PENERAPAN CIRI GEOMETRIC PADA DETEKSI DAN VERIFIKASI TANDA TANGAN *OFFLINE*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Wenny Ramadha Putri
NIM: 135150207113008

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2017

PENGESAHAN

PENERAPAN CIRI *GEOMETRIC* PADA DETEKSI DAN VERIFIKASI TANDA TANGAN *OFFLINE*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Wenny Ramadha Putri
NIM: 135150207113008

Skripsi ini telah diuji dan dinyatakan lulus pada
10 Agustus 2017
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	*Dosen Pembimbing II*
Agus Wahyu Widodo, S.T., M.Cs.	Bayu Rahayudi, S.T., M.T.
NIP: 19740805 200112 1 001	NIP: 19740712 200604 1 000

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 10 Agustus 2017

Wenny Ramadha Putri

NIM: 135150207113008

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Penerapan Ciri *Geometric* pada Deteksi dan Verifikasi Tanda Tangan Offline" dengan baik dan tepat pada waktunya. Dalam penyelesaian skripsi ini tidak luput dari berbagai hambatan dan kesulitan karena keterbatasan kemampuan yang penulis miliki. Maka terselesainya skripsi ini tentu tidak terlepas dari bantuan dan dukungan berbagai pihak. Karenanya, pada kesempatan ini penulis ingin mengucapkan terimakasih kepada semua pihak diantaranya:

1. Agus Wahyu Widodo, S.T, M.Cs selaku dosen pembimbing pertama yang telah membimbing, memberi saran dan motivasi kepada penulis selama proses penyusunan skripsi.
2. Bayu Rahayudi, S.T.,M.T selaku dosen pembimbing kedua yang telah membimbing, memberi saran serta motivasi kepada penulis selama penyusunan skripsi ini.
3. Seluruh dosen dan civitas akademika Fakultas Ilmu Komputer yang telah memberikan wawasan, bantuan dan dukungan yang sangat bermanfaat bagi penulis.
4. Kedua orang tua, saudara, dan keluarga penulis yang telah memberikan fasilitas, dukungan dan semangat yang luar biasa, serta doa-doa yang tiada henti kepada penulis.
5. Seluruh teman-teman seperjuangan, Sobat TUJUH, KEMAL, Kos by Accident, SABEB ENTERTAINMENT, dan Informatika UB angkatan 2013 yang telah memberikan dukungan dan semangat kepada penulis.
6. Semua pihak yang tidak dapat disebutkan satu persatu sehingga terselesainya skripsi ini.

Penulis menyadari dalam penyusunan skripsi ini masih banyak terdapat kesalahan. Penulis mengharapkan kritik dan saran untuk perbaikan yang lebih baik. Penulis berharap semoga laporan skripsi ini dapat memberikan manfaat bagi penulis serta pembaca.

Malang, 10 Agustus 2017

Penulis

wennyramadha@gmail.com

ABSTRAK

Berbagai upaya dalam mengamankan informasi personal telah banyak dilakukan secara tradisional maupun biometric. Dan di antara berbagai cara untuk melindungi informasi, tanda tangan merupakan yang paling banyak digunakan dalam mengidentifikasi dan memverifikasi informasi personal. Untuk itu perlu dilakukan upaya untuk dapat mengenali apakah tanda tangan tersebut asli atau palsu dengan melakukan deteksi dan verifikasi. Dalam melakukan proses deteksi digunakan langkah-langkah yang terdiri dari *preprocessing*, ekstraksi ciri geometric, dan klasifikasi dengan metode modified-K Nearest Neighbour sebagai cara untuk melakukan verifikasi tanda tangan. Proses *preprocessing* terdiri dari filtering, binarisasi, thinning, cropping, dan resize. Kemudian dilakukan proses ekstraksi cirri geometric. Sebelum melakukan ekstraksi, dilakukan zoning terhadap citra dengan 3 teknik berbeda yaitu teknik vertikal, horizontal, dan zoning 4 bagian. Setelah itu dilakukan klasifikasi untuk proses verifikasi tanda tangan. Hasilnya adalah dengan melakukan pengujian terhadap teknik zoning untuk mengetahui nilai FRR dan FAR dari masing-masing teknik tersebut. Nilai FRR terkecil yang diperoleh adalah 54% dan nilai FAR terkecil adalah 7%. Nilai tersebut didapatkan dengan menerapkan teknik zoning vertikal. Hal tersebut menunjukkan bahwa sistem memiliki kemampuan yang baik dalam melakukan proses verifikasi terhadap tanda tangan palsu. Sedangkan dalam proses verifikasi tanda tangan asli kemampuan sistem masih rendah. Maka sesuai dengan hasil yang didapatkan, untuk meningkatkan kemampuan sistem dapat dilakukan perbaikan pada proses preprocessing citra.

Kata kunci: *deteksi dan verifikasi tanda tangan, ciri geometric, modified-K Nearest Neighbour*

ABSTRACT

Attempts at securing personal information have been done in both traditional and biometric ways. And among the various ways to protect information, signatures are the most widely used in identifying and verifying personal information. Therefore, efforts should be made to be able to recognize whether the signature is genuine or false by performing detection and verification. In performing the detection process used steps consisting of preprocessing, geometric extraction features, and classification with the modified-K approach method of Nearest Neighbors as a way of verifying signatures. The preprocessing process consists of filtering, binarization, thinning, cropping, and resizing. Then extraction process of geometric features. Before performing the extraction, zoning on the image with 3 different techniques are vertical, horizontal, and zoning 4 parts. After that is done classification for signature verification process. The result is by testing the zoning technique to determine the value of FRR and FAR of each technique. The smallest FRR value obtained is 54% and the smallest FAR value is 7%. The value is obtained by applying the vertical zoning technique. This shows that the system has a good ability in performing the verification process against fake signatures. While in the process of verification of the original signature the ability of the system is still low. So in accordance with the results obtained, to improve the ability of the system can be improved on the process of preprocessing the image.

Keywords: *signature detection and verification, geometric features, modified-K Nearest Neighbor*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Keamanan informasi merupakan kemutlakan yang tidak boleh dilupakan seiring dengan perkembangan teknologi yang kian pesat. Hal tersebut dikarenakan dalam kehidupan sehari-hari banyak transaksi yang melibatkan informasi personal. Sebagai contohnya pada bidang perbankan, sistem pertahanan, birokrasi, dll yang dituntut untuk harus terus meningkatkan sistem keamanan demi menjaga kerahasiaan informasi pelanggan. Berbagai upaya dalam mengamankan informasi personal telah banyak dilakukan secara tradisional maupun *biometric*. Secara tradisional yang umum digunakan antara lain dalam bentuk *smartcard,* nomor PIN, dan *password*. Sedangkan bentuk biometric yaitu diantaranya tanda tangan, retina mata, suara dan sidik jari (Deore & Handore, 2015). Dan di antara berbagai cara untuk melindungi informasi, tanda tangan merupakan yang paling banyak digunakan dalam mengidentifikasi dan memverifikasi informasi personal (Chandra & Maheskar, 2016).

Penggunaan tanda tangan sebagai sarana legalitas memiliki banyak kelebihan. Selain karena telah diterima secara luas dalam masyarakat (Chandra & Maheskar, 2016; Widodo & Harjoko, 2015), tanda tangan adalah salah satu otentikasi biometric yang tidak mudah berubah (Deore & Handore, 2015), lebih ekonomis, dan membutuhkan media penyimpanan yang tidak terlalu besar (Lakshmi & Nayak, 2013). Untuk proses deteksi dan verifikasi, tanda tangan dapat dibedakan ke dalam 2 jenis yaitu tanda tangan *offline* dan tanda tangan *online*. Tanda tangan *offline* merupakan tanda tangan yang ditulis dengan tangan dan kemudian dipindai sehingga menjadi format digital. Sedangkan tanda tangan *online* didapatkan melalui perangkat digital seperti tablet PC dan *touchscreen* dengan pena digital. Untuk fitur-fitur yang terdapat pada masing-masing jenis tanda tangan memiliki keunikan yang berbeda. Seperti pada tanda tangan *online*, fitur diekstrak pada saat tanda tangan itu dibuat. Sebaliknya, pada tanda tangan *offline* ekstraksi fitur dapat dilakukan kapanpun (Deore & Handore, 2015). Jenis-jenis fitur yang dimiliki oleh kedua jenis tanda tangan pun berbeda karakterisktiknya. Dikarenakan menggunakan perangkat digital, maka fitur pada tanda tangan *online* meliputi kecepatan penulisan tanda tangan, tekanan pena, ketebalan goresan, dll. Pada tanda tangan *offline*, fitur berupa data 2 Dimensi yang mana dalam proses ekstraksinya memiliki tantangan tersendiri (Chandra & Maheskar, 2016; Deore & Handore, 2015).

Dalam penggunaannya sebagai sarana otentikasi informasi, nyatanya tanda tangan *offline* lebih banyak digunakan secara luas. Banyak dokumen-dokumen resmi yang dalam proses verifikasinya masih ditanda tangani secara manual setiap hari (Widodo & Harjoko, 2015).Tanda tangan *offline* lebih fleksibel dibanding *online* karena tidak memerlukan sistem masukan tertentu (Alvarez, et al., 2016). Peningkatan nilai efisiensi dalam deteksi dan verifikasi tanda tangan

terus diupayakan untuk pengembangan penelitian di bidang ini (Angadi, et al., 2014). Upaya untuk meningkatkan nilai akurasi dengan menggunakan pendekatan ekstraksi ciri pun terus dilakukan. Subhash Chandra (2016) mencoba untuk terus mengembangkan penelitian dengan menerapkan ciri dan kombinasi metode verifikasi yang berbeda. Algoritma *Backpropagation* dan ciri *geometric* dipilih untuk dapat meningktkan nilai akurasi deteksi dan verifikasi tanda tangan *offline*. Penelitian tersebut berhasil dilakukan dengan pencapaian nilai akurasi sebesar 89,24% (Chandra & Maheskar, 2016).

Penelitian lain dikembangkan dari segi metode untuk melakukan klasifikasi. Seperti yang dilakukan oleh Saniya Ansari (2015), melakukan klasifikasi dengan metode *K Nearest Negihbour* berdasarkan 4 ciri yang salah satunya adalah ciri *geometric*, berhasil mendapatkan akurasi sebesar 79,1% (Ansari & Sutar, 2015). Pengolahan citra dengan metode klasfikasi *modified-K Nearest Neighbour* terus dilakukan. Pada tahun yang sama, Salouan melakukan pengenalan terhadap karakter Yunani dengan menggunakan metode K *Nearest Neighbour* dan menghasilkan akurasi yang cukup tinggi. Nilai akurasi yang didapatkan adalah 70-90% pada pengenalan tiap karakter (Salouan, et al., 2015). Hal tersebut menunjukkan bahwa verifikasi dengan menggunakan metode modified-*K Nearest Neighbour* berhasil dilakukan dan menghasilkan akurasi yang tinggi. Sedangkan ciri *geometric* merupakan ciri global yang berupa ukuran dan bentuk dari sebuah tanda tangan (Chandra & Maheskar, 2016). Pada penelitian lain yang dilakukan oleh Kowsalya dan Periyasamy, 2016 yang berjudul *Handwritten Tamil Character Recognition Using Geometric Feature Extraction Approach* menghasilkan nilai akurasi sebesar 94% dalam mengenali kata dalam tulisan tangan dengan jumlah masukan sebanyak 100 kata (Kowsalya & Periyasamy, 2016). Fitur *geometric* yang digunakan dalam penelitian tersebut yaitu *zoning*, *starters*, *intersection and minor starters*, dan *character traversal*. Di tahun yang sama, penggunaan ciri *geometric* untuk melakukan pengenalan karakter hindi yang dilakukan oleh Neha Assiwal dan Dr. Neetu Sharma berhasil dilakukan. Ciri yang digunakan antara lain *zoning*, *euler number*, regional *area*, dan *eccentricity* (Assiwal & Sharma, 2016). Sedangkan pada penelitian ini ciri *geometric* yang digunakan adalah standar deviasi, *skewness, kurtosis*, *center of gravity, pixels density*, dan *eccentricity.*

Berdasarkan ulasan di atas penulis mengusulkan tentang **Penerapan Ciri *Geometric* Pada Deteksi dan Verifikasi Tanda tangan *Offline***. Harapannya sistem ini dapat membantu proses deteksi dan verifikasi tanda tangan *offline* dengan mudah dan menghasilkan akurasi yang tinggi.

1.2 Rumusan masalah

Berdasarkan permasalahan yang diangkat pada bagian latar belakang, rumusan masalah dikhususkan pada:

1. Bagaimana deteksi dan verifikasi tanda tangan *offline* dengan menerapkan ciri *geometric* dengan menggunakan metode *modified-K Nearest Neighbor (*m-KNN)?

2. Bagaimana ekstraksi ciri *geometric* untuk deteksi dan verifikasi tanda tangan *offline*?
3. Bagaimana tingkat akurasi penerapan ciri *geometric* pada deteksi dan verifikasi tanda tangan *offline*?

1.3 Tujuan

Adapun tujuan umum yang dapat diperoleh dari penelitian ini adalah melakukan deteksi dan verifikasi tanda tangan *offline*.

Adapun tujuan khusus yang dapat diperoleh dari penelitian ini adalah:

1. Mengetahui cara melakukan deteksi dan verifikasi tanda tangan *offline* dengan menerapkan ciri *geometric*
2. Mengetahui cara melakukan ekstraksi ciri *geometric* untuk deteksi dan verifikasi tanda tangan *offline*
3. Mengukur tingkat akurasi penerapan ciri *geometric* pada deteksi dan verifikasi tanda tangan *offline*

1.4 Manfaat

Manfaat yang dapat diperoleh dari penelitian ini adalah:

1. Bagi penulis, sebagai penerapan ilmu yang telah didapatkan selama menjalani perkuliahan di Jurusan Informatika / Ilmu Komputer konsentrasi Komputasi Cerdas
2. Penulis mendapatkan pemahaman lebih lanjut dalam proses deteksi dan verifikasi tanda tangan *offline*
3. Penulis mendapat pemahaman lebih lanjut tentang penerapan ciri *geometric* untuk melakukan deteksi dan verifikasi tanda tangan *offline*
4. Dapat diketahui tingkat akurasi yang dihasilkan dalam penerapan ciri *geometric* pada deteksi dan verifikasi tanda tangan *offline*
5. Sebagai masukan bagi proses pengenalan ciri biometric tanda tangan

1.5 Batasan masalah

Agar permasalahan yang dirumuskan lebih terfokus, maka penelitian ini dibatasi dalam hal:

1. Tanda tangan *offline* yang digunakan adalah data yang bersumber dari penelitian Sistem Verifikasi Tanda Tangan Offline Berdasar Ciri Histogram of Oriented Gradient (HOG) dan Histogram of Curvature (HoC) (Widodo & Harjoko, 2015)
2. Ciri *geometric* yang digunakan adalah standar deviasi, *skewness, kurtosis*, *center of gravity, pixels density*, dan *eccentricity.*
3. Implementasi menggunakan bahasa pemrograman Python

4. Pengukuran akurasi menggunakan metode *False Acceptance Rate* (*FAR*) dan *False Rejection Rate* (*FRR*)
5. Metode yang digunakan untuk melakukan verifikasi adalah *modified-K Nearest Neighbor (*m-KNN)

1.6 Sistematika pembahasan

Sistematika penulisan pada skripsi ini adalah:

BAB I PENDAHULUAN

Menguraikan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika pembahasan.

BAB II LANDASAN KEPUSTAKAAN

Menguraikan tentang dasar teori dan referensi yang mendasari penelitian.

BAB III METODE PENELITIAN

Menguraikan tentang metode dan langkah kerja yang dilakukan dalam penulisan proposal skripsi ini.

BAB IV PERANCANGAN SISTEM

Menjelaskan langkah-langkah deteksi dan verifikasi tanda tangan, perancangan antarmuka, dan perhitungan manual dengan berdasarkan ciri *geometric*.

BAB V IMPLEMENTASI

Mengimplementasi hasil rancangan sehingga dapat dilakukan deteksi dan verifikasi tanda tangan *offline* dengan metode *modified-K Nearest Neighbour (m-KNN).*

BAB VI PENGUJIAN DAN ANALISIS

Menguraikan hasil pengujian dan analisis terhadap hasil deteksi dan verifikasi tanda tangan offline berdasarkan ciri *geometric* menggunakan metode *modified-K Nearest Neighbour (m-KNN).*

BAB VII PENUTUP

Membuat kesimpulan dari keseluruhan uraian bab-bab sebelumnya, serta saran-saran dari hasil yang diperoleh, yang diharapkan dapat bermanfaat dalam pengembangan selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Penelitian tentang tanda tangan *offline* telah banyak dilakukan sebelumnya. Hal ini menyangkut teknik ekstraksi ciri dan juga metode verifikasi sehingga dapat menghasilkan nilai akurasi yang tinggi. Berbagai cara untuk melakukan ekstraksi ciri telah coba diimplementasikan yaitu diantaranya ciri *geometric*, *maximum horizontal and vertical histogram, center of mass, aspect ratio,* dan *miscellaneous*.

Mendeteksi dan memverifikasi tanda tangan *offline* dengan mengkombinasikan ciri *geometric* dan metode *backproopagation* telah dilakukan. Penelitian dilakukan dengan langkah-langkah yaitu, *image preprocessing*, *feature extraction* dan *recognition*. Pada proses *image preprocessing*, citra tanda tangan diberikan perlakuan khusus yang bertujuan untuk mempersiapkan citra tersebut menuju tahapan selanjutnya. Proses ini penting dilakukan karena dapat meningkatkan kualitas citra dengan cara mengurangi *noise*, membuat tiap-tiap citra memiliki kesamaan sehingga mudah untuk dilakukan ekstraksi fitur. Tahapan selanjutnya adalah ekstraksi fitur, yang mana pada penelitian ini menggunakan 4 ciri *geometric* yaitu: *eccentricity*, *skewness*, *kurtosis*, dan *orientation*. Dengan kombinasi tersebut dihasilkan nilai akurasi mencapai 78,8% (Odeh & Khalil, 2011).

Upaya untuk meningkatkan nilai akurasi terus dilakukan. Pada tahun 2012, Ashwini Pansare dan Shalini Bhatia melakukan penelitian dengan menggunakan *maximum horizontal and vertical histogram, center of mass,* dan *aspect ratio* sebagai cara untuk ekstraksi ciri tanda tangan *offline*. Pada penelitian ini terjadi peningkatan nilai akurasi yaitu 82,66%. Agar tanda tangan dapat diverifikasi, dilakukan pelatihan dengan metode *backpropagation* kepada 300 data (Pansare & Bhatia, 2012).

Penelitian tanda tangan *offline* yang berfokus pada ranah ekstraksi ciri menarik untuk dilakukan. Suhail M. Odeh dan Manal Khalil mencoba melakukan penelitian dengan menggunakan ciri *geometric* tanda tangan *offline*. Hasilnya menunjukkan akurasi yang cukup baik dengan nilai 78.8% (Odeh & Khalil, 2011). Namun penelitian tidak berhenti pada eksplorasi ciri melainkan juga pada metode yang digunakan dalam verifikasi. Seperti penelitian yang dilakukan oleh Saniya Ansari dan Udayasingh Sutar pada tahun 2015, menggunakan *K Nearest Neighbour* untuk proses klasifikasi tulisan tangan. Peneltian tersebut berhasil mendapatkan nilai akurasi sebesar 79% (Ansari & Sutar, 2015). Pada tahun yang sama, dilakukan penelitian dengan menggunakan modifikasi pada metode *K Nearest Neighbour* untuk melakukan pengenalan karakter Yunani. Akurasi yang didapatkan terbilang tinggi yaitu 70-90% untuk pengenalan tiap karakter (Salouan, et al., 2015).

Berdasarkan penelitian yang sudah ada, maka dapat dilakukan perbandingan sesuai dengan Tabel 2.1.

Tabel 2.1 Kajian Pustaka

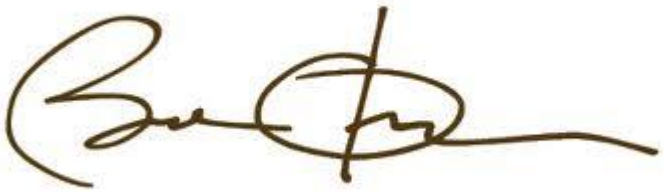
No	Judul	Objek	Ciri	Mekanisme Pengenalan	Hasil
1	Offline Signature Verification and Recognition: Neural Network Approach (Odeh & Khalil, 2011)	Tanda tangan *offline*	*Geometric feature: eccentricity, skewness, kurtosis, orientation*	- *Preprocessing* : segmentasi, binarisasi, *thinning* - *Feature extraction* - *Recognition* dengan metode *backpropagation*	Akurasi 78,8%
2	Handwritten Signature Verification using Neural Network (Pansare & Bhatia, 2012)	Tanda tangan *offline*	*maximum horizontal and vertical histogram, center of mass,* dan *aspect ratio*	- *Preprocessing* : *binarization, resizing, thinning, bounding box* - *Feature extraction* - *Training*	*FAR* 14,66%, *FRR* 20%, Akurasi 82,66%
3	Offline Signature Verification based on *Geometric* Feature Extraction using Artificial Neural Network (Chandra & Maheskar, 2016)	Tanda tangan *offline*	*Geometric feature: area, center of gravity, standard deviation, skewness, kurtosis, even pixels*	- *Data acquisition* - *Preprocessing: RGB to grayscale conversion, binarization, cropping* - *Feature extraction* - *Training and verification*	*FAR* 10.62%, *FRR* 10.91%, Akurasi 89,24%
4	Devanagari Handwritten Character Recognition using Hybrid Features Extraction and Feed Forward Neural	Devanagari Handwritten Character	*Geometric features, regional features, gradient features, dan distance*	- *Preprocessing: Gaussian filter,* segmentasi dengan binarisasi, *canny edge detection,* dilasi,	Akurasi SVM 86.34%, akurasi KNN 79.1%, akurasi FFNN 91.3%

	Network Classifier (FFNN) (Ansari & Sutar, 2015)		*transform features*	erosi, *bounding box* - *Feature extraction* - *Classification:* SVM, KNN, FFNN	
5	Printed Noisy Greek Characters Recognition Using Hidden Markov Model, Kohonen Network, K Nearest Neighbours and Fuzzy Logic (Salouan, et al., 2015)	Noisy Greek Characters	*Krawtchouk invariant moment*	- *Preprocessing: filtering,* binarisasi - *Feature extraction* - *Classification: Hidden Markov Model, Kohonen Network, modified K Nearest Neighbour*	Akurasi 70-90%

2.2 Dasar Teori

2.2.1 Tanda Tangan

Tanda tangan secara tradisional merupakan cara untuk memberikan bukti atas tanda dan tindakan seseorang, terhadap sebuah dokumen atau niat yang menunjukkan sebuah identitas dan maksud. Tanda tangan (bahasa Latin: *signare*, menandatangani) adalah representasi identitas yang dapat berupa tulisan tangan tentang nama seseorang, nama panggilan maupun inisial. Tanda tangan dibuat agar mudah untuk mengidentifikasi seseorang (Wikipedia, 2017).



Gambar 2.1 Gambar Tanda Tangan *Offline*

Berdasarkan cara pembuatan, tanda tangan dibedakan menjadi 2 yaitu tanda tangan *online* dan *offline.* Tanda tangan *online* adalah tanda tangan yang pembuatannya menggunakan perangkat digital seperti tablet PC dan *touchscreen* dengan pena digital. Sedangkan tanda tangan *offline* dibuat dengan tulisan tangan yang kemudian dipindai sehingga menjadi format digital. Fitur yang terdapat pada kedua jenis tanda tangan pun berbeda. Pada tanda tangan *online* fitur-fitur diekstrak ketika tanda tangan tersebut dibuat. Sebaliknya, pada tanda tangan *offline* fiturnya merupakan fitur yang sudah melekat pada citra tanda tangan dan dapat diekstrak kapanpun (Deore & Handore, 2015). Hal tersebut mengakibatkan tiap-tiap fitur pada masing-masing jenis tanda tangan memiliki karakteristik dan keunikan yang berbeda. Fitur-fitur yang umum terdapat pada tanda tangan *online* adalah kecepatan penulisan tanda tangan, tekanan pena, ketebalan goresan, dll. Sedangkan pada tanda tangan *offline*, fitur berupa data 2 dimensi yang dalam proses ekstraksinya memiliki tantangan tersendiri (Chandra & Maheskar, 2016; Deore & Handore, 2015).

2.2.2 Citra Digital

Citra digital adalah larik (*array*) yang pada deretan bit tertentu yang merepresentasikan nilai-nilai yang kompleks. Citra digital berupa matrik fungsi f(x,y) dengan f merupakan amplitude di titik koordinat (x,y). Secara keseluruhan nilai x,y dan amplitude f bernilai *finite* dan diskrit, maka suatu citra disebut citra digital. Representasi citra digital dalam bentuk matriks dapat dilihat pada Gambar 2.1 (Darma, 2010).

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \ldots & f(0,N-1) \\ f(1,0) & f(1,1) & \ldots & f(1,N-1) \\ \ldots & \ldots & \ldots & \ldots \\ f(M-1,0) & f(M-1,1) & \ldots & f(M-1,N-1) \end{bmatrix}$$

Gambar 2.2 Matriks Citra Digital

2.2.3 Pengolahan Citra Digital

Pengolahan citra digital adalah metode untuk pemrosesan citra dengan cara memanipulasi sesuai dengan keinginan untuk mendapatkan informasi tertentu. Terdapat 2 cara untuk melakukan pengolahan citra yaitu dengan teknik ruang 2 dimensi (spasial dan frekuensi) dan teknik tingkat keabuan. Teknik ruang 2 dimensi bertujuan untuk menentukan resolusi citra sedangkan teknik tingkat keabuan digunakan untuk mengetahui resolusi intensitas citra yang dihasilkan (Darma, 2010). Tujuan utama dilakukannya pengolahan citra digital adalah untuk memperbaiki kualitas citra dan mengekstrak ciri dominan yang merepresentasikan suatu citra (Noercholis, et al., 2013). Proses pengolahan citra digital pada penelitian ini terbagi menjadi 3 tahap yaitu tahap *pre-processing*, *feature extraction*, dan *recognition*.

2.2.3.1 Pre-processing

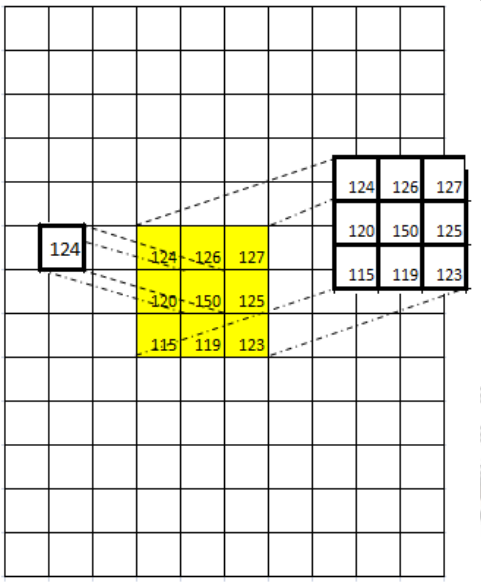
Proses *image preprocessing* adalah tahapan yang dilakukan untuk mempersiapkan citra menuju tahapan selajutnya. Pada tahapan ini berisi teknik untuk memanipulasi dan memodifikasi citra. *Pre-processing* penting dilakukan karena pada tahap ini citra akan diseragamkan sehingga memiliki kesamaan yang baku. Selain itu, pada proses ini juga dilakukan reduksi *noise* yang akan meningkatkan kualitas citra (Odeh & Khalil, 2011). Pada penelitian ini, tahapan pre-processing dilakukan melalui 5 tahapan yaitu *filtering*, binarisasi, *thinning*, *cropping,* dan *resize*.

1. *Filtering*

   Sebuah citra yang tersusun atas piksel-piksel terkadang mengandung piksel-piksel yang tidak merepresentasikan citra tersebut. Hal itu sering disebut sebagai *noise*. Maka untuk mendapatkan piksel yang representatif diperlukan sebuah teknik untuk mengurang *noise* tersebut. *Filtering* disebut juga *conditioning* atau *enhancement* berawal dari gagasan tentang keinginan untuk mengekstrak struktur dari sebuah citra, namun didalam nya terkandung variansi-variansi yang tidak diinginkan (Shapiro & Stockman, 2001).

   Median filter merupakan salah satu teknik *filtering* yang menerapkan nilai median piksel tetangga untuk menggantikan nilai piksel yang dimaksud. Median merupakan nilai tengah dari sekumpulan data dalam suatu citra. Untuk dapat mengetahui median dari nilai piksel tetangga, maka harus dilakukan pengurutan terlebih dahulu terhadap nilai-nilai tersebut. Nilai tengah dari hasil pengurutan disebut sebagai median yang nanti digunakan

untuk menggantikan nilai piksel yan dimaksud. Ilustrasi median filter dengan matriks ukuran 3x3 sesuai dengan Gambar 2.3.



Nilai piksel tetangga:

115, 119, 120, 123, 124, 125, 126, 127, 150

Nilai median:

124

Gambar 2.3 Median Filter

2. Binarisasi

Binarisasi adalah proses untuk merubah citra menjadi citra biner. Citra biner adalah citra yang hanya memiliki dua nilai yaitu 0 dan 1. Metode yang digunakan untuk melakukan binarisasi adalah *thresholding Otsu* (Huang, et al., 2005). Metode Otsu adalah metode yang menentukan batas ambang citra *gray level* dengan melakukan analisis diskriminan. Analisis diskriminan bekerja dengan cara menentukan variabel yang dapat membedakan *foreground* dengan *background* (Putra, 2004).

Intensitas dari citra *grayscale* dilambangkan dengan *L* (1, 2, …, L) sedangkan ambang batas atau *threshold* dilambangkan dengan *t*. Probabilitas piksel *grayscale* secara matematis dimodelkan sesuai Persamaan 2.1.

$$p(i) = n_i/N \quad (2.1)$$

Dengan:

$p(i)$ = Probabilitas piksel ke-i

n_i = Jumlah piksel pada level ke-i

N = Total jumlah piksel pada citra

Nilai intensitas dibagi menjadi dua kelas yaitu *q1* dan *q2*. Daerah yang termasuk ke dalam *q1* adalah piksel dengan level (1, 2, …, t) sedangkan daerah *q2* meliputi piksel level (t+1, t+2, …, L). Persamaan 2.2 merepresentasikan perhitungan *q1* dan *q2*. Setiap *q1* dan *q2* kemudian dihitung nilai rata-ratanya sesuai dengan Persamaan 2.3.

$$q_1(t) = \sum_{i=1}^{t} p(i) \qquad q_2(t) = \sum_{i=t+1}^{L} p(i) \quad (2.2)$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{i\,p(i)}{q_1(t)} \qquad \mu_2(t) = \sum_{i=t+1}^{L} \frac{i\,p(i)}{q_2(t)} \quad (2.3)$$

Masing-masing bagian *q1* dan *q2* kemudian dihitung nilai variannya. Secara matematis perhitungan varian sesuai Persamaan 2.4.

$$\sigma_1^2(t) = \sum_{i=1}^{t}[i - \mu_1(t)]^2 \frac{p(i)}{q_1(t)}$$
$$\sigma_2^2(t) = \sum_{i=t+1}^{L}[i - \mu_2(t)]^2 \frac{p(i)}{q_2(t)} \quad (2.4)$$

Kemudian dilakukan perhitungan untuk menentukan variansi dalam kelas (*within-class variance*). Model matematis perhitungan *within-class variance* sesuai Persamaan 2.5.

$$\sigma^2 w(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \quad (2.5)$$

Untuk menentukan ambang batas atau *threshold t* yang tepat diperoleh berdasarkan nilai *within-class variance*. Nilai tersebut dicari yang paling minimum sesuai dengan Persamaan 2.6 (Greensted, 2010) (Gonzalez & Woods, 2008).

$$t = \min(\sigma^2 w(t)) \quad (2.6)$$

3. *Thinning*

Pada proses ini merubah citra menjadi lebih tipis. Proses *thinning* ini disebut sebagai erosi. *Thinning* dilakukan untuk mengurangi jumlah piksel pada garis-garis yang membentuk citra sehingga menjadi 1 piksel saja. Hal ini mengakibatkan citra memiliki nilai ketebalan yang sama. Metode yang digunakan untuk melakukan proses *thinning* adalah Zhang Suen (Widodo & Harjoko, 2015). Algoritma Zhang Suen merupakan algoritma yang memiliki 2 sub iterasi. Pada sub iterasi yang pertama, piksel A(i,j) akan dihapus jika memenuhi kondisi (Soni & Kaur, 2016):

1. Konektivitas berjumlah 1
2. Memiliki minimum 2 dan maksimum 6 tetangga berwarna hitam
3. Minimum 1 diantara piksel A(i, j+1), A(i-1, j), dan A(i, j-1) berwarna putih
4. Minimum 1 diantara piksel A(i-1, j), A(i+1, j), dan A(i, j-1) berwarna putih

Pada sub iterasi kedua, kondisi ke 3 dan ke 4 berubah.

1. Konektivitas berjumlah 1
2. Memiliki minimum 2 dan maksimum 6 tetangga berwarna hitam
3. Minimum 1 diantara piksel A(i-1, j), A(i, j+1), dan A(i+1, j) berwarna putih
4. Minimum 1 diantara piksel A(i, j+1), A(i+1, j), dan A(i, j-1) berwarna putih

Proses dijalankan dengan menggunakan matriks ber-ordo 3 untuk melakukan perhitungan yang menentukan apakah suatu piksel pada citra dapat bertahan atau tidak. Jika semua kondisi terpenuhi oleh suatu piksel, maka piksel tersebut akan dihapus. Dan iterasi berhenti jika sudah tidak ada lagi piksel yang dianalisis.

4. Cropping

Proses ini bertujuan untuk memotong citra sehingga area yang akan digunakan untuk tahap selanjutnya hanya berisi citra tanda tangan saja. *Cropping* dilakukan dengan cara mencari nilai koordinat maksimum dan

minimum suatu piksel. Kemudian dilakukan pengurangan koordinat maksimum dengan koordinat minimum. Koordinat minimum berada pada piksel dengan koordinat (0, 0) atau kiri atas (AlTarawneh, et al., 2010).

5. *Resizing Citra*

   Resize dilakukan untuk meminimasilisir keberagaman ukuran antara citra yang satu dengan citra yang lain. Dengan ukuran yang sama, maka akan mempermudah proses sehingga hasilnya tidak memiliki perbedaan yang signifikan. Resize ukuran ini dilakukan secara linear yaitu perbandingan antara tinggi dan lebar suatu citra adalah sebanding (Widodo & Harjoko, 2015).

2.2.3.2 Feature Extraction

Tahapan ekstaksi fitur merupakan langkah kedua dalam proses pengolahan citra digital. Tujuan dilakukannya proses ini adalah untuk mengetahui ciri yang akan digunakan dalam rangka mengenali tanda tangan (Odeh & Khalil, 2011). Pada penelitian ini digunakan 6 fitur yaitu standar deviasi, *skewness, kurtosis*, *center of gravity, pixels density*, dan *eccentricity*.

Sebelum dilakukan ekstraksi terhadap ciri yang sudah disebutkan di atas, citra akan dibagi menjadi beberapa zona. Proses tersebut disebut dengan *zoning* citra. *Zoning* dilakukan dengan tujuan agar citra terbagi menjadi wilayah-wilayah dimana ciri local dari masing-masing wilayah dapat diekstrak. Pada penelitian ini, *zoning* dilakukan dengan 3 jenis variasi yaitu zona vertical, zona horizontal, dan zona 4. Pada zona vertical, citra dibagi secara vertical menjadi 3 bagian dan menghasilkan cirri lokal sebanyak 18 ciri (Pansare & Bhatia, 2012). Untuk zona horizontal, pembagian citra dilakukan dengan membagi menjadi 3 secara horizontal. Seperti halnya dengan zona vertical, zona horizontal menghasilkan cirri lokal sebanyak 18 ciri. Dan yang terakhir adalah citra dibagi menjadi 4 bagian sehingga menghasilkan cirri lokal sebanyak 24 ciri (Impedovo, et al., 2012).

1. Standar deviasi

   Standar deviasi digunakan untuk mengukur sebaran data dan kedekatan data terhadap nilai rata-rata. Pada pengolahan citra digital, standar deviasi digunakan untuk menajamkan tepian citra. Secara matematis, standar deviasi dapat direpresentasikan sesuai Persamaan 2.7.

$$\boldsymbol{\sigma} = \sqrt{\sum_{i=0}^{L-1} (\boldsymbol{r_i} - \boldsymbol{\mu})^2 \boldsymbol{p}(\boldsymbol{r_i})} \quad (2.7)$$

   Dengan:

   [0, *L-1*] = Batas intensitas

   r_i = Intensitas ke-i

   $p(r_i)$ = Probabilitas intensitas ri

   μ = Rata-rata intensitas

   Nilai probabilitas intensitas ri $(p(r_i))$ dapat direpresentasikan secara matematis sesuai dengan Persamaan 2.8.

$$\boldsymbol{p(r_i) = {n_i}/{M * N}} \quad (2.8)$$

Dengan:
n_i = Jumlah piksel dengan intensitas ri
$M * N$= Ukuran Matriks dari citra

Untuk nilai rata-rata intensitas (μ) didapatkan dari Persamaan 2.9.

$$\boldsymbol{\mu = \sum_{i=0}^{L-1}(r_i)p(r_i)} \quad (2.9)$$

2. Skewness

Skewness adalah ukuran dari ketidaksimetrian suatu citra. Data set citra disebut simetri apabila bagian kanan dan kirinya bernilai sama terhadap titik pusat (NIST/SEMATECH, 2010). Pengukuran *skewness* penting dilakukan karena karakteristik kerumitan suatu tanda tangan berbeda-beda. Pada tanda tangan yang memiliki tingkat kerumitan yang cukup tinggi, liku yang dimilikinya dapat diukur tinggi maupun lebarnya. Pengukuran aspek tersebut sangatlah penting untuk proses perbandingan (Odeh & Khalil, 2011). Model matematis pengukuran *skewness* dapat dilihat pada Persamaan 2.10.

$$\boldsymbol{skewness = \frac{\sum_{i=0}^{L-1}(r_i-\mu)^3 p(r_i)}{s^3}} \quad (2.10)$$

Dengan:
μ= Rata-rata
r_i = intensitas ke-i
s = Standar deviasi
$p(r_i)$ = probababilitas ri

3. Kurtosis

Data dengan k*urtosis* yang rendah adalah data yang memiliki kecenderungan ekor-cahaya atau kurangnya *outlier*. Sebaliknya, data dengan kurtosis tinggi memiliki kecenderungan berat-ekor atau *outlier.* Berdasarkan hal tersebut, maka dapat dikatakan kurtosis adalah sebuah ukuran sedikit atau banyaknya *outlier* relatif terhadap distribusi normal (NIST/SEMATECH, 2010). Pengukuran kurtosis menitikberatkan pada puncak dari segmen suatu citra tanda tangan (Odeh & Khalil, 2011). Persamaan 2.11 menunjukkan model matematis dari *kurtosis*.

$$\boldsymbol{kurtosis = \frac{\sum_{i=0}^{L-1}(r_i-\mu)^4 p(r_i)}{s^4}} \quad (2.11)$$

Dengan:
μ= Rata-rata
r_i = intensitas ke-i
s = Standar deviasi
$p(r_i)$ = probababilitas ri

4. *Center of Gravity*

Untuk mengetahui dimanakah pusat suatu citra maka harus diketahui nilai dari c*enter of gravity* (CoG) citra tersebut. Hal ini dikarenakan CoG adalah lokasi pusat rata-rata berat suatu obyek. Persamaan 2.12 merepresentasikan perhitungan matematis CoG. Dimana xi adalah koordinat ke-i pada sumbu x dan yi adalah koordinat ke-i pada sumbu y (Khaleel, et al., 2013).

$$Xcog = \frac{\sum_{i=1}^{n} xi}{n} \qquad Ycog = \frac{\sum_{i=1}^{n} yi}{n} \tag{2.12}$$

Dengan :
Xcog = koordinat x *center of gravity*
Ycog = koordinat y *center of gravity*
xi = koordinat ke-i sumbu x citra
yi = koordinat ke-i sumbu y citra
n =banyak nya piksel obyek

5. *Pixels Density*

Pixels Density atau *energy density* merupakan fitur local yang didapatkan dengan cara menghitung jumlah piksel obyek (Sharma & Shrivastav, 2011). Secara matematis, perhitungan pixels density dapat dimodelkan sesuai dengan Persamaan 2.13 (Wai & Aung, 2013).

$$PD = \frac{A}{n} \tag{2.13}$$

Dengan :
PD = *pixels density*
A = area, luas citra
N = jumlah piksel obyek

6. *Eccentricity*

Ciri *eccentricity* adalah bagian dari ekstraksi ciri *shape descriptor*. Secara matematis perhitungan eccentricity dimodelkan sesuai dengan Persamaan 2.14 (Park, 2011).

$$E = \frac{L}{W} \tag{2.14}$$

Dengan:
E = *Eccentricity*
L = panjang citra
W = lebar citra

2.2.3.3 Recognition

Recognition atau pengenalan adalah tahapan terakhir dari penelitian deteksi dan verifikasi tanda tangan *offline*. Pada penelitian ini metode *modified-K*

Nearest Neighbour digunakan untuk melakukan pengenalan. Berikut adalah langkah yang dilakukan:

- Proses pengenalan dimulai dengan memasukkan data citra tanda tangan baru yang bukan berasal dari database
- Citra tanda tanda tersebut dibandingkan dengan citra tanda tangan yang ada pada database dengan menghitung nilai standar deviasi, *skewness, kurtosis*, *center of gravity, pixels density*, dan *eccentricity*
- Kemudian dihitung selisih nya terhadap semua kelas dan nilai yang minimum merepresentasikan kelas hasil *recognition*

2.2.4 Normalisasi Data

Normalisasi data digunakan dengan tujuan untuk membuat data memiliki rentang nilai yang sama. Dikarenakan data yang bernilai lebih besar dari yang lain bisa jadi menyebabkan data yang lebih kecil menjadi terkesampingkan (Basheer & Hajmeer, 2000). Metode untuk membuat data menjadi normal telah banyak digunakan salah satunya adalah dengan menjadikan data memiliki rentang 0 sampai 1. Namun, metode tersebut menyebabkan perlambatan pembelajaran dikarenakan kejenuhan fungsi sigmoid. Maka dilakukan normalisasi data dengan metode yang menjadikan data memiliki rentang 0.1 sampai 0.9 (Hassoun, 1995; Basheer & Hajmeer, 2000). Secara matematis, normalisasi data ditunjukkan oleh Persamaan 2.15.

$$X_i = (\lambda_2 - \lambda_1)\, x\, \frac{(z_i - z_i^{min})}{(z_i^{max} - z_i^{min})} + \lambda_1 \quad (2.15)$$

Dengan:

X_i = data normal
λ_2 = rentang tertinggi (0.9)
λ_1 = rentang terendah (0.1)
Z_i = data asli
z_i^{min} = nilai minimum data asli
z_i^{max} = nilai maksimum data asli

2.2.5 Modified-K Nearest Neighbour Algorithm

Dikenal sebagai salah satu metode yang paling sederhana dan efisien, *K Nearest Neighbour (*KNN*)* melakukan proses klasifikasi tanpa memerlukan proses pembelajaran. KNN bekerja dengan membandingkan jarak antara data latih dengan data uji (Tomar & Agarwal, 2013; Salouan, et al., 2015). Nilai yang telah didapatkan dari perhitungan jarak kemudian diurutkan berdasarkan jumlah k tetangga terdekat. Penentuan kelas dilakukan dengan *voting* jumlah kelas terbanyak yang muncul diantara k tetangga terdekat (Salouan, et al., 2015).

Meski kelas sudah dapat ditentukan dengan langkah yang sudah dijelaskan sebelumnya, namun permasalahan muncul ketika nilai k merupakan bilangan genap. Dengan menggunakan metode voting, permasalahan klasifikasi kelas tidak dapat dipecahkan karena label kelas akan selalu seimbang (Larose,

2005). Untuk itu perlu dilakukan modifikasi terhadap proses *voting* untuk penentuan kelas data uji. Sehingga sebuah algoritma baru lahir dari proses modifikasi tersebut dan disebut sebagai *modified-KNearest Neighbor* (m-KNN). Modifikasi dilakukan dengan memberikan pembobotan pada nilai jarak yang sudah dihitung. Berikut adalah langkah-langkah dalam melakukan klasifikasi dengan menggunakan metode m-KNN (Salouan, et al., 2015; Larose, 2005):

1. Melakukan perhitungan jarak

   Yang pertama dilakukan adalah menghitung jarak antara data latih dan data uji. Perhitungan ini dilakukan terhadap data uji ke semua data latih yang ada. Untuk mendapatkan nilai jarak, digunakan perhitungan dengan *Euclidean Distance* sesuai dengan Persamaan 2.16.

   $$d(x,y) = \sqrt{\sum_k (X_k - x_k)^2} \quad (2.16)$$

   Dengan :
   d(x,y) = jarak data latih dengan data uji
   X = data latih
   x = data uji
   k = indeks data ke 1,…..k

2. Menentukan k tetangga terdekat

   Untuk menentukan k tetangga terdekat, yang harus dilakukan adalah mengurutkan jarak dari yang terkecil ke yang terbesar. Kemudian dipilih jarak terdekat sebanyak k, misal k=3 maka diambil 3 jarak terdekat untuk kemudian digunakan pada proses selanjutnya.

3. Hitung pembobotan jarak

   Pembobotan jarak dilakukan pada data yang sudah didapatkan pada proses sebelumnya. Pembobotan ini digunakan untuk mengetahui bobot masing-masing jarak sehingga dapat dilakukan *voting* pada proses selanjutnya. Untuk menghitung pembobotan jarak dengan menggunakan Persamaan 2.17.

   $$W_i = \begin{cases} \frac{d(X,x_k) - d(X,x_i)}{d(X,x_k) - d(X,x_1)} & , jika\ d(X,x_k) \neq d(X,x_1) \\ 1 & , lainnya \end{cases} \quad (2.17)$$

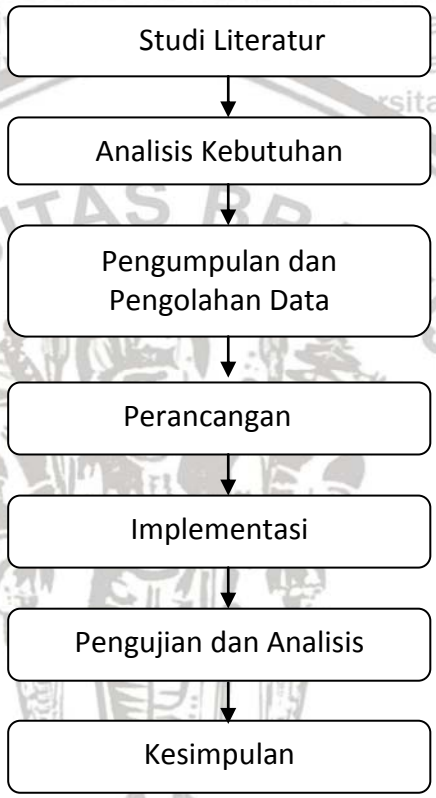
   Dengan :
   W_i = bobot jarak ke-i

4. *Voting* label kelas

   Penentuan label kelas dilakukan dengan *voting* kelas yang memiliki jumlah bobot terbanyak. Maka, kelas tersebut merupakan kelas yang merepresentasikan data uji.

BAB 3 METODOLOGI

Bab ini menjelaskan tentang metode, rancangan sistem dan langkah-langkah yang digunakan dalam melakukan penerapan ciri *geometric* pada deteksi dan verifikasi tanda tangan *offline*. Gambar 3.1 menjelaskan tentang alur penelitian skripsi yang dimulai dengan studi literatur, lalu dilakukan analisis kebutuhan, pengumpulan dan pengolahan data, kemudian dilakukan perancangan, lalu selanjutnya adalah implementasi, dan hasilnya dilakukan pengujian dan analisis lalu terakhir dilakukan penarikan kesimpulan.



Gambar 3.1 Diagram Alir Metode Penelitian

3.1 Studi Literatur

Studi literature menjelaskan tentang dasar teori berbagai bidang ilmu yang berhubungan dengan penerapan ciri *geometric* pada deteksi dan verifikasi tanda tangan *offline*. Studi ini didasarkan pada *review* jurnal nasional dan internasional, skripsi terdahulu, maupun website tentang teori:

- Tanda Tangan
- Citra Biner dan Pengolahan Citra Digital
- Metode *modified-K Nearest Neighbour* (m-KNN)

3.2 Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk mengevaluasi permasalahan-permasalahan yang terjadi dan diharapkan dapat meminimalisir kesalahan yang akan timbul. Kebutuhan tersebut adalah data citra tanda tangan *offline* yang diambil dari penelitian *Sistem Verifikasi Tanda Tangan Offline Berdasar Ciri Histogram of Oriented Gradient (HOG) dan Histogram of Curvature* (HoC) (Widodo & Harjoko, 2015). Tanda tangan *offline* yang digunakan merupakan tanda tangan orang Indonesia.

3.3 Pengumpulan dan Pengolahan Data

3.3.1 Pengumpulan Data

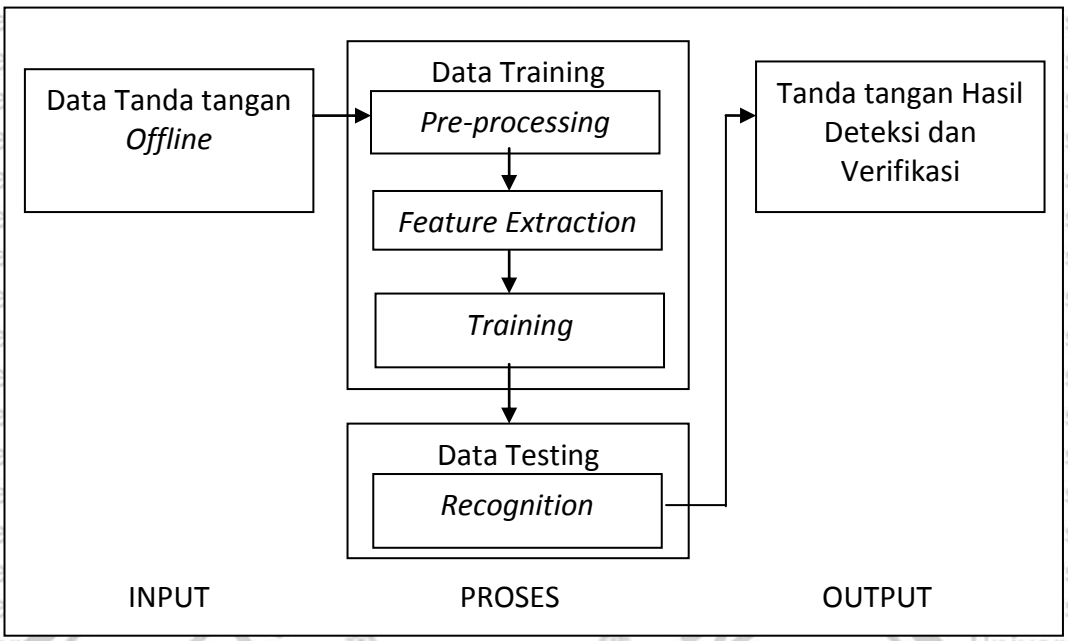
Pengumpulan data dilakukan untuk mendapatkan data-data yang sesuai dengan kebutuhan dari permasalahan. Maka dari itu teknik yang dilakukan untuk pengumpulan data pada penelitian skripsi ini adalah dengan menggunakan teknik *sampling*. Pada penelitian ini proses *sampling* bertujuan untuk mendapatkan data dengan mengambil sebagian data atau sampel yang merepresentasikan suatu populasi. Data yang dikumpulkan berupa citra tanda tangan *offline*. Data ini didapat dari sampel yang berjumlah 74 orang dengan masing-masing orang memiliki 24 tanda tangan asli dan 20 tanda tangan palsu.

3.3.2 Pengolahan Data

Proses pengolahan data pada penelitian skripsi ini dilakukan melalui beberapa tahapan yaitu *image pre-processing, feature extraction,* dan *neural network approach*. Pada tahapan *image pre-processing* data citra tanda tangan diolah dengan cara dilakukan manipulasi dan modifikasi sehingga data siap untuk memasuki tahapan selanjutnya. Hasil dari *pre-processing* kemudian diambil fitur-fitur penting yang memberikan informasi yang dapat membedakan data satu dengan data yang lain. Kemudian setelah didapatkan fiturnya dilakukan *training* untuk mengenali tanda tangan dengan menggunakan metode *modified-K Nearest Neighbour* (m-KNN).

3.4 Perancangan Sistem

Desain yang dimaksud adalah desain *prototype* untuk penerapan ciri *geometric* pada deteksi dan verifikasi tanda tangan *offline*. Desain ini bertujuan untuk memberikan gambaran tentang sistem secara keseluruhan setelah benar-benar dilakukan implementasi. Pada penelitian skripsi ini desain berupa diagram blok.



Gambar 3.2 Diagram Blok Perancangan Sistem

3.5 Implementasi

Implementasi sistem mengacu pada hasil analisis dan perancangan yang sudah didefinisikan sebelumnya. Proses implementasi sistem dilakukan dengan menggunakan JetBrains Pycharm Community Edition 2016.3 IDE. Adapun langkah-langkah dalam melakukan implementasi sistem ini antara lain:

- Pembuatan antarmuka pengguna
- Memasukkan data citra tanda tangan *offline* sebagai inputan sistem
- Penerapan ciri *geometric* untuk proses deteksi dan verifikasi tanda tangan *offline*

Adapun scenario yang dijalankan dalam sistem yaitu:

- Pada tampilan pengguna menampilkan kolom untuk citra tanda tangan, kolom nilai standar deviasi, nilai *skewness*, nilai *kurtosis*, nilai *center of gravity*, nilai *pixels density*, dan nilai *eccentricity,* kolom status, dan kolom nilai kemiripan dengan data latih
- Pengguna melakukan input citra tanda tangan sebagai data uji untuk dibandingkan dengan citra tanda tangan yang sudah dilatih
- Citra tanda tangan inputan user diolah melalui tahapan *pre-processing*, *feature extraction*, dan *recognition*. Pada tahapan *feature extraction* diambil ciri *geometric* untuk kemudian dilakukan *recognition*.

Citra tanda tangan yang sudah berhasil dilakukan *recognition* kemudian dilakukan klasifikasi apakah citra tersebut asli atau palsu dengan menggunakan metode *modified-K Nearest Neighbour* (m-KNN).

3.6 Pengujian dan Analisis

Pengujian yang dilakukan adalah pengujian terhadap hasil implementasi yang terdiri dari pengujian performa dan pengujian nilai akurasi. Pengujian dilakukan dengan tujuan untuk dapat mengetahui keluaran yang dihasilkan oleh sistem sudah sesuai dengan yang diharapkan atau tidak.

- Pengujian performa

  Pengujian performa adalah pengujian yang dilakukan untuk mengetahui performa dari proses verifikasi tanda tangan. Pengujian ini menggunakan dua metode yaitu *false rejection rate* (*FRR*) dan *false acceptance rate* (*FAR*). *False rejection rate* (*FRR*) terjadi ketika kondisi menunjukkan bahwa suatu citra tanda tangan yang asli terdeteksi sebagai tanda tangan palsu. Sedangkan *false acceptance rate* (*FAR*) digunakan ketika suatu citra tanda tangan palsu terdeteksi sebagai tanda tangan asli.

- Pengujian akurasi

  Pengujian akurasi dilakukan dengan tujuan untuk mengetahui seberapa akurat metode yang digunakan dalam memecahkan masalah. Dalam permasalahan ini akurasi ditunjukkan dengan suatu nilai tertentu dalam bentuk persen, yang mencerminkan data uji apakah sesuai dengan data latih. Akurasi dilakukan dengan menghitung nilai *AER* berdasarkan nilai *FAR* dan *FRR* sesuai Persamaan3.1 (Chandra & Maheskar, 2016; Widodo & Harjoko, 2015).

$$AER = \frac{(100-FAR)+(100-FRR)}{2} \quad (3.1)$$

Hasil pengujian performa dan akurasi kemudian dianalisis sehingga dapat diketahui apakah sistem yang dibangun sudah sesuai dengan kebutuhan yaitu dapat dilakukan deteksi dan verifikasi tanda tangan *offline* berdasarkan ciri *geometric*nya. Hasil analisis akan dijadikan pertimbangan untuk proses penarikan kesimpulan atau evaluasi untuk perbaika selanjutnya.

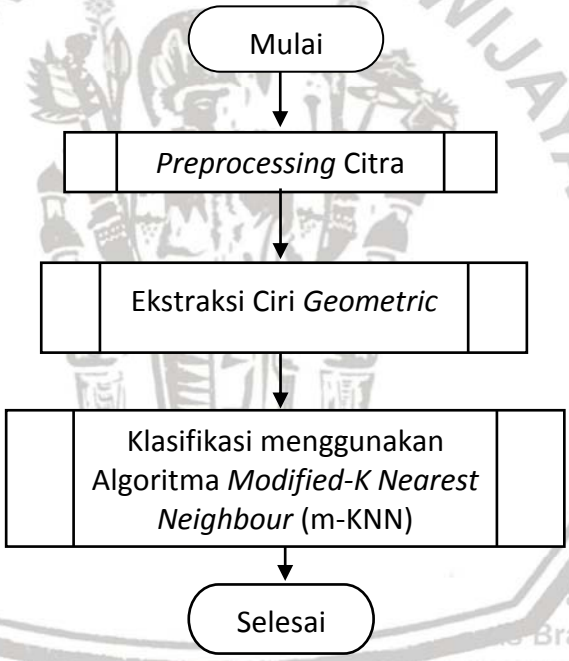
3.7 Kesimpulan

Pengambilan kesimpulan dilakukan setelah keseluruhan tahapan analisis kebutuhan, pengumpulan dan pengolahan data, implementasi, dan pengujian dan analisis telah dilalui. Kesimpulan adalah hasil dari pengujian dan analisis metode untuk mengetahui apakah rumusan masalah dapat terselesaikan atau tidak. Kemudian tahap akhir penulisan penelitian adalah saran yang bertujuan memperbaiki kesalahan dengan maksud untuk memberikan pertimbangan dan pandangan bagi proses penelitian selanjutnya.

BAB 4 PERANCANGAN

4.1 Perancangan Sistem

Perancangan sistem pengenalan tanda tangan *offline* dilakukan dengan mengimplementasikan ekstraksi ciri *geometric* dan klasifikasi menggunakan algoritma *modified-K Nearest Neighbour*(m-KNN). Sebelum dilakukan klasifikasi dengan algoritma m-KNN, citra diolah dengan cara dilakukan *preprocessing* yang kemudian berlanjut pada ekstraksi ciri *geometric*. *Preprocessing* citra merupakan tahapan awal dalam melakukan pengolahan citra digital. Pada tahapan ini, citra diberi perlakuan tertentu sehingga mudah untuk mengekstrak cirinya. Kemudian tahapan selanjutnya ada ekstraksi ciri yang bertujuan untuk mendapatkan fitur yang terdapat pada tanda tangan. Ekstraksi ciri yang digunakan untuk deteksi dan verifikasi tanda tangan adalah ciri *geometric*. Lalu tahapan terakhir adalah klasifiksi tanda tangan dengan menggunakan algoritma m-KNN. Langkah-langkah proses deteksi dan verifikasi tanda tangan sesuai dengan Gambar 4.1.

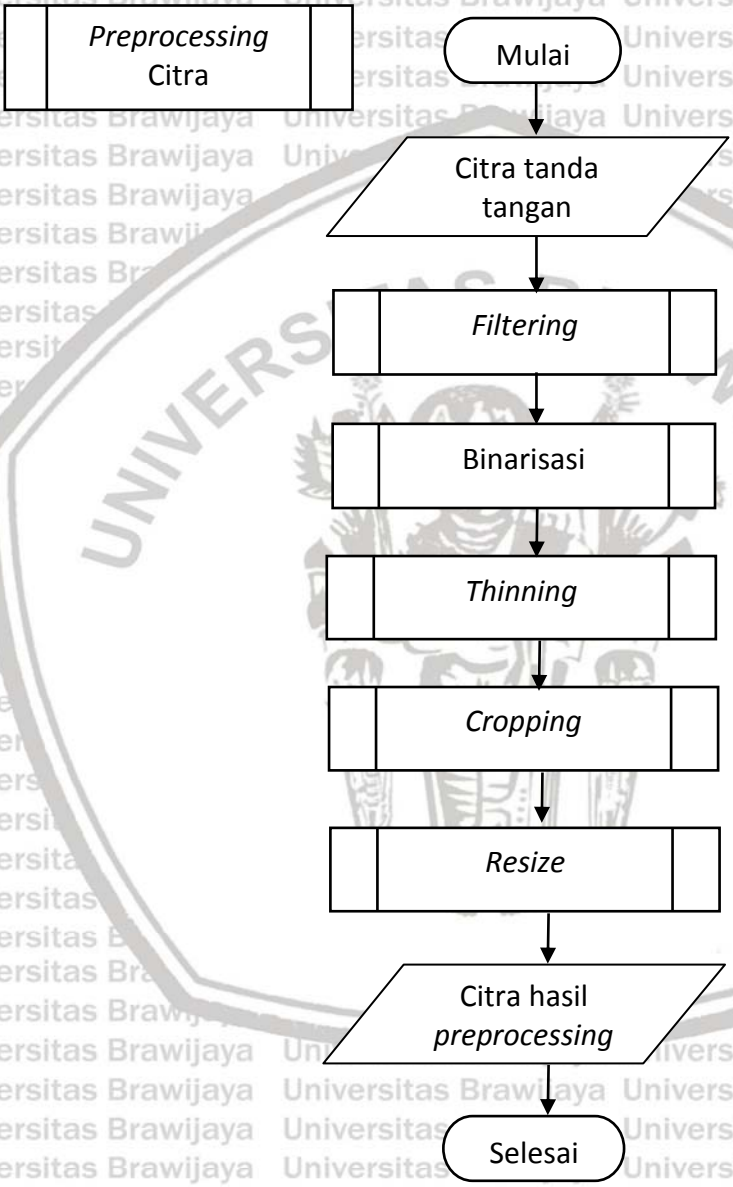


Gambar 4.1 Diagram Alir Perancangan Sistem

4.2 Preprocessing Citra

Proses preprocessing citra adalah tahapan yang dilakukan untuk mempersiapkan citra menuju tahapan ekstraksi ciri. Tahapan pertama yang dilakukan pada citra adalah dengan melakukan *grayscale* citra. Kemudian dilakukan proses binarisasi yaitu mengubah citra yang memiliki derajat keabuan tertentu menjadi citra dengan warna hitam atau putih atau citra biner. Sehingga dengan dilakukan proses tersebut dapat diketahui manakah daerah yang merupakan *background* dari suatu citra. Dalam melakukan proses binarisasi

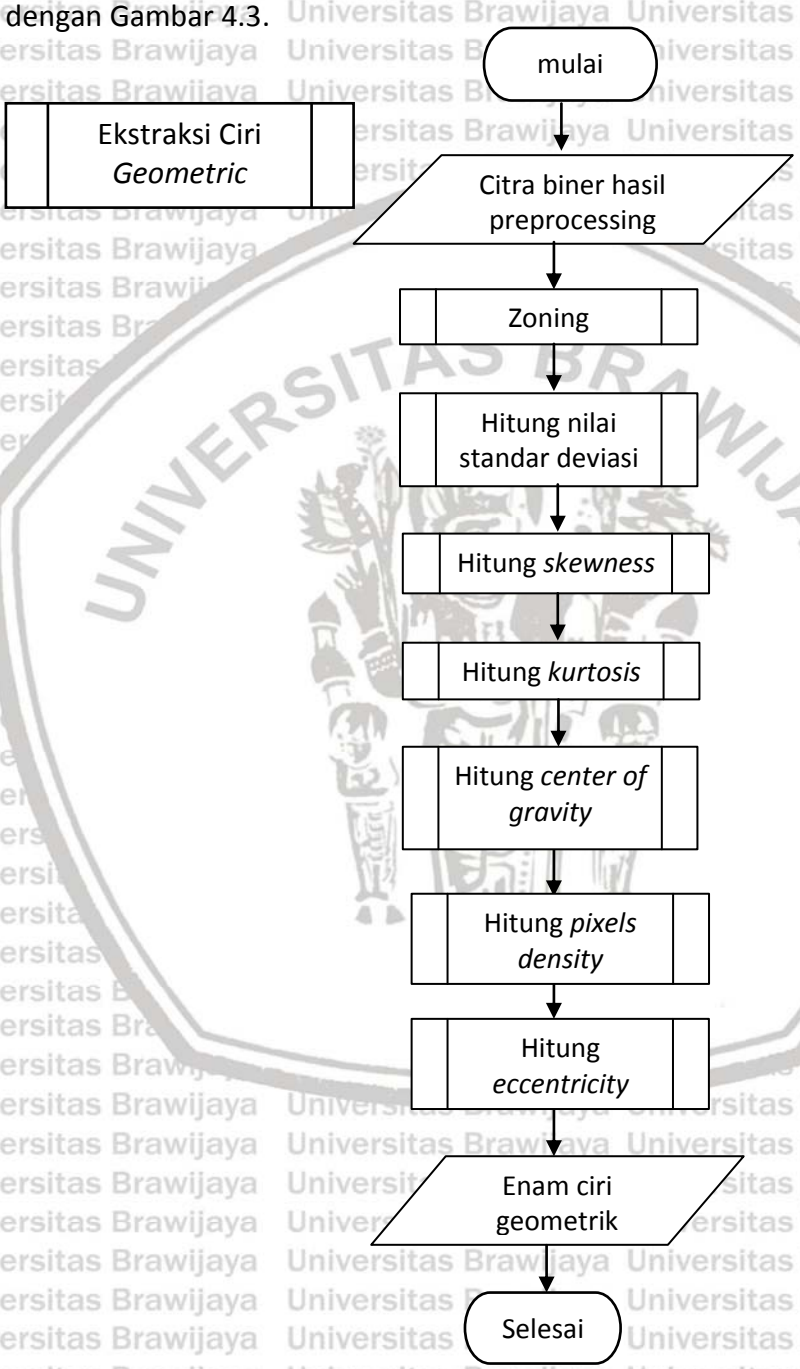
digunakan metode *thresholding otsu.* Selanjutnya setelah citra sudah dalam bentuk biner, dilakukan proses *thinning*. Proses *thinning* menggunakan algoritma *Zhang Suen* yang bertujuan untuk penipisan citra. Kemudian citra akan di normalisasi dengan melakukan *cropping* sehingga citra hanya terdiri dari daerah yang berisi piksel tanda tangan. Kemudian dilakukan *resize* citra sehingga semua citra memiliki ukuran yang sama yaitu 210x100. Proses *preprocessing* secara keseluruhan sesuai dengan Gambar 4.2.



Gambar 4.2 Diagram Alir Proses *Preprocessing*

4.3 Ekstraksi Ciri *Geometric*

Proses ekstraksi ciri *geometric* merupakan proses pengambilan ciri berdasarkan nilai standar deviasi, *skewness, kurtosis*, *center of gravity, pixels density*, dan *eccentricity* sebuah citra. Sebelum citra diambil cirinya, maka citra dibagi menjadi beberapa zona yang disebut proses *zoning*. Pada penelitian ini, citra akan dilakukan *zoning* menjadi 3 bagian. Alur ekstraksi ciri *geometric* sesuai dengan Gambar 4.3.



Gambar 4.3 Diagram Alir Proses Ekstraksi Ciri

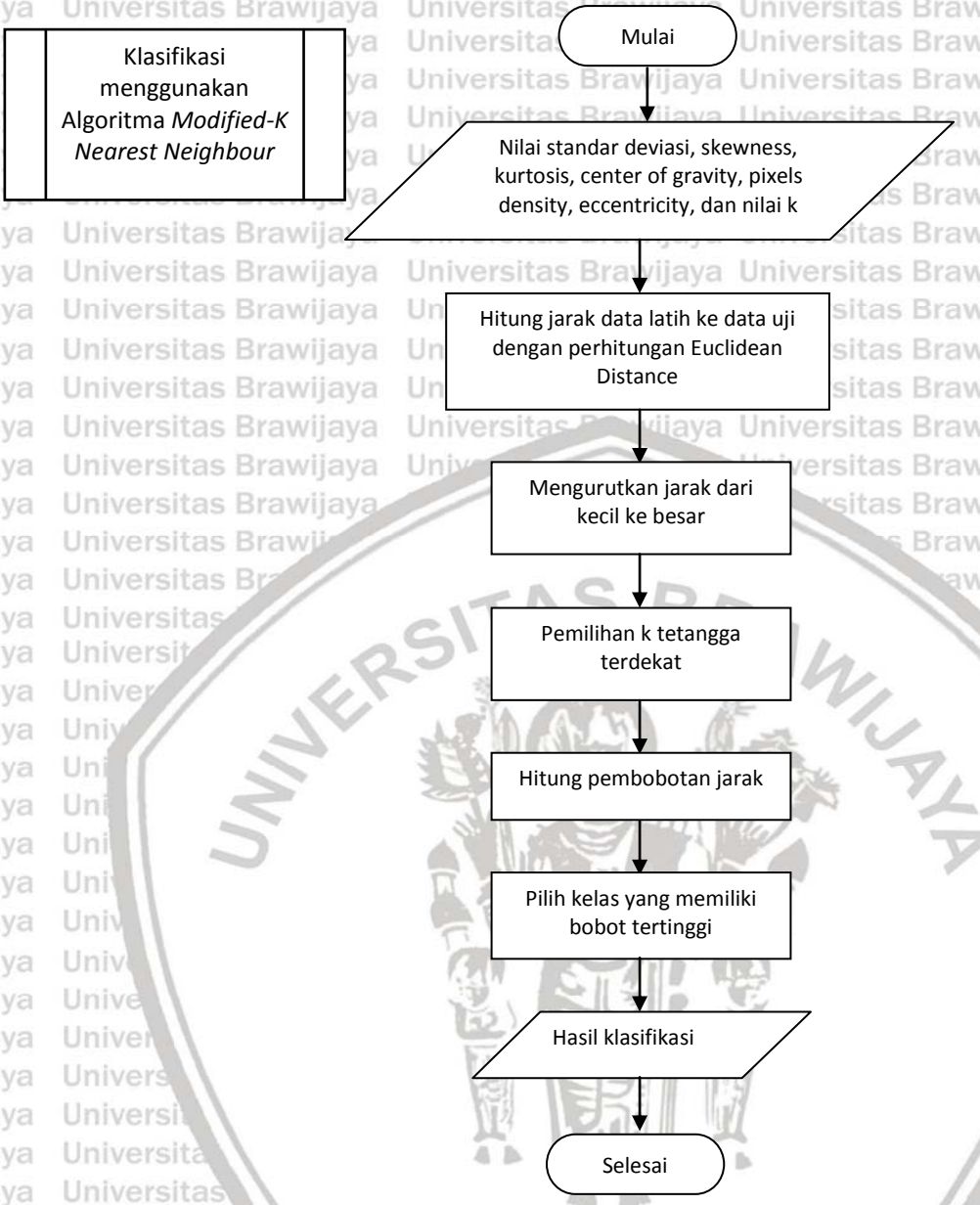
Tahapan yang dilakukan adalah sebagai berikut:

1. Melakukan *zoning* citra
   Citra dibagi menjadi zona-zona untuk diekstrak ciri local disetiap zona.
2. Hitung standar deviasi
   Tahapan pertama adalah menghitung standar deviasi sebuah citra. Standar deviasi adalah persebaran piksel citra dan seberapa dekat piksel tersebut terhadap nilai rata-rata.
3. Hitung nilai *skewness*
   Selanjutnya adalah menghitung nilai *skewness* berdasarkan nilai standar deviasi citra tersebut.
4. Hitung *kurtosis*
   Kurtosis dihasilkan dari perhitungan nilai *skewness* dan standar deviasi.
5. Hitung *center of gravity*
   Setelah itu menghitung *center of gravity* dari citra untuk mengetahui pusat citra sebagai ciri lokal
6. Hitung *pixels density.*
   Kemudian melakukan perhitungan untuk mendapatkan nilai *pixels density*.
7. Hitung *eccentricity*
   Tahapan terakhir dari ekstraksi ciri *geometric* adalah menghitung nilai *eccentricity* dari citra.

4.4 Klasifikasi *Modified-K Nearest Neighbour*

Proses klasfikasi citra dilakukan dengan menggunakan algoritma klasifikasi *modified-K Nearest Neighbour*. Ciri yang sudah diekstrak pada proses ekstraksi ciri menjadi masukan untuk algoritma ini. Alur klasifikasi dengan menggunakan metode *modified-K Nearest Neighbour* (m-KNN) sesuai Gambar 4.4 dengan langkah sebagai berikut:

1. Mulai
2. Melakukan perhitungan jarak semua data latih terhadap data uji dengan menggunakan perhitungan *Euclidean Distance*
3. Mengurutkan jarak dari kecil ke besar
4. Melakukan pemilihan k tetangga terdekat
5. Melakukan perhitungan pembobotan jarak
6. Melakukan klasifikasi dengan memilih kelas yang memiliki bobot tertinggi
7. selesai



Gambar 4.4 Diagram Alir Algoritma Modified-K Nearest Neighbour

4.5 Perhitungan Manual

Perhitungan manual merupakan langkah-langkah perhitungan data citra secara detail dengan menggunakan Ms.Excel sesuai dengan landasan teori pada Bab 2. Perhitungan manual dilakukan sebagai rancangan sebelum implementasi ke dalam sistem. Citra yang digunakan dalam proses perhitungan adalah citra pada Gambar 4.5 dengan dimensi 25x25 piksel.

Gambar 4.5 Citra tanda tangan perhitungan manual

4.5.1 Perhitungan Median Filter

Proses *filtering* pada penelitian ini menggunakan metode *median filter* dengan *mask* berupa matriks berukuran 3x3. Proses perhitungan dimulai dengan menelusuri piksel-piksel dari citra. Penelusuran dilakukan oleh mask dan ketika menemui piksel yang tidak diinginkan, kemudian piksel tetangga dari piksel *noise* diurutkan dari kecil ke besar. Nilai piksel noise kemudian digantikan dengan nilai tengah atau *median* dari nilai piksel tetangga yang sudah diurutkan sebelumnya. Proses filtering secara manual dilakukan pada Gambar 4.6 dan hasilnya sesuai dengan Tabel 4.1.

Gambar 4.6 Citra Tanda tangan perhitungan manual

4.5.2 Perhitungan Binarisasi menggunakan Metode Otsu

Berdasarkan intensitas *grayscale* Tabel 4.2, maka selanjutnya dilakukan perhitungan histogram derajat keabuan. Tabel 4.3 menunjukkan nilai histogram citra tanda tangan.

Dimana (i) adalah derajat keabuan sedangkan n(i) adalah jumlah piksel derajat keabuan ke-i. Proses binarisasi dengan metode *otsu* sesuai langkah-langkah berikut:

1. Menghitung probabilitas derajat keabuan (i) dalam histogram sesuai dengan Persamaan2.1 seperti berikut:

$$p(i) = n(i)/N ,$$

$$p(100) = \frac{1}{1024} = 0.0098$$

Perhitungan p(i) dilakukan sebanyak nilai histogram citra, pada penelitian ini p(i) dihitung hingga p(255). Hasil perhitungan p(i) ditunjukkan Tabel 4.4 dan Tabel 4.5.

2. Citra dibagi menjadi 2 daerah yaitu *foreground* dan *background* dengan *threshold* t. Masing-masing daerah dihitung nilai q(t), μ(t), $\sigma^2(t)$, dan $\sigma^2 w(t)$sesuai Persamaan 2.2, 2.3, dan 2.4 Seperti berikut:

$$q_1(t) = \sum_{i=1}^{t} p(i), \qquad q_2(t) = \sum_{i=1}^{t} p(i)$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{i\,p(i)}{q_1(t)}, \qquad \mu_2(t) = \sum_{i=1}^{t} \frac{i\,p(i)}{q_1(t)}$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{p(i)}{q_1(t)}, \qquad \sigma_2^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{p(i)}{q_1(t)}$$

Hasil perhitungan ditunjukkan pada Tabel 4.6. dan Tabel 4.7.

Tabel 4.1 Hasil Perhitungan Median Filter

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	255	255	255	255	255	255	255	255	255	252	252	255	255	255	255
2	255	255	255	255	255	255	255	255	203	153	165	167	244	255	255
3	255	255	255	255	255	255	255	243	141	249	255	179	200	255	255
4	255	255	255	255	255	255	255	193	184	255	255	188	200	255	255
5	255	255	255	255	255	255	255	157	231	255	255	180	205	255	255
6	255	255	255	255	255	255	253	149	251	255	255	147	238	255	255
7	255	255	255	255	255	255	240	153	255	255	190	173	255	255	255
8	254	248	251	255	255	255	222	169	255	183	159	252	255	255	255
9	240	169	165	168	175	196	154	124	162	193	253	255	255	255	255
10	255	255	253	243	235	239	171	184	255	255	255	255	255	255	255
11	255	255	255	255	255	255	191	206	255	255	255	255	255	255	255
12	255	255	255	255	255	255	197	219	255	255	255	255	255	255	255
13	255	255	255	255	255	255	176	216	255	255	255	254	255	255	255
14	255	255	255	255	255	255	172	215	255	255	255	255	255	255	255
15	255	255	255	255	255	255	172	215	255	255	255	255	255	255	255

Tabel 4.2 Tabel Nilai Intensitas Citra Grayscale

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
2	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
3	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
4	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
5	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
6	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
7	255	255	255	255	253	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	253	255	255	255	255
8	255	255	162	59	50	92	171	246	255	255	255	255	255	255	255	255	255	243	166	89	50	62	170	255	255
9	255	199	0	0	0	0	0	50	187	255	255	255	255	255	255	255	180	44	0	0	0	0	2	208	255
10	255	111	56	198	212	169	84	1	0	125	248	255	255	255	245	117	0	4	90	172	212	195	47	125	255
11	255	69	170	255	255	255	255	197	54	0	71	223	255	217	62	0	61	203	255	255	255	255	156	80	255
12	254	57	199	255	255	255	255	243	111	0	34	136	27	0	121	246	255	255	255	255	255	255	187	66	255
13	254	54	210	255	255	255	255	255	255	162	2	0	6	172	255	255	255	255	255	255	255	255	198	67	255
14	254	53	208	255	255	255	255	255	255	229	86	0	20	0	94	234	255	255	255	255	255	255	197	66	255
15	254	57	196	255	255	255	255	255	187	35	0	109	224	100	0	42	194	255	255	255	255	255	185	66	255
16	255	76	132	255	255	255	215	111	0	12	156	255	255	255	148	7	4	117	218	255	255	255	118	88	255
17	255	141	0	74	93	58	6	0	54	204	255	255	255	255	255	197	47	0	8	60	93	71	0	154	255
18	255	235	59	0	0	0	41	139	244	255	255	255	255	255	255	255	240	133	37	0	0	0	68	240	255
19	255	255	245	180	167	199	246	255	255	255	255	255	255	255	255	255	255	255	245	196	167	183	247	255	255
20	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
21	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
22	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
23	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
24	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
25	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255

Tabel 4.3 Histogram Citra Grayscale

I	n(i)	I	n(i)	I	n(i)	I	n(i)
0	31	61	1	132	1	203	1
1	1	62	2	133	1	204	1
2	2	66	3	136	1	208	2
4	2	67	1	139	1	210	1
6	2	68	1	141	1	212	2
7	1	69	1	148	1	215	1
8	1	71	2	154	1	217	1
12	1	74	1	156	2	218	1
20	1	76	1	162	2	223	1
27	1	80	1	166	1	224	1
34	1	84	1	167	2	229	1
35	1	86	1	169	1	234	1
37	1	88	1	170	2	235	1
41	1	89	1	171	1	240	2
42	1	90	1	172	2	243	2
44	1	92	1	180	2	244	1
47	2	93	2	183	1	245	3
50	3	94	1	185	1	246	3
53	1	100	1	187	3	247	1
54	3	109	1	194	1	248	1
56	1	111	3	195	1	250	1
57	2	117	2	196	2	251	1
58	1	118	1	197	3	253	2
59	2	121	1	198	2	254	2
60	1	125	2	199	3	255	453
						N	625

Tabel 4.4 Hasil Perhitungan Probabilitas p(i)

	(i)	n(i)	p(i)		(i)	n(i)	p(i)
1	0	31	0.0496	26	61	1	0.0016
2	1	1	0.0016	27	62	2	0.0032
3	2	2	0.0032	28	66	3	0.0048
4	4	2	0.0032	29	67	1	0.0016
5	6	2	0.0032	30	68	1	0.0016
6	7	1	0.0016	31	69	1	0.0016
7	8	1	0.0016	32	71	2	0.0032
8	12	1	0.0016	33	74	1	0.0016
9	20	1	0.0016	34	76	1	0.0016
10	27	1	0.0016	35	80	1	0.0016
11	34	1	0.0016	36	84	1	0.0016

12	35	1	0.0016	37	86	1	0.0016
13	37	1	0.0016	38	88	1	0.0016
14	41	1	0.0016	39	89	1	0.0016
15	42	1	0.0016	40	90	1	0.0016
16	44	1	0.0016	41	92	1	0.0016
17	47	2	0.0032	42	93	2	0.0032
18	50	3	0.0048	43	94	1	0.0016
19	53	1	0.0016	44	100	1	0.0016
20	54	3	0.0048	45	109	1	0.0016
21	56	1	0.0016	46	111	3	0.0048
22	57	2	0.0032	47	117	2	0.0032
23	58	1	0.0016	58	118	1	0.0016
24	59	2	0.0032	49	121	1	0.0016
25	60	1	0.0016	50	125	2	0.0032

Tabel 4.5 Hasil Perhitungan Probabilitas p(i)

	(i)	n(i)	p(i)		(i)	n(i)	p(i)
51	132	1	0.0016	76	203	1	0.0016
52	133	1	0.0016	77	204	1	0.0016
53	136	1	0.0016	78	208	2	0.0032
54	139	1	0.0016	79	210	1	0.0016
55	141	1	0.0016	80	212	2	0.0032
56	148	1	0.0016	81	215	1	0.0016
57	154	1	0.0016	82	217	1	0.0016
58	156	2	0.0032	83	218	1	0.0016
59	162	2	0.0032	84	223	1	0.0016
60	166	1	0.0016	85	224	1	0.0016
61	167	2	0.0032	86	229	1	0.0016
62	169	1	0.0016	87	234	1	0.0016
63	170	2	0.0032	88	235	1	0.0016
64	171	1	0.0016	89	240	2	0.0032
65	172	2	0.0032	90	243	2	0.0032
66	180	2	0.0032	91	244	1	0.0016
67	183	1	0.0016	92	245	3	0.0048
68	185	1	0.0016	93	246	3	0.0048
69	187	3	0.0048	94	247	1	0.0016
70	194	1	0.0016	95	248	1	0.0016
71	195	1	0.0016	96	250	1	0.0016

72	196	2	0.0032	97	251	1	0.0016
73	197	3	0.0048	98	253	2	0.0032
74	198	2	0.0032	99	254	2	0.0032
75	199	3	0.0048	100	255	453	0.7248

Tabel 4.6 Hasil Perhitungan q1(t), μ1(t), dan $\sigma^2 1(t)$dengan t=50

	q1	μ1(t)	$\sigma^2 1(t)$		q1	μ1(t)	$\sigma^2 1(t)$
1	0.0496	0	0	26	0.1056	23.28554	599.9283
2	0.0512	0.03125	0.029327	27	0.1088	25.10907	639.956
3	0.0544	0.148897	0.230891	28	0.1136	27.8978	701.2987
4	0.0576	0.371119	0.96249	29	0.1152	28.82836	721.5358
5	0.0608	0.686909	2.448223	30	0.1168	29.75987	741.5674
6	0.0624	0.866396	3.412867	31	0.1184	30.6923	761.3983
7	0.064	1.066396	4.614738	32	0.1216	32.56072	800.2819
8	0.0656	1.359079	7.376426	33	0.1232	33.52176	821.561
9	0.0672	1.835269	15.23256	34	0.1248	34.49612	843.6452
10	0.0688	2.463176	29.23385	35	0.1264	35.50878	868.7018
11	0.0704	3.235904	50.74362	36	0.128	36.55878	896.8352
12	0.072	4.013681	72.08033	37	0.1296	37.6205	925.7312
13	0.0736	4.818029	94.59509	38	0.1312	38.69367	955.3789
14	0.0752	5.69037	121.1221	39	0.1328	39.76596	984.5836
15	0.0768	6.56537	147.2807	40	0.1344	40.83739	1013.357
16	0.0784	7.463329	174.5242	41	0.136	41.91975	1042.863
17	0.0816	9.306466	230.2419	42	0.1392	44.05768	1097.929
18	0.0864	12.08424	310.1088	43	0.1408	45.12586	1125.073
19	0.088	13.04788	339.1301	44	0.1424	46.24945	1157.535
20	0.0928	15.84098	414.4462	45	0.144	47.46056	1199.614
21	0.0944	16.79014	440.504	46	0.1488	51.04121	1315.583
22	0.0976	18.65899	488.7019	47	0.152	53.50437	1400.461
23	0.0992	19.59447	512.4919	48	0.1536	54.73353	1442.155
24	0.1024	21.43822	556.5821	49	0.1552	55.98096	1485.738
25	0.104	22.3613	578.3771	50	0.1584	58.50621	1575.059

Tabel 4.7 Hasil Perhitungan q2(t), μ2(t), dan $\sigma^2 2(t)$ dengan t=50

	q2	μ2(t)	$\sigma^2 2(t)$		q2	μ2(t)	$\sigma^2 2(t)$
51	0.0016	132	0	76	0.064	641.0975	207230.3
52	0.0032	198.5	2145.125	77	0.0656	646.0731	211996.9
53	0.0048	243.8333	6021.134	78	0.0688	655.7476	221321.4
54	0.0064	278.5833	10892.01	79	0.0704	660.5203	225934.3
55	0.008	306.7833	16388.83	80	0.0736	669.7377	235044.1
56	0.0096	331.45	21997.82	81	0.0752	674.3122	239532.7
57	0.0112	353.45	27680.72	82	0.0768	678.833	243976.3
58	0.0144	388.1167	39653.64	83	0.0784	683.282	248394.4
59	0.0176	417.5712	51529.39	84	0.08	687.742	252714.1
60	0.0192	431.4045	57399.36	85	0.0816	692.1341	257011.1
61	0.0224	455.2617	69270.04	86	0.0832	696.538	261214.8
62	0.024	466.5284	75171.58	87	0.0848	700.9531	265328.9
63	0.0272	486.5284	86958.67	88	0.0864	705.3049	269424.9
64	0.0288	496.0284	92827.75	89	0.0896	713.8763	277444.9
65	0.032	513.2284	104471.4	90	0.0928	722.2557	285365.1
66	0.0352	529.592	115581.8	91	0.0944	726.3912	289309.2
67	0.0368	537.5485	121047.3	92	0.0992	738.2461	301081.4
68	0.0384	545.2568	126455	93	0.104	749.5999	312786.6
69	0.0432	566.0346	142418	94	0.1056	753.3424	316671.2
70	0.0448	572.9632	147547	95	0.1072	757.0439	320538.7
71	0.0464	579.6873	152649.9	96	0.1088	760.7203	324374.5
72	0.0496	592.3325	162784.1	97	0.1104	764.358	328193.9
73	0.0544	609.7148	177813.5	98	0.1136	771.4848	335766.5
74	0.0576	620.7148	187740.6	99	0.1168	778.4437	343301.8
75	0.0624	636.0225	202432.1	100	0.8416	998.0539	818804.7

3. Menghitung nilai within-class variance (WCV) sesuai Persamaan 2.5 sebagai berikut:

$$\sigma^2 w(t) = q_1(t)\,\sigma_1^2(t) + q_2(t)\,\sigma_2^2(t)$$

Maka, nilai WCV nya adalah:

$$\sigma^2 w(t) = q_1(t)\,\sigma_1^2(t) + q_2(t)\,\sigma_2^2(t)$$

$\sigma^2 w(t)$ =(0.09179688 * 262574.8) + (0.908203 * 1437417) = 1329570

4. Mencari nilai minimum dari within-class variance (WCV) dengan menggunakan Persamaan 2.6 sebagai berikut:

$$t = \min\,(\sigma^2 w(t))$$

Hasil perhitungan sesuai dengan Tabel 4.8.

Tabel 4.8 Hasil Perhitungan Minimum Within-Class Variance (WCV)

	t=50 (125)	t=98 (253)	t=99 (254)	t=100 (255)
q1 (t)	0.1584	0.272	0.2752	1
μ1(t)	58.50621	163.3003	845.3138	351.0778
$\sigma^2 1(t)$	1575.059	5779.504	501572	12559.62
q2 (t)	0.8416	0.728	0.7248	0
μ2(t)	998.0539	507.8791	255	0
$\sigma^2 2(t)$	818804.7	63666.76	8.08E-28	0
$\sigma^2 w(t)$	689355.5	47921.43	1615.158	12559.62

Berdasarkan perhitungan untuk proses binarisasi dengan menggunakan *Otsu Thresholding* didapatkan nilai *threshold* yaitu piksel 254. Nilai threshold tersebut digunakan untuk merubah citra *grayscale* menjadi citra biner seperti pada Tabel 4.9.

4.5.3 Perhitungan Thinning menggunakan Algoritma Zhang Suen

Proses *thinning* dilakukan dengan menggunakan metode Zhang Suen yang bertujuan agar citra hanya mengandung 1 piksel penyusun saja. Hasil *thinning* citra ditunjukkan oleh Tabel 4.10.

4.5.4 Proses Cropping Citra

Cropping citra dilakukan agar citra hanya berisi tanda tangan saja. Proses ini diawali dengan mencari nilai koordinat maksimum dan minimum yang dimiliki oleh sebuah citra. Kemudian dicari manakah yang termasuk koordinat dari piksel penyusun obyek, bukan *background*. Lalu koordinat citra dikurangi dengan koordinat obyek tersebut. Piksel yang memiliki koordinat sebelum koordinat obyek akan dihapus. Hasil cropping citra ditunjukkan oleh Tabel 4.11.

Tabel 4.9 Gambar Citra Biner Hasil Perhitungan Otsu Thresholding

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
8	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	1
9	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
10	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1
11	1	0	0	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0	0	1
12	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1
13	1	0	0	1	1	1	1	1	1	1	0	0	0	6	0	1	1	1	1	1	1	1	0	0	1
14	1	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	1
15	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1
16	1	0	0	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0	0	1
17	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1
18	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
19	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1
20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
21	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
23	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
25	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

4.5.5 Proses Resize Citra

Proses *resize* citra membuat semua citra memiliki dimensi ukuran yang sama. Dilakukan perhitungan untuk resize citra dengan ukuran 21x10. Hasil resize citra ditunjukkan oleh Tabel 4.12.

4.5.6 Perhitungan Zoning Citra

Sebelum dilakukan ekstraksi ciri, citra dibagi menjadi beberapa zona untuk kemudian dapat diekstrak ciri lokalnya. Pada peneltian ini citra dibagi menjadi 3 zona sesuai dengan Tabel 4.13, Tabel 4.14, dan Tabel 4.15.

4.5.7 Perhitungan Ekstraksi Ciri *Geometric*

Dalam proses perhitungan ekstraksi ciri, ciri yang digunakan adalah standar deviasi, *skewness, kurtosis*, *center of gravity, pixels density*, dan *eccentricity*.

1. Menghitung standar deviasi

   Perhitungan standar deviasi dilakukan dengan menghitung beberapa nilai yaitu *mean* (rata-rata) dan *probabilitas* sesuai dengan Persamaan 2.8 dan Persamaan 2.9.

   $$\boldsymbol{p(r_i) = {n_i}/{M * N}}$$

   $$\boldsymbol{\mu = \sum_{i=0}^{L-1}(r_i)p(r_i)}$$

   Setelah mendapatkan nilai di atas, perhitungan standar deviasi dilakukan seperti Persamaan 2.7 dan hasilnya ditunjukkan oleh Tabel 4.16.

   $$\boldsymbol{\sigma = \sqrt{\sum_{i=0}^{L-1}(r_i - \mu)^2 p(r_i)}}$$

2. Menghitung nilai *skewness*

   Menghitung *skewness* dilakukan seperti Persamaan 2.10 dan hasilnya sesuai dengan Tabel 4.17.

   $$\boldsymbol{skewness = \frac{\sum_{i=0}^{L-1}(r_i - \bar{\mu})^3 p(r_i)}{\sigma^3}}$$

3. Menghitung *kurtosis*

   Menghitung *kurtosis* dilakukan seperti Persamaan 2.11 dan hasilnya sesuai dengan Tabel 4.18.

   $$\boldsymbol{kurtosis = \frac{\sum_{i=0}^{L-1}(r_i - \bar{\mu})^4 p(r_i)}{\sigma^4}}$$

Tabel 4.10 Tabel Hasil Thinning Citra

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	1
11	1	1	0	1	1	1	1	0	0	1	1	1	1	1	1	1	0	0	1	1	1	1	0	1	1
12	1	1	0	1	1	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	1	1	0	1	1
13	1	1	0	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1
14	1	1	0	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1
15	1	1	0	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	0	1	1
16	1	1	0	1	1	1	1	1	1	0	0	1	1	1	0	0	1	1	1	1	1	1	0	1	1
17	1	1	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	1	1
18	1	1	0	1	1	1	0	0	0	1	1	1	1	1	1	1	0	0	0	1	1	1	0	1	1
19	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1
20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
21	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
23	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
25	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tabel 4.11 Hasil Cropping Citra

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
2	0	1	1	1	1	0	0	1	1	1	1	1	1	1	0	0	1	1	1	1	0
3	0	1	1	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	1	1	0
4	0	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	0
5	0	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	0
6	0	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	0
7	0	1	1	1	1	1	1	0	0	1	1	1	0	0	1	1	1	1	1	1	0
8	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0
9	0	1	1	1	0	0	0	1	1	1	1	1	1	1	0	0	0	1	1	1	0
10	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0

Tabel 4.12 Hasil Resize Citra

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
2	0	1	1	1	1	0	0	1	1	1	1	1	1	1	0	0	1	1	1	1	0
3	0	1	1	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	1	1	0
4	0	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	0
5	0	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	0
6	0	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	0
7	0	1	1	1	1	1	1	0	0	1	1	1	0	0	1	1	1	1	1	1	0
8	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0
9	0	1	1	1	0	0	0	1	1	1	1	1	1	1	0	0	0	1	1	1	0
10	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0

Tabel 4.13 Zona I Citra

	1	2	3	4	5	6	7
1	0	0	0	0	0	0	1
2	0	1	1	1	1	0	0
3	0	1	1	1	1	1	0
4	0	1	1	1	1	1	1
5	0	1	1	1	1	1	1
6	0	1	1	1	1	1	1
7	0	1	1	1	1	1	1
8	0	1	1	1	1	1	0
9	0	1	1	1	0	0	0
10	0	0	0	0	0	1	1

Tabel 4.14 Zona II Citra

	8	9	10	11	12	13	14
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
3	0	0	1	1	1	0	0
4	1	0	0	0	0	0	1
5	1	0	0	0	0	0	1
6	1	0	1	1	1	0	1
7	0	0	1	1	1	0	0
8	0	1	1	1	1	1	0
9	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1

Tabel 4.15 Zona III Citra

	15	16	17	18	19	20	21
1	1	0	0	0	0	0	0
2	0	0	1	1	1	1	0
3	0	1	1	1	1	1	0
4	1	1	1	1	1	1	0
5	1	1	1	1	1	1	0
6	1	1	1	1	1	1	0
7	1	1	1	1	1	1	0
8	0	1	1	1	1	1	0
9	0	0	0	1	1	1	0
10	1	1	0	0	0	0	0

Tabel 4.16 Hasil Perhitungan Standar Deviasi

Zona	Standar deviasi
I	0.483186701
II	0.44469664
III	0.483186701

Tabel 4.17 Hasil Perhitungan Nilai Skewness

Zona	Skewness
I	-0.53218
II	-1.02799
III	-0.53218

Tabel 4.18 Hasil Perhitungan Kurtosis

Zona	Kurtosis
I	1.283217
II	2.05676
III	1.283217

4. Menghitung *Center of Gravity*

Menghitung *center of gravity* dilakukan seperti Persamaan 2.12 dan hasilnya sesuai dengan Tabel 4.19.

$$Xcog = \frac{\sum_{i=1}^{n} xi}{n} \qquad Ycog = \frac{\sum_{i=1}^{n} yi}{n}$$

Tabel 4.19 Hasil Perhitungan Center of Gravity (CoG)

Zona	CoG
I	1
II	0
III	1

5. Menghitung *Pixels Density*

Perhitungan *pixels density* sesuai dengan Persamaan 2.13 dan hasilnya ditunjukkan oleh Tabel 4.20.

$$PD = \frac{A}{n}$$

Tabel 4.20 Hasil Perhitungan *Pixels Density*

Zona	PD
I	2.692307692
II	3.684210526
III	2.692307692

6. Menghitung *Eccentricity*

Perhitungan eccentricity sesuai dengan Persamaan 2.14 dan hasilnya ditunjukkan oleh Tabel 4.21.

$$E = \frac{L}{W}$$

Tabel 4.21 Hasil Perhitungan Eccentricity

Zona	E
I	0.7
II	0.7
III	0.7

4.5.8 Perhitungan Manual Klasifikasi dengan *modified-K Nearest Neighbour* (m-KNN)

Perhitungan algoritma *modified*-K *Nearest Neighbour* (m-KNN) merupakan perhitungan untuk klasifikasi yang didasarkan pada jarak terdekat dengan tetangga sejumlah k. Inputan dari algoritma ini adalah ciri yang sudah didapatkan melalui proses ekstraksi ciri. Berikut adalah langkah-langkah yang dilakukan pada perhitungan manual klasifikasi menggunakan algoritma *modified-K Nearest Neighbour* (m-KNN). Sebelum melakukan perhitungan, data harus dinormalisasi terlebih dahulu. Data normal memiliki rentang nilai 0.1 sampai 0.9. Perhitungan manual dilakukan dengan menggunakan data latih sebanyak 10 data dan data uji sebanyak 2 data. Tabel 4.22 dan Tabel 4.23 secara berturut-turut merepresentasikan data latih dan data uji. Sedangkan data normal ditunjukkan oleh Tabel 4.24 dan Tabel 4.25.

Kemudian dilakukan perhitungan jarak data latih dengan data uji menggunakan *Euclidean Distance* sesuai dengan Persamaan 2.16. setelah itu mengurutkan jarak dari yang terkecil sampai yang terbesar. Hasil perhitungan jarak dan hasil pengurutan jarak ditunjukkan Tabel 4.26 dan 4.27. Setelah itu dipilih k (k=4) tetangga terdekat untuk kemudian dilakukan pembobotan jarak. Perhitungan pembobotan jarak sesuai dengan Persamaan 2.17. Hasil pehitungan bobot jarak ditunjukkan Tabel 4.28. Proses klasifikasi dilakukan dengan menjumlahkan bobot sesuai dengan kelasnya. Maka kelas yang memiliki bobot tertinggi merupakan kelas hasil klasifikasi data uji. Hasil perhitungan klasifikasi ditunjukkan Tabel 4.29.

Tabel 4.22 Data Latih Asli Citra Tanda Tangan Offline

No	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	kelas
1	0.158	6.011	37.132	255	1.026	0.7	0.202	4.531	21.534	255	1.044	0.7	0.307	2.575	7.628	255	1.117	0.7	1
2	0.155	6.123	38.488	255	1.025	0.7	0.225	3.972	16.773	0	1.056	0.7	0.284	2.902	9.424	255	1.097	0.7	1
3	0.147	6.517	43.477	255	1.022	0.7	0.191	4.833	24.355	255	1.040	0.7	0.287	2.853	9.141	255	1.100	0.7	1
4	0.258	3.320	12.021	255	1.077	0.7	0.349	2.050	5.201	0	1.166	0.7	0.113	8.599	74.936	255	1.013	0.7	2
5	0.238	3.700	14.691	255	1.064	0.7	0.240	3.653	14.345	255	1.065	0.7	0.312	2.502	7.261	255	1.123	0.7	2
6	0.217	4.157	18.283	255	1.052	0.7	0.305	2.605	7.784	255	1.115	0.7	0.241	3.633	14.195	255	1.066	0.7	2
7	0.249	3.477	13.093	255	1.071	0.7	0.342	2.138	5.571	0	1.156	0.7	0.231	3.833	15.692	255	1.060	0.7	2
8	0.229	3.874	16.008	255	1.059	0.7	0.317	2.445	6.977	255	1.127	0.7	0.269	3.129	10.790	255	1.085	0.7	3
9	0.244	3.582	13.833	0	1.068	0.7	0.315	2.467	7.088	255	1.126	0.7	0.060	16.643	278.004	255	1.004	0.7	3
10	0.233	3.804	15.474	255	1.061	0.7	0.322	2.381	6.669	255	1.133	0.7	0.266	3.189	11.167	255	1.083	0.7	3

Tabel 4.23 Data Uji Asli Citra Tanda Tangan Offline

No	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	kelas
1	0.211	4.296	19.456	255	1.049	0.7	0.312	2.502	7.261	255	1.123	0.7	0.186	4.993	25.926	0	1.037	0.7	2
2	0.153	6.240	39.941	255	1.024	0.7	0.236	3.743	15.012	255	1.063	0.7	0.267	3.166	11.023	255	1.084	0.7	1

Tabel 4.24 Data Latih Normal Citra Tanda Tangan Offline

No	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	kelas
1	0.180	0.773	0.739	0.9	0.155	0.1	0.154	0.813	0.782	0.9	0.131	0.1	0.883	0.104	0.101	0.1	0.864	0.1	1
2	0.162	0.801	0.773	0.9	0.142	0.1	0.271	0.652	0.583	0.1	0.207	0.1	0.810	0.123	0.106	0.1	0.725	0.1	1
3	0.100	0.900	0.900	0.9	0.100	0.1	0.100	0.900	0.900	0.9	0.100	0.1	0.820	0.120	0.106	0.1	0.744	0.1	1
4	0.900	0.100	0.100	0.9	0.900	0.1	0.900	0.100	0.100	0.1	0.900	0.1	0.270	0.445	0.300	0.1	0.164	0.1	2
5	0.754	0.195	0.168	0.9	0.706	0.1	0.348	0.561	0.482	0.9	0.264	0.1	0.900	0.100	0.100	0.1	0.900	0.1	2
6	0.604	0.310	0.259	0.9	0.531	0.1	0.674	0.260	0.208	0.9	0.581	0.1	0.675	0.164	0.120	0.1	0.519	0.1	2
7	0.837	0.139	0.127	0.9	0.813	0.1	0.861	0.125	0.115	0.1	0.838	0.1	0.644	0.175	0.125	0.1	0.479	0.1	2
8	0.694	0.239	0.201	0.9	0.633	0.1	0.735	0.214	0.174	0.9	0.657	0.1	0.764	0.135	0.110	0.1	0.649	0.1	3
9	0.797	0.166	0.146	0.1	0.761	0.1	0.726	0.220	0.179	0.9	0.646	0.1	0.100	0.900	0.900	0.1	0.100	0.1	3
10	0.718	0.221	0.188	0.9	0.661	0.1	0.760	0.195	0.161	0.9	0.691	0.1	0.753	0.139	0.112	0.1	0.631	0.1	3

Tabel 4.25 Data Uji Normal Citra Tanda Tangan Offline

No	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	kelas
1	0.9	0.1	0.1	0.1	0.9	0.1	0.9	0.1	0.1	0.1	0.9	0.1	0.1	0.9	0.9	0.1	0.1	0.1	2
2	0.1	0.9	0.9	0.1	0.1	0.1	0.1	0.9	0.9	0.1	0.1	0.1	0.9	0.1	0.1	0.9	0.9	0.1	1

Tabel 4.26 Hasil Pehitungan Jarak dengan Euclidean Distance

Data Uji ke-	Data Latih ke-	Nilai Jarak	Data Uji ke-	Data Latih ke-	Nilai Jarak
1	1	2.793556	2	1	1.413664
	2	2.499797		2	1.246097
	3	2.92798		3	1.396842
	4	1.113623		4	2.738113
	5	2.231499		5	2.031446
	6	1.849099		6	2.183672
	7	1.491788		7	2.469094
	8	1.864935		8	2.297252
	9	0.889902		9	2.746378
	10	1.837833		10	2.346863

Tabel 4.27 Hasil Pengurutan Jarak

Data Uji ke-	Data latih ke-	Nilai Jarak	Kelas	Data Uji ke-	Data latih ke-	Nilai Jarak	Kelas
1	9	0.889902	3	2	2	1.246097	1
	4	1.113623	2		3	1.396842	1
	7	1.491788	2		1	1.413664	1
	10	1.837833	3		5	2.031446	2
	6	1.849099	2		6	2.183672	2
	8	1.864935	3		8	2.297252	3
	5	2.231499	2		10	2.346863	3
	2	2.499797	1		7	2.469094	2
	1	2.793556	1		4	2.738113	2
	3	2.92798	1		9	2.746378	3

Tabel 4.28 Perhitungan Pembobotan Jarak

Data Uji ke-	Jarak (data ke-)	Nilai Bobot	Kelas	Data Uji ke-	Jarak (data ke-)	Nilai Bobot	Kelas
1	D(X9)	1	3	2	D(X2)	1	1
	D(X4)	0.763991	2		D(X3)	0.808054	1
	D(X7)	0.365053	2		D(X1)	0.786634	1
	D(X10)	0	3		D(X5)	0	2

Tabel 4.29 Hasil Klasifikasi dengan m-KNN

Data Uji ke-	Kelas	Nilai Bobot kelas	Data Uji ke-	Kelas	Nilai Bobot kelas
1	Kelas 1	0	2	Kelas 1	2.594688
	Kelas 2	1.129044		Kelas 2	0
	Kelas 3	1		Kelas 3	0

Berdasarkan perhitungan dengan melakukan pembobotan jarak, masing-masing kelas dijumlahkan bobotnya kemudian dipilih kelas yang memiliki nilai bobot tertinggi. Setelah diketahui nilai kelas dengan bobot tertinggi, maka kelas itulah yang merupakan kelas data uji. Pada Tabel 4.29 menunjukkan bahwa kelas dari data uji pertama adalah kelas 2, sedangkan kelas dari data uji kedua adalah kelas 1.

4.6 Perancangan Pengujian

Perancangan pengujian bertujuan untuk membuat skenario pengujian dengan tepat sehingga sistem yang telah dibuat dapat divalidasi. Pada penelitian ini pengujian dilakukan untuk menguji nilai *FAR* dan *FRR* sistem. Pengujian *FAR* adalah pengujian untuk mengetahui jumlah data uji palsu yang dianggap asli oleh sistem. Sedangkan pengujian *FRR* adalah pengujian untuk mengetahui jumlah data uji asli yang dianggap palsu oleh sistem. Pengujian dilakukan terhadap data tanda tangan dari Indonesia. Skenario pengujian terdiri dari 2 macam, yaitu pengujian fitting dan pengujian ciri local dengan ciri global.

4.6.1 Perancangan Pengujian *Fitting*

Pengujian *fitting* dilakukan dengan tujuan untuk mengetahui apakah sistem yang dibuat sudah benar atau tidak. Pada pengujian ini, data latih juga digunakan sebagai data uji. Pengujian ini dilakukan pada citra data latih dengan dimensi ukuran 210x100. Penentuan nilai akurasi pada pengujian ini dilakukan dengan

menghitung nilai *FRR* dikarenakan pengujian dilakukan dengan menggunakan data citra asli sebagai data uji. Skenario pengujian *fitting* sesuai dengan Tabel 4.30.

Tabel 4.30 Skenario Pengujian *Fitting*

	Jumlah Data Latih	Nilai K	Zona		
			Zona Vertikal	Zona Horizontal	Zona 4
%FRR	5	K=1			
		K=2			
		K=3			
		K=4			
		K=5			
	10	K=1			
		K=2			
		K=3			
		K=4			
		K=5			
	14	K=1			
		K=2			
		K=3			
		K=4			
		K=5			

4.6.2 Perancangan Pengujian Jumlah Data Latih

Pengujian data latih bertujuan untuk mengetahui nilai akurasi yang didapatkan apakah berpengaruh dengan jumlah data latih yang digunakan. Dalam pengujian ini, data latih yang digunakan terdiri dari 20 kelas dengan masing-masing kelas memiliki variasi jumlah yaitu 5, 10, dan 14. Sedangkan data uji yang digunakan terdiri dari 20 kelas dengan masing-masing kelas memiliki jumlah 10 tanda tangan. Hasil dari pengujian jumlah data latih dengan nilai akurasi yang tertinggi digunakan sebagai data untuk pengujian selanjutnya. Skenario pengujian jumlah data latih sesuai degan Tabel 4.31.

Tabel 4.31 Skenario Pengujian Jumlah Data Latih

Zona	Jumlah Data Latih	*%FRR*	*%FAR*
Zona Vertikal	5		
	10		

	14		
Zona Horizontal	5		
	10		
	14		
Zona 4	5		
	10		
	14		

4.6.3 Perancangan Pengujian Nilai K

Tujuan dilakukan pengujian terhadap nilai k adalah untuk mengetahui berapa nilai k yang optimal sehingga dapat menghasilkan akurasi yang terbaik. Nilai k yang digunakan dalam pengujian bernilai 1 sampai dengan 15. Hasil terbaik dari pengujian ini akan digunakan untuk melakukan pengujian selanjutnya. Tabel 4.32 menunjukkan skenario pengujian nilai k.

Tabel 4.32 Skenario Pengujian Nilai k

Zona	Nilai K	%FRR	%FAR
Zona Vertikal	1		
	2		
	:		
	:		
Zona Horizontal	1		
	2		
	:		
	:		
Zona 4	1		
	2		
	:		
	:		
Zona Global	1		
	2		
	:		
	:		

4.6.4 Perancangan Pengujian Variasi Ciri

Pengujian variasi ciri bertujuan untuk mengetahui ciri *geometric* yang berpengaruh terhadap nilai *FAR* dan *FRR*. Pada pengujian ini dilakukan kombinasi ciri dengan jumlah ciri yaitu 1 ciri, 2 ciri, hingga 6 ciri. Pengujian ini dilakukan

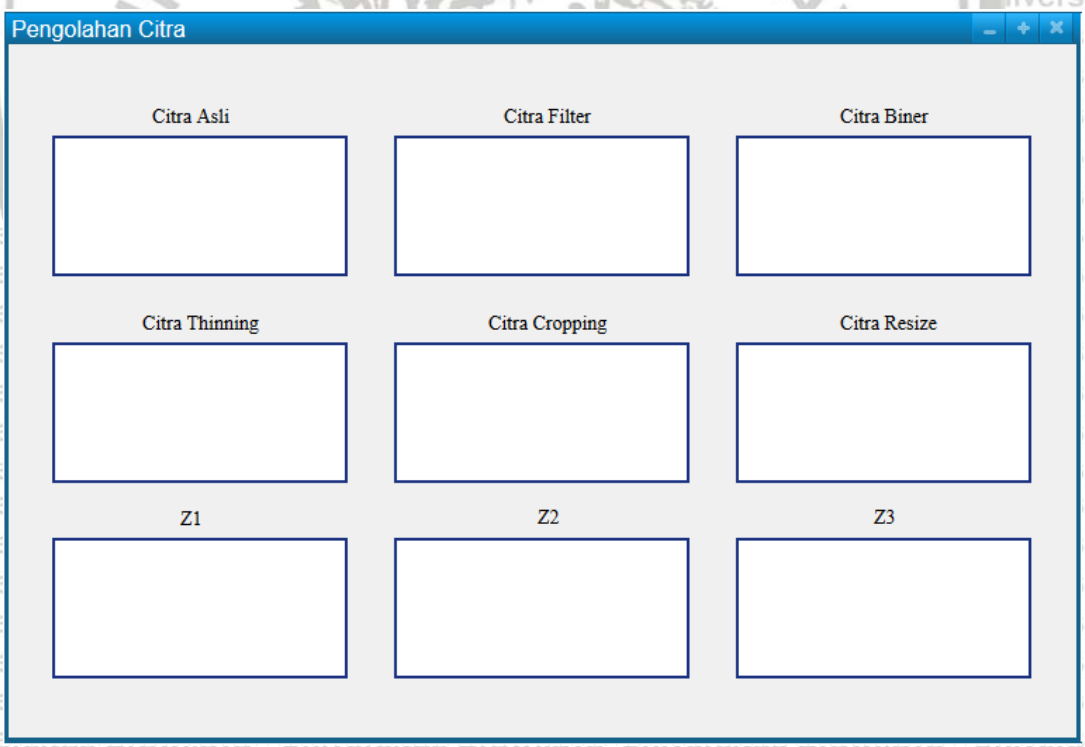
dengan menggunakan jumlah data latih, zona dan nilai *k* yang terbaik hasil dari pengujian sebelumnya. Skenario pengujian variasi ciri sesuai dengan Tabel 4.33.

Tabel 4.33 Skenario Pengujian Variasi Ciri

No	Variasi Ciri ke-	%FRR	%FAR
1	1		
2	1,2		
3	1,2,3		
:	:		

4.7 Perancangan Antarmuka

Perancangan antarmuka membahas rancangan tampilan program yang akan dibuat. Antarmuka dibuat dengan tujuan agar program dapat berkomunikasi dengan baik dengan pengguna. Perancangan antarmuka program ditunjukkan Gambar 4.7.



Gambar 4.7 Perancangan Antarmuka Program

BAB 5 IMPLEMENTASI

5.1 Lingkungan Implementasi

Lingkungan implementasi yang dimaksud dalam sub bab ini adalah lingkungan implementasi perangkat lunak dan perangkat keras.

5.1.1 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan sistem deteksi dan verifikasi tanda tangan adalah:

1. Sistem Operasi Windows 8.1 Pro with Media Center (64-bit)
2. Python 2.7.13
3. JetBrains PyCharm Community Edition 2016.3
4. OpenCV 2.4.12

5.1.2 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem deteksi dan verifikasi tanda tangan adalah:

1. Intel® Core™ i5-3230M CPU @2.60GHz
2. Memory (RAM) 2.00GB
3. 500GB HDD

5.2 Implementasi Sistem

Sub bab implementasi sistem menjelaskan tentang implementasi untuk sitem deteksi dan verifikasi tanda tangan dan algoritma modified-K Nearest Neighbour. Implementasi meliputi implementasi kode program dan implementasi antarmuka.

5.2.1 Implementasi Filtering Citra

Filtering merupakan proses yang bertujuan untuk mengurangi *noise* pada citra sehingga dapat mengingkatkan kualitas citra. Metode yang digunakan untuk *filtering* adalah median *filter*. Implementasi kode proses filtering sesuai Gambar 5.1.

```
1   def filtering(image):
2       img_filter = image[:]
3       for j in range(len(image)):
4           for i in range(j):
5               img_filter[j, i] = image[j, i]
6       members = [image[0, 0]] * 9
7       for j in range(1, image.shape[0] - 1):
8           for i in range(1, image.shape[1] - 1):
9               members[0] = image[j - 1, i - 1]
10              members[1] = image[j, i - 1]
11              members[2] = image[j + 1, i - 1]
12              members[3] = image[j - 1, i]
```

```
13              members[4] = image[j, i]
14              members[5] = image[j + 1, i]
15              members[6] = image[j - 1, i + 1]
16              members[7] = image[j, i + 1]
17              members[8] = image[j + 1, i + 1]
18
19              members.sort()
20              img_filter[j, i] = members[4]
21      #cv2.imshow("filter", img_filter)
22      return img_filter
```

Gambar 5.1 Implementasi Kode Proses Filtering

Penjelasan kode program Gambar 5.1:

1. Baris 2-6 adalah inisialisasi image untuk perhitungan filtering
2. Baris 7-17 adalah proses untuk mencari nilai dari piksel-piksel tetangga
3. Baris 19 adalah proses sorting atau pengurutan nilai dari piksel-piksel tetangga
4. Baris 20 adalah proses pengambilan nilai yang sudah diurutkan untuk kemudian dijadikan sebagai nilai baru. Nilai yang diambil adalah median dari semua nilai yang sudah diurutkan yang mana nilai itu berada pada indeks ke empat

5.2.2 Implementasi Binarisasi Citra

Pada proses binarisasi citra, untuk mendapatkan nilai *threshold* dilakukan dengan menerapkan metode *thresholding* Otsu. Implementasi kode program proses binarisasi sesuai Gambar 5.2.

```
1   def thresholding(image):
2       hist = cv2.calcHist([image], [0], None, [256], [0, 256])
3       hist_norm = hist.ravel() / hist.max()
4       Q = hist_norm.cumsum()
5       bins = np.arange(256)
6       fn_min = np.inf
7       thresh = -1
8       for i in xrange(1, 256):
9           p1, p2 = np.hsplit(hist_norm, [i])  # probabilities
10          q1, q2 = Q[i], Q[255] - Q[i]  # cum sum of classes
11          b1, b2 = np.hsplit(bins, [i])  # weights
12          # finding means and variances
13          m1, m2 = np.sum(p1 * b1) / q1, np.sum(p2 * b2) / q2
14          v1, v2 = np.sum(((b1 - m1) ** 2) * p1) / q1,
15  np.sum(((b2 - m2) ** 2) * p2) / q2
16          # calculates the minimization function
17          fn = v1 * q1 + v2 * q2
18          if fn < fn_min:
19              fn_min = fn
20              thresh = i
21      im_bw = np.zeros(image.shape, np.uint8)
22      l, t = np.shape(image)
23      #binarisasi
24      for i in range(l):
25          for j in range(t):
26              if (image[i, j] < thresh):
27                  im_bw[i, j] = 0
28              else:
29                  im_bw[i, j] = 255
```

```
30      #cv2.imshow("biner", im_bw)
31      return im_bw
```

Gambar 5.2 Implementasi Kode Proses Binarisasi

Penjelasan kode program Gambar 5.2:

1. Baris 2-7 adalah proses perhitungan histogram citra yang akan digunakan untuk proses binarisasi citra
2. Baris 8-20 adalah pencarian nilai dengan menggunakan metode *thresholding* Otsu
3. Baris 21-22 adalah proses inisialsasi matriks untuk image biner
4. Baris 23-29 adalah proses binarisasi. Yaitu mengganti nilai piksel manjadi 0 jika nilai nya lebih kecil dari threshold yang sudah didapat. Jika nilainya lebih besar sama dengan maka diganti menjadi 255.

5.2.3 Implementasi Thinning Citra

Proses *thinning* menyebabkan citra menjadi lebih tipis karena hanya mengandung 1 piksel pembentuk citra. Hal ini dilakukan agar ukuran piksel citra menjadi seragam. Metode yang digunakan untuk proses *thinning* adalah metode *Zhang Suen*. Implementasi kode program proses *thinning* citra sesuai Gambar 5.3.

```
1   def thinning(image):
2       iter = 0
3       lb, ti = np.shape(image)
4       lb = lb - 1
5       ti = ti - 1
6
7       M = np.zeros(image.shape, np.uint8)
8       for i in range(1, lb):
9           for j in range(1, ti):
10              # matriks tetangganya,
11              p2 = image[i - 1, j]
12              p3 = image[i - 1, j + 1]
13              p4 = image[i, j + 1]
14              p5 = image[i + 1, j + 1]
15              p6 = image[i + 1, j]
16              p7 = image[i + 1, j - 1]
17              p8 = image[i, j - 1]
18              p9 = image[i - 1, j - 1]
19
20              a1 = p2 / 255
21              a2 = p3 / 255
22              a3 = p4 / 255
23              a4 = p5 / 255
24              a5 = p6 / 255
25              a6 = p7 / 255
26              a7 = p8 / 255
27              a8 = p9 / 255
28
29              # koneksi
30              c1 = 0  # p2 == 0 & & p3 == 1
31              c2 = 0  # p3 == 0 & & p4 == 1
32              c3 = 0  # p4 == 0 & & p5 == 1
33              c4 = 0  # p5 == 0 & & p6 == 1
34              c5 = 0  # p6 == 0 & & p7 == 1
```

```
35              c6 = 0  # p7 == 0 & & p8 == 1
36              c7 = 0  # p8 == 0 & & p9 == 1
37              c8 = 0  # p9 == 0 & & p2 == 1
38
39              if (p2 == 0 and p3 == 255):
40                  c1 = 1
41              if (p3 == 0 and p4 == 255):
42                  c2 = 1
43              if (p4 == 0 and p5 == 255):
44                  c3 = 1
45              if (p5 == 0 and p6 == 255):
46                  c4 = 1
47              if (p6 == 0 and p7 == 255):
48                  c5 = 1
49              if (p7 == 0 and p8 == 255):
50                  c6 = 1
51              if (p8 == 0 and p9 == 255):
52                  c7 = 1
53              if (p9 == 0 and p2 == 255):
54                  c8 = 1
55
56              A = c1 + c2 + c3 + c4 + c5 + c6 + c7 + c8  #
57  koneksi
58              B = a1 + a2 + a3 + a4 + a5 + a6 + a7 + a8  #
59  piksel tetangga yang berwarna hitam
60
61              iterMod = iter % 2
62
63              # kondisi ke 3 dan 4
64              if (iterMod == 1):
65                  m1 = p2 * p4 * p6
66              else:
67                  m1 = p2 * p4 * p8
68
69              if (iterMod == 1):
70                  m2 = p4 * p6 * p8
71              else:
72                  m2 = p2 * p6 * p8
73
74              iter = iter + 1
75
76              # cek kondisinya
77              if (A == 1 and (B >= 2 and B <= 6) and m1 == 0
78  and m2 == 0):
79                  M[i, j] = 255
80
81      img_thin = image + M
82
83      #cv2.imshow("thinning", img_thin)
84      return img_thin
```

Gambar 5.3 Implementasi Kode Proses Thinning

Penjelasan kode program Gambar 5.3:

1. Baris 2-5 adalah inisialisasi variable untuk proses *thinning* citra
2. Baris 7 adalah inisialisasi matriks Mask
3. Baris 10-27 adalah proses mencari nilai dari piksel tetangga

4. Baris 30-54 adalah proses mencari koneksi pada piksel tetangga. Ketika nilai piksel tetangga secara beurutan 0 1 maka dihitung koneksinya adalah 1
5. Baris 56-74 adalah proses mencari kondisi yang diperlukan untuk melakukan *thinning* citra sesuai dengan algoritma pada Sub bab 2.2.3.1 poin *thinning*.
6. Baris 76-78 adalah proses pengecekan kondisi
7. Baris 79 adalah proses pengubahan nilai piksel yang semula bernilai 0 menjadi 255
8. Baris 81 adalah proses pengaplikasian Mask ke citra asli sehingga didapatkan citra yang sudah di *thinning*

5.2.4 Implementasi Cropping dan Resizing Citra

Proses *cropping* dan *resize* citra bertujuan agar citra memiliki keberagaman minmum. Proses *resize* citra dilakukan dengan menggunakan *library* openCV. Implementasi kode program untuk cropping dan resize citra secara berturut ditunjukkan Gambar 5.4 dan Gambar 5.5.

```
1   def cropping(self,image):
2       crop = np.zeros(image.shape, np.uint8)
3       l, t = np.shape(image)
4       done = False
5       # sumbu x
6       for x in xrange(len(crop)):
7           for y in xrange(len(crop[x])):
8               if image[x][y] == 0:
9                   t_atas = x
10                  done = True
11                  break
12          if done:
13              break
14
15      for x in xrange(len(crop)):
16          for y in range(t_atas, len(crop[x])):
17              if image[x][y] == 0:
18                  t_bawah = x + 1
19
20      # y
21      done = False
22      for y in xrange(len(crop[0])):
23          for x in xrange(t_atas, t_bawah):
24              if image[x][y] == 0:
25                  t_kiri = y - 1
26                  done = True
27                  break
28          if done:
29              break
30
31      for y in xrange(len(crop[0])):
32          for x in xrange(t_atas, t_bawah):
33              if image[x][y] == 0:
34                  t_kanan = y + 1
35
36      img_crop = image[t_atas:t_bawah, t_kiri:t_kanan]
```

```
37      return  img_crop
```

Gambar 5.4 Implementasi Kode Proses Cropping

Penjelasan kode program Gambar 5.4:

1. Baris 2-3 adalah inisialisasi matriks untuk menyimpan nilai dari hasil *cropping*
2. Baris 6-18 adalah proses untuk mencari nilai batas atas dan batas bawah obyek dari suatu citra. Pencarian batas ini berdasarkan sumbu x
3. Baris 21-34 adalah proses pencarian nilai batas kanan dan kiri obyek suatu citra. Pencarian ini didasarkan pada sumbu y
4. Baris 36 adalah proses *cropping* citra. *Cropping* citra didasarkan pada batas-batas yang sudah dicari pada proses sebelumnya

```
1   def resize(image):
2       h, w = image.shape[:2]
3       img_resize = cv2.resize(image, (210, 100),
4   interpolation=cv2.INTER_LINEAR)
5       cv2.imshow("resize", img_resize)
6
7       resize2 = np.zeros(img_resize.shape, np.uint8)
8       l, t = np.shape(img_resize)
9       for i in range(l):
10          for j in range(t):
11              if (img_resize[i, j] < 255):
12                  resize2[i, j] = 0
13              else:
14                  resize2[i, j] = 255
15      cv2.imshow("resize2", resize2)
16      return resize2
```

Gambar 5.5 Implementasi Kode Proses Resize

Penjelasan kode program Gambar 5.5 :

1) Baris 2-4 adalah proses *resizing* citra dengan menggunakan fungsi *bilinear interpolation* pada library opencv
2) Baris 7-15 adalah proses normalisasi piksel citra setelah dilakukan *resize* dengan *bilinear interpolation*

5.2.5 Implementasi Ekstraksi Ciri *Geometric*

Sebelum dilakukan ekstraksi ciri *geometric*, citra terlebih dahulu dibagi menjadi zona-zona yang disebut sebagai proses *Zoning*. Pada penelitian ini, citra dibagi menjadi 3 zona. Implementasi kode program untuk proses *Zoning* ditunjukkan Gambar 5.6.

```
1   def zoning(image):
2       Z1 = np.zeros((100, 70), np.uint8)
3       Z2 = np.zeros((100, 70), np.uint8)
4       Z3 = np.zeros((100, 70), np.uint8)
5
6       for i in range(700):
7           for j in range(100):
8               Z1[j, i] = image[j, i]
9
10      for i in range(70):
```

```
11          for j in range(100):
12              ii = 70 + i
13              Z2[j, i] = image[j, ii]
14
15      for i in range(70):
16          for j in range(100):
17              ii = 139 + i
18              # print jj
19              Z3[j, i] = image[j, ii]
20
21      return Z1, Z2, Z3
```

Gambar 5.6 Implementasi Kode Proses *Zoning*

Penjelasan kode program Gambar 5.6:

1. Baris 2-4 adalah proses inisialisasi matrik untuk masing-masing zona 1, zona 2, dan zona 3
2. Baris 6-18 adalah proses pembagian citra menjadi 3 zona

Setelah dilakukan *zoning*, maka dimulai proses ekstraksi ciri *geometric*. Ciri yang diekstrak antara lain standar deviasi, *skewness*, *kurtosis*, *centroid*, *pixels density*, dan *eccentricity*. Ciri ini berdasarkan pada *geometric* citra yang mana merupakan aspek bentuk dan ukuran. Implementasi kode program untuk proses ekstraksi ciri *geometric* ditunjukkan Gambar 5.7.

```
1   def ekstraksi (image):
2       w, h = np.shape(image)
3       countw = 0
4       countb = 0
5       counti, countj=0
6       for i in range(w):
7           for j in range(h):
8               if (image[i][j] == 0):
9                   counti += i  # untuk koordinat centroid
10                  countj += j  # untuk koordinat centroid
11                  countb += 1
12              else:
13                  countw += 1
14      print countw, countb
15      total = countb + countw
16      print total
17
18      # hitung standar deviasi
19      meanW = countw / total
20      meanB = countb / total
21
22      miuW = meanW * 0
23      miuB = meanB * 1
24
25      miu = miuW + miuB
26
27      stdW = ((0 - miu) ** 2) * meanW
28      stdB = ((1 - miu) ** 2) * meanB
29      std = (stdW + stdB) ** (1 / 2)
30      print 'standar deviasi= ', std
31
32      # hitung skewness biasa
```

```
33      skewW = stdW * (0 - miu)
34      skewB = stdB * (1 - miu)
35      skew = (skewB + skewW) / ((std) ** 3)
36      print 'skewness= ', skew
37
38      # menghitung kurtosis
39      kurtoW = skewW * (0 - miu)
40      kurtoB = skewB * (1 - miu)
41      kurtosis = (kurtoB + kurtoW) / ((std) ** 4)
42      print 'kurtosis= ', kurtosis
43
44      #hitung centroid
45      cox = counti / countb
46      coy = countj / countb
47      bulatx = int(round(cox))
48      bulaty = int(round(coy))
49      cog=image[bulatx][bulaty]
50      print 'koor x', cox, 'koor y', coy
51      print 'x bulat', bulatx, 'y bulat', bulaty
52      print 'centroid', cog
53
54      # hitung piksel density
55      w, h = np.shape(image)
56      print w
57      print h
58      pd=w*h/countb
59      #hitung eccentricity
60      Ecc = h / w
61      print 'Eccentricity', Ecc
62      return std, skew, kurtosis, cog, pd, Ecc
63
```

Gambar 5.7 Implementasi Kode Ekstraksi Ciri *Geometric*

Penjelasan kode program gambar 5.7:

1) Baris 2-14 adalah proses pengambilan histogram dari citra hasil preprocessing
2) Baris 16-29 adalah proses ekstraksi ciri standar deviasi
3) Baris 31-35 adalah proses ekstraksi ciri *skewness*
4) Baris 38-42 adalah proses perhitungan ekstraksi ciri *kurtosis*
5) Baris 45-52 adalah proses perhitungan ekstraksi ciri *centroid*
6) Baris 55-58 adalah proses perhitungan ekstraksi ciri *pixels density*
7) Baris 61-62 adalah proses perhitungan ekstraksi ciri *eccentricity*

5.2.6 Implementasi Algoritma Modified-K Nearest Neighbour

Algoritma *modified-K Nearest Neighbour* digunakan untuk melakukan proses klasifikasi citra. Implementasi kode program untuk proses klasifikasi dengan algoritma *modified-K Nearest Neighbour* ditunjukkan Gambar 5.8.

```
1   import csv
2   import math
3
4   def read_latih():
5       count =0
6       with open('data_14ttd_normal.csv', 'r') as fileciri:
7           datafile = csv.reader(fileciri)
8           data_latih = []
```

```
        for data in datafile:
            tmp=[]
            for item in data:
                tmp.append(float(item))
            data_latih.append(tmp)
            count+=1
    return data_latih

def read_uji():
    count=0
    with open('uji_palsu_normal.csv', 'r') as fileciri:
        datafile = csv.reader(fileciri)
        data_uji = []
        for data in datafile:
            tmp=[]
            for item in data:
                tmp.append(float(item))
            data_uji.append(tmp)
            count+=1
    return data_uji

#hitung euclidean
def Euclid(data_latih, data_uji):
    nilai_dist = []
    kelas_latih = []
    for i in range(len(data_uji)):
        for j in range(len(data_latih)):
            nilai_jarak = 0.0
            for k in range(0,18):
                nilai_jarak += pow((data_uji[i][k] -
data_latih[j][k]),2)
            dist= math.sqrt(nilai_jarak)
            nilai_dist.append(dist)
            kelas_latih.append(data_latih[j][18])

    return nilai_dist,kelas_latih


def splitEuclid(nilai_dist,kelas_latih,length):
    new_Dist=[]
    new_Kelas=[]
    for i in range(len(nilai_dist)):
        if ((i % length) == 0):
            new_Dist.append([])
            new_Kelas.append([])
            for j in range(0 + i, i + length):
                new_Dist[(i / length)].append(nilai_dist[j])
                new_Kelas[(i /
length)].append(kelas_latih[j])
    return new_Dist,new_Kelas

#sorting eucli
def sort_Euclid(nilai_dist, kelas_latih):
    for m in range(len(nilai_dist)):
        l=m
        while l>0:
            if(nilai_dist[l]<nilai_dist[l-1]):
              tmpDist=nilai_dist[l]
              tmpKelas=kelas_latih[l]
```

```
                nilai_dist[l]=nilai_dist[l-1]
                kelas_latih[l]=kelas_latih[l-1]
                nilai_dist[l-1]=tmpDist
                kelas_latih[l-1]=tmpKelas
            l=l-1
    return nilai_dist,kelas_latih


def select(nilai_dist,kelas_latih,k):
    sorting_dist = []
    sorting_kelas = []

    for i in range(k):
        sorting_dist.append(nilai_dist[i])
        sorting_kelas.append(kelas_latih[i])
    return sorting_dist,sorting_kelas

#pembobotan jarak
def bobot_jarak(sorting_dist, sorting_kelas):
    bobot = []
    hasil_bobot = [0] * 94
    pointer = [False] * 94
    dk=max(sorting_dist)
    d1=min(sorting_dist)

    for i in range(len(sorting_dist)):
        if ((dk-d1)==0):
            hasil=1
        else:
            hasil = (dk-sorting_dist[i])/(dk-d1)
        bobot.append(hasil)

    for m in range(len(sorting_kelas)):
        l=m
        while l>0:
            if(sorting_kelas[l]<sorting_kelas[l-1]):
              tmpkk=sorting_kelas[l]
              tmpkl=bobot[l]
              sorting_kelas[l]=sorting_kelas[l-1]
              bobot[l]=bobot[l-1]
              sorting_kelas[l-1]=tmpkk
              bobot[l-1]=tmpkl
            l=l-1

    # penjumlahan bobot yang berada pada kelas yang sama
    for i in range(len(sorting_dist)):
        if(hasil_bobot[int(sorting_kelas[i]) - 1] == 0):
            hasil_bobot[int(sorting_kelas[i]) - 1] = bobot[i]
        if(pointer[int(sorting_kelas[i])-1] !=  True):
            for j in range(len(sorting_dist)):
                if(i!=j):
                    if(sorting_kelas[i] == sorting_kelas[j]):
                        # print bobot[j]
                        hasil_bobot[int(sorting_kelas[i] -
1)] += bobot[j]
            pointer[int(sorting_kelas[i] - 1)] = True
    return hasil_bobot

# klasifikasi kelas
```

```
127 def klasifikasi(hasil_bobot):
128     for i in range(len(hasil_bobot)):
129         if(hasil_bobot[i] == max(hasil_bobot)):
130             print "Data uji di klasifikasikan di kelas ", (i
131 + 1)
```

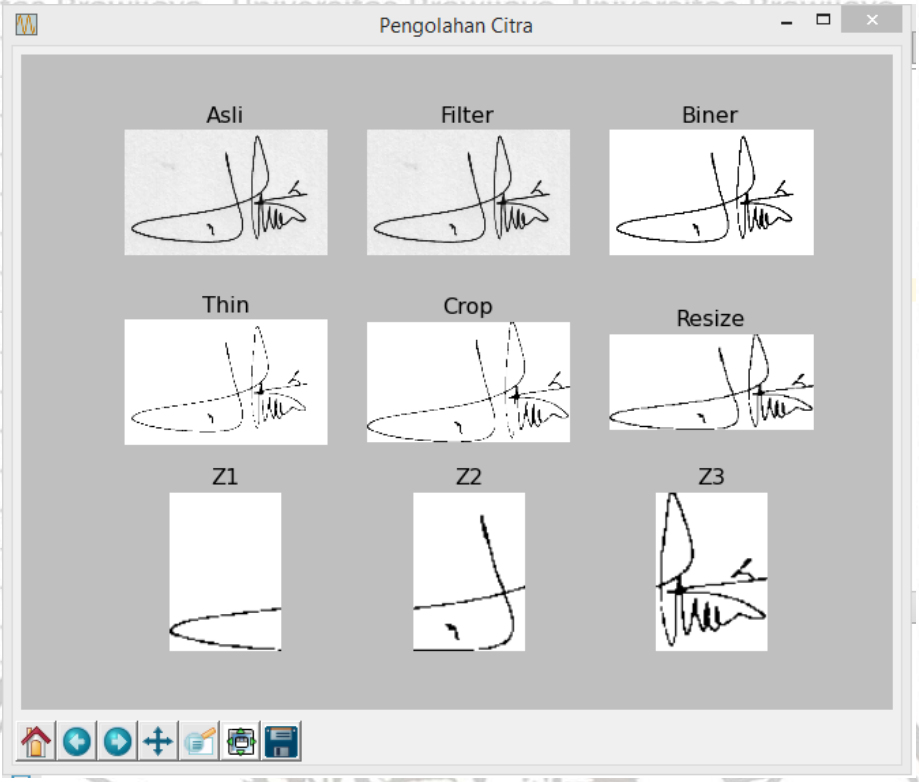
Gambar 5.8 Implementasi Kode Klasifikasi dengan mKNN

Penjelasan kode program Gambar 5.8:

1) Baris 4-15 adalah proses pembacaan data latih
2) Baris 17-28 adalah proses pembacaan data uji
3) Baris 31-44 adalah proses menghitung jarak dengan menggunakan *Euclidean Distance*
4) Baris 47-58 adalah proses pemisahan data nilai jarak sesuai dengan banyaknya data latih sehingga terbagi jelas nilai jarak antara data uji pertama dan data uji yang lain
5) Baris 61-73 adalah proses *sorting* jarak dari yang terkecil ke yang terbesar
6) Baris 76-83 adalah proses pemilihan jarak yang sudah diurutkan sebanyak nilai k tetangga terdekat
7) Baris 86-124 adalah proses pembobotan jarak
8) Baris 127-131 adalah proses klasifikasi untuk menentukan kelas dari data uji

5.3 Implementasi Antarmuka

Implementasi antarmuka membahas hasil implementasi tampilan program. Pada tampilan program ditampilkan semua proses *pre-processing* citra yang terdiri dari citra asli, citra hasil *filtering*, citra biner, citra hasil *thinning*, citra hasil *cropping*, citra yang sudah dilakukan *resize*, dan tiap-tiap zona hasil proses *zoning*. Tampilan program ditunjukkan oleh Gambar 5.9.



Gambar 5.9 Implementasi Antarmuka Program

BAB 6 PENGUJIAN DAN ANALISIS

6.1 Pengujian

Pengujian dilakukan dengan tujuan untuk dapat mengetahui apakah sistem telah dibuat dengan benar. Selain itu, pengujian juga dilakukan agar dapat mengetahui tingkat akurasi sistem untuk melakukan deteksi dan verifikasi tanda tangan dengan menerapkan ciri *geometric* serta algoritma *modified-K Nearest Neighbour*. Proses pengujian dilakukan dengan menggunakan data tanda tangan Indonesia.

Pada penelitian ini terdiri dari 3 buah scenario pengujian yaitu pengujian fitting, pengujian jumlah data latih, dan pengujian nilai k. Setiap pengujian dihitung performa dan akurasinya menggunakan nilai *FAR* dan *FRR*. Data yang digunakan dibagi menjadi dua yaitu data latih dan data uji. Data latih terdiri dari 20 kelas dengan variasi jumlah data tiap-tiap kelasnya. Sedangkan data uji terdiri dari 20 kelas dengan tiap-tiap kelas memiliki 10 data sehingga jumlah data uji sebanyak 200 data.

6.2 Pengujian *Fitting*

6.2.1 Skenario Pengujian *Fitting*

Pengujian *fitting* dilakukan dengan melakukan pengujian terhadap data latih yang sekaligus digunakan sebagai data uji. Dalam melakukn pengujian *fitting*, variasi jumlah data latih setiap kelas yaitu 5 data latih, 10 data latih, dan 14 data latih. Sedangkan nilai k yang digunakan untuk setiap variasi jumlah data latih adalah k=1, k=2, k=3, k=4, dan k=5. Untuk pengujian *fitting* yang digunakan adaah pengujian *FRR*. Hasil pengujian *fitting* sesuai dengan Tabel 6.1.

Tabel 6.1 Hasil Pengujian *Fitting*

	Jumlah Data Latih	Nilai K	Zona		
			Zona Vertikal	Zona Horizontal	Zona 4
%FRR	5	K=1	0	0	0
		K=2	0	0	0
		K=3	0	0	0
		K=4	2	1	1
		K=5	3	4	4
	10	K=1	0	0	0
		K=2	0	0	0
		K=3	0	0	0

		K=4	0.5	1	0.5
		K=5	2.5	2	2
	14	K=1	0	0	0
		K=2	0	0	0
		K=3	0	0	0
		K=4	0	0.7	0
		K=5	2.14	1.78	2.14

6.2.2 Analisis Pengujian *Fitting*

Sesuai dengan hasil yang tertera pada Tabel6.1 didapatkan nilai prosentase *FRR* nol (0) ketika k=1, k=2 , dan k=3 untuk data Indonesia pada semua variasi jumlah data latih. Namun, nilai tersebut mulai berubah ketika nilai k=4 dan k=5. Maka, berdasarkan hasil tersebut dapat disimpulkan bahwa algoritma yang digunakan pada sistem sudah benar dan dapat digunaka untuk pengujian selanjutnya.

6.3 Pengujian Jumlah Data Latih

6.3.1 Skenario Pengujian Jumlah Data Latih

Pada pengujian jumlah data latih, dilakukan variasi berjumlah 3 variasi untuk setiap kelas yaitu 5 data latih, 10 data latih, dan 14 data latih untuk data Indonesia. Untuk mendapatkan nilai variasi tersebut dilakukan pemilihan secara acak terhadap data latih yang berjumlah 24 data setiap kelas. Yang pertama dilakukan adalah memilah data menjadi 2 jenis data yaitu data latih dan data uji. Pada penelitian ini data latih yang terpilih sebanyak 14 data yang digunakan sebagai variasi data pertama. Kemudian dari 14 data latih, dipilih 5 data secara acak sebagai variasi kedua. Sisa dari data latih kemudian dipilih sebanyak 5 data, dan kemudian digabungkan dengan data variasi kedua menjadi data variasi ketiga. Sehingga didapatkan nilai variasi data latih yang diinginkan. Proses tersebut diterapkan untuk semua kelas.

Pada pengujian jumlah data latih, dilakukan pengujian *FAR* dan *FRR*. Untuk pengujian *FAR* data yang digunakan sebagai data uji adalah data tanda tangan palsu yang berjumlah 200 data. Sedangkan pengujian *FRR* digunakan data tanda tangan asli yang sudah kita pilah sebagai data uji. Pengujian ini dilakukan dengan menerapkan nilai k=4 dan hasil dari pengujian digunakan untuk melakukan pengujian selanjutnya. Hasil pengujian jumlah data latih sesuai dengan Tabel 6.2.

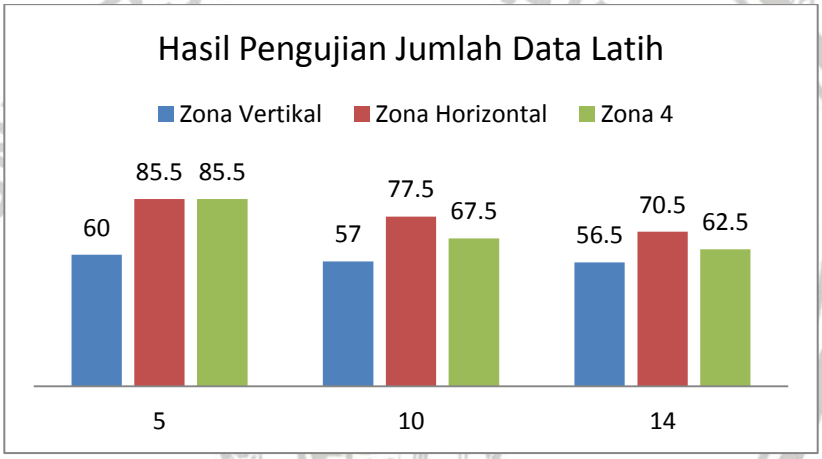
Tabel 6.2 Hasil Pengujian Jumlah Data Latih

Zona	Jumlah Data Latih	*%FRR*	*%FAR*
	5	60	12

Zona Vertikal	10	57	10
	14	56.5	10
Zona Horizontal	5	85.5	12
	10	77.5	16
	14	70.5	14
Zona 4	5	85.5	6.5
	10	67.5	11.5
	14	62.5	12

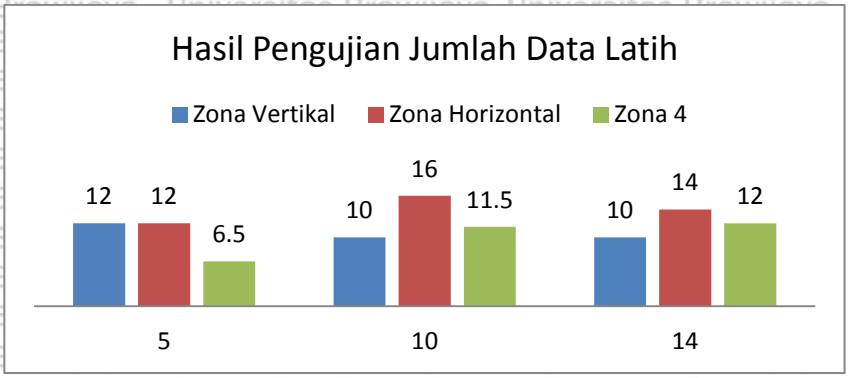
6.3.2 Analisis Pengujian Jumlah Data Latih

Berdasarkan hasil yang tertera pada Tabel 6.2 dapat digambarkan dengan diagram batang seperti pada Gambar 6.1 dan Gambar 6.2.



Gambar 6.1 Diagram Hasil Pengujian FRR Jumlah Data Latih

Nilai *FRR* menunjukkan seberapa banyak kesalahan klasifikasi terhadap data asli. Berdasarkan Gambar 6.1, nilai *FRR* terus mengalami penurunan ketika jumlah data latih semakin banyak dan hal tersebut terjadi pada semua zona. Nilai *FRR* terkecil didapatkan ketika jumlah data latih berjumlah 14. Pada Zona Vertikal, nilai FRR terkecil adalah 56.5% sedangkan pada Zona Horizontal nilai FRR terkecil adalah 70.5%. Begitupun dengan Zona 4 yang mendapatkan nilai terkecil ketika FRR=62.5%. Hasil pengujian dengan kondisi seperti pada penelitian ini sesuai dengan kondisi yang didapatkan pada penelitian sebelumnya (Widodo & Harjoko, 2015). Hal tersebut dikarenakan, semakin banyak data latih, maka semakin banyak pula referensi yang dimiliki oleh sistem. Sehingga dalam melakukan klasifikasi akan lebih tepat karena setiap data uji dibandingkan dengan banyak data.



Gambar 6.2 Diagram Hasil Pengujian FAR Jumlah Data Latih

Sedangkan nilai *FAR* menunjukkan seberapa banyak kesalahan penerimaan sistem terhadap data palsu. Nilai FAR pada zona vertical terus mengalami penurunan seiring dengan bertambahnya jumlah data latih. Sedangkan pada zona horizontal, nilai FAR justru mengalami fluktuasi meskipun jumlah data latih yang semakin bertambah. Hal yang dialami pada zona vertikal menunjukkan bahwa semakin sedikit data palsu yang dikenali sebagai data asli. Hal ini dikarenakan, data yang digunakan diperoleh dengan cara ekstraksi ciri lokal yang tentu menghasilkan ciri yang lebih spesifik dari ciri global. Sedangkan pada zona horizontal, kondisi yang terjadi dimungkinkan karena proses zoning yang kurang baik sehingga ciri yang dihasilkan menjadi kurang representative. Berbeda dengan yang dialami oleh zona 4, nilai FAR yang didapatkan berbanding lurus dengan bertambahnya jumlah data latih. Kondisi yang dialami oleh zona 4 selaras dengan kondisi yang terjadi pada penelitian sebelumnya (Widodo & Harjoko, 2015). Hal tersebut menunjukkan bahwa semakin banyak cirri lokal mengakibatkan proses pengenalan tanda tangan palsu menjadi lebih sulit. Ditambah lagi dengan jumlah data latih yang banyak membuat sistem memiliki referensi yang semakin banyak pula sehingga proses pengenalan menjadi lebih susah.

6.4 Pengujian Nilai *K*

Pengujian ini merupakan pengujian yang dilakukan untuk mengetahui pengaruh nilai k yang berbeda-beda dalam penerapan metode klasifikasi *modified-K Nearest Neighbour* terhadap nilai *FAR* dan *FRR*. Pengujian nilai k ini menjelaskan tentang hasil skenario pengujian nilai *k* dan analisis pengujian nilai *k*.

6.4.1 Scenario Pengujian Nilai *K*

Dalam melakukan pengujian nilai *k*, rentang nilai yang digunakan adalah 1 sampai 10 dengan menyertakan semua nilai genap maupun nilai ganjil. Pengujian ini menggunakan jumlah data latih terbaik yang sudah diuji pada pengujian sebelumnya. Hasil pengujian nilai *k* sesuai dengan Tabel 6.3.

Tabel 6.3 Hasil Pengujian Nilai *K*

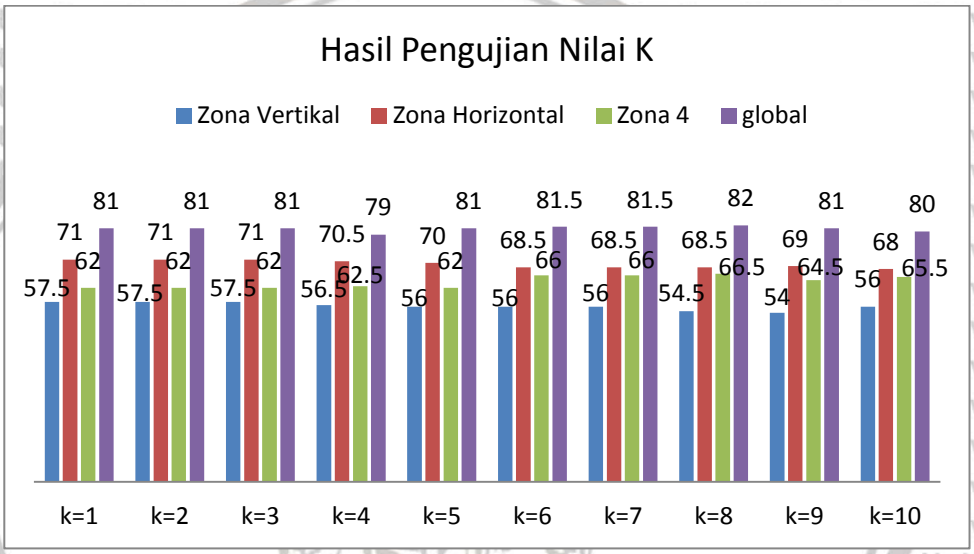
Zona	Nilai k	*%FRR*	*%FAR*
	1	57.5	9.5

Zona Vertikal	2	57.5	10
	3	57.5	10
	4	56.5	9.5
	5	56	9.5
	6	56	7
	7	56	8
	8	54.5	9.5
	9	54	9.5
	10	56	9.5
Zona Horizontal	1	71	12
	2	71	12
	3	71	12
	4	70.5	14
	5	70	15.5
	6	68.5	16
	7	68.5	16
	8	68.5	15.5
	9	69	15
	10	68	14.5
Zona 4	1	62	11.5
	2	62	11.5
	3	62	11.5
	4	62.5	12
	5	62	11.5
	6	66	12.5
	7	66	12.5
	8	66.5	12.5
	9	64.5	11.5
	10	65.5	12.5
Zona Global	1	81	5
	2	81	5
	3	81	5
	4	79	4.5

	5	81	4.5
	6	81.5	5.5
	7	81.5	4
	8	82	6
	9	81	6
	10	80	6

6.4.2 Analisis Pengujian Nilai *K*

Berdasarkan hasil yang tertera pada Tabel 6.3 dapat digambarkan dengan diagram batang seperti pada Gambar 6.3 dan Gambar 6.4.

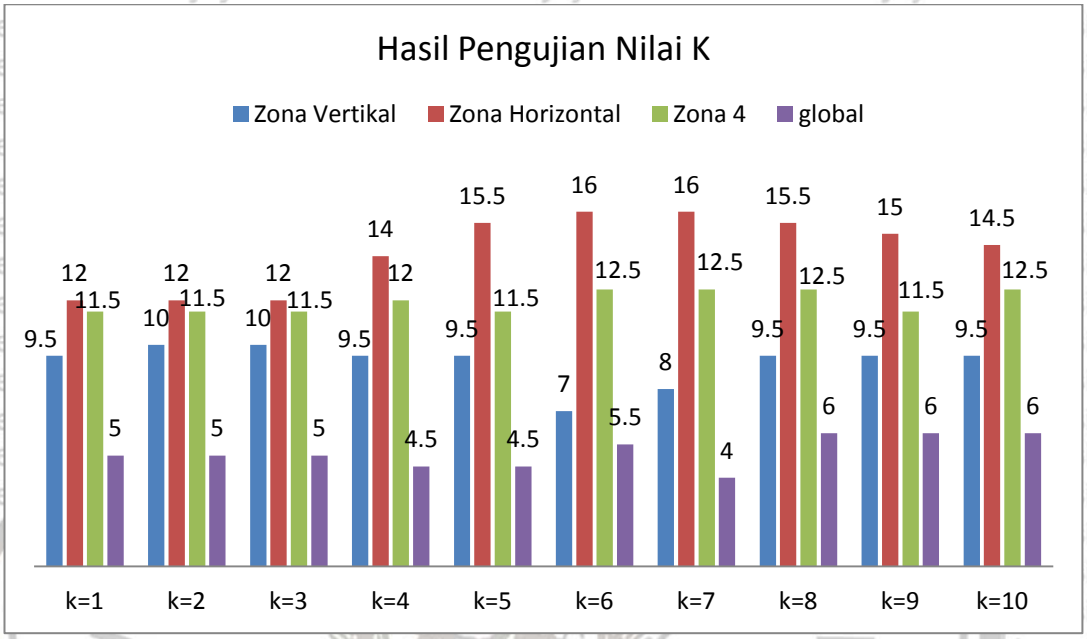


Gambar 6.3 Diagram Hasil Pengujian FRR Nilai *K*

Gambar 6.3 menunjukkan bahwa k terbaik tidak dipengaruhi oleh besar maupun kecil nilainya. Untuk Zona Vertikal, nilai *FRR* terkecil ketika k=9 dan nilai tersebut terus mengalami kenaikan saat k bernilai lebih kecil. Ketika nilai k lebih besar dari 9, nilai FRR pun ikut mengalami kenaikan. Hal ini dikarenakan, ketika nilai k kecil, maka yang terjadi adalah kondisi *underfitting*. Yaitu kondisi yang menyebabkan sistem tidak dapat mengenali dengan baik karena diberikan batasan yang sangat kecil. Sedangkan pada nilai k yang semakin besar, kondisi *overfitting* terjadi. Hal ini dikarenakan dalam melakukan pengenalan, sistem melakukan eksplorasi kesemua data yang ada dalam batasan yang besar tersebut. Sedangkan untuk Zona Horizontal, nilai FRR terkecil ketika k=10 dan nilai tersebut semakin bertambah saat nilai k semakin kecil. Hal tersebut menunjukkan bahwa sistem akan mengenali tanda tangan dengan benar ketika nilai k semakin besar. Hal ini dikarenakan sistem membutuhkan referensi yang lebih banyak dalam melakukan klasifikasi.

Dan untuk Zona 4, nilai FRR terkecil berada pada k=1 sampai dengan k=5. Ketika nilai k bertambah, maka nilai FRR pun ikut bertambah. Hal tersebut

menunjukkan bahwa proses pengenalan dengan memberikan batasan yang luas kepada sistem akan membuat sistem melakukan eksplorasi secara berlebihan sehingga membuat pengenalan menjadi semakin tidak akurat. Hal ini dikarenakan, data uji akan dibandingkan ke semua data yang ada pada rentang batasan yang diberikan.



Gambar 6.4 Diagram Hasil Pengujian FAR Nilai *K*

Berdasarkan Gambar 6.4 nilai FAR terkecil pada Zona Vertikal adalah ketika k=7. Sama hal nya dengan kondisi yang terjadi pada pegujian nilai FRR, pada Zona Vertikal semakin besar nilai k tidak berpengaruh pada semakin tinggi nilai FAR demikian pula sebaliknya. Pada Zona vertikal yang terjadi justru nilai FAR menjadi tinggi saat nilai k semakin kecil dan bahkan semakin besar. Ketika berada pada nilai k yang kecil atau kurang dari k optimum, kondisi yang terjadi adalah *underfitting*. Dan ketika berada pada nilai k yang lebih besar dari k optimu, kondisi yang terjadi adalah *overfitting*. Kedua kondisi tersebut mmebuat sistem tidak dapat melakukan pengenalan dengan baik sehingga mengakibatkan nilai FAR yang semakin tinggi. Sedangkan pada Zona Horizontal, yang terjadi adalah kecenderungan bertambahnya nilai FAR ketika nilai k juga bertambah. Hal ini menunjukkan, dalam melakukan pengenalan tanda tangan, sistem akan memberikan hasil yang baik ketika nilai k semakin kecil. Ketika sistem diberikaan batasan yang besar, makan sistem akan melakukan eksplorasi kepada seluruh data yang ada pada batasan tersebut.

Pada Zona 4 terjadi kondisi dimana nilai k yang mengalami pertambahan nilai secara fluktuatif. Hal ini menunjukkan bahwa variasi mKNN tidak mampu menoleransi perubahan yang terjadi pada citra tanda tangan sehingga hasil pengenalan memiliki nilai yang cenderung berubah-ubah.

6.5 Pengujian Variasi Ciri

6.5.1 Skenario Pengujian Variasi Ciri

Pengujian variasi ciri bertujuan untuk mengetahui ciri *geometric* yang berpengaruh terhadap nilai *FAR* dan *FRR*. Pada pengujian ini digunakan variasi ciri yang terdiri dari 1 ciri, 2 ciri, 3 ciri hingga 6 ciri *geometric* untuk menghasilkan nilai *FAR* dan *FRR* terbaik. Pemilihan variasi citra dilakukan secara acak dikarenakan tidak menggunakan metode seleksi fitur. Pengujian ini dilakukan dengan menggunakan jumlah data latih, zona, dan nilai *k* terbaik yang dihasilkan dari pengujian sebelumnya. Hasil pengujian variasi ciri ditunjukkan oleh Tabel 6.4.

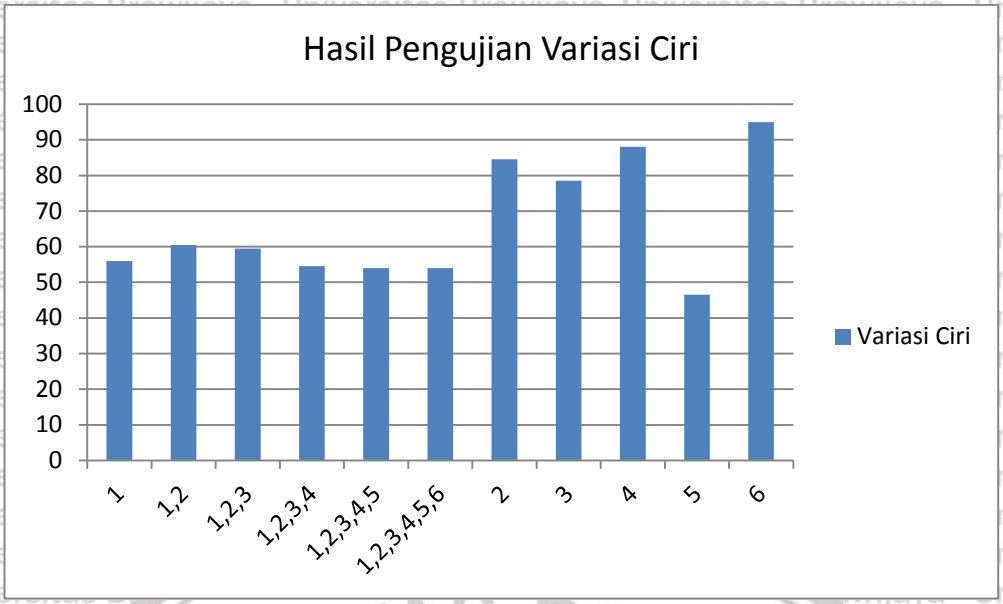
Tabel 6.4 Hasil Pengujian Variasi Ciri

No	Variasi Penggunaan Ciri	%FRR	%FAR
1	1	56	7.5
2	1,2	60.5	7
3	1,2,3	59.5	7.5
4	1,2,3,4	54.5	7
5	1,2,3,4,5	54	8.5
6	1,2,3,4,5,6	54	7
7	2	84.5	3
8	3	78.5	4.5
9	4	88	7
10	5	46.5	6
11	6	95	5

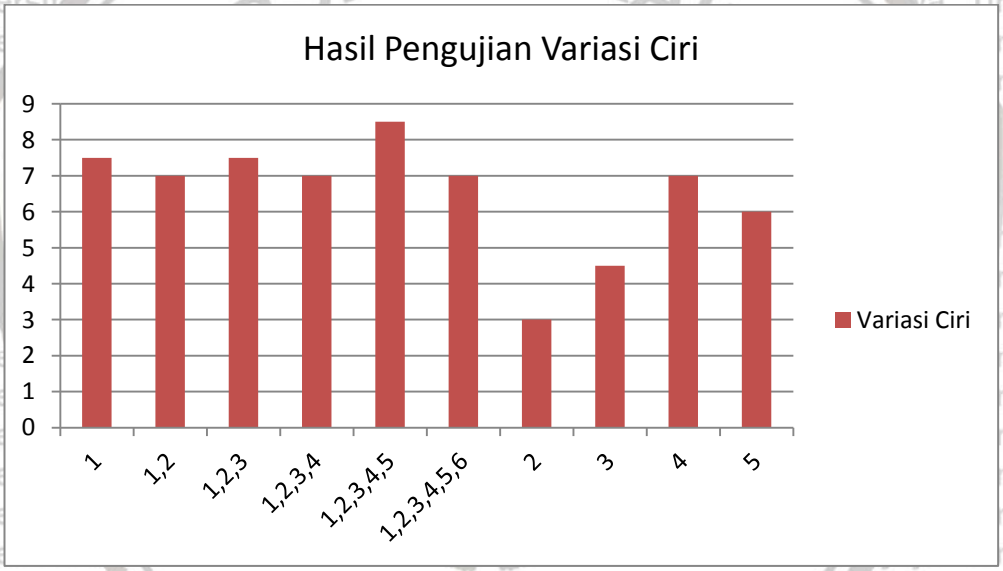
6.5.2 Analisis Pengujian Variasi Ciri

Berdasarkan hasil yang tertera pada Tabel 6.4 dapat digambarkan dengan diagram batang seperti pada Gambar 6.5 dan Gambar 6.6.

Berdasarkan Gambar 6.5 nilai *FRR* terkecil diperoleh ketika menggunakan ciri 5 yaitu sebesar 46.5%. Sedangkan nilai *FRR* terbesar adalah 95% didapat ketika hanya menggunakan 1 ciri yaitu ciri *eccentricity*. Nilai yang rendah tersebut dikarenakan variasi ciri yang digunakan adalah *pixels density* dimana ciri ini hanya merepresentasikan kerapatan piksel setiap citra.



Gambar 6.5 Diagram Hasil Pengujian FRR Variasi Ciri



Gambar 6.6 Diagram Hasil Pengujian FAR Variasi Ciri

Berdasarkan Gambar 6.6 nilai *FAR* terkecil didapatkan ketika melakukan variasi ciri dengan menggunakan ciri 2 yaitu ciri *skewness*. Sedangkan nilai *FAR* terbesar diperoleh ketika menggunakan 5 ciri. Hal tersebut menunjukkan bahwa dalam melakukan pengenalan tanda tangan palsu, ciri *skewness* sudah cukup representatif untuk membuat perbedaan terhadap citra yang lain.

BAB 7 PENUTUP

7.1 Kesimpulan

Kesimpulan yang bisa diambil berdasarkan penelitian penerapan ciri *geometric* untuk deteksi dan verifikasi tanda tangan offline adalah sebagai berikut.

1. Proses deteksi dan verifikasi tanda tangan *offline* dilakukan dengan 3 tahapan yaitu preprocessing, ekstraksi ciri, dan klasifikasi menggunakan metode *modified-K Nearest Neighbor*. Preprocessing merupakan
2. tahapan yang terdiri dari proses filtering, binarisasi, thinning, cropping, dan resize. Kemudian dilakukan ekstraksi ciri *geometric*, dan yang terakhir adalah melakukan klasifikasi untuk verifikasi tanda tangan
3. Dalam melakukan ekstraksi ciri *geometric*, terlebih dahulu dilakukan *zoning* atau pembagian area menjadi 3. Kemudian dilakukan ekstraksi pada masing-masing zona untuk menghasilkan 6 ciri *geometric* yaitu standar deviasi, *skewness*, *kurtosis*, *center of gravity*, *pixels density*, dan *eccentricity*. Proses zoning menggunakan 3 teknik yaitu teknik vertikal, horizontal, dan zoning 4 bagian. Pada teknik vertikal dan horizontal, ciri lokal yang dihasilkan berjumlah 18 ciri untuk masing-masing teknik sedangkan zoning 4 bagian menghasilkan 24 ciri lokal.
4. Kemampuan sistem untuk mengenali tanda tangan asli terbilang rendah, hal ini dikarenakan nilai *FRR* minimum yang dihasilkan oleh sistem adalah sebesar 46.5% sedangkan nilai *FAR* minimum sebesar 3%. Dengan begitu dalam melakukan pengenalan terhadap tanda tangan palsu, kemampuan sistem sudah baik. Dengan hasil tersebut didapatkan nilai *AER* sebesar 75.25%. Kedua nilai tersebut dihasilkan pada Zona Vertikal yang mana menunjukkan bahwa variasi Zona terbaik adalah ketika citra dilakukan *zoning* secara vertikal. Sedangkan variasi ciri yang paling baik untuk dapat melakukan pengenalan tanda tangan asli adalah dengan menggunakan ciri *pixels density* saja. Untuk pengenalan tanda tangan palsu, variasi cirri terbaik adalah dengan menggunakan ciri *skewness*.

7.2 Saran

Berdasarkan penelitian yang telah dilakukan, maka saran untuk penelitian selanjutnya sebagai berikut.

1. Nilai akurasi yang didapatkan tergolong rendah. Hal ini dikarenakan nilai %FRR yang dihasilkan masih tinggi. Untuk membuat nilai %FRR menjadi lebih rendah, dapat dilakukan dengan menambah variasi jumlah data latih lebih banyak, pembobotan ciri lokal, pemberian *threshold* jarak dan juga dapat menggunakan metode lain dalam melakukan klasfikasi.
2. Perlu dilakukan penelitian lebih lanjut untuk berfokus pada proses *preprocessing* ciri geometri sehingga ciri yang dihasilkan dapat lebih *representative* dan proses pengenalan menjadi lebih baik.

DAFTAR PUSTAKA

AlTarawneh, M. S., Al-Mahadeen, B. & AlTarawneh, I. H., 2010. Signature Region of Interest using Auto Cropping. *JCSI International Journal of Computer Science Issues,* 7(2).

Alvarez, G., Sheffer, B. & Bryant, M., 2016. Offline Signature Verification with Convolutional Neural Network. *Stanford University,* pp. 1-8.

Angadi, S. A., Gour, S. & Bhajantri, G., 2014. Offline Signature Recognition System using Radon Transform. *International Conference on Signal and Image Processing,* Volume 5, pp. 56-61.

Ansari, S. & Sutar, U., 2015. Devanagari Handwritten Character Recognition using Hybrid Features Extraction and Feed Forward Neural Network Classifier (FFNN). *International Journal of Computer Applications,* 129(7), pp. 22-27.

Assiwal, N. & Sharma, D. N., 2016. A Geometric Feature Extraction Technique for Hindi Handwritten Character Recognition. *IJSTE - International Journal of Science Technology & Engineering,* pp. 295-302.

Basheer, I. A. & Hajmeer, H. M., 2000. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods,* Volume 43, pp. 3-31.

Chandra, S. & Maheskar, S., 2016. Offline Signature Verification Based on Geometric Feature Extraction Using Artificial Neural Network. *Recent Advance in Information Technology.*

Cilimkovic, M., n.d. Neural Networks and Back Propagation Algorithm. *www.dataminingmasters.com.*

Darma, P., 2010. *Pengolahan Citra Digital.* I ed. Yogyakarta: Penerbit ANDI.

Deore, M. R. & Handore, S. M., 2015. Offline Signature Recognition : Artificial Neural Network Approach. *ICCSP,* pp. 1708-1712.

Fausett, L. V., 1994. *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications.* s.l.:Prentice-Hall.

Gonzales, R. &. W. R., 2008. *Digital Image Processing Third Edition.* 3 ed. New Jersey: Pearson Education Inc.

Gonzalez, R. C. & Woods, R. E., 2008. *Digital Image Processing.* 3rd ed. s.l.:Prentice-Hall.

Greensted, A., 2010. *http://www.labbookpages.co.uk.* [Online] Available at: http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html [Accessed 25 March 2017].

Hassoun, M., 1995. *Fundamentals of Artificial Neural Networks.* s.l.:MIT Press.

Huang, Q., Gao, W. & Cai, W., 2005. Thresholding Technique with Adaptive Window Selection for Uneven Lighting Image. *Pattern Recognition,* pp. 801-808.

Impedovo, D., Pirlo, G. & Modugno, R., 2012. New Advancements in Zoning-Based Recognition of Handwritten Characters. *International Conference on Frontiers in Handwriting Recognition,* pp. 661-665.

Ismail, I. A., Ramadan, M. A., Danf, T. E. & Samak, A. H., 2008. Automatic Signature And Verification Using Principal Components Analysis. *Fifth International Conference on Computer Graphics, Imaging and Visualisation,* pp. 356-361.

Karouni, A. D. B. B. S., 2010. Offline Signature Recognition using Neural Networks Approach. *Procedia Computer Science 3,* Volume 3, pp. 155-161.

Khaleel, H. H. et al., 2013. Vessel Centerline Extraction Using New Center of Gravity Equations. *IAENG International Journal of Computer Science.*

Kowsalya, S. & Periyasamy, P. S., 2016. Handwritten Tamil Character Recognition Using Geometric Feature Extraction Approach. *Asian Journal of Information Technology 15 (20),* pp. 4124-4128.

Lakshmi, K. V. & Nayak, S., 2013. Offline Signature Verification Using Neural Networks. *Internationa Advance Computing Conference,* Volume 3, pp. 1065-1069.

Larose, D. T., 2005. *Discovering Knowledge in Data An Introduction to Data Mining.* New Jersey: Wiley Interscience.

NIST/SEMATECH, 2010. *e-Handbook of Statistical Methods.* s.l.:http://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm.

Noercholis, A., Muslim, A. A. & Maftuch, 2013. Ekstraksi Fitur Roundness untuk Menghitung Jumlah Leukosit dalam Citra Sel Darah Ikan. *Jurnal EECCIS,* 7(1), pp. 35-40.

Odeh, S. M. & Khalil, M., 2011. Offline Signature Verification and Recognition: Neural Network Approach. *IEEE,* pp. 34-38.

Pansare, A. & Bhatia, S., 2012. Handwritten Signature Verification using Neural Netwrok. *International Journal of Applied Information Systems,* 1(2), pp. 44-49.

Park, F., 2011. Shape Descriptor/ Feature Extraction Techniques. *UCI iCAMP .*

Putra, D., 2004. Binerisasi Citra Tangan dengan Metode Otsu. *Journal Teknik Elektro, Fakultas Teknik, Universitas Udayana,* 3(2), pp. 11-13.

Quraishi, M. D. A. &. R. S., 2013. A Novel Siganture Verification and Authentication System Using Image Transformation and Artificial Neural Network. *IEEE.*

Rojas, R., 2005. *Neural Networks: A Systematic Introduction.* s.l.:Springer.

Salouan, Safi, S. & Bouikhalene, B., 2015. Printed Noisy Greek Characters Recognition Using Hidden Markov Model, Kohonen Network, K Nearest

Neighbours and Fuzzy Logic. *International Journal of Signal Processing, Image Processing and Pattern Recognition,* 8(10), pp. 241-256.

Shapiro, L. G. & Stockman, G. C., 2001. *Computer Vision.* Washington: Pearson.

Sharma, R. & Shrivastav, M., 2011. An Offline Signature Verification System Using Neural Network Based on Angle Feature and Energy Density. *International Journal on Emerging Technologies,* 2(2), pp. 84-89.

Soni, S. & Kaur, S., 2016. To Propose an Improvement in Zhang-Suen Algorithm for Image Thinning in Image Processing. *IJSTE,* 3(1), pp. 74-81.

Tomar, D. & Agarwal, S., 2013. A survey on Data Mining approaches for Healthcare. *International Journal of Bio-Science and Bio-Technology,* 5(5), pp. 241-266.

Wai, H. H. & Aung, S. L., 2013. Feature Extraction for Offline Signature Verification System. *IJCCER International Journal of Computer & Communication Engineering Research,* I(3), pp. 84-87.

Widodo, A. W. & Harjoko, A., 2015. Sistem Verifikasi Tanda Tangan Offline Berdasar Ciri Histogram of Oriented Gradient (HOG) dan Histogram of Curvature (HoC). *Jurnal Teknologi Informasi dan Ilmu Komputer,* 2(1), pp. 1-10.

Wikipedia, 2017. Signature. 2 March.