PENGECAMAN KARAKTER DIGITAL PADA PLAT NOMOR KENDARAAN DALAM PERPARKIRAN

Dwiki Jatikusumo, Hanny Hikmayanti[1)], Setyawan Widyarto[2)]
[1)] Universitas Budi Luhur Jakarta
Jl. Ciledug Raya Jakarta Selatan 12260 DKI Jakarta Telp (021) 585 3753
[1)] e-mail : dwikijk@gmail.com, hanny.hikmayanti@yahoo.com
[2)] Faculty of Computer Science and Information Technology, Universiti Selangor, Bestari Jaya Campus, Jalan Timur Tambahan, 45600 Bestari Jaya, Selangor Darul Ehsan, Malaysia, Tel:603-32805121 Faks:603-32806015, e-mail: swidyarto@unisel.edu.my

Abstrak
Era saat ini sudah menjadi penerapan teknologi informasi sebagai mempermudah pekerjaan manusia. Dalam hal ini salah satunya adalah sistem perpakiran yang ada pada saat ini sudah mempergunakan teknologi informasi dalam penerapannya, tetapi permasalahan dalam sistem perpakiran masih mempergunakan manusia untuk pengoperasiannya untuk memasukkan data angka dan huruf plat nomor kendaraan ke database yang digunakan, serta dalam pengecekkan nomor kendaraan plat nomornya. Oleh karena itu, dalam pengembangan sistem perpakiran ditambah dengan digital character recognition dapat mengatasi hal tersebut. Pada penelitian ini, bertujuan mempermudah dalam mengisi data angka dan huruf dengan menggunakan digital character recognition pada gambar yang diperoleh dari kamera digital yang mempotret plat nomor kendaraan menjadi hasil citra digital dikonversikan menjadi data karakter angka dan huruf. Dalam pengembangannya menggunakan Delphi sebagai bahasa pemrogramannya.

***Kata Kunci** : Sistem perpakiran, Digital character recognition, Citra digital, Delphi*

1. PENDAHULUAN

Proses pengambilan data plat nomor kendaraan dalam sistem perpakiran saat ini masih menggunakan manusia untuk melakukan hal tersebut. Penggunaan sistem perparkiran yang dikombinasikan dengan *digital character recogniton* dapat mempermudah dalam mengisi data pada plat nomor kendaraan.
Digital character recogniton merupakan salah satu teknik dalam melakukan proses pengolahan citra. Pengolahan citra merupakan suatu metode atau teknik yang dapat digunakan untuk memproses citra atau gambar dengan memanipulasi gambar tersebut untuk menjadi sebuah data yang diinginkan untuk mendapatkan sebuah informasi. Pengolahan citra dapat dimanfaatkan untuk berbagai keperluan salah satunya adalah sistem perparkiran.
Dalam mendapatkan informasi data kendaraan berupa nomor plat kendaraan, dibutuhkan beberapa proses yang dilakukan. Proses-proses tersebut antara lain: pengambilan gambar kendaraan, penentuan posisi dari plat nomor kendaraan, pengecaman karakter yang terdapat pada plat nomor kendaraan. Proses mengubah data gambar menjadi data karakter diharapkan dapat mempermudah dan mempercepat proses parkir kendaraan bermotor dengan otomatisasi sistem pengelolaan parkir, yang bisa mengefisiensikan pemakaian sumber daya manusia.

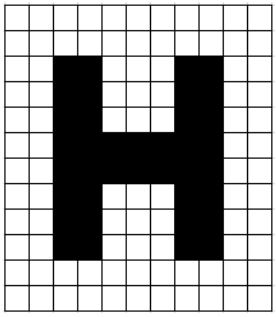
2. TINJAUAN PUSTAKA

A. Pengolahan citra digital
Citra digital merupakan perubahan citra kontinu kedalam bentuk diskrit, baik koordinat maupun intensitas cahayanya. (Pratt, 2007)
Proses yang dapat dilakukan dalam pengolahan citra digital adalah:

a. Konversi dari citra berwarna ke dalam cita abu-abu.
b. Perubahan dari citra berwarna atau citra abu-abu ke dalam citra biner. Proses ini bisa dilakukan dengan deteksi tepi atau filtering.

Operasi pengolahan citra digital umumnya dilakukan dengan tujuan memperbaiki kualitas suatu gambar sehingga dapat dengan mudah diinterpretasi oleh mata manusia dan untuk mengolah informasi yang terdapat pada suatu gambar untuk keperluan pengenalan objek secara otomatis.
Berdasarkan warna-warna penyusunnya, citra digital dapat dibagi menjadi tiga macam (Munir, 2005) yaitu:

a. Citra biner, yaitu citra yang hanya terdiri atas dua warna, yaitu hitam dan putih. Oleh karena itu, setiap pixel pada citra biner cukup direpresentasikan dengan 1 bit.



Gambar 1. Citra biner

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	1	0	0
0	0	1	1	0	0	0	1	1	0	0
0	0	1	1	0	0	0	1	1	0	0
0	0	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	0	0
0	0	1	1	0	0	0	1	1	0	0
0	0	1	1	0	0	0	1	1	0	0
0	0	1	1	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Gambar 2. Representasi citra biner

Alasan penggunaan citra biner adalah karena citra biner memiliki sejumlah keuntungan sebagai berikut:
a. Kebutuhan memori kecil karena nilai derajat keabuan hanya membutuhkan representasi 1 bit.
b. Waktu pemrosesan lebih cepat di bandingkan dengan citra hitam putih ataupun warna.

b. Citra grayscale, yaitu citra yang nilai pixel-nya merepresentasikan derajat keabuan atau intensitas warna putih. Citra grayscale merupakan citra satu kanal, dimana citra f(x,y) merupakan fungsi tingkat keabuan dari hitam keputih, x menyatakan variable kolom atau posisi pixel di garis jelajah dan y menyatakan variable kolom atau posisi pixel di garis jelajah. Intensitas f dari gambar hitam putih pada titik (x,y) disebut derajat keabuan (grey level), yang dalam hal ini derajat keabuannya bergerak dari hitam keputih. Derajat keabuan memiliki rentang nilai dari Imin sampai Imax, atau Imin < f < Imax, selang (Imin, Imax) disebut skala keabuan.
Biasanya selang (Imin, Imax) sering digeser untuk alasan-alasan praktis menjadi selang [0,L], yang dalam hal ini nilai intensitas 0 menyatakan hitam, nilai intensitas L meyatakan putih, sedangkan nilai intensitas antara 0 sampai L bergeser dari hitam ke putih. Sebagai contoh citra grayscale dengan 256 level artinya mempunyai skala abu dari 0 sampai 255 atau [0,255], yang dalam hal ini intensitas 0 menyatakan hitam, intensitas 255 menyatakan putih, dan nilai antara 0 sampai 255 menyatakan warna keabuan yang terletak antara hitam dan putih.

c. Citra berwarna, yaitu citra yang nilai pixel-nya merepresentasikan warna tertentu Banyaknya warna yang mungkin digunakan bergantung kepada kedalaman pixel citra yang bersangkutan. Citra berwarna direpresentasikan dalam beberapa kanal (channel) yang menyatakan komponen-komponen warna penyusunnya. Banyaknya kanal yang digunakan bergantung pada model warna yang digunakan pada citra tersebut.
Setiap piksel pada citra warna mewakili warna yang merupakan kombinasi dari tiga warna dasar (RGB = Red Green Blue). Setiap warna dasar menggunakan penyimpanan 8 bit =1 byte, yang berarti setiap warna mempunyai gradiasi sebanyak 255 warna. Berarti setiap piksel mempunyai kombinasi warna sebanyak 28 . 28 . 28 = 216=16 juta warna lebih. itulah sebabnya format ini dinamakan true color karena mempunyai jumlah warna yang cukup besar sehingga bias dikatakan hampir mencangkup semua warna di alam. (Sutoyoso, 2009)
Penyimpanan citra true color di dalam memori berbeda dengan citra grayscale. Setiap piksel dari citra grayscale 256 gradasi warna diwakili oleh 1 byte. Sedangkan 1 piksel citra true color diwakili oleh 3 byte, dimana masing-masing byte mempresentasikan warna merah (Red), hijau (Green), dan Biru (Blue). (Sutoyoso, 2009)

B. Format BMP

Format gambar paling sederhana adalah BMP (Bitmap). Bitmap adalah format gambar asli di sistem operasi Microsoft Windows. Bitmap mendukung gambar dengan 1, 4, 8, 16, 24, dan 32 bit per pixel, meskipun file Bitmap menggunakan 16 dan 32 bit per pixel hal ini jarang terjadi. Bitmap juga mendukung kompresi untuk 4 dan 8 bit per pixel. Namun, Bitmap kompresi adalah penggunaan hanya dengan blok besar dengan warna identik, sehingga nilai yang sangat terbatas. Sangat jarang untuk bitmap berada dalam format terkompresi. (Miano, 1999) Kriteria yang paling penting dari citra ini adalah kedalaman warna yaitu berapa banyak bit per pixel yang didefinisikan dari sebuah warna (Munir, 2005). Bitmap dengan mengikuti kriteria tadi maka dapat dilihat: a. 8 bit = 256 warna (256 gray scales). b. 24 bit = 16.777.216 warna Secara umum dapat dikatakan semakin banyaknya warna, maka akan diperlukan keamanan yang ketat atau tinggi dikarenakan bitmap memiliki area yang sangat luas dalam sebuah warna yang seharusnya dihindarkan.

Dilihat dari kedalaman atau kejelasan dari sebuah warna, bitmap dapat mengambil sejumlah data tersembunyi dengan perbandingan sebagai berikut (ukuran ratio dari bitmap dalam byte = ukuran dari data yang disembunyikan) :

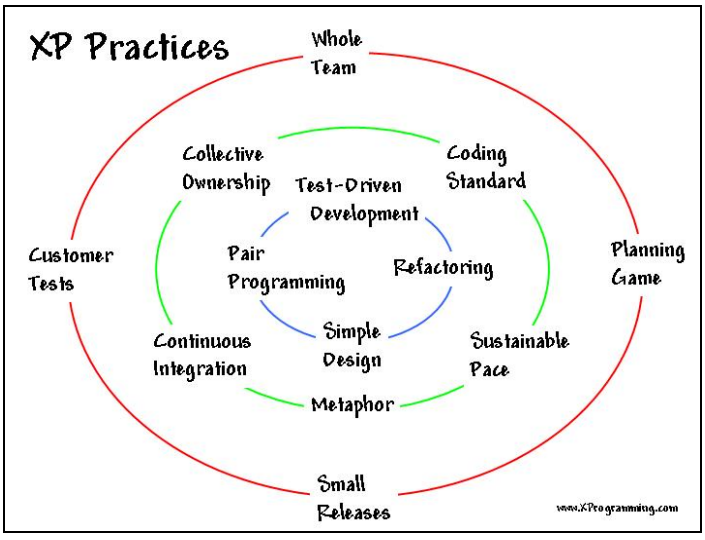
1. 8 bit = 256 warna : 8 : 1
2. 24 bit = 16.777.216 warna : 8 : 1

Perbandingan tersebut diperoleh dari penentuan LSB dalam suatu byte, untuk citra 8 bit letak LSB adalah pada bit terakhir sedangkan untuk citra 24 bit letak LSB adalah pada bit ke-8, bit ke-16 dan bit ke 24 dimana masing-masing byte mewakili warna merah (red), warna hijau (green) dan warna biru (blue). Manipulasi pada bitmap tidak dapat dikonvert atau diubah ke dalam bentuk format grafik yang lain, karena data tersembunyi dalam file tersebut akan hilang. Format menggunakan metode komperesi yang lain (seperti JPEG) tidak di gunakan dalam skripsi ini. Mengurangi ukuran dari carrier file sangatlah penting untuk melakukan transmisi online, yaitu dengan menggunakan utilitas kompresi (seperti : ARZ, LZH, PKZIP, WinZip), dikarenakan kerja mereka tidak terlalu berat. (Sutoyoso, 2009)

3. METODE PENELITIAN

Metode yang digunakan dalam penilitian ini adalah *Extreme Programming*. *Extreme Programming* atau XP adalah sebuah pendekatan atau model pengembangan perangkat lunak yang mencoba menyederhanakan berbagai tahapan dalam proses pengembangan tersebut sehingga menjadi lebih adaptif dan fleksibel. Walaupun menggunakan kata *programming*, XP bukan hanya berfokus pada *coding* tetapi meliputi seluruh area pengembangan perangkat lunak.(Widhiartha, 2008)

Aspek dasar XP terdiri dari berbagai teknik atau metode yang diterapkan (Beck, 1999) pada *C3 Project*. Teknik-teknik tersebut dapat dijelaskan pada gambar berikut ini:



Gambar 3. Aspek dasar *Extreme Programming*

1. *The Planning Game*
   Pendekatan XP dalam perencanaan sangat mirip dengan metode yang diterapkan pada *RAD (Rapid Application Development)*. Proses pendek dan cepat, mengutamakan aspek teknik, memisahkan unsur bisnis dengan unsur teknis dan pertemuan intensif antara klien dengan *developer*. Pada XP proses ini menggunakan terminologi "*game*" karena Beck menyarankan untuk menggunakan teknik *score card* dalam

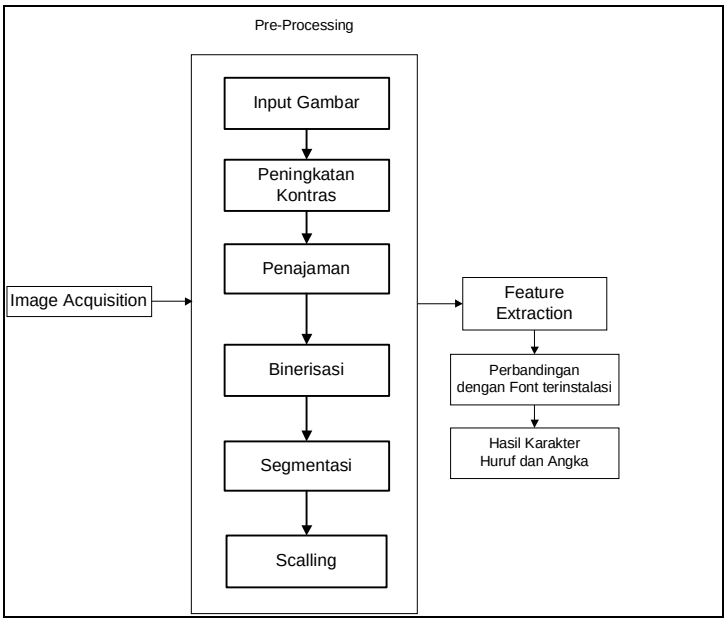
menentukan *requirements*. Semakin sulit aspek teknis yang dibutuhkan semakin tinggi pula skor pada kartu rencana tersebut.

2. *Small Releases*
   Setiap *release* dilakukan dalam lingkup sekecil mungkin pada XP. Setiap *developer* menyelesaikan sebuah unit atau bagian dari perangkat lunak maka hasil tersebut harus segera dipresentasikan dan didiskusikan dengan klien. Jika memungkinkan untuk menerapkan unit tersebut pada perusahaan, hal itu juga dapat dilakukan sekaligus sebagai tes awal dari penerapan keseluruhan sistem. Kendati demikian hal ini tidak selalu perlu dilakukan karena harus dihitung terlebih dahulu sumberdaya yang dibutuhkan. Apakah lebih menguntungkan langsung melakukan tes terhadap unit tersebut atau melakukan tes setelah unit tersebut terintegrasi secara sempurna pada sistem.
3. *Metaphor*
   Metaphor pada dasarnya sama dengan arsitektur perangkat lunak. Keduanya menggambarkan visi yang luas terhadap tujuan dari pengembangan perangkat lunak. Beck sendiri seperti para penandatangan Agile Manifesto lainnya bercita-cita menyederhanakan proses pengembangan perangkat lunak yang saat ini sudah dianggap terlalu rumit. Arsitektur yang saat ini banyak berisi diagram dan kode semacam UML dianggap terlalu rumit untuk dimengerti, terutama oleh klien. *Metaphor*, walaupun mirip dengan arsitektur lebih bersifat naratif dan deskriptif. Dengan demikian diharapkan komunikasi antara klien dengan *developer* akan berlangsung lebih baik dan lancar dengan penggunaan *metaphor*.
4. *Simple Design*
   Sebagai salah seorang penandatangan Agile Manifesto, Beck adalah seorang yang tidak menyukai desain yang rumit dalam sebuah pengembangan perangkat lunak. Tidak heran jika dia memasukkan *Simple Design* sebagai salah satu unsur XP. Pada XP desain dibuat dalam lingkup kecil dan sederhana. Tidak perlu melakukan antisipasi terhadap berbagai perubahan di kemudian hari. Dengan desain yang simpel apabila terjadi perubahan maka membuat desain baru untuk mengatasi perubahan tersebut dapat dengan mudah dilakukan dan resiko kegagalan desain dapat diperkecil.
5. *Refactoring*
   Refactoring adalah salah satu aspek paling khas dari XP. *Refactoring* seperti didefinisikan oleh Martin Fowler adalah "Melakukan perubahan pada kode program dari perangkat lunak dengan tujuan meningkatkan kualitas dari struktur program tersebut tanpa mengubah cara program tersebut bekerja". *Refactoring* sendiri sangat sesuai untuk menjadi bagian XP karena *Refactoring* mengusung konsep penyederhanaan dari proses desain maupun struktur baris kode program. Dengan *Refactoring* tim pengembang dapat melakukan berbagai usaha untuk meningkatkan kualitas program tanpa kembali mengulang-ulang proses desain. Fowler adalah salah satu kolega dekat dari Kent Beck karena itu tidak mengherankan bahwa cara berpikir mereka terhadap proses pengembangan perangkat lunak sangat mirip satu dengan lainnya.
6. *Testing*
   XP menganut paradigma berbeda dalam hal tes dengan model pengembangan perangkat lunak lainnya. Jika pada pengembangan perangkat lunak lainnya tes baru dikembangkan setelah perangkat lunak selesai menjalani proses *coding* maka pada XP tim pengembang harus membuat terlebih dahulu tes yang hendak dijalani oleh perangkat lunak. Berbagai model tes yang mengantisipasi penerapan perangkat lunak pada sistem dikembangkan terlebih dahulu. Saat proses *coding* selesai dilakukan maka perangkat lunak diuji dengan model tes yang telah dibuat tersebut. Pengetesan akan jauh lebih baik apabila dilakukan pada setiap unit perangkat lunak dalam lingkup sekecil mungkin daripada menunggu sampai seluruh perangkat lunak selesai dibuat. Dengan memahami tahap ini kita dapat melihat bahwa siklus pada XP adalah *requirement analysis, test, code,* dan *design*. Sekilas terlihat hal ini tidak mungkin dilakukan tetapi pada kenyataannya memang gambaran inilah yang paling dapat menjelaskan tentang XP.
7. *Pair Programming*
   Pair programming adalah melakukan proses menulis program dengan berpasangan. Dua orang programer saling bekerjasama di komputer yang sama untuk menyelesaikan sebuah unit. Dengan melakukan ini maka keduanya selalu dapat berdiskusi dan saling melakukan koreksi apabila ada kesalahan dalam penulisan program. Aspek ini mungkin akan sulit dijalankan oleh para programer yang memiliki ego tinggi dan sering tidak nyaman untuk berbagi komputer bersama rekannnya.
8. *Collective Ownership*
   Tidak ada satupun baris kode program yang hanya dipahami oleh satu orang programer. XP menuntut para programer untuk berbagi pengetahuan untuk tiap baris program bahkan beserta hak untuk mengubahnya. Dengan pemahaman yang sama terhadap keseluruhan program, ketergantungan pada programer tertentu ataupun berbagai hambatan akibat perbedaan gaya menulis program dapat diperkecil. Pada level yang lebih tinggi bahkan dimungkinkan para programer dapat bertukar unit yang dibangunnya.

9. *Coding Standards*
   Pair programming dan *collective ownership* hanya akan dapat berjalan dengan baik apabila para programer memiliki pemahaman yang sama terhadap penulisan kode program. Dengan adanya *coding standards* yang telah disepakati terlebih dahulu maka pemahaman terhadap program akan menjadi mudah untuk semua programer dalam tim. Hal ini dapat diterapkan sebagai contoh pada penamaan variabel dan penggunaan tipe data yang sama untuk tiap elemen semua *record* atau *array* pada program.
10. *Continous Integration*
    Melakukan *build* setiap hari kerja menjadi sebuah model yang disukai oleh berbagai tim pengembang perangkat lunak. Hal ini terutama didorong oleh keberhasilan penerapan sistem ini oleh Microsoft dan telah sering dipublikasikan. Dengan melakukan *build* sesering mungkin berbagai kesalahan pada program dapat dideteksi dan diperbaiki secepat mungkin. Apabila banyak tim pengembang perangkat lunak meyakini bahwa *build* sekali sehari adalah minimum maka pada XP hal tersebut adalah maksimum. Pada XP tim disarankan untuk melakukan *build* sesering mungkin misalnya setiap 4 jam atau bahkan lebih cepat lagi.
11. *40-hours Week*
    Beck berpendapat bekerja 8 jam sehari dan 5 hari seminggu adalah maksimal untuk tiap programer. Lebih dari itu programer akan cenderung membuat berbagai *error* pada baris-baris kode programnya karena kelelahan.
12. *On-Site Customer*
    Sebuah pendekatan klasik, di mana XP menganjurkan bahwa ada anggota dari klien yang terlibat pada proses pengembangan perangkat lunak. Yang lebih penting lagi ia harus ada di tempat pemrogaman dan turut serta dalam proses build dan test yang dilakukan. Apabila ada kesalahan dalam pengembangan diharapkan klien dapat segera memberikan masukan untuk koreksinya.

4. HASIL DAN PEMBAHASAN

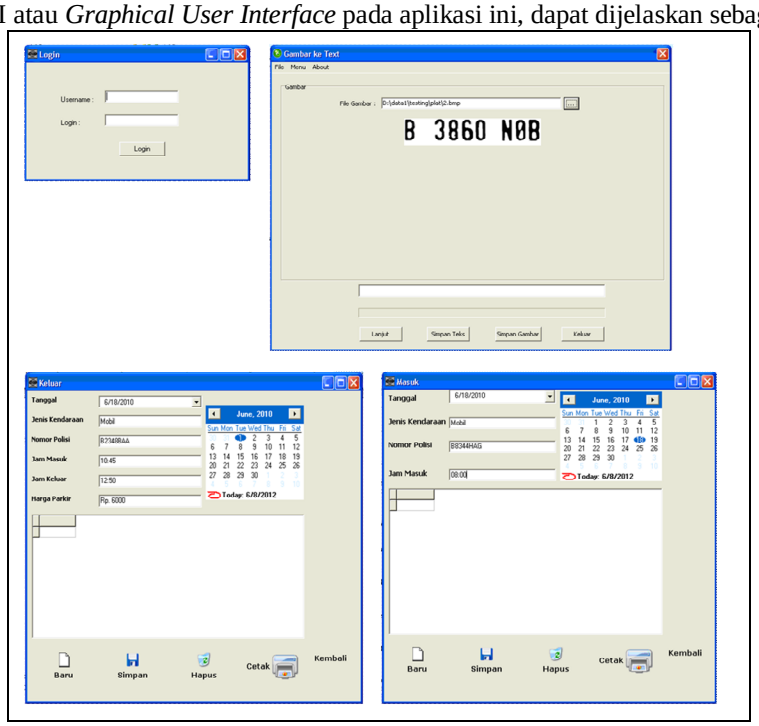
Dalam pembuatan sistem ini bertujuan untuk merubah bentuk dari citra plat nomor menjadi sebuah karakter. Pada awalnya pengguna memasukkan input data berupa citra. Citra masukan adalah citra dengan format *bitmap grayscale* karena sistem hanya dibatasi untuk memproses citra *bitmap grayscale*.
Carta alir berikut merupakan implementasi dari perubahan bentuk citra plat nomor menjadi sebuah karakter yang melalui proses pengambilan citra menggunakan alat bantu kamera atau *image acquisition*, selanjutnya diproses *pre-processing*, yaitu input gambar, peningkatan kontras, penajaman, binerisasi, segmentasi, dan scalling. Kemudian melalui proses *feature extraction* merupakan suatu pengambilan ciri dari suatu bentuk. Dalam hal ini adalah gambar karakter selanjutnya nilai yang didapatkan akan dianalisis untuk proses selanjutnya. Selanjutnya proses perbandingan dengan *font* yang sudah terintalasi pada komputer, yang nantinya didapatkan hasilnya berupa karakter. Gambar di bawah merupakan alur proses pengecaman karakter digital ini berjalan.



Gambar 4. Alur proses proses pengecaman karakter digital

Dalam melakukan percobaan menggunakan simulasi dengan jenis font *DIN Medium* yang sudah diinstalasi pada komputer yang diuji cobakan, karena kemiripan bentuk angka dan hurufnya. Percobaan dilakukan dengan

sepuluh citra dengan jenis *font* yang sama. Kemudian, menggunakan lima gambar dengan jenis *font Arial*, dan dengan lima citra plat nomor sebenarnya. Aplikasi yang dibuat menggunakan dua jenis progam untuk mengambil karakter dari citra (Malkoff, 2006), dan sistem perparkiran untuk menghitung total kendaraan yang masuk. Warna dari latar belakangnya dibuat putih dan tulisan dibuat hitam, karena program yang dibuat belum memenuhi untuk berwarna latar belakang hitam dan tulisan putih.
Contoh tampilan GUI atau *Graphical User Interface* pada aplikasi ini, dapat dijelaskan sebagai berikut :



Gambar 5. Tampilan GUI

Tampilan GUI di atas termasuk *form* login, *form* utama, *form* masuk kendaraan, dan *form* keluar kendaraan.
Percobaan dengan jenis *font* yang sama, jenis font yang berbeda, dan gambar plat nomor sebenarnya, dapat dijelaskan dengan tabel-tabel sebagai berikut :

Tabel 1. Percobaan gambar jenis *font* sama

No.	Percobaan	Hasil	Tingkat Keberhasilan	Waktu
1.	B 8475 KBN	B8475KBN	100%	48 s
2.	B 7564 JAQ	B7584JA0	75%	49 s
3.	B 1475 KLD	B1475KUD	87.5%	48 s
4.	B 2907 RWG	B2q07RWS	75%	48 s
5.	B 7542 VS	B7542VS	100%	40 s
6.	B 842 RO	B842R0	100%	32 s
7.	B 9764 KLP	Bq784KUP	62.5%	49 s
8.	B 3602 JJA	B3802JJA	100%	52 s
9.	B 555 OK	B5550K	100%	36 s
10.	B 1 PW	B1PW	100%	24 s

Tabel 2. Percobaan gambar jenis *font* berbeda (*Arial*)

No.	Percobaan	Hasil	Tingkat Keberhasilan	Waktu
1.	B 8239 YTX	II239Y	50%	44 s
2.	B 777 VGA	I777	42.86%	31 s
3.	B 128 KH	I12I	33.33%	35 s
4.	B 4703 GFD	I4703GFD	87.5%	48 s
5.	B 5391 IUA	I5391IUA	87.5%	49 s

Tabel 3. Percobaan gambar plat nomor sebenarnya

No.	Percobaan	Hasil	Tingkat Keberhasilan	Waktu
1.	B 3860 N0B	I3IIIIII	12.5%	2 m 13 s
2.	B 510 VW	r	0%	1 m 49 s
3.	B 1620 PFV	B	12.5%	2 m 10 s
4.	B 6349 TAP	II	0%	2 m 20 s
5.	B 6501 SGD	II	0%	2 m 20 s

Hasil dari percobaan diatas dapat dijelaskan rata-rata tingkat keberhasilan untuk gambar jenis *font* yang sama (100%+ 75% + 87.5% + 75% + 100% + 100% + 62.5% + 100% + 100% + 100%) / 10 = 80%. Rata-rata tingkat keberhasilan untuk gambar dengan jenis *font Arial* (50% + 42.86% + 33.33% + 87.5% + 87.5%) / 5 = 64.82%. Rata-rata tingkat keberhasilan untuk gambar plat nomor sebenarnya (12.5% + 0% + 12.5% + 0% + 0%) / 5 = 5%.

5. KESIMPULAN DAN SARAN

Setelah melakukan pengujian dan implementasi terhadap aplikasi yang telah dibuat, dapat disimpulkan sebagai berikut:

a. Dari hasil penelitian yang dilakukan, masih beberapa gambar yang bisa terdeteksi dan tidak bisa dideteksi mendapatkan karakter huruf angka dengan aplikasi ini.
b. Dalam hal pendeteksian untuk aplikasi ini masih tahap simulasi plat nomor dari warna latar belakang putih dan warna tulisan hitam, sehingga belum dapat memberikan kepastian 100% pendeteksian huruf dan angka.

Saran untuk pengembangan selanjutnya, dapat dijelaskan sebagai berikut :

a. Pendeteksian dalam huruf dan angka sebaiknya langsung difoto plat nomor dengan kamera langsung beserta tampak depan kendaraan.
b. Dapat dikembangkan secara mobile untuk smartphone, seperti Android, Blackberry, IOS, Windows Phone, dan sebagainya.
c. Walaupun pada projek ini bahasa yang digunakan adalah Dhelpi tetapi penggunaan MATLAB dengan fasilitas *Image Acquistion Tool Box* dan *Image Processing Tool Box* dapat juga diterapkan untuk projek yang sama.

DAFTAR PUSTAKA

A. Akoum, B. Daya, and P. Chauvet, 2010, *A New Algorithmic Approach for Detection and Identification of Vehicle Plate Numbers*, International Journal of Computer Technology and Electronics Engineering (IJCTEE) Vol. 1, Issue 1, Saida, pp 99-108

Asthana, Stuti, Niresh Sharma, and Rajdeep Singh, 2010, *Vehicle number plate recognition using multiple layer back propagation neural networks*, International Journal of Computer Technology and Electronics Engineering (IJCTEE) Vol. 1, Issue 1, Bhopal, pp 35-38

Beck, Kent, 1999, *Extreme Programming Explained: Embrace Change*, Addison-Wesley

Humayun K. Sulehria, Ye Zhang, Danish Irfan, and Atif Karim Sulehria, 2008, *Vehicle Number Plate Recognition Using Hybrid Mathematical Morphological Techniques*, Proceedings of the 7th WSEAS International Conference on SIGNAL PROCESSING (SIP'08), Istanbul, pp 127-133

Gonzalez, Rafael C., William K, 2004, *Digital Image Processing*, Pearson Education, Inc., New Jersey

M. M. Rashid, A. Musa, M. Ataur Rahman, and N. Farahana, A. Farhana, 2012, *Automatic Parking Management System and Parking Fee Collection Based on Number Plate Recognition*, International Journal of Machine Learning and Computing, Vol. 2, No. 2, Kuala Lumpur, pp 93-98

Malkoff, Dennis, 2006, *Get image chars*, http://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=1796&lngWId=7, diakses pada tanggal 27 Mei 2012.

Miano, John. 1999. *Compressed Image File Format : JPEG, PNG, GIF, XBM, BMP*. Addison Wesley Longman, Inc, Massachusetts

Ozbay, Serkan, and Ergun Ercelebi, 2005, *Automatic Vehicle Identification by Plate Recognition*, World Academy of Science, Engineering and Technology, Gaziantep, pp 222-225

P. K. Suri, Ekta Walia, and Amit Verma, 2010, *Vehicle Number Plate Detection using Sobel Edge Detection Technique*, International Journal of Computer Science and Technology, IJCST Vol. 1, Issue 2, Haryana, pp 179-182

Pratt, William K., 2007, *Digital Image Processing*, 4th ed, John Wiley & Sons, Inc., New Jersey

Sutoyoso, T., 2009, *Teori Pengolahan Citra*, Andi, Yogyakarta

Munir, Rinaldi, 2005, *Pengolahan Citra Digital*, Informatika Bandung, Bandung

Widhiartha, Putu, 2008, *Extreme Programming – Melakukan Pengembangan Perangkat Lunak dengan Lebih Sederhana*, http://ilmukomputer.org/2008/05/28/extreme-programming-%E2%80%93-melakukan-pengembangan-perangkat-lunak-dengan-lebih-sederhana/, diakses pada tanggal 28 Mei 2012