Aplikasi Ekstraksi Fitur Citra Hufur Jawa Berdasarkan Morfologinya

Meiliana Indrawijaya, Liliana, Rudy Adipranata
Petra Christian University
Siwalankerto 121-131 Surabaya
+62 31 8439040, 8394830-31
meiliindrawijaya@gmail.com

ABSTRAK

Ekstraksi fitur merupakan topik penting dalam klasifikasi, karena fitur-fitur yang baik akan dapat meningkatkan tingkat akurasi untuk pengenalan suatu objek. Bahasa Jawa adalah bahasa yang sering digunakan oleh masyarakat di Pulau Jawa, Indonesia. Ekstraksi fitur huruf Jawa memiliki tingkat kesulitan sendiri karena huruf Jawa terdiri dari banyak jenis huruf dan kombinasinya. Dalam skripsi ini dikembangkan aplikasi yang dapat mengekstrak fitur tulisan tangan huruf Jawa.

Adapun proses yang dilakukan adalah sebagai berikut: gambar huruf Jawa yang sudah disegmentasi sebelumnya, di-*skeleton* dengan menggunakan metode Zhang Suen. Hasil *skeletonizing* ini nantinya akan digunakan untuk mengekstrak fitur gambar huruf Jawa, yang terdiri dari fitur *loop*, garis lurus dan kurva lengkung. Deteksi fitur *loop* menggunakan metode *flood fill*, garis dengan metode *hough transform*, dan kurva menggunakan metode *Hough Transform* dengan algoritma untuk deteksi kurva

Input berupa gambar huruf Jawa yang sudah disegmentasi sebelumnya. *Output* berupa jenis dan jumlah fitur yang dikenali dan dapat disimpan dalam format .csv. Aplikasi ini dibuat dengan bahasa pemrograman C# dengan Microsoft Visual Studio 2010 sebagai IDE-nya.

Kata Kunci: Huruf Jawa, *skeletonizing*, Ekstraksi Fitur, *Image Processing*

ABSTRACT

Feature extraction is an important topic in the classification. Well detected features will be able to increase the level of accuracy for the recognition of an object. Javanese language is a language that is often used by people on the island of Java, Indonesia. Feature extraction of Javanese letter has its own difficulty level for Javanese letter consists of many types of letters and combinations. In this paper, developed an application that can extract a handwritten Javanese letter features.

The process is carried out as follows: image that has been previously segmented, skeleton by using the method of Zhang Suen. The results of this skeletonizing will be used to extract image features, which consists of loops, straight lines and curves arch. Loop detection using flood-fill algorithm, line detection using Hough Transform and curve detection using Hough Transform with curve detection algorithm.

The input is Javanese letters image that has been segmented previously. The output is the type and number of features that are recognized and can be saved in .csv format. This application is made to the programming language C # with Microsoft Visual Studio 2010 as the IDE.

Keywords: *Javanese letter, skeletonizing, Feature Extraction, Image Processing.*

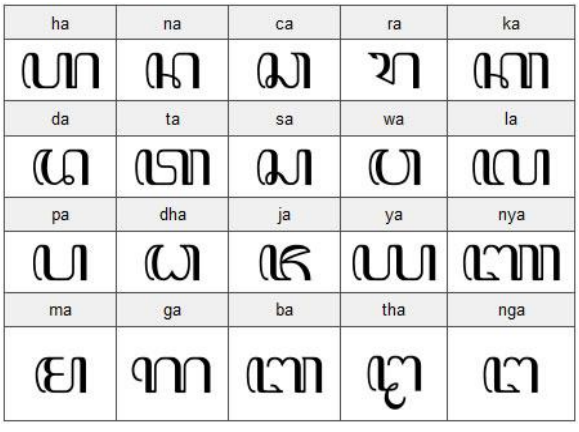
1. INTRODUCTION

Bangsa Indonesia merupakan bangsa besar yang terdiri dari banyak suku bangsa. Salah satunya budaya Jawa berupa aksara Jawa yang digunakan di Pulau Jawa, Indonesia. Dalam rangka memudahkan dalam pelestarian aksara Jawa tersebut, maka diperlukan sebuah aplikasi yang mampu membantu mengenali aksara Jawa dari sebuah gambar/citra yang ada. Pengenalan huruf-huruf Jawa mempunyai tingkat kesulitan sendiri karena adanya huruf-huruf dasar, vokal, pasangan, dan karakter-karakter pembantu lainnya. Pada penelitian ini, peneliti akan mengembangkan aplikasi untuk mengekstrak fitur aksara Jawa yang berdasarkan ciri bentuknya (morfologinya).

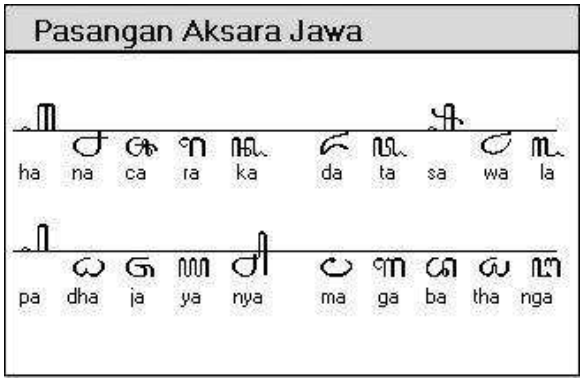
2. LANDASAN TEORI

2.1 Huruf Jawa

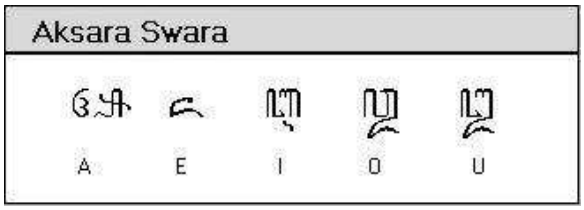
Aksara Jawa berbeda dengan huruf Latin yang biasa digunakan. Aksara Jawa terdiri dari aksara Carakan (Gambar 1), aksara Pasangan (Gambar 2), aksara Swara (Gambar 3), aksara Rekan (Gambar 4), aksara Murda (Gambar 5), aksara Wilangan (Gambar 6) dan Pelengkap/Sandhangan (Gambar 7). [6]



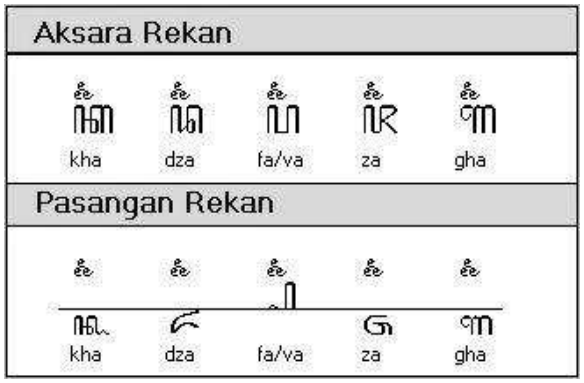
Gambar 1. Aksara Carakan [4]



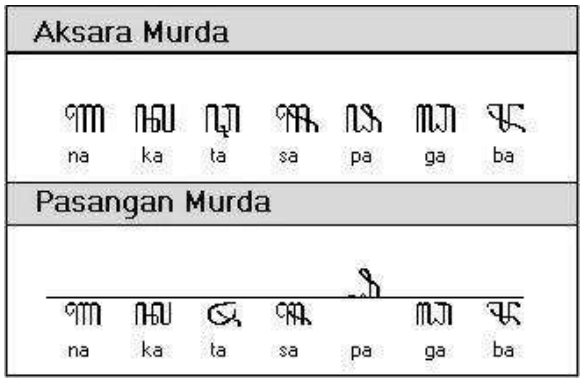
Gambar 2. Aksara Pasangan [4]



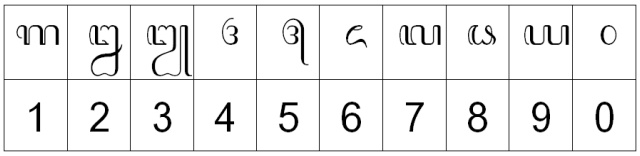
Gambar 3. Aksara Swara [4]



Gambar 4. Aksara Rekan [4]



Gambar 5. Aksara Murda [4]



Gambar 6. Aksara Wilangan [9]

Nama Sandhangan	Aksara Jawa	Keterangan
Wulu		tanda vokal i
Suku		tanda vokal u
Taling		tanda vokal é
Pepet		tanda vokal e
Taling tarung		tanda vokal o
Layar		tanda ganti konsonan r
Wignyan		tanda ganti konsonan h
Cecak		tanda ganti konsonan ng
Pangkon/pangku		tanda penghilang vokal
Péngkal		tanda ganti konsonan ya
Cakra		tanda ganti konsonan ra
Keret		tanda ganti konsonan re

Gambar 7. Pelengkap/Sandhangan [4]

2.2 Skeletonizing

Skeletonizing merupakan salah satu *image processing* yang digunakan untuk mengurangi *pixel* dari suatu *image* dengan tetap mempertahankan informasi, karakteristik dan *pixel* penting dari objek tersebut. Hal ini diimplementasikan dengan mengubah *image* awal dengan pola *binary* menjadi representasi kerangka (*skeletal representation*) *image* tersebut. Tujuan dari *skeletonizing* adalah membuat gambar yang lebih sederhana sehingga gambar tersebut dapat dianalisis lebih lanjut dalah hal bentuk dan kecocokannya maupun untuk dibandingkan dengan gambar lainnya untuk dikenali.

Ada dua syarat yang digunakan untuk menentukan apakah sebuah *pixel* bisa dihapus atau tidak. Syarat yang pertama adalah sebagai berikut [7]:

- Sebuah *pixel* dapat dihapus hanya jika dia memiliki lebih dari satu dan kurang dari 7 tetangga.

- Sebuah *pixel* dapat dihapus hanya jika memiliki *connectivity number* sama dengan satu.
- Sebuah *pixel* dapat dihapus hanya jika setidaknya satu dari tetangganya yang berada di arah 1, 3, atau 5 adalah *pixel* yang menandakan *background*.
- Sebuah *pixel* dapat dihapus hanya jika salah satu tetangganya yang berada pada arah 3, 5, atau 7 adalah *pixel* yang menandakan *background*.

Syarat yang kedua kurang lebih sama seperti syarat yang pertama namun berbeda pada dua langkah terakhir yaitu:

- Sebuah *pixel* dapat dihapus hanya jika setidaknya satu dari tetangganya yang berada di arah 7, 1, atau 3 adalah *pixel* yang menandakan *background*.
- Sebuah *pixel* dapat dihapus hanya jika salah satu tetangganya yang berada pada arah 1, 5, atau 7 adalah *pixel* yang menandakan *background*.

2.3 Ekstraksi Fitur

Pengolahan citra *digital* adalah sebuah disiplin ilmu yang mempelajari hal-hal berkaitan dengan perbaikan kualitas gambar (peningkatan kontras, transformasi warna, restorasi citra), transformasi gambar (rotasi, translasi, skala, transformasi geometrik), melakukan proses penarikan informasi atau deskripsi objek atau pengenalan objek yang terkandung pada citra, melakukan kompresi atau reduksi data untuk tujuan penyimpanan data, transmisi data, dan waktu proses data [8]. Ekstraksi fitur merupakan salah satu bagian dari pengolahan citra *digital.*

Ekstraksi fitur adalah proses untuk menemukan pemetaan dari fitur-fitur asli ke dalam fitur- fitur baru yang diharapkan dapat menghasilkan keterpisahan kelas secara lebih baik [3]. Ekstraksi fitur merupakan topik penting dalam klasifikasi, karena fitur-fitur yang baik akan sanggup meningkatkan tingkat akurasi, sementara fitur-fitur yang tidak baik cenderung memperburuk tingkat akurasi. [2]

Untuk pengenalan *loop* akan digunakan metode *flood fill*, sedangkan untuk pengenalan garis dan kurva akan digunakan metode *hough transform.*

Hough Transform merupakan suatu teknik untuk menentukan lokasi suatu bentuk dalam citra. *Hough Transform* dicetuskan pertama kali oleh P.V.C Hough (1962), dilihat potensinya sebagai salah satu algoritma dalam pemrosesan citra oleh Rosenfeld (1969), kemudian diimplementasikan untuk mendeteksi garis dalam citra Duda (1972).

Hough Transform melakukan pemetaan terhadap titik-titik pada citra ke dalam parameter *space* (*hough transform space*) berdasar kan suatu fungsi yang mendefinisikan bentuk yang ingin dideteksi, kemudian melakukan *voting* pada suatu elemen *array* yang disebut *accumulator array*. *Hough transform* umumnya digunakan untuk melakukan ekstrasi garis, lingkaran atau *elips* pada citra, namun dalam perkembangnnya, *Hough transform* juga telah dapat digunakan untuk melakukan ekstraksi bentuk-bentuk yang lebih kompleks.

Pada tahun 1972, Duda dan Hart [1] mengusulkan persamaan polar untuk garis dengan parameter besar rho dan orientasi theta

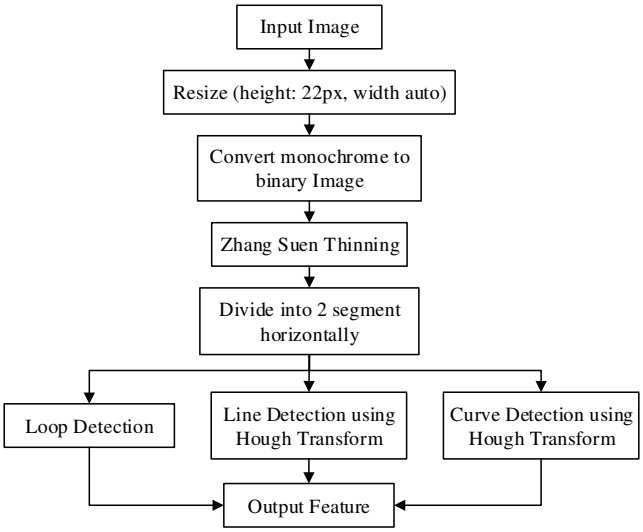
$$\rho = x \cos\theta + y \sin\theta \quad (1)$$

Algoritma untuk pengenalan kurva ini diajukan oleh M.Z. Mat Jafri and F. Deravi pada tahun 1994 (Jafri & Deravi, 1994). Algoritma ini digunakan untuk deteksi kurva parabola berdasarkan metode *Hough Transform*. Pendekatan ini menggunakan titik dari kurva sebagai parameter yang juga menunjukkan posisi kelengkungan maksimum kurva parabola. Operator Sobel digunakan untuk pendekatan gradien. Untuk mendeteksi kurva parabola di posisi mana saja, sebuah transformasi koordinat matriks digunakan untuk menurunkan persamaan parabola baru yang melibatkan orientasi kurva parabola. [5]

3. DESAIN SISTEM

3.1 Garis Besar Sistem Kerja Perangkat Lunak

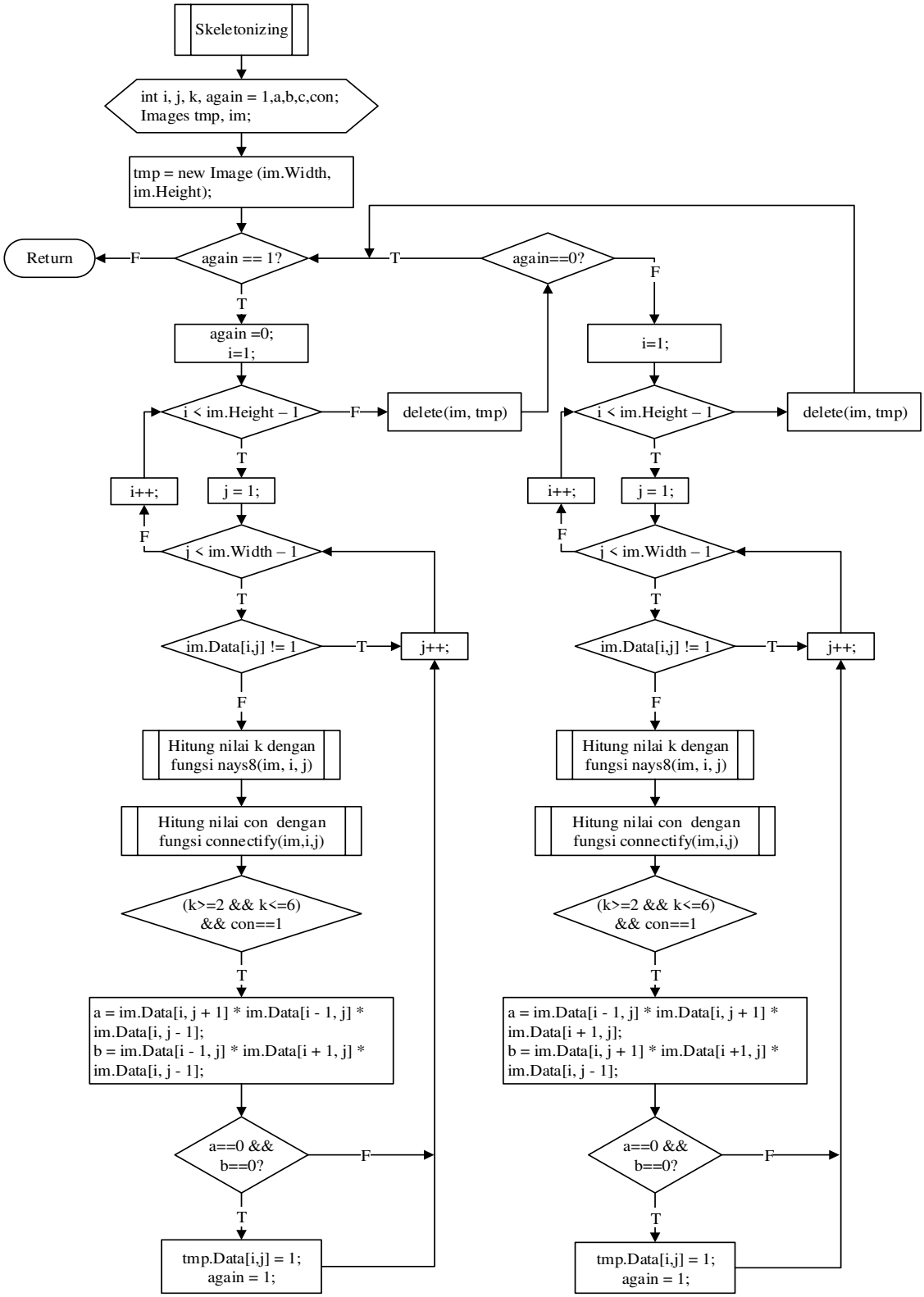
Sistem perangkat lunak untuk ekstraksi fitur citra huruf Jawa ini dimulai dengan proses *input* gambar. Setelah itu gambar di-*resize* dan diubah dari *monochrome* menjadi *binary.* Selanjutnya gambar melewati proses *skeletonizing* untuk mendapatkan kerangka dasar objek berukuran 1 *pixel* dan dibagi menjadi 2 segmen secara horizontal. Gambar tersebut digunakan untuk ekstraksi fitur *loop*, fitur garis dan fitur kurva pada citra huruf Jawa. Secara garis besar, sistem perangkat lunak ini dapat digambarkan seperti blok diagram yang ditunjukkan pada Gambar 8 berikut:



Gambar 8. Blok diagram garis besar sistem kerja perangkat lunak

3.2 Garis Besar Fitur *Skeletonizing*

Untuk melakukan proses *skeletonizing* terdapat beberapa tahapan yang harus dilakukan, sesuai dengan aturan Zhang Suen (dapat dilihat pada poin 2.2). Adapun rancangan sistem kerja perangkat lunak untuk modul *skeletonizing* secara garis besar ditunjukan oleh Gambar 9 berikut:



Gambar 9. Diagram Alir proses *Skeletonizing*

4. IMPLEMENTASI SISTEM

Dalam pembuatan skripsi ini, sebagian proses yang ada dibuat menggunakan *environment* pendukung dengan menggunakan library EmguCV.

Pertama-tama, gambar akan di *resize* ke ukuran yang lebih kecil. Selanjutnya dijalankan fungsi *skeletonizing* yang berfungsi untuk menghilangkan ketebalan garis tanpa menghilangkan bagian yang penting dalam suatu objek. Proses *skeletonizing* dimulai dengan iterasi pertama untuk menandai *pixel-pixel* yang dapat dihapus. Selanjutnya, *pixel-pixel* yang telah ditandai akan dihapus dengan fungsi *delete*. Jika masih ada *pixel* yang memenuhi syarat untuk dihapus, maka proses *mark and delete* akan berjalan hingga kriteria tidak terpenuhi lagi (sudah tidak ada *pixel* yang *redundant*). Setelah proses *mark and delete* selesai, dilanjutkan dengan fungsi *stair* untuk menghaluskan hasil *skeletonizing* objek.

Selanjutnya gambar akan melalui proses deteksi fitur *loop*. Proses dimulai dengan membingkai gambar dengan 1 baris tambahan bernilai *background*. Selanjutnya, dijalankan fungsi *floodfill* pada salah satu titik sudut gambar, yaitu pada koordinat 0, 0. Kemudian akan dilakukan iterasi untuk mengecek apakah masih terdapat *pixel background* pada gambar. Jika masih terdapat *pixel* bernilai *background*, maka fungsi *floodFill* akan dijalankan kembali dengan titik tersebut sebagai *seedPoint*. Iterasi dihentikan jika sudah tidak ada lagi *pixel background* pada gambar. Jumlah *loop* didapatkan dengan menghitung jumlah iterasi, dimana fungsi *floodfill* menghasilkan nilai diatas atau sama dengan minimal area.

Selanjutnya untuk deteksi garis, akan dijalankan metode *Hough Transform*. Fungsi *votingRhoValue* berfungsi untuk menghitung nilai Rho untuk tiap *pixel* dalam gambar. Selanjutnya, dijalankan fungsi *findLocalMaxima* untuk mendapatkan nilai maksimum lokalnya. Fungsi *filterlines* digunakan untuk membersihkan *noise*. Hasil garis yang terdeteksi akan diolah sesuai dengan nilai parameter, yaitu minimum panjang garis, maksimum *gap* antar garis.

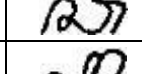
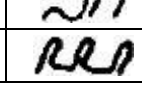
Untuk deteksi fitur kurva, dijalankan metode yang sama dengan *Hough Transform* untuk deteksi garis

5. PENGUJIAN SISTEM

5.1 Pengujian terhadap Kecepatan Proses *Skeletonizing* dengan Ukuran Gambar yang Berbeda

Pengujian terhadap waktu proses *skeletonizing* pada gambar *input* yang sama dengan ukuran yang berbeda dapat dilihat pada Tabel 1 di bawah ini.

Tabel 1. Hasil pengujian terhadap kecepatan proses *skeletonizing* dengan ukuran gambar yang berbeda

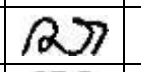
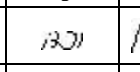
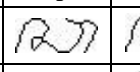
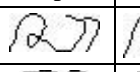
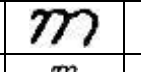
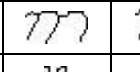
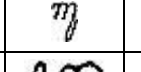
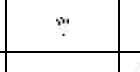
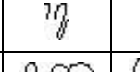
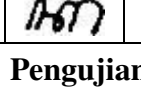
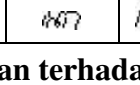
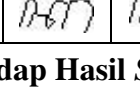
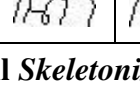
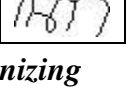
No	Normal	10 px	20 px	30 px	40 px
1		5 ms	6 ms	9 ms	13 ms
2		5 ms	6 ms	10 ms	12 ms
3		5 ms	5 ms	10 ms	10 ms

Dari hasil pengujian dapat disimpulkan bahwa semakin besar ukuran gambar, maka proses *skeletonizing* membutuhkan waktu proses yang lebih banyak.

5.2 Pengujian terhadap Hasil *Skeletonizing* dengan Ukuran Gambar yang Berbeda

Pengujian ini adalah pengujian terhadap hasil *skeletonizing image* pada gambar *input* dengan ukuran yang berbeda-beda (dapat dilihat pada Tabel 2). Pengujian dilakukan pada gambar dengan ukuran tinggi tertentu, sedangkan lebarnya menyesuaikan rasio lebar gambar asli terhadap tinggi gambar. Pada pengujian ini diambil ukuran tinggi sebesar 10 *pixel*, 15 *pixel*, 20 *pixel*, 22 *pixel*, 24 *pixel*, dan 26 *pixel*. Gambar yang semakin besar menunjukkan hasil yang lebih detail. Gambar dengan ukuran kecil menyebabkan beberapa bagian penting dalam *skeleton* yang hilang.

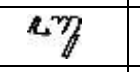
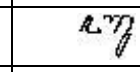
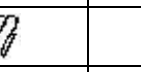
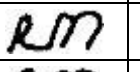
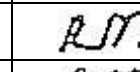
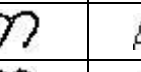
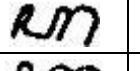
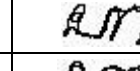
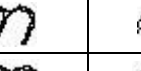
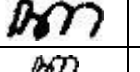
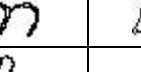
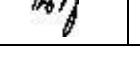
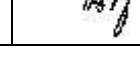
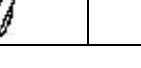
Tabel 2. Hasil pengujian proses *skeletonizing* terhadap gambar *input* dengan ukuran yang berbeda

No	Normal	*Height* 10px	*Height* 20px	*Height* 22px	*Height* 26px
1					
2					
3					
4					

5.3 Pengujian terhadap Hasil *Skeletonizing* dengan Parameter yang Berbeda

Penggunaan parameter *window size* diharapkan dapat mengatasi secara langsung masalah *loop* yang tidak berlubang dan 2 garis yang menempel. Tabel 3 menunjukkan hasil *skeletonizing* dengan gambar berukuran tinggi 22 *pixel* terhadap perubahan parameter *window size* 3x3. Dari hasil pengujian, ukuran *window size* yang terlalu kecil menyebabkan garis yang seharusnya tidak berlubang menjadi berlubang, sehingga hasil *skeleton* tidak mencerminkan kerangka asli objek.

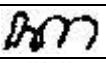
Tabel 3. Hasil pengujian *skeletonizing* dengan *window size* 3x3

No	Gambar Normal	Gambar *Pre_Process*	Gambar hasil *Skeletonizing*
1			
2			
3			
4			
5			

Tabel 4 di bawah ini menunjukkan hasil *skeletonizing* terhadap perubahan parameter *window size* 5x7. Bagian *loop* yang seharusnya berlubang sudah berlubang, dan bagian yang seharusnya tidak berlubang juga tetap tidak berlubang. Untuk

ukuran gambar dengan tinggi 22px, *window size* 5x7 sudah menunjukkan hasil yang stabil. Secara visual, hasil *skeleton* lebih sesuai dengan kerangka objek aslinya.

Tabel 4. Hasil pengujian *skeletonizing* dengan *window size* 5x7

No	Gambar Normal	Gambar Pre_Process	Gambar *Skeleton*
1			
2			
3			
4			
5			

5.4 Pengujian hasil *Line Detection*

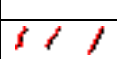
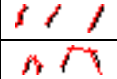
Hasil pengujian *Line Detection* dengan menggunakan metode *Hough Transform* sebelum difilter dapat dilihat pada Tabel 5 di bawah ini. Garis-garis yang sejenis masih dideteksi menjadi beberapa garis, sehingga garis yang seharusnya hanya berjumlah satu garis di deteksi lebih dari satu garis.

Tabel 5 Hasil Pengujian Line Detection Dengan Metode Hough Transform Sebelum Difilter

No	Gambar	Gambar Asli	Gambar Hasil	Gambar Union	Jumlah Garis
1					20
					211
2					113
					135

Hasil pengujian *Line Detection* dengan menggunakan metode *Hough Transform* setelah difilter dapat dilihat pada Tabel 6 di bawah ini. Setelah difilter, garis-garis yang sejenis dianggap menjadi satu garis, sehingga hasil deteksi garis lebih akurat (lebih sesuai dengan hasil objek).

Tabel 6. Hasil Pengujian Line Detection Dengan Metode Hough Transform Setelah Difilter

No	Gambar	Gambar Asli	Gambar Hasil	Gambar Union	Jumlah Garis
1					4
					4
2					6
					7

5.5 Pengujian terhadap Hasil Ekstraksi Fitur Garis dengan Parameter yang Berbeda

Pengujian berikut adalah pengujian terhadap perubahan nilai parameter pada *Line Detection*. Parameter yang diuji pada jurnal ini terdiri dari *Theta Resolution* dan *Threshold.* Parameter *Theta Resolution* merupakan resolusi sudut dalam satuan derajat. Pengujian terhadap pengaruh parameter *Theta Resolution* dapat dilihat pada Tabel 7. Parameter *Theta Resolution* ini berpengaruh pada increment nilai theta/angle, artinya garis akan dideteksi setiap n-derajat nilai theta/angle. Dari hasil pengujian, dapat disimpulkan bahwa semakin kecil nilai Theta Resolution, maka semakin detail garis yang dapat terdeteksi.

Tabel 7. Hasil pengujian pengaruh parameter *Theta Resolution* terhadap *Line Detection*

Parameter	*Image*	*Image* Hasil	*Image Union*	Jumlah *Line*
Theta = 1				4
				5
Theta = 2				8
				8
Theta = 3				7
				7

Parameter *Threshold* merupakan batasan nilai minimum suatu nilai dalam *accumulator array* yang dapat dinyatakan sebagai garis. *Threshold* terdiri dari dua jenis, yaitu *by Percentage* dan *by Value.* Nilai *threshold by Percentage* didapatkan dari n% dari nilai maksimum pada *accumulator array*. Pengujian pengaruh parameter *Threshold by Value* dapat dilihat pada Tabel 8 dan *Threshold by Percentage* dapat dilihat pada Tabel 9. Dari hasil pengujian, dapat disimpulkan bahwa threshold yang terlalu kecil menyebabkan *noise* terdeteksi. Akan tetapi, threshold yang terlalu besar menyebabkan garis pendek tidak terdeteksi.

Tabel 8. Hasil pengujian pengaruh parameter *Threshold by Value* terhadap *Line Detection*

Parameter	*Image*	*Image* Hasil	*Image Union*	Jumlah *Line*
Threshold = 1				6
				8
Threshold = 2				4
				5
Threshold = 3				0
				4

Tabel 9. Hasil pengujian pengaruh parameter *Threshold by Percentage* terhadap *Line Detection*

Parameter	***Image***	***Image* Hasil**	***Image Union***	**Jumlah *Line***
Threshold = 90%				0
				0
Threshold = 60%				0
				0
Threshold = 50%				2
				2

Tabel 10 menunjukkan perbandingan *line detection* dengan metode *Hough Transform* dengan filter dan *Hough Transform* pada *library* EmguCV. Nomor 1 dan 2 adalah hasil *Hough Transform* dengan filter. Nomor 3 dan 4 adalah hasil *line detection library*. Dari hasil pengujian, hasil deteksi garis tidak terlalu berbeda jauh antara metode *Hough Transform* dengan filter dan *Hough Transform* dengan *library* EmguCV. Oleh karena itu, metode *Hough Transform* dengan filter ini dapat digunakan untuk ekstraksi fitur garis pada citra Huruf Jawa.

Tabel 10. Perbandingan *line detection* yang telah dibuat dengan yang ada pada *library* Emgu CV

No	**Normal**	***Line* yang Terdeteksi**	***Image* Gabungan**
1			
2			
3			
4			

6. KESIMPULAN

Berdasarkan hasil pengujian, dapat disimpulkan beberapa hal sebagai berikut:

- Penambahan parameter window space pada skeletonizing dapat mengatasi masalah loop yang tidak berlubang dan 2 garis pararel yang berhimpit.
- Pembagian gambar menjadi 2 segmen dapat menekan kesalahan deteksi loop.
- Penambahan parameter minimum area pada deteksi loop dapat menekan kesalahan deteksi noise sebagai loop.
- Garis yang sejenis dapat dihilangkan/diabaikan dengan penambahan filterisasi pada metode pengenalan garis Hough Transform, sehingga dengan hasil garis yang terdeteksi lebih akurat dibandingkan dengan tanpa filterisasi.
- Kegagalan pada deteksi kurva disebabkan karena tulisan tangan yang seharusnya lengkung tetapi relatif cukup datar, sehingga luput dari deteksi kurva

7. REFERENCES

[1] Duda, R., & Hart, P. 1972. Use of the Hough Transform to Detect Lines and Curves in Pictures. *Communications of the ACM, 15*, 11.

[2] Gonzalez, R., & Woods, R. 2008. *Digital Image Processing*, 3rd ed. New Jersey, USA: Pearson Prentice Hall.

[3] Guo, L., Rivero, D., Dorado, J., Munteanu, C., & Pazos, A. 2011. *Automatic Feature Extraction Using Genetic Programming: An Application to Epileptic EEG Classification, Expert Systems with Applications.*

[4] Hastuti, D. 2011, April 11. *Mari belajar (lagi) Menulis Aksara Jawa*. Retrieved Mei 14, 2014, from http://dhenokhastuti.com/2011/04/11/mari-belajar-lagi-menulis-aksara-jawa/

[5] Jafri, M., & Deravi, F. 1994. Efficient algorithm for the detection of parabolic curves. *Vision Geometry III*, 53-62.

[6] Nusantara Ranger. n.d.. Retrieved September 28, 2014, from http://nusantaranger.com/referensi/buku-elang/chapter-4merah/aksara-jawa-hanacaraka/.

[7] Parker, J. R. 1993. *Practical Computer Vision Using C* (1 ed.). New York, NY, USA: John Wiley & Sons, Inc.

[8] Sutoyo, T., Mulyanto, E., Suhartono, V., Nurhayati, O., & Wijanarto. 2009. *Teori Pengolahan Citra Digital*. Yogyakarta: Penerbit: Andi.

[9] Syahra, U. 2013, April 22. *Mengingat Kembali Aksara Jawa*. Retrieved Mei 14, 2014, from http://urviasyahra20.blogspot.com/2013/04/materi-bahasa-jawa-nyinau-aksara-jawa.html