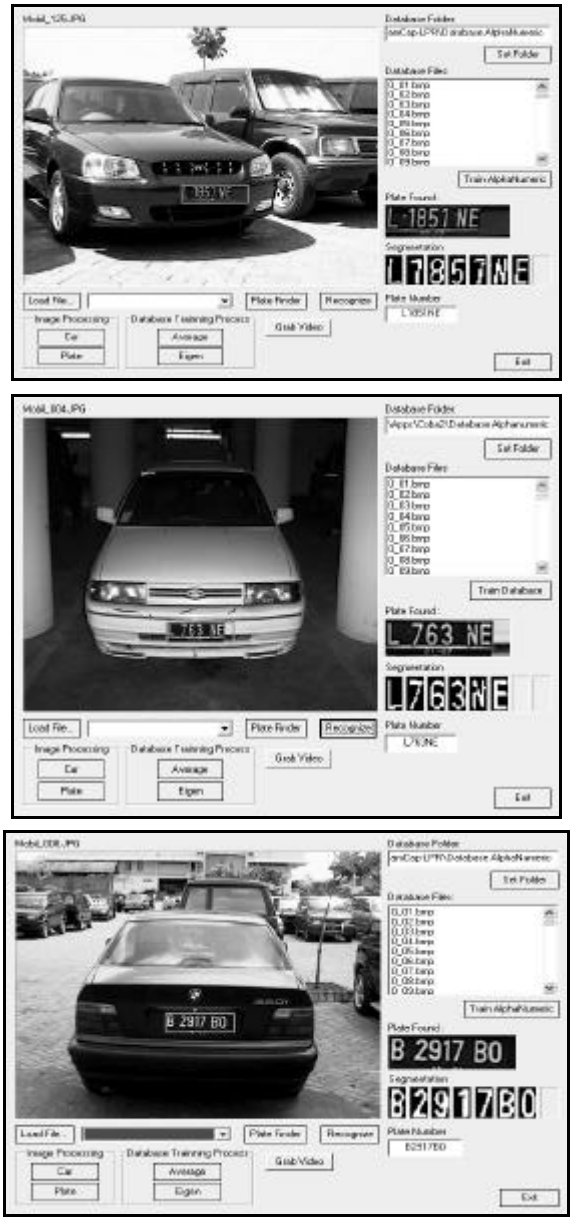
- *Geometric Methods*
- *Recognition*
- *Measures*
- *Segmentation*
- *Utilities*
- *Features*
- *Image Pyramids*
- *Camera*
- *Tracking*
- *Fitting*
- *Matrix*
- *Image Processing*

Open Source Computer Vision Library dibuat berdasarkan fungsi-fungsi dasar dan library dari Intel® *Image Processing Library.* Intel® *Image Processing Library* menyediakan sekumpulan fungsi-fungsi C sangat teroptimasi yang meng-implementasikan fungsi-fungsi *image processing* pada prosessor berarsitektur Intel.

Hasil-Hasil Percobaan

Pengujian sistem dilakukan menggunakan image diam maupun file video dengan resolusi 640x480 pixel. Image diam terdiri dari 257 buah memuat beberapa tipe kendaraan. Pengambilan image posisi kendaraan pun bervariasi (terlihat dari arah depan, belakang, miring kiri, miring kanan serta variasi jarak)

Gambar 9 adalah contoh tampilan program pengenalan plat nomor yang dibuat.



Gambar 9. Contoh Tampilan Program.

Pada proses pencarian letak plat nomor (*Plate Finder*), hasil yang didapat adalah sebagai berikut:

Tabel 1. Hasil Proses Pencarian Plat Nomor.

Posisi Kendaraan	Jml. Kendaraan	Hasil Proses Plate Finder			
		Berhasil			Tidak Berhasil
		Tepat	Kurang	Salah	
Total (Depan/Belakang/Miring)	257	178 (69,26%)	24 (9,33%)	4 (1,55%)	51 (19,84%)
Depan	175	127 (72,57%)	15 (8,57%)	2 (1,14%)	31 (17,71%)
Belakang	82	51 (62,19%)	9 (10,97%)	2 (2,43%)	20 (24,39%)
Miring (Depan/Belakang)	65	40 (61,53%)	7 (10,76%)	2 (3,07%)	16 (24,61%)

Pengujian selanjutnya adalah dengan melakukan proses segmentasi terhadap 202 plat yang ditemukan tersebut. Hasilnya adalah sebagai berikut:

Tabel 2. Hasil Keberhasilan Segmentasi.

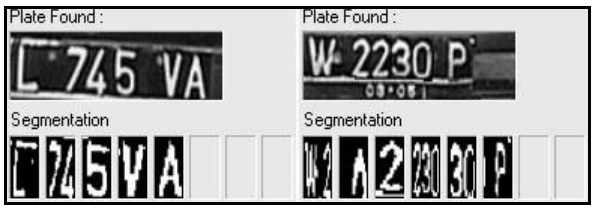
Jumlah Plat	Tepat	Kurang
202	119 (58,91%)	83 (41,09%)

Dari 202 plat nomor yang ditemukan, karakter yang ada pada plat nomor tersebut adalah sebanyak 1224 karakter. Pengujian selanjutnya adalah menguji kondisi (terbaca atau tidak) 1224 karakter yang disegmentasi dari proses sebelumnya. Hasilnya adalah sebagai berikut:

Tabel 3. Hasil Pembacaan Segmentasi.

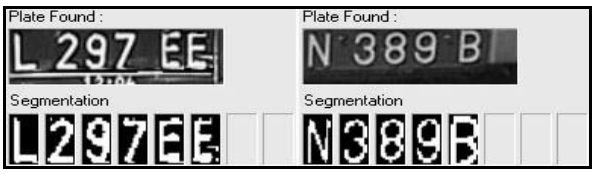
Jumlah Karakter	Terbaca	Tidak Terbaca
1224	1051 (85,87%)	173 (14,13%)

Berikut ini adalah contoh dari hasil proses segmentasi karakter yang kurang sempurna:



Gambar 10. Contoh Hasil Sementasi Yang Gagal.

Sedangkan berikut ini adalah contoh dari proses segmentasi yang sempurna:



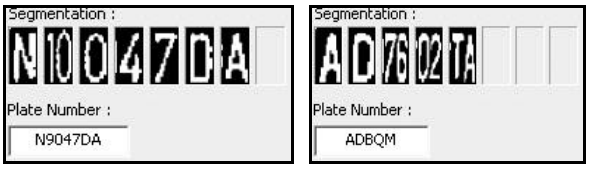
Gambar 11. Contoh Hasil Sementasi Yang Sempurna.

Pengujian selanjutnya adalah menguji dari 1051 karakter yang terbaca apakah dapat dikenali sistem dengan benar. Hasilnya adalah sebagai berikut:

Tabel 4. Hasil Pengenalan Karakter.

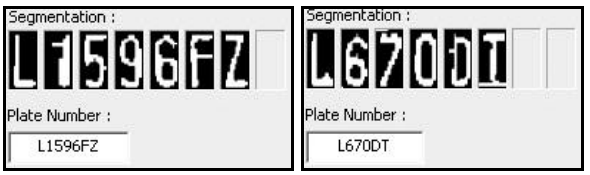
Jumlah Karakter	Benar	Salah
1051	995 (94,67%)	56 (5,37%)

Berikut adalah contoh proses pengenalan yang kurang sempurna:



Gambar 12. Contoh Pengenalan Yang Gagal.

Sedangkan dibwah ini adalah contoh proses pengenalan yang sempurna:



Gambar 13. Contoh Pengenalan Yang Sempurna.

Secara keseluruhan dari hasil pengujian menggunakan file gambar, *performance* dari sistem ini memberikan beberapa hasil sebagai berikut:

Tabel 5. Hasil Keseluruhan Sistem.

Keterangan	Tingkat Keberhasilan
Sistem mendeteksi letak plat dengan sempurna dan mensegmentasi serta mengenalinya dengan benar (dari 257 kendaraan)	42,80%
Pendeteksian letak plat nomor secara sempurna (dari 257 kendaraan)	69,26%
Proses segmentasi plat nomor kendaraan secara sempurna (dari 202 plat)	58,91%
Proses segmentasi karakter secara sempurna (dari 1224 karakter)	85,86%
Proses pengenalan karakter dengan benar (dari 1051 karakter)	94,67%

Disini tampak *Nearest Neighbour Classifier* dengan bantuan pengolahan feature menggunakan metode PCA, memiliki keakuratan pengenalan karakter sebesar 94,67% dari 1051 karakter yang dicoba untuk dikenali.

Kesimpulan

1. Sistem ini secara keseluruhan dapat mendeteksi dan mengenali plat nomor kendaraan dengan akurasi 42,80% (110 kendaraan) dengan jumlah *database* sebanyak 609 karakter untuk pengujian sebanyak 257 kendaraan (tanpa memperhatikan kondisi kendaraan).
2. Pendeteksian letak plat nomor kendaraan secara sempurna menghasilkan akurasi sebesar 69,26% (178 kendaraan) dari 257 kendaraan yang diujikan.
3. Algoritma pendeteksian letak plat nomor kendaraan dan segmentasi plat nomor ini sangat dipengaruhi sekali oleh beberapa faktor berikut :
   - Posisi kendaraan (miring, terlihat atas/ bawah, depan/belakang)
   - Variasi cahaya sekitar
   - Kondisi plat nomor kendaraan (cat yang tidak jelas/kusam, karakter yang saling terhubung, rusak, dan sebagainya)

   Dan apabila pada proses segmentasi terdapat karakter yang saling berhubungan maka akan dapat terekstraksi bersama. Hal ini tentu saja dapat menyebabkan proses pengenalan yang salah.
4. Metode PCA ini terbukti cukup handal digunakan sebagai metode ekstraksi feature. Jumlah variasi image training yang banyak akan membuat sistem pengenalan menjadi lebih baik.

Daftar Pustaka

[1] C. Coetzee, C. Botha and D. Weber, PC Based Number Plate Recognition System.

[2] J.Chang and N.C. Griswold, "*A Hierarchical Multilayer Perceptron Neural Network for The Recognition of the Automobile License Plate Number*", Proc. SPIE, vol. 2664, pp. 138-144, 1996.

[3] Jesús Molina, Jose M. Mossi and Antonio Albiol, Development Of A Plate Reader For A Surveillance System.

[4] Intel® Image Processing Library Reference Manual. , Intel Corporation, 2000.

[5] Open SourceComputer Vision Library Reference Manual, Intel Corporation, 2001.

[6] S. Draghici, A Neural Network Based Artificial Vision System for License Plate Recognition.