# IMPLEMENTASI EKSTRAKSI FITUR JUMLAH *KEYPOINT DESCRIPTOR* PADA PENGENALAN TANDA TANGAN DENGAN ALGORITME *LEARNING VECTOR QUANTIZATION*

**SKRIPSI**

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Imada Nur Afifah
NIM: 145150200111084

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

# PENGESAHAN

IMPLEMENTASI EKSTRAKSI FITUR JUMLAH *KEYPOINT DESCRIPTOR* PADA PENGENALAN TANDA TANGAN DENGAN ALGORITME *LEARNING VECTOR QUANTIZATION*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Imada Nur Afifah
NIM: 145150200111084

Skripsi ini telah diuji dan dinyatakan lulus pada
20 April 2018
Telah diperiksa dan disetujui oleh:

| Mengetahui<br>Ketua Jurusan Teknik Informatika | Dosen Pembimbing |
|---|---|
| Tri Astoto Kurniawan, S.T, M.T, Ph.D | Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D. |
| NIP: 19710518 200312 1 001 | NIP: 19720919 199702 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, 4 Mei 2018

Imada Nur Afifah

NIM: 145150200111084

# KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT karena limpahan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Implementasi Ekstraksi Fitur Jumlah *Keypoint Descriptor* pada Pengenalan Tanda Tangan dengan Algoritme *Learning Vector Quantization*". Tujuan dalam penulisan skripsi ini adalah untuk memenuhi persyaratan perolehan gelar Sarjana Komputer Fakultas Ilmu Komputer Universitas Brawijaya Malang.

Dalam penyusunan skripsi ini tidak lepas dari bantuan dan dukungan dari beberapa pihak. Sehingga penulis mengucapkan terima kasih kepada:

1. Bapak Wayan Firdaus Mahmudy, S.Si, M.T., Ph.D, selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya dan Dosen Pembimbing I yang telah membimbing, memberi arahan, dukungan, dan saran selama proses pengerjaan dan penyusunan skripsi penulis.
2. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D, selaku Ketua Jurusan Teknik Informatika dan Bapak M. Tanzil Furqon, S.Kom, M.ComSc, selaku Sekretaris Jurusan Teknik Informatika.
3. Bapak M. Abd. Moentolib dan Ibu Asih Supeni, selaku orang tua penulis, serta saudara penulis yaitu Shintya Dwi Nur'aini, M. Nurrahman Fanani, dan Nihayah Nurrohmah yang telah memberikan dukungan berupa do'a, semangat, saran dan bantuan lainnya selama proses pengerjaan skripsi ini.
4. Seluruh Dosen Fakultas Ilmu Komputer Universitas Brawijaya yang telah mengajarkan dan membimbing serta membagikan ilmu-ilmu yang bermanfaat kepada penulis selama menjadi mahasiswa Teknik Informatika.
5. Seluruh teman-teman Teknik Informatika khususnya Fitri Anggarsari, Silvia Aprilla, Ratna Tri Utami, Wanda Athira Luqyana, Chandra Yogi Adhitama, Egi Muliandri, Rayza Arfian, Rizky Haryandi Rahman, dan Rahmat Yani yang selalu memberikan bantuan berupa doa, tenaga, dukungan dan informasi yang sangat bermanfaat dalam kelancaran pengerjaan skripsi.
6. Dua puluh lima mahasiswa Teknik Informatika dari angkatan 2014, 2015, dan 2016 yang bersedia memberikan tanda tangan sebagai objek penelitian pada skripsi penulis.
7. Seluruh anggota Eksekutif Mahasiswa Teknik Informatika (EMIF) 2016/2017 yang memberikan dukungan berupa informasi dan saran selama pengerjaan dan penyelesaian skripsi ini.
8. Semua pihak yang telah membantu dalam proses pengerjaan, penyusunan dan penyelesaian skripsi ini.

Semoga dengan penulisan skripsi ini diharapkan dapat berguna bagi semua pihak. Dalam penulisan skripsi terdapat banyak kekurangan, maka penulis dengan

terbuka menerima kritik dan saran yang membangun serta sebagai pedoman untuk perbaikan naskah skripsi ini.

Malang, 4 Mei 2018

Penulis

imadanurafifah@gmail.com

# ABSTRAK

Pengenalan tanda tangan sangat penting untuk proses verifikasi tanda tangan. Salah satu metode pengenalan tanda tangan adalah implementasi algoritme *Learning Vector Quantization* (LVQ) pada pengenalan tanda tangan dengan ekstraksi fitur jumlah *keypoint descriptor* menggunakan metode *Scale Invariant Features Transform* (SIFT) dan fitur fraktal global. Dalam proses pelatihan pengenalan tanda tangan digunakan fitur-fitur seperti nilai maksimum piksel hitam dari histogram horizontal dan vertikal, massa pusat objek, normalisasi luas objek tanda tangan, rasio aspek, fitur *three surface*, fitur *six fold surface* dan fitur transisi serta tambahan fitur berupa jumlah *keypoint descriptor* yang didapatkan dari hasil penggunaan metode *Scale Invariant Features Transform* (SIFT). Dataset yang digunakan antara lain 100 citra untuk data latih dan 100 citra untuk data uji dari 20 kelas yang berbeda. Serta 25 citra diluar kelas yang dilatih sebagai data yang tidak dikenali. Hasil pengujian diperoleh 71,2% dari pengujian tanpa *threshold*, 56% dari pengujian dengan nilai *threshold* menggunakan nilai maksimum jarak terpendek antara citra dengan kelas, 45,6% dari pengujian dengan nilai *threshold* menggunakan nilai maksimum kedua dari jarak terpendek antara citra dengan kelas.

**Kata Kunci:** *sift, Learning Vector Quantization, pengenalan, tanda tangan*

# ABSTRACT

*Signature recognition is important for signature verification process. One of the signature recognition method is implementation Learning Vector Quantization (LVQ) for signature recognition with number of keypoint descriptor features extraction method using Scale Invariant Features Transform (SIFT) and fractal global. In the train process, this research used some features such as maximum of black pixel in horizontal and vertical histogram, center of mass, normalized area of signature, aspect ratio, tri surface feature, the Six Fold Surface feature, transition feature and additional features called number of keypoint descriptors. Number of keypoint descriptors are output of Scale Invariant Features Transform (SIFT) method. The dataset used is 100 images for training data and 100 images for testing data from 20 different classes. And 25 images from out of trained class as unknown data. The result of algorithm testing is 71,2% from testing of non-threshold process, 56% from testing process with maximum value of minimum euclidean distance between data and class as threshold value, 45,6% % from testing process with second maximum value of minimum euclidean distance between data and class as threshold value.*

**Keywords:** *sift, Learning Vector Quantization, recognition, signature*

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# DAFTAR LAMPIRAN

# BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan berisikan kumpulan kajian pustaka dan dasar teori yang digunakan dalam penelitian ini. Pada bab kajian pustaka terdapat pembahasaan dari penelitian-penelitian yang pernah dilakukan dan memiliki hubungan dengan permasalahan dalam penilitian ini. Untuk dasar teori adalah dasar penelitian yang berupa pembahasan dari kumpulan teori-teori.

## 2.1 Kajian Pustaka

Pada Tabel 2.1 menunjukkan kajian pustaka berupa penjelasan dari penelitian-penelitian sebelumnya tentang metode pengenalan tanda tangan dan ektraksi fiturnya.

**Tabel 2.1 Kajian Pustaka**

| No. | Judul | Penulis | Kajian Pustaka |
|---|---|---|---|
| 1. | *Hand Written Signature Recognition and Verification using Neural Network* (Kumar, et al., 2013) | Pradeep Kumar, Shekhar Singh, Ashwani Garg, Nishant Prabhat | Pada penelitian ini pengenalan dan verifikasi tanda tangan menggunakan Algoritme pembelajaran *error backpropagation*. Ekstraksi fitur yang digunakan antara lain nilai maksimum histogram horizontal dan vertikal, massa pusat objek, normalisasi luas objek tanda tangan, rasio aspek, fitur *tri surface*, fitur *Six Fold Surface* dan fitur transisi. Hasil dari penelitian ini adalah mampu mengenali seluruh citra latih sehingga akurasi terhadap data latih sebesar 100% dan 82,66% dalam pengenalan terhadap data uji. |
| 2. | *Offline Signature Recognition using Back Propagation Neural Network* (Rahmi, et al., 2016) | Asyrofa Rahmi, Vivi Nur Wijayaningrum, Wayan Firdaus Mahmudy, Ani M. A. K. Parewe | Pada penelitian pengenalan tanda tangan ini menggunakan metode *backpropagation* dengan ekstraksi fitur menggunakan metode fitur grid citra segmentasi. Hasil dari penelitian ini adalah *Learning Rate* awal sebesar 0,64, banyak iterasi yaitu 100 iterasi, *hidden layer* sebanyak 3 layer, dan akurasi sebesar 63%. |
| 3. | Perbandingan Metode SURF | Felix Pidha Hilman | Penelitian ini menggunakan perbandingan dua metode untuk |

| | | | |
|---|---|---|---|
| | dan SIFT dalam Sistem Identifikasi Tanda Tangan (Hilman, 2015) | | proses identifikasi tanda tangan. Metode yang digunakan adalah *Speed Up Robust Features* (SURF) dan *Scale Invariant Feature Transform* (SIFT). Untuk proses klasifikasi didukung dengan penggunaan metode *k-Nearest Neighbour*. Hasil penelitian ini berupa pengaruh jumlah point pada proses klasifikasi dan pengaruh nilai *k*. Hasil pengujian dari penelitian adalah ektraksi ciri menggunakan metode SIFT lebih baik dari pada SURF. |
| 4. | Implementasi *Learning Vector Quantization* (LVQ) untuk Klasifikasi Kualitas Air Sungai (Hamidi, et al., 2017) | Rifwan Hamidi, M. Tanzil Furqon, Bayu Rahayudi | Pada penelitian ini menggunakan Algoritme *Learning Vector Quantization* (LVQ) untuk mengklasifikasi kualitas air sungai. Ekstrasi fitur yang digunakan sebanyak 7 fitur. Hasil keluaran dari penelitian ini adalah kelas klasifikasi air sungai yaitu berupa kelas memenuhi baku mutu, tercemar tingkat ringan, tingkat sedang dan tingkat berat. Untuk hasil akurasinya yaitu sebesar 81,9% pada data latih dan 81,13% pada data uji. |
| 5. | Jaringan Syaraf Tiruan *Learning Vector Quantization* untuk Aplikasi Pengenalan Tanda Tangan (Qur'ani & Rosmalinda, 2010) | Difla Yustisia Qur'ani, Safrina Rosmalinda | Pada penelitian ini pelatihan pola-pola tanda tangan menggunakan Algoritme *Learning Vector Quantization* dengan *input* masukkan berupa ekstraksi ciri menggunakan deteksi garis tepi pada citra segmentasi. Hasil keluaran dari sistem pengenalan tanda tangan ini adalah identifikasi tanda tangan dan kecocokan tanda tangan. Dari hasil pengujian yang dilakukan diperoleh akurasi sebesar 98% dengan kesalahan pengenalan yang disebabkan perbedaan lokasi objek tanda tangan data uji dengan data yang dilatih. |

| | | | |
|---|---|---|---|
| 6. | Perbandingan Antara Metode *Kohonen Neural Network* dengan Metode *Learing Vector Quantization* Pada Pengenalan Pola Tanda Tangan (Prabowo, et al., 2006) | Anindito Prabowo, Eko Adi Sarwoko dan Djalal Er Riyanto | Pada penelitian ini pola-pola tanda tangan diklasifikasi menggunakan metode LVQ dan Kohonen. Ekstraksi fitur yang digunakan sama antara metode LVQ dan Kohonen yaitu dengan menggunakan proses *downsample*. Hasil dari penelitian yaitu tingkat keakuratan metode LVQ dalam klasifikasi tanda tangan lebih baik dari metode Kohonen. Dalam hal waktu eksekusi, metode LVQ lebih lama daripada Kohonen karena proses pelatihan dipengaruhi oleh banyaknya pola pelatihan. |

Dari penelitian-penelitian sebelumnya, maka penulis melakukan penelitian dengan menggunakan metode *Learning Vector Quantization* (LVQ) untuk proses pengenalan tanda tangan dengan tambahan ekstraksi fitur jumlah *Keypoint Descriptor* dengan menggunakan *Scale Invariant Features Transform* (SIFT). Dengan ekstraksi fitur yang lain berupa nilai maksimum histogram horizontal dan vertikal, massa pusat objek, normalisasi luas objek tanda tangan, rasio aspek, fitur *tri surface*, fitur *Six Fold Surface*, dan fitur transisi. Dengan penambahan fitur dan penggunaan metode *Learning Vector Quantization* diharapkan dapat mengenali tanda tangan yang sudah dilatih oleh sistem.

## 2.2 Tanda Tangan (*Signature*)

Tanda tangan merupakan salah satu yang dapat mengenali identitas seseorang dalam menandatangani suatu dokumen dengan sifat yang kemungkinan kecil untuk berubah dan tidak mudah dihapuskan, serta memiliki perbedaan bentuk dan karakteristik antara setiap orang (Kumar, et al., 2013). Tanda tangan memiliki pola karakter yang dapat dikenali oleh komputer. Akan tetapi pengenalan pola pada tanda tangan mengalami kesulitan karena tingkat kompleksitasnya dan bentuk yang rumit setiap individu (Prabowo, et al., 2006).

## 2.3 *Image Preprocessing*

*Image preprocessing* dapat juga disebut dengan pengolahan citra merupakan dasar dari berbagai aplikasi nyata seperti dalam pengenalan pola (Kadir & Susanto, 2013). Sebelum dilakukannya *preprocessing* citra harus melakukan akuisisi citra. Akuisisi citra adalah tahap awal mendapatkan citra digital dari data analog (Sutramiani, Putra, & Sudarma, 2015).

### 2.3.1 Eliminasi Latar (*Background Elimination*)

*Background Elimination* dapat dilakukan dengan cara melakukan konversi dari citra berwarna (*image* RGB) ke citra biner dengan menggunakan nilai batas (*threshold*) (Kumar, 2012). Sebelum konversi ke citra biner, citra akan dikonversi ke citra tingkat keabuan atau citra *grayscale* terlebih dahulu.

Citra tingkat keabuan biasanya digunakan lebih sering untuk pengolahan citra di dalam komputer dibandingkan dengan RGB (Kadir & Susanto, 2013, hal. 25-31). Suatu citra diharuskan diubah kebentuk citra biner untuk menyesuaikan dengan proses yang akan dilakukan pada citra tersebut. Citra *grayscale* adalah citra berskala keabuan. Cara untuk mengonversi citra RGB menjadi citra *grayscale* adalah dengan mengalikan nilai komponen merah (*red* atau *R*) dengan nilai konstanta untuk komponen merah, dijumlahkan dengan nilai kompenen hijau (*green* atau *G*) dengan nilai konstanta untuk komponen hijau, dijumlahkan lagi dengan nilai komponen biru (*blue* atau *B*) nilai konstanta untuk komponen biru, yang jumlah nilai konstanta untuk komponen merah, hijau, dan biru adalah 1 (Kadir & Susanto, 2013).

$$I = a \times R + b \times G + c \times B, \quad a + b + c = 1 \tag{2.1}$$

Citra biner adalah citra yang hanya mengandung dua unsur warna yaitu 1 yang mewakili hitam dan 0 yang mewakili putih (Kadir & Susanto, 2013). Biasanya citra biner digunakan untuk memperoleh garis tepi dari suatu objek. Konversi citra *grayscale* ke citra biner menggunakan *threshold* dengan nilai *threshold* sebesar 0,5.

$$citra = \begin{matrix} p_{ij} < T & 0 \\ p_{ij} > T & 1 \end{matrix} \tag{2.2}$$

Keterangan:

$p_{ij}$ = Nilai *grayscale* piksel

$T$ = Nilai *threshold*

### 2.3.2 *Image Resizing*

Ukuran citra dari setiap data yang digunakan dalam penelitian harus homogen dan konsisten baik ukuran panjang maupun lebarnya. Salah satu ukuran citra yang dapat digunakan adalah 256 x 256 piksel. Hasil *image resizing* dapat dilihat pada Gambar .

Gambar 2.1 Hasil *Image Resizing*

### 2.3.3 *Thinning*

*Thinning* merupakan salah satu operasi morfologi yang berguna untuk mengurangi ketebalan pada objek didalam citra dengan hasil keluaran berupa *skeleton* atau rangka (Kadir & Susanto, 2013). Salah satu metode *thinning* adalah *Thinning* Zhang-Suen. Metode *thinning* Zhang-Suen merupakan metode *thinning* dengan menggunakan dua kali iterasi yaitu iterasi pertama untuk menghapus titik batas dari selatan-timur dan *vertex* dari utara-barat pada objek, sedangkan iterasi kedua untuk menghapus titik batas bagian utara barat dan *vertex* selatan-timur dari objek (Sudarma & Sutramiani, 2014).

Gambar 2.2 Hasil Proses *Thinning*

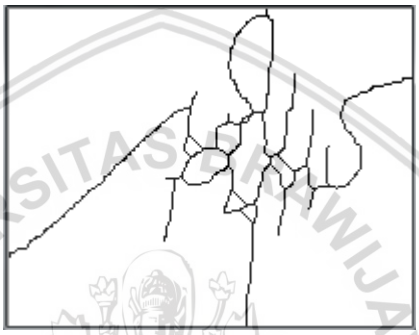
Persyaratan perubahan atau penghapusan nilai piksel untuk menjadi kerangka memiliki beberapa syarat dalam method *Thinning* Zhang-Suen yaitu:

1. Sub-Iterasi 1
   a Piksel berwarna hitam (0) dan memiliki 8 tetangga.
   b Jumlah piksel tetangga yang berwarna hitam lebih besar dari 2 dan lebih kecil sama dengan 6.
   c Perubahan transisi dari putih ke hitam sebanyak 1.
   d Tetangga P2 dan P4 dan P6 adalah putih.
   e Tetangga P4 dan P6 dan P8 adalah putih
2. Sub-Iterasi 2
   a Piksel berwarna hitam (0) dan memiliki 8 tetangga.

b Jumlah piksel tetangga yang berwarna hitam lebih besar dari 2 dan lebih kecil sama dengan 6.
c Perubahan transisi dari putih kehitam sebanyak 1.
d Tetangga P2 dan P4 dan P8 adalah putih.
e Tetangga P2 dan P6 dan P8 adalah putih

### *2.3.4 Cropping*

Proses *cropping* adalah proses pemotongan citra. Proses pemotongan citra dilakukan dengan mengurangi area citra dan membentuk persegi panjang yang berisi hanya objek citra saja. Proses *cropping* ini dapat membantu mengurangi proses pengolahan citra baik dalam segi waktu dan perhitungan (Kumar, et al., 2013). Hasil *cropping* citra ditunjukkan pada Gambar .



Gambar 2.3 Hasil Proses *Cropping*

## 2.4 Ekstraksi Fitur

Ekstraksi fitur merupakan proses terpenting dalam pengembangan sistem pengenalan tanda tangan karena ekstraksi fitur adalah kunci untuk mengidentifikasi dan pembeda antara tanda tangan milik orang satu dengan yang lain (Kumar, 2012). Beberapa fitur yang diekstraksi yaitu maksimum horizontal and vertikal histogram, *center of mass*, luas objek ternormalisasi, rasio aspek, *tri surface feature*, *the Six Fold Surface feature*, fitur transisi (Kumar, et al., 2013).

### 2.4.1 Nilai Maksimum dari Histogram Piksel Horizontal dan Vertikal

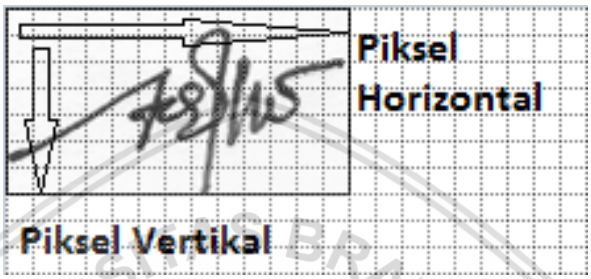
Histogram adalah suatu diagram yang merepresentasikan frekuensi setiap nilai intensitas yang ada dalam citra. Pada skala keabuan atau *grayscale*, jumlah tingkatan nilai keabuan sebanyak 256 yang dituliskan dalam range 0-255 (Kadir & Susanto, 2013, hal. 36-53).

Kegunaan dari histogram yang memiliki peran penting dalam *image preprocessing*, antara lain:

a. Untuk mengamati persebaran intensitas warna dalam citra, sehingga dapat digunakan untuk pengambilan keputusan atau proses lebih lanjut dari praproses.
b. Untuk penentuan batas-batas dalam pemisahan objek dengan latar belakang.
c. Untuk identifikasi citra dapat dilakukan dengan memberikan persentase komposisi warna dan tekstur intensitas.

Histogram tidak dapat menebak bentuk objek yang ada dalam suatu citra karena histogram tidak merepresentasikan susunan posisi warna piksel dalam citra (Kadir & Susanto, 2013, hal. 40).

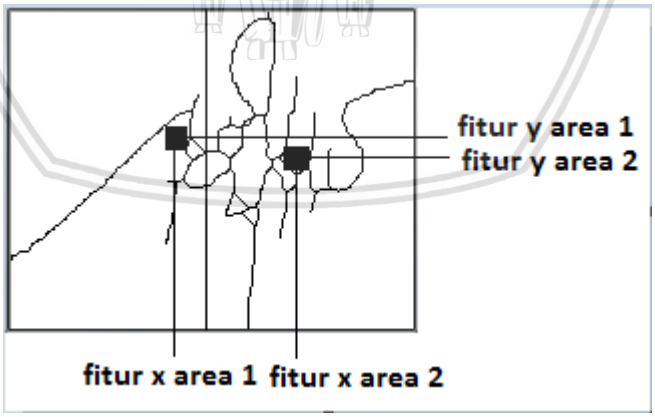
Fitur nilai maksimum dari histogram piksel horizontal diperoleh dari pehitungan jumlah seluruh piksel setiap baris dan mengambil nilai terbesar dari hasil perhitungan jumlah seluruh piksel setiap baris (Kumar, 2012). Sama dengan fitur nilai maksimum dari histogram piksel vertikal yaitu mengambil nilai maksimum dari kolom yang memiliki jumlah piksel terbesar. Representasi perhitungan dari nilai maksimum piksel secara horizontal dan vertikal ditunjukkan pada Gambar.




Gambar 2.4 Gambaran Fitur Jumlah Piksel secara Horizontal dan Vertikal

### 2.4.2 Nilai Massa Pusat

Fitur nilai massa pusat didapatkan dari perhitungan massa pusat masing-masing area dari citra yang dibagi menjadi dua bagian yang sama (Kumar, et al., 2013). Perhitungan nilai massa pusat menggunakan nilai rata-rata koordinat setiap piksel yang menyusun objek (Kadir & Susanto, 2013). Nilai massa pusat memperoleh dua nilai yaitu nilai pada axis horizontal dan axis vertikal. Gambaran perolehan massa pusat ditunjukkan pada Gambar.

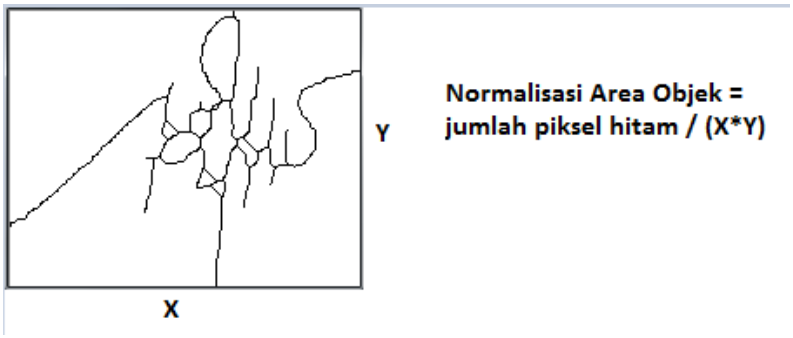



Gambar 2.5 Gambaran Fitur Massa Pusat

### *2.4.3 Normalized Area of Signature*

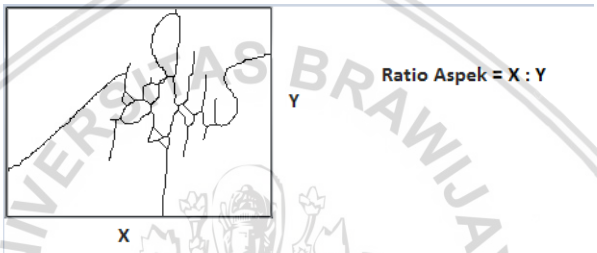
Fitur normalisasi luas objek (tanda tangan) adalah suatu fitur yang didapatkan dari jumlah piksel yang menyusul objek dibagi dengan banyak piksel penyusun citra (Kumar, et al., 2013). Perhitungan normalisasi area objek dapat ditunjukkan dengan Gambar.

$$NormalizedArea = \frac{\text{jumlah piksel penyusun objek}}{\text{banyak piksel penyusun citra}} \quad (2.3)$$




Gambar 2.6 Gambaran Fitur Normalisasi Area Objek

### 2.4.4 Rasio Aspek




Gambar 2.7 Gambaran Fitur Rasio Aspek

Fitur rasio aspek adalah nilai dari pembagian antara lebar objek dengan panjang objek dalam suatu citra. Gambaran rasio aspek ditunjukkan pada Gambar.

$$RasioAspek = \frac{\max \text{ dari histogram pixel horizontal}}{\max \text{ dari histogram pixel vertikal}} \quad (2.4)$$

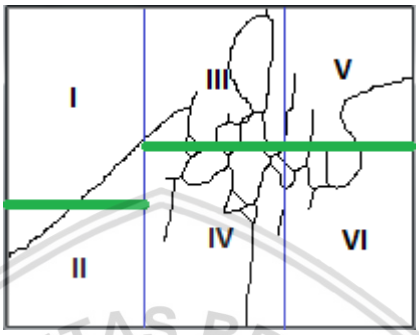
### *2.4.5* Fitur *Three Surface*

Fitur *three surface* merupakan pembagian satu citra menjadi tiga area, lalu setiap area dihitung normalisasi luas yang mengandung susunan objek seperti Persamaan (2.3) (Kumar, et al., 2013). Pada ekstraksi fitur *three surface* diperoleh fitur sebanyak 3 fitur. Fitur *three surface* ditunjukkan pada Gambar.


Gambar 2.8 Gambaran Fitur *Three Surface*

### 2.4.6 *The Six Fold Surface Feature*

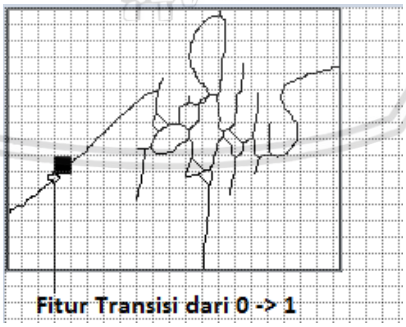
Langkah awal pada fitur *six fold furface* adalah membagi citra menjadi tiga area yang sama. Setiap area ditentukan pusat massanya. Setelah pusat massa diketahui, selanjutnya adalah membuat garis horizontal diatas dan di bawah pusat massa. *Output* dari fitur *six fold surface* adalah 6 fitur yaitu dari setiap nilai jumlah piksel yang dilewati garis horizontal (Kumar, et al., 2013). Fitur *six fold surface* ditunjukkan pada Gambar.




Gambar 2.9 Gambaran Fitur *Six Fold Surface*

### 2.4.7 Fitur transisi

Fitur transisi akan menghasilkan 10 fitur (Kumar, et al., 2013). Fitur pertama dan kedua merupakan rasio dari jumlah kolom yang mengalami transisi nilai piksel dari 0 ke 1 atau 1 ke 0 dengan jumlah kolom penyusun citra dengan arah transisi dari kiri ke kanan. Untuk fitur lainnya perhitungan sama seperti transisi arah kiri ke kanan akan tetapi arah diganti menjadi kanan ke kiri, atas ke bawah, bawa ke atas. Fitur yang terakhir adalah mengitung jumlah transisi dari 0 ke 1 dan 1 ke 0. Gambaran proses perhitungan dari citra untuk fitur *transisi* ditunjukkan pada Gambar.




Gambar 2.10 Gambaran Fitur Transisi

## 2.5 *Scale Invariant Features Transform* (SIFT)

Algoritme *Scale Invariant Features Transform* (SIFT) adalah sebuah Algoritme untuk metode ekstraksi fitur pada citra dengan mengubah citra menjadi fitur lokal yang kemudian akan digunakan sebagai fitur untuk mendeteksi objek yang diinginkan (Lowe, 2004). Beberapa kelebihan dari metode SIFT dalam deteksi objek antara lain:

1. Hasil ekstraksi fitur bersifat *Invariant* terhadap ukuran, *tranlasi* dan rotasi dua dimensi.
2. Hasil ekstraksi bersifat *Invariant* sebagian terhadap perubahan iluminasi dan perubahan sudut pandang tiga dimensi.
3. Mampu mengekstrak banyak *keypoint* dari citra yang tipikal.
4. Hasil ektraksi fitur benar-benar mencirikan secara khusus (*distinctive*).

Tahapan-tahapan dalam metode SIFT yaitu:

1. Mencari nilai ekstrim pada ruang skala

Langkah pertama untuk menetukan *keypoint* adalah identifikasi lokasi dan skala dengan salah satu cara yaitu memvariasi nilai $\sigma$ pada objek citra yang sama untuk memperoleh perubahan skala yang *Invariant* dari citra dan dapat dijadikan sebagai fitur yang stabil (Lowe, 2004).

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (2.5)$$

Keterangan :

$L(x, y, \sigma)$, merupakan ruang skala dari citra

$G(x, y, \sigma)$, merupakan hasil konvolusi dari skala variabel *gaussian*

$I(x, y)$, merupakan citra asli

Dimana untuk perhitungan konvolusi dari skala variabel gaussian ditunjukkan oleh persamaan (2.9).

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \quad (2.6)$$

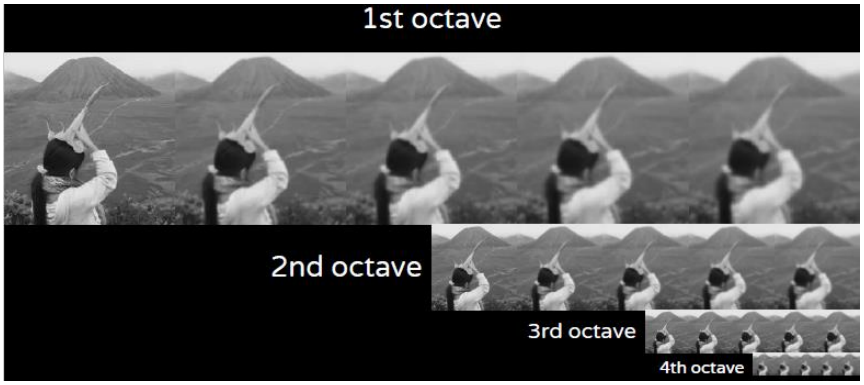
Untuk efisiensi dalam pengkodean dengan bahasa pemrograman, maka proses konvolusi menggunakan variabel *k* sebagai perkalian ke nilai $\sigma$ yang memiliki arti yang sama yaitu untuk membuat nilai $\sigma$ menjadi bervariasi (Lowe, 2004).

$$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y) \quad (2.7)$$

$\sigma$ $\sigma = 1*\sigma$ $\sigma = 2*\sigma$ $\sigma = 3*\sigma$ $\sigma = 4*\sigma$

Gambar 2.11 Hasil *gaussian* blur dengan variasi nilai $\sigma$ dari fungsi $\sigma = k\sigma$

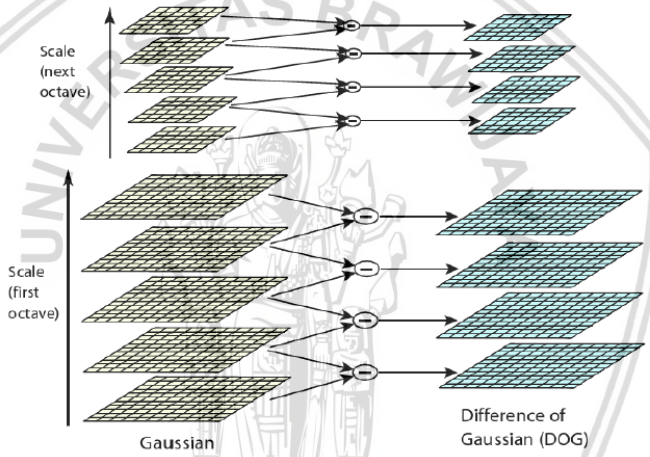
Satu kumpulan citra dengan beberapa variasi nilai $\sigma$ disebut dengan satu *octave*. Suatu ruang skala dibutuhkan kurang lebih 4-5 *octave*. Setiap perpindahan ke bawah suatu *octave* memiliki ukuran yaitu setengah kali dari *octave* sebelumnya.




Gambar 2.12 *Octave* pada Ruang Skala

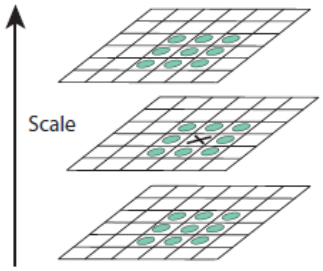
Pada setiap dua citra bersebelahan dalam satu *octave* dihitung *Difference of Gaussian* (DOG).

$$D(x, y, \sigma_D) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (2.8)$$




Gambar 2.13 Hasil *Difference of Gaussian* (DoG)
(Lowe, 2004)

Menentukan nilai maksimum dan minimum dari citra *Difference of Gaussian* (DoG) yang dideteksi oleh perbandingan antara 26 piksel tetangga di 3x3 citra *Difference of Gaussian* (DoG).




Gambar 2.14 Hasil nilai maksimum dan minimum dari citra *Difference of Gaussian* (DoG)
(Lowe, 2004)

2. Menentukan Lokalisasi *Keypoint*

Lokalisasi *keypoint* dapat menggunakan akurasi subpiksel dengan persamaan ekspansi Taylor.

$$z = \left( \frac{\partial^2 D}{\partial x^2} \right)^{-1} \frac{\partial D}{\partial x} \tag{2.9}$$

Keterangan:

$z$ , merupakan posisi nilai ekstrim

Nilai *keypoint* ekstrim dapat dihitung dengan persamaan (2.13).

$$D(z) = D + \frac{1}{2} \left( \frac{\partial D^{-1}}{\partial x} \right) z \tag{2.10}$$

Selanjutnya adalah penghapusan *keypoint* yang tidak memenuhi *threshold* yaitu 0,03 dan penghapusan *keypoint* yang berada pada *edge* dengan menggunakan matriks Hessian.

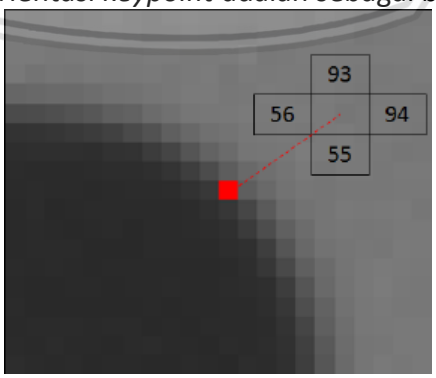
$$H = \begin{bmatrix} D_{XX} & D_{XY} \\ D_{YX} & D_{YY} \end{bmatrix}$$

$$\frac{D_{XX} + D_{YY}}{D_{XX} D_{YY} - (D_{XY})^2} < \frac{(r+1)^2}{r} \tag{2.11}$$

Nilai *r* adalah *threshold* untuk rasio kurvatur, dimana jika *keypoint* di bawah nilai *r* maka akan dihapus (Agustina & Mukhlash, 2012).

3. Penentuan orientasi

Setelah *keypoint* telah diketahui maka selanjutnya dilakukan proses orientasi dari *keypoint* tersebut. Orientasi yang telah ditemukan disetiap *keypoint* di kumpulkan ke grup dimana grup tersebut merupakan representasi dari gradient histogram. Jika dalam histogram gradien terdapat nilai paling maksimum atau lebih dari 80% maka akan nilai tersebut akan dipilih sebagai orientasi dari *keypoint*. Jika ada dua nilai maka, satu nilai akan dideskripsikan sebagai *keypoint* baru. Contoh manualisasi dari orientasi *keypoint* adalah sebagai berikut:




Gambar 2.15 Gambar Perhitungan Orientasi Suatu Piksel

Contoh langkah-langkah perhitungan dari orientasi:

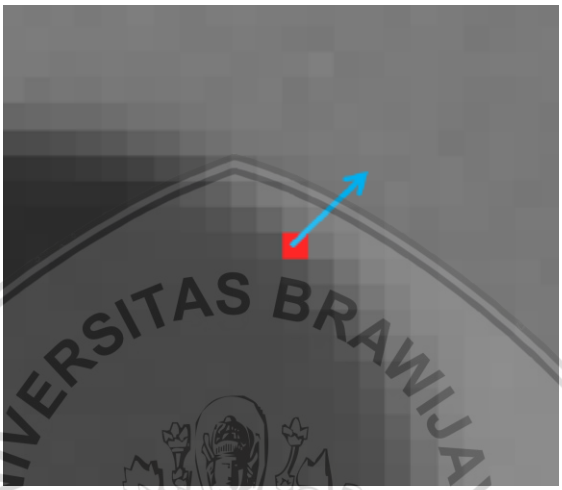
$$\text{vertical change} : 93 - 55 = 38$$

$$\text{horizontal change} : 94 - 56 = 38$$

$$\text{gradient vector} : \begin{bmatrix} 38 \\ 38 \end{bmatrix}$$

$$\text{magnitude} : \sqrt{38^2 + 38^2} = 53.74$$

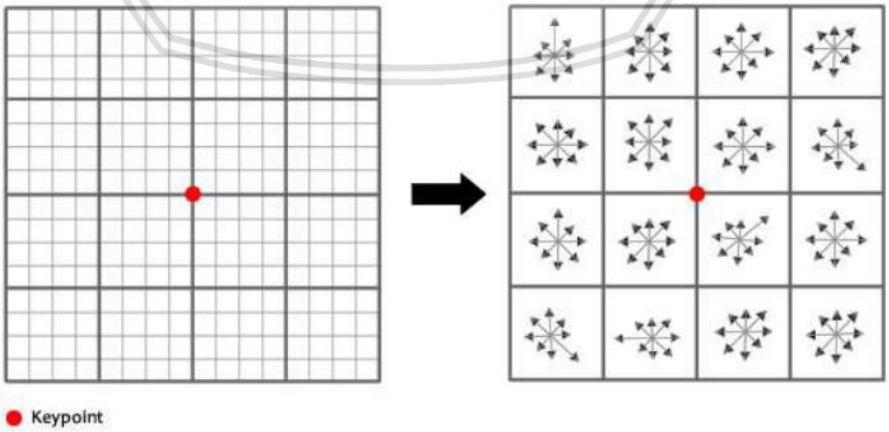
$$\text{orientation} : ar\tan\left(\frac{38}{38}\right) = 45°$$



Gambar 2.16 Gambar Orientasi Piksel

4. Deskripsi *keypoint*

Fitur seperti sebuah *fingerprint* sehingga harus bersifat unik. Pada *mask* atau *kernel* yang berukuran 16x16 posisi *keypoint* ada di tengah. Setiap *kernel* dibagi menjadi 4x4 bagian, yang setiap kotak bagian tersebut terdapat nilai *magnitude* dan orientasi yang telah dihitung pada langkah ke-3. Sehingga deskripsi dari *keypoint* adalah seluruh nilai *magnitude* dan orientasi pada *mask* atau *kernel* 4x4 bagian pada setiap bagiannya.




Gambar 2.17 *Keypoint Descriptor*

5. *Keypoint Matching*

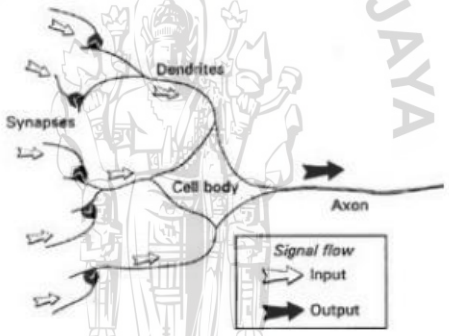
*Keypoint matching* adalah proses mencocokan *keypoint* satu gambar ke gambar lainnya. Salah satu metode yang digunakan adalah *Nearest Neighbor Distance Ratio* (NNDR). Langkah awal yang digunakan adalah mengitung *euclidean distance* dari satu *keypoint* pada gambar pertama dengan semua *keypoint* pada gambar kedua.

$$d = \| f1 - f2 \| \quad (2.12)$$

## 2.6 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan atau *Artificial Neural Network* adalah kumpulan sistem yang terdiri dari elemen, unit atau *node* yang saling berhubungan dengan memiliki fungsi yang berdasarkan pada jaringan syaraf manusia serta memiliki kemampuan mengolah dan menyimpan informasi dengan metode mempelajari atau learning (Gurney, 1997). JST telah dikembangkan sebagai generalisasi model matematika dari aspek jaringan syaraf biologis (Rojas, 1996), yaitu didasarkan dengan asumsi :

1. Pemrosesan informasi terjadi pada elemen-elemen yang disebut neuron.
2. Sinyal-sinyal merambat diantara neuron melalui interkoneksi.
3. Setiap interkoneksi memiliki bobot yang bersesuaian yang pada kebanyakan jaringan syaraf berfungsi untuk mengalikan sinyal yang dikirim.

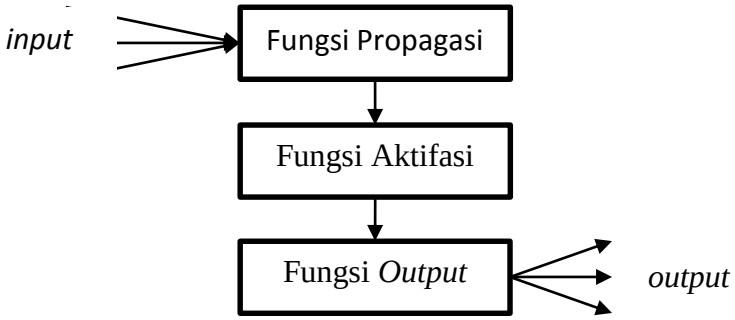



Gambar 2.18 Neuron pada jaringan syaraf manusia
(Gurney, 1997)

Pada jaringan syaraf manusia, setiap neuron menerapkan fungsi aktifasi pada masukkan jaringan untuk menentukan sinyal keluaran. Alur dari neuron adalah sinyal elektrik yang diterima sinapsis dari neuron sebelumnya atau dari sensor motorik akan di lanjutkan ke dendrit, lalu ke *body cell* untuk dioleh informasinya dan jika belum bisa diolah maka akan diteruskan ke akson untuk dialirkan ke neuran selanjutnya (Gurney, 1997).

Prototipe jaringan syaraf tiruan memiliki kesamaan struktur dengan neuron yang ada di jaringan syaraf manusia (Gurney, 1997). Sinapsis pada jaringan syaraf tiruan dimodelkan dengan *weight* atau bobot yang akan dikalikan dengan *input*. Selanjutnya adalah menjumlahkan hasil dari perkalian setiap sinapsis yang diibaratkan proses pengolahan informasi pada *body cell* bisa juga disebut fungsi *propagation*. Untuk menghasilkan ke nilai *output* hasil dari fungsi propagation dilakukan *transform* untuk dijadikan *input* lagi dengan menggunakan fungsi aktifasi. Jika masih ada layer lagi maka *ouput* dari fungsi aktifasi harus dilakukan

*transform* lagi untuk dijadikan sebagai masukkan sehingga disebut dengan fungsi *output*.




Gambar 2.19 *Prototype* Jaringan Syaraf Tiruan

## 2.7 Algoritme *Learning Vector Quantization* (LVQ)

*Learning Vector Quantization* (LVQ) adalah sebuah Algoritme dari jaringan syaraf tiruan yang digunakan untuk melakukan pembelajaran pada lapisan kompetitif yang terwarisi (Qur'ani & Rosmalinda, 2010). Algoritme *Learning Vector Quantization* (LVQ) dapat digunakan untuk proses klasifikasi. Metode yang dilakukan dapat melalui pendekatan *supervised*. Dalam proses klasifikasi menggunakan LVQ dibutuhkan pembelajaran sehingga ada data pembelajaran (data latih) dalam metode ini. Hasil yang dikeluarkan pada proses pembelajaran adalah bobot tiap fitur yang digunakan dalam data. Nantinya bobot ini akan digunakan dalam proses pertimbangan dan pengambilan keputusan dalam proses pengujian (data uji). Pengambilkan keputusan berupa klasifikasi data uji pada kelas tanda tangan yang ada.

Algoritme LVQ dibagi menjadi 3 algoritme yaitu LVQ1, LVQ2, dan LVQ3 (Kohonen, et al., 1996). Langkah-langkah dalam pelatihan algoritme LVQ1 (Qur'ani & Rosmalinda, 2010) adalah sebagai berikut :

1. Menginisialisasi bobot setiap fitur dari setiap kelas klasifikasi ($w_{ij}$), nilai maksimum iterasi, nilai *error* minimum, nilai *Learning Rate* (α) awal dan nilai pengurang *Learning Rate* (α).
2. Memasukkan seluruh nilai *input* ($x_{ij}$) dan target ($T_j$). Nilai *input* adalah nilai dari ekstraksi ciri satu data latih yang sudah ternormalisasi. Dan nilai target adalah nilai target pada data latih.
3. Menginisialisasi kondisi awal parameter yaitu nilai *epoch* atau nilai indek awal iterasi dengan nilai 0.
4. Melakukan iterasi dengan syarat iterasi bejalan jika nilai *epoch* lebih kecil dari nilai maksimum iterasi dan nilai α lebih besar dari nilai *error* minimum. Langkah nomor 4 ini, pada saat satu kali iterasi menjalankan langkah 5-8.
5. Menghitung *euclidean distance* ($S_j$) antara fitur dengan bobot fitur dalam satu kelas.

$$S_j = \left\| x_{ij} - w_{ij} \right\| \qquad (2.13)$$

6. Menghitung Nilai minimum dari $S_j$. Dan menentukan hasil kelas klasifikasi ($C_j$), dari perolehan nilai minimum $S_j$.
7. Memperbarui $w_{ij}$ dengan ketentuan :

   Jika $T_j$ = $C_j$ maka

   $$w_j(t+1) = w_j(t) + \alpha(t)(x(t) - w_j(t)) \quad (2.14)$$

   Jika $T_j$ ≠ $C_j$ maka

   $$w_j(t+1) = w_j(t) - \alpha(t)(x(t) - w_j(t)) \quad (2.15)$$

8. Mengurangi nilai *Learning Rate* atau α.

   $$\alpha(baru) = \alpha(lama) - (dec.\alpha * \alpha(lama)) \quad (2.16)$$

9. Memberhentikan iterasi jika sudah melebihi syarat iterasi.

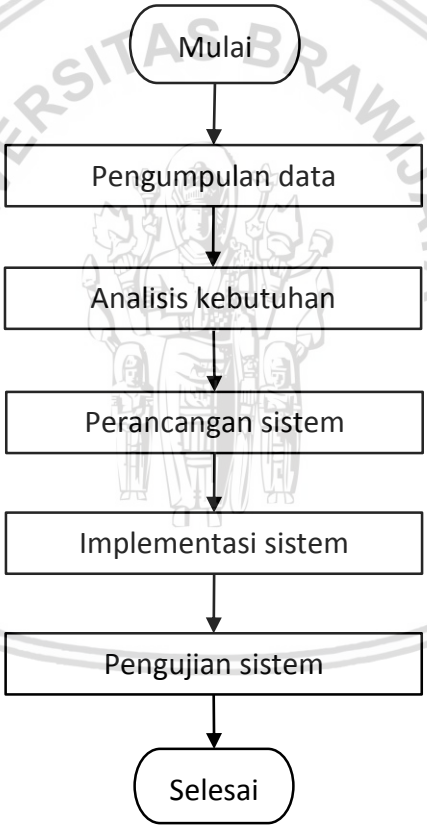
# BAB 3 METODOLOGI

Bab metodologi merupakan bab yang berisikan metode-metode dan *objek* yang akan digunakan untuk melakukan penelitian ini. Alur-alur dalam metodologi adalah alur dalam pembuatan sistem pengenalan tanda tangan menggunakan algoritme *Learning Vector Quantization* (LVQ) dengan penambahan fitur dengan metode *Scale Invariant Features Transform (SIFT).*

## 3.1 Tahapan Penelitian

Penelitian yang akan dilakukan merupakan penelitian non-implementatif, maka didalam penelitian terdapat proses awal berupa analisis kebutuhan, pengolahan data, perancangan sistem, implementasi sistem hingga pengujian sistem.

Tahap-tahap penelitian digambarkan dalam diagram alir pada Gambar 3.1.




**Gambar 3.1 Diagram alir tahapan penelitian**

Analisis Kebutuhan dilakukan sebagai aktivitas untuk menemukan dan mendefinisikan kebutuhan fungsional sistem dengan tujuan agar sistem yang dibangun dapat berjalan sesuai dengan tujuan. Kebutuhan fungsional dari sistem ini adalah dapat mendeteksi posisi tanda tangan dan dapat mengenali tanda tangan tersebut. Kebutuhan non-fungsional dari sistem ini adalah keefektifan atau kecepatan dalam penyelesaian dan validitas dari pengenalan tanda tangan yang

dapat dihitung dari membedakan hasil dengan metode pengenalan tanda tangan yang lain.

Tahap pengumpulan data berupa teknik pengumpulan data untuk penelitian. Untuk teknik pengumpulan data, data berupa citra atau gambar tanda tangan dari beberapa orang yang setiap orangnya melakukan lebih dari satu kali tanda tangan asli yang dimilikinya. Citra atau gambar tanda tangan palsu yang seperti dengan aslinya tapi berbeda orang yang menandatanganinya.

Perancangan sistem dilakukan sebelum implementasi sistem. Perancangan sistem merupakan suatu proses represesntasi penyelesaian masalah pengenalan tanda tangan menggunakan algoritme *Learning Vector Quantization* (LVQ) beserta proses ektraksi fitur-fitur yang digunakan.

Pengujian sistem bertujuan untuk menguji algoritme *Learning Vector Quantization* (LVQ) yang digunakan untuk pengenalan tanda tangan. Pengujian algoritme ini berfokus pada parameter algoritme *Learning Vector Quantization* (LVQ) yang digunakan untuk memperoleh solusi hasil terbaik.

## 3.2 Teknik Pengumpulan Data

Tahap ini dilakukan untuk menghimpun citra-citra tanda tangan yang akan dijadikan data latih maupun data uji. Citra yang digunakan pada penelitian ini tergolong data primer yaitu data yang diperoleh dari sumber data secara langsung. Data yang diambil berupa tanda tangan dari beberapa orang berbeda. Citra tanda tangan tersebut dikelompokkan berdasarkan pemilik tanda tangan.

Teknik pengumpulan data, data berupa citra atau gambar tanda tangan dari 25 orang dengan jumlah tanda tangan sebanyak 10 tanda tangan per orang. Tanda tangan yang diambil memiliki keberagaman bentuk dan rotasi walaupun dalam satu kelas. Jumlah dataset antara lain yaitu 100 citra untuk data uji, 100 citra untuk data latih, dan 25 citra untuk data latih yang tidak memiliki kelas yang dilatih dalam proses pelatihan.

## 3.3 Algoritme yang Digunakan

Algoritme yang digunakan pada penelitian ini adalah *Learning Vector Quantization* (LVQ) sesuai yang diuraikan pada Bab 2 karena terbukti memiliki akurasi yang tinggi pada pengenalan tanda tangan dibandingkan dengan menggunakan algoritme *Kohonen* (Prabowo, et al., 2006). Fitur yang digunakan berupa fiktur fraktal global. Tambahan fitur yang digunakan adalah fitur jumlah *keypoint descriptor* yang merupakan hasil *Scale Invariant Features Transform* (SIFT) yang terbukti dapat digunakan untuk pengenalan tanda tangan dengan akurasi sebesar 63% (Hilman, 2015).

Implementasi algoritme menggunakan bahasa pemrograman Python dan OpenCV 3 karena lebih mudah dan memiliki *source code* lebih pendek yang mudah dimengerti serta terdapat fitur-fitur yang efektif yang telah disediakan bahasa Python dan OpenCV untuk memproses citra digital. Dan untuk tampilan *user interface* menggunakan Visual Basic.

## 3.4 Kebutuhan Sistem

Kebutuhan sistem perangkat lunak agar berjalan baik maka pengujian sistem perangkat lunak ini dijalan pada laptop dengan spesifikasi sebagai berikut :

1. Prosesor Intel®Core™ i5-2540M CPU @ 2.60 GHz
2. RAM 4,00 GB
3. Harddisk 500 GB
4. Monitor
5. Keyboard

Dan untuk perangkat lunak yang menunjang sistem antara lain :

1. Sistem operasi Windows 7 32 bit sebagai lingkup kerja sistem.
2. Python 2.7 untuk penulisan *source code* pada proses implementasi.
3. OpenCV 3.1.
4. phpMyAdmin untuk *database* yang digunakan.

## 3.5 Pengujian Algoritme

Tahap pengujian Algoritme merupakan tahap untuk menguji sistem yang telah diimplementasikan berdasarkan perancangan yang telah dibuat. Pengujian dilakukan dengan mengamati akurasi dan reliabilitas sistem sebagai parameternya. Pengujian dilakukan dengan menggunakan 200 dataset citra tanda tangan dari 20 orang. Pengambilan data latih dengan cara menggunakan 5 citra dari setiap orang, sehingga total data latih sebesar 100 citra. Dan untuk data uji sebesar 100 citra dari sisa dataset yang belum digunakan untuk data latih. Dan untuk tambahan ada 5 citra dari 5 orang yang berbeda yang tidak dimasukkan dalam proses pelatihan pengenalan tanda tangan. Data 5 citra ini digunakan untuk pengujian tanda tangan yang tidak dikenali. Berikut merupakan uraian beberapa skenario pengujian yang dilakukan antara lain :

1. Pengujian Variasi Fitur. Dilakukan dengan menggunakan sepuluh citra. Stiap citra dihitung fitur-fiturnya yang berjumlah 28 fitur. Lalu dilakukan proses normalisasi. Hasil normalisasi dari fitur tersebut yang digunakan dalam pengujian variasi fitur menggunakan perhitungan standar deviasi.
2. Pengujian Nilai *Learning Rate* Awal. Pengujian ini dilakukan dengan menggunakan 10 nilai *Learning Rate* dengan rentang nilai antara 0,05 sampai dengan 0,5 dengan nilai kelipatan 0,05.
3. Pengujian Nilai Pengurang *Learning Rate*. Pengujian ini dilakukan dengan menggunakan 10 nilai pengurang *Learning Rate* yaitu rentang angka 0,025 sampai dengan 0,25 dengan kelipatan 0,025.
4. Pengujian Nilai Maksimum Iterasi. Pengujian iterasi dilakukan dengan menggunakan 10 nilai untuk nilai maksimum iterasi yaitu 1, 5, 10, 20, 50, 75, 100, 125, 150, 200.
5. Pengujian Data Uji dengan Nilai Parameter Optimal. Data uji yang dipakasi dalam pengujian ini dibagi menjadi dua macam yaitu data uji dengan kelas yang sudah dilatihkan dan data uji dengan kelas yang belum pernah

dilatihkan. Pengujian ini memakai *threshold* dengan nilai maksimum dari nilai jarak minimum tiap kelas.

6. Pengujian Nilai *Threshold*. Pengujian nilai *threshold* dilakukan dengan dua metode. Metode pertama menggunakan nilai maksimum dari nilai jarak terkecil pada setiap kelas. Dan metode kedua menggunakan nilai maksimum ke-2 dari nilai jarak terkecil pada setiap kelas.
7. Pengujian Implementasi Fitur *Keypoint Descriptor*. Pengujian ini dilakukan dengan menguji sistem jika menggunakan tambahan fitur jumlah *keypoint descriptor* dan tanpa tambahan fitur jumlah *keypoint descriptor*.

Fitur jumlah *keypoint descriptor* dari satu citra dengan pencarian *keypoint descriptor* menggunakan metode *Scale Invariant Features Transform* (SIFT) (Lowe, 2004). Jumlah *keypoint descriptor* pada citra didasarkan pada banyak *keypoint descriptor* setiap citra.

Pada tahapan inisialisasi fitur selain proses pencarian nilai fitur atau ektraksi fitur terdapat proses lain. Proses tersebut adalah proses normalisasi nilai fitur. Proses normalisasi fitur dilakukan dengan menggunakan cara *min-maks*. Salah satu ciri(j) pada satu citra(i) akan dikurang dengan nilai minimum ciri yang ada pada fitur tersebut, lalu dibagi dengan selisih nilai maksimum ciri dan minimum ciri. Hasil proses normalisasi ini selanjutkan menjadi *input* pada proses pelatihan sistem. Proses normalisasi dapat dilihat pada persamaan 4.1.

$$NormalisasiFitur = \frac{fitur_{ij} - \min(fitur_j)}{\max(fitur_j) - \min(fitur_j)} \quad (4.1)$$

### 4.3.3 Inisialisasi Bobot

Proses inisialisasi bobot dilakukan pada setiap fitur disetiap citra. Metode inisialisasi bobot pada penelitian pengenalan tanda tangan ini menggunakan metode pengambilkan nilai fitur dari salah satu citra dari setiap kelas untuk dijadikan bobot fitur awal. Contoh hasil proses inisialisasi bobot awal pada dua kelas klasifikasi dapat ditunjukkan pada Tabel 4.3.

**Tabel 4.3 Inisialisasi Bobot**

| Bobot (w) | Kelas Klasifikasi | |
|---|---|---|
| | 1 | 2 |
| w1 | 0.126214 | 0.058252 |
| w2 | 0.11236 | 0.314607 |
| w3 | 0.84127 | 0.84127 |
| w4 | 0.262136 | 0.38835 |
| w5 | 0.521739 | 0.521739 |
| w6 | 0.159091 | 0.215909 |
| w7 | 0.200709 | 0.138374 |
| w8 | 0.121747 | 0.11941 |
| w9 | 0.008827 | 0.002212 |
| w10 | 0.474 | 0.36406 |
| w11 | 0.31325 | 0.304734 |
| w12 | 0.037879 | 0.015152 |
| w13 | 0.046053 | 0.029605 |
| w14 | 0.759322 | 0.583051 |
| w15 | 0.550964 | 0.341598 |
| w16 | 0.313889 | 0.283333 |
| w17 | 0.3225 | 0.275 |
| w18 | 0.398049 | 0.159966 |

| | | |
|---|---|---|
| w19 | 0.113384 | 0.099034 |
| w20 | 0.036358 | 0.055051 |
| w21 | 0.077041 | 0.064594 |
| w22 | 0.2678 | 0.115214 |
| w23 | 0.119891 | 0.103284 |
| w24 | 0.038922 | 0.03793 |
| w25 | 0.081106 | 0.066808 |
| w26 | 0.33258 | 0.212309 |
| w27 | 0.332956 | 0.211212 |
| w28 | 0.571429 | 0.142857 |

### 4.3.4 *Euclidean Distance*

Proses perhitungan fungsi aktifasi dari algoritme *Learning Vector Quantization* (LVQ) menggunakan persamaan *Euclidean Distance*. Dimana hasil dari *Euclidean Distance* ini merupakan jarak kesamaan yang ditunjukkan oleh kelas citra. Jika hasil perhitungan proses *euclidean distance* adalah minimum dari seluruh data maka citra tersebut masuk ke kelas citra sesuai bobot fitur yang digunakan dalam perhitungan proses *euclidean distance*.

$$s_j = \sqrt{\sum_{i=1}^{28} \left(x_i - w_{ji}\right)^2} \qquad (4.1)$$

Keterangan:
j, merupakan kelas citra
s, merupakan *euclidean distance*
x, merupakan nilai fitur yang didapat dari citra
w, bobot setiap fitur

### 4.3.5 *Update* Bobot Fitur

Pada proses *euclidean distance*, jika menghasilkan nilai paling minimum dari kelas citra maka kelas tersebut merupakan kelas dari citra yang di proses. Selanjutnya adalah mengubah bobot-bobot setiap fitur yang ada di kelas hasil proses *euclidean distance* yang bernilai minimum. Dalam alurnya, jika sesuai dengan data asli dari pakar maka bobot akan didekatnya. Sedangkan jika tidak sesuai maka bobot-bobot fitur pada kelas tersebut harus dijauhkan (Qur'ani & Rosmalinda, 2010).

Jika kelas citra hasil proses komputasi sama dengan pakar maka

$$w_j(t+1) = w_j(t) + \alpha(t)(x(t) - w_j(t)) \qquad (2.7)$$

Jika kelas citra hasil proses komputasi tidak sama dengan pakar maka

$$w_j(t+1) = w_j(t) - \alpha(t)(x(t) - w_j(t)) \qquad (2.8)$$

### 4.3.6 *Threshold*

Pada penelitian ini, *threshold* digunakan untuk membatasi nilai jarak minimum pada citra dimana kelas yang dihasilkan sistem sesuai dengan kelas target. Tujuan dari nilai *threshold* ini adalah untuk memperkecil kemungkinan sebuah tanda tangan memasuki kelas yang bukan kelas aslinya. Nilai *threshold* ditentukan berdasarkan banyak kelas klasifikasi yang dilatih. Dalam penelitian ini menggunakan dua metode pencarian nilai *threshold*, yaitu :

1. Nilai *Threshold* dengan Menggunakan Nilai Maksimum dari Nilai *Euclidean*.

   Tahapan-tahapan dari metode ini adalah:

   a Menghitung nilai jarak minimum dan menentukan kelas klasifikasi citra.

   Pada Tabel 4.4 dapat dilihat bahwa lima citra latih dengan kelas target 1 sesuai dengan kelas yang dihasilkan sistem. Sedangkan pada kelas target 2 pada citra ke-2 masih belum sesuai antara kelas target dengan kelas sistem, sehingga diberikan nilai -.

**Tabel 4.4 Nilai Jarak Minimum Citra Setiap Kelas**

| Kelas | Nilai Jarak Minimum Citra ke- | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| 1 | 0.516611 | 0.328474 | 0.526076 | 0.381019 | 0.424685 |
| 2 | 0.608821 | - | 0.747745 | 0.391102 | 0.553268 |

   b Menentukan nilai terbesar nilai jarak minimum setiap kelas.

**Tabel 4.5 Nilai *Threshold* Setiap Kelas**

| Kelas | Nilai *Threshold* |
|---|---|
| 1 | 0.526075728 |
| 2 | 0.747744746 |

2. Nilai *Threshold* dengan Menggunakan Nilai Maksimum Ke-2 dari Nilai *Euclidean*.

   Pada metode kedua untuk mencari nilai *threshold* menggunakan nilai terbesar kedua dari jarak minimum citra yang sesuai antara kelas target dengan kelas sistem. Hasil dari metode ini ditunjukkan pada Tabel 4.6.

**Tabel 4.6 Nilai *Threshold* Setiap Kelas**

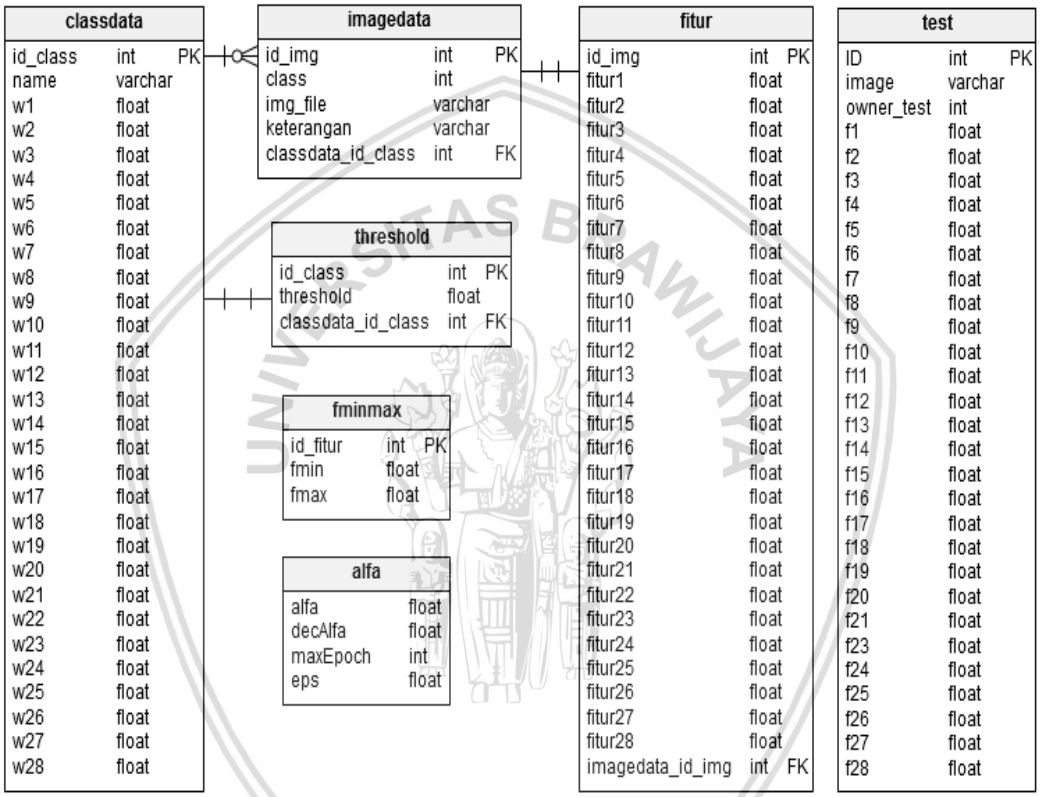
| Kelas | Nilai *Threshold* |
|---|---|
| 1 | 0.516610732 |
| 2 | 0.608821114 |

### 4.3.7 Update *Learning Rate* atau Nilai α

Proses mengubah laju pembelajaran setiap trainning data dan setiap data dimasukkan. Laju pembelajaran didapat dari proses perhitungan nilai laju pembelajaran dahulu dikali dengan satu dikurangi iterasi sekarang dibagi terasi seluruhnya.

$$\alpha = \alpha * (1 - (decAlfa)) \quad (2.9)$$

## 4.4 Perancangan *Database*




Gambar 4.17 Struktur *Database*

Pada perancangan *database* terdapat tiga tabel yang akan digunakan. Tabel pertama merupakan tabel yang berisikan nama pemilik tanda tangan, nomor id citra sebagai kelas, dan 28 fitur yang digunakan untuk pengenalan tanda tangan. Tabel kedua, merupakan tabel yang berisi nomor id citra, nomor id citra yang dicocokan, dan kolom jumlah *keypoint descriptor*. Tabel terakhir merupakan tabel yang berisikan bobot-bobot setiap fitur sehingga berjumlah 28 bobot dengan kunci dari tabel adalah nomor id citra.

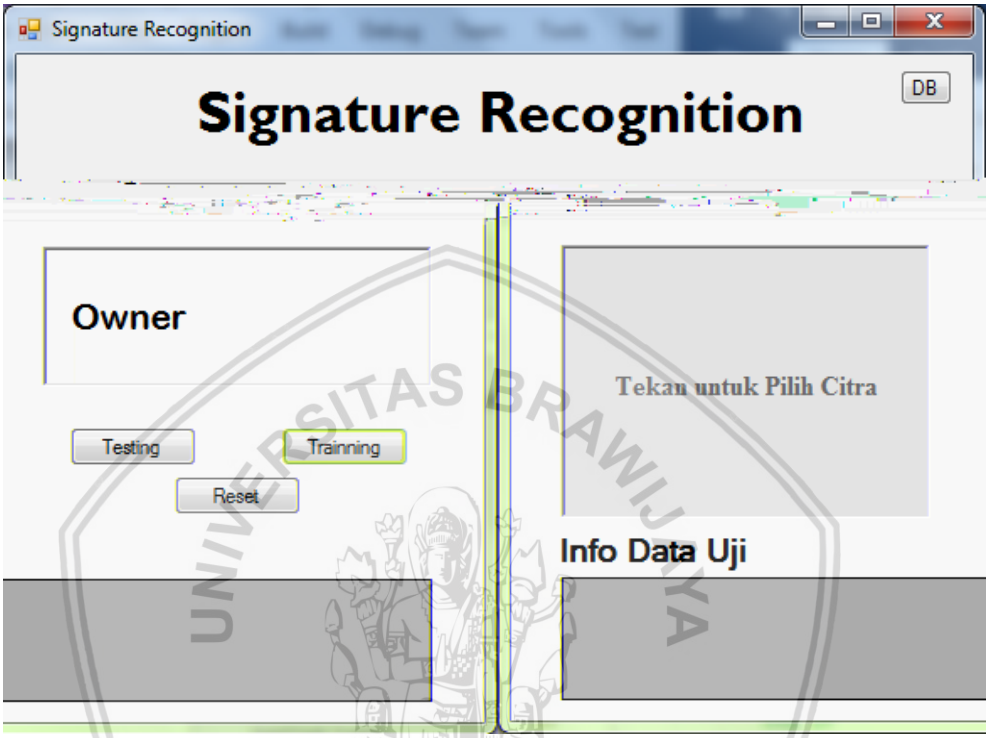
## 4.5 Perancangan *User interface*

Pada sub bab perancangan *User interface* merupakan penjelasan mengenai perancangan *User interface* yang akan digunakan pada proses implementasi. *User*

*interface* memiliki rancangan yang terdiri dari dua *form* yaitu form utama dan form nilai fitur.

### 4.5.1 *Form* Utama

Pada *form* utama terdapat pemasukkan citra untuk data uji. Istilah kelas adalah nama pemilik dari tanda tangan.



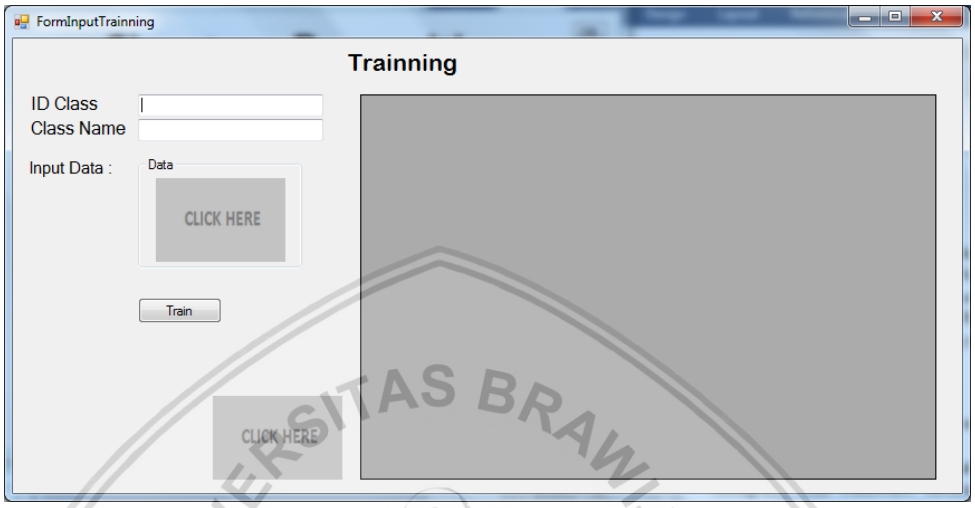

Gambar 4.18 Perancangan Form Utama

Keterangan:

1. Citra uji yang ingin diuji kepemilikan tanda tangannya. Dengan cara menekan kotak dibawah tulisan citra uji yang terdapat perkataan 'Tekan untuk Pilih Citra'. Citra akan dipilih melalui open *dialog box* dan citra yang dipilih akan ditampilkan pada kotak.
2. Hasil pengujian berupa nama pemilik tanda tangan hasil pengolahan sistem pengenalan tanda tangan. Jika data belum pernah dilatih, maka hasil *output* berupa data tidak dikenali.
3. Tombol *Testing* adalah untuk proses testing citra uji. Tombol testing ditekan setelah citra yang akan diuji sudah dipilih dan ditampilkan pada kotak citra uji.
4. Tombol *Trainning* adalah tombol untuk proses pelatihan citra latih dan informasi data latih.
5. Tombol *Reset* merupakan tombol untuk menghapus semua informasi data uji dan mengembalikan ke tampilan awal. Serta tombol *reset* bertujuan menghapus data pada tabel test didalam *database*.
6. Tabel Info Data Uji merupakan tabel untuk menampilkan informasi data uji seperti nama citra uji, kelas hasil klasifikasi, dan nilai setiap fitur dalam citra uji.

### 4.5.2 *Form Input* Data Latih

Pada *form Input* Data Latih merupakan *user interface* untuk proses pelatihan sistem pengenalan tanda tangan dan informasi data latih sistem pengenalan tanda tangan. Dalam *form input* data latih ini dapat melakukan *input* data latih.




Gambar 4.19 Perancangan Form *Input* Data Latih

Keterangan :

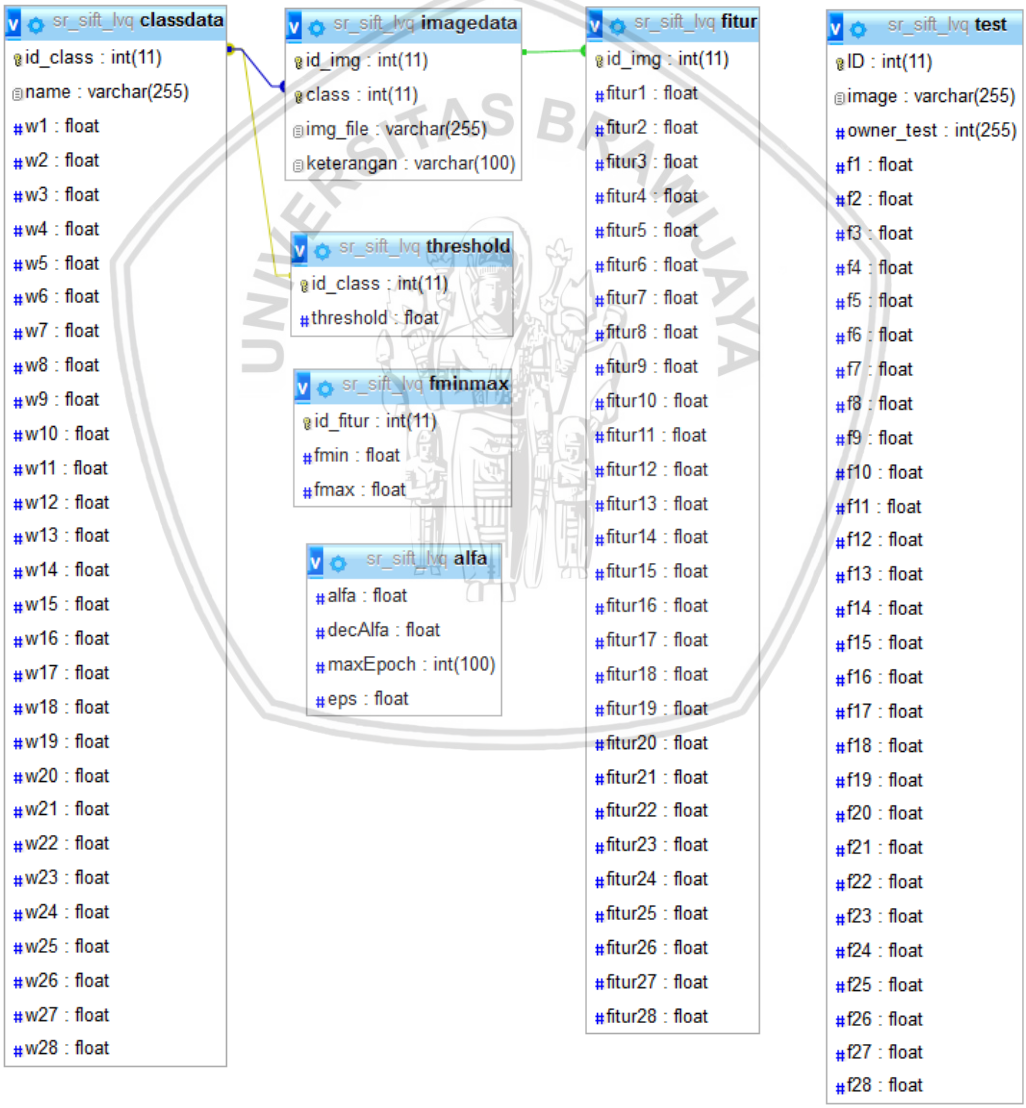
1. Kolom ID *Class* adalah kolom untuk memasukkan id *Class* pada citra yang akan dimasukkan kedalam data latih.
2. Kolom *Class Name* adalah kolom untuk memasukkan nama pemilik dari citra yang akan dimasukkan kedalam data latih.
3. *Input* data dilakukan dengan cara menekan kotak yang bertulisan *click here* yang selanjutkan akan menampilkan *open dialog box*. Dan citra yang dipilih akan ditampilkan pada kotak data.
4. Tombol *train* adalah tombol untuk proses pelatihan.
5. Tabel data *train* ditunjukkan pada samping kolom pemasukkan data. Data yang ditampilkan adalah fitur-fitur setiap citra latih.

# BAB 5 IMPLEMENTASI

Pada bab implementasi merupakan penjelasan tentang uraian struktur *database*, struktur *Class* dan implementasi algoritme pada kode program. Bab implementasi ini dibuat berdasarkan perancangan yang dijelaskan pada bab perancangan.

## 5.1 Struktur *Database*

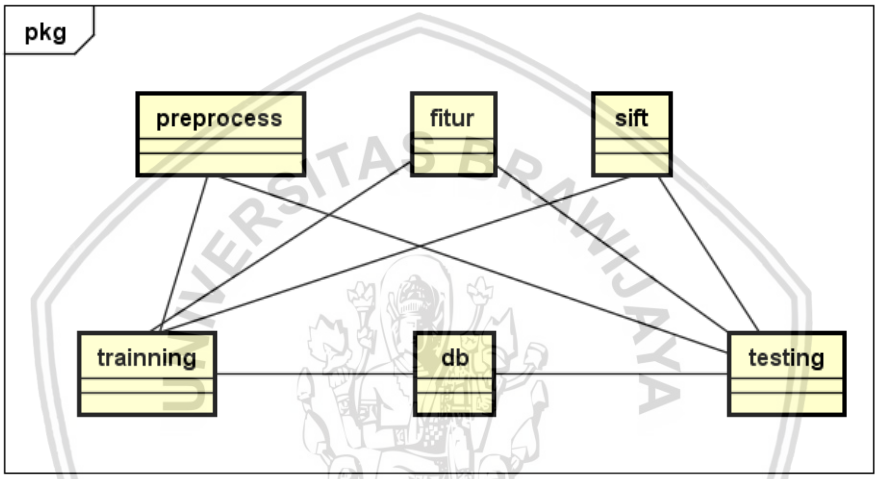
Pembangunan sistem pengenalan tanda tangan menggunakan Algoritme *Learning Vector Quantization* (LVQ) dan ektraksi citra dengan menggunakan metode *Scale Invariant Features Transform*(SIFT) terdapat struktur *database* yang ditunjukkan pada Gambar 5.1.




Gambar 5.1 Struktur *Database*

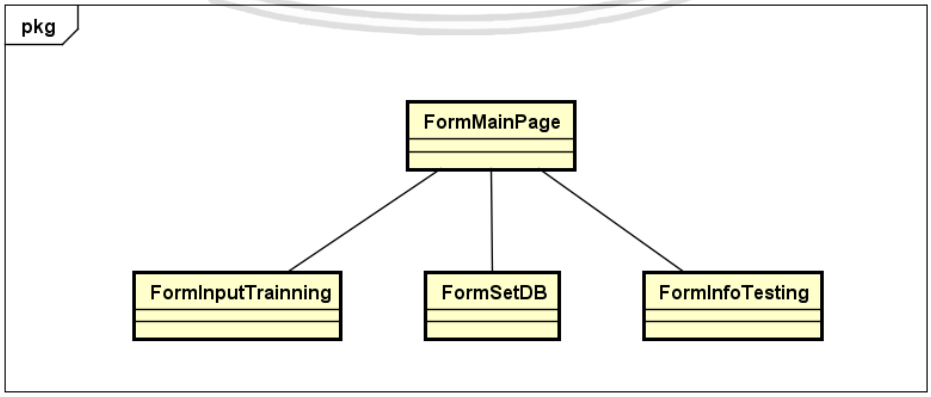
## 5.2 Struktur *Class*

Struktur *class* pada sistem pengenalan tanda tangan menggunakan algoritme *Learning Vector Quantization* (LVQ) dan ektraksi citra dengan menggunakan metode *Scale Invariant Features Transform*(SIFT) dapat ditunjukkan pada Gambar 5.2. Pada stuktur *class* sistem pengenalan tanda tangan terbagi menjadi enam kelas yaitu *class* preprocess yang digunakan untuk melakukan *image preprocessing* sebelum diolah, *class* fitur yang berisikan *method* pengolahan fitur-fitur untuk pengenalan tanda tangan dari citra yang sudah di praproses, *class* sift yaitu kelas untuk mengolah fitur jumlah *keypoint descriptor* menggunakan metode *Scale Invariant Features Transform* (SIFT), *class trainning* merupakan *class* untuk proses *trainning* pengenalan tanda tangan, *class* db, dan *Class testing*.



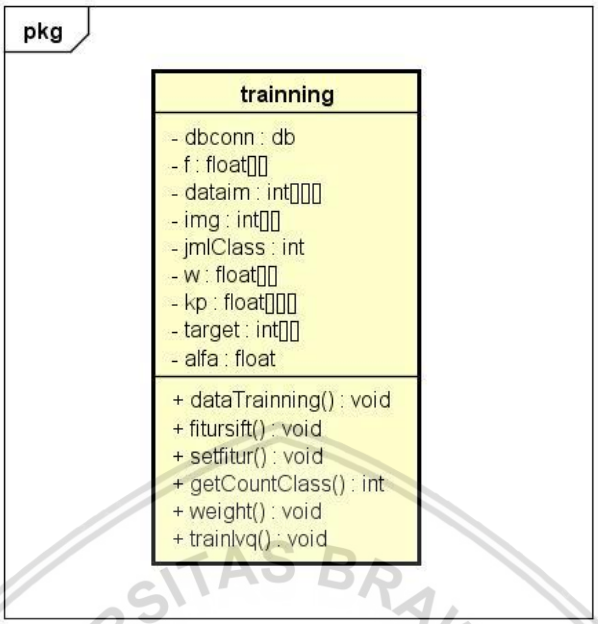

Gambar 5.2 Struktur *Class*

Pada Gambar 5.3 merupakan gambar yang untuk struktur *class user interface* pada sistem pengenalan tanda tangan. Pada *class user interface* terdapat empat *form* yaitu FormMainPage, FormInputTrainning, FormSetDB, dan FormInfoTesting.




Gambar 5.3 Struktur *Class User interface*

### 5.2.1 Struktur *Class Trainning*




Gambar 5.4 Struktur *Class Trainning*

Pada Gambar 5.4 merupakan struktur *class trainning* untuk proses pelatihan. *Class* trainning berisikan mehod-method yang digunakan untuk proses pelatihan pengenalan tanda tangan. *Method-method* yang berada pada *class* trainning antara lain :

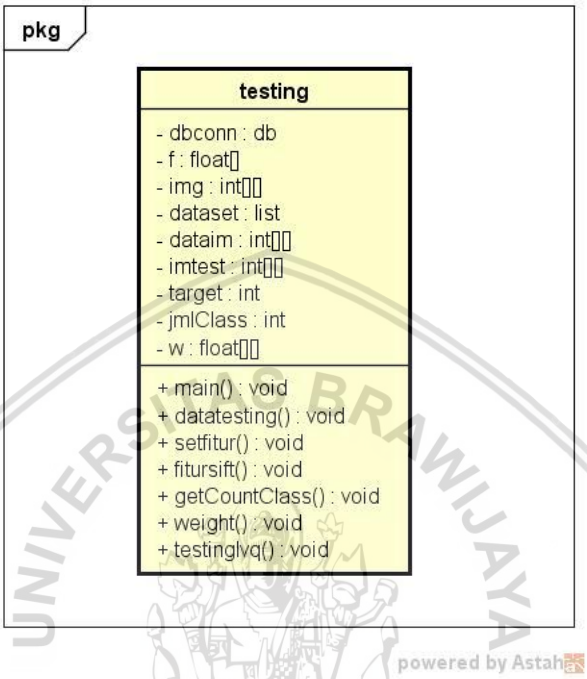
1. Method dataTrainning() merupakan method untuk inisialisasi data trainning.
2. Method fitursift() merupakan method untuk mengolah fitur jumlah *keypoint descriptor* citra.
3. Method setfitur() merupakan method untuk ekstraksi fitur dari citra latih.
4. Method getCount*Class*() merupakan method untuk mendapat banyaknya kelas yang ada pada dataset.
5. Method weight() merupakan method untuk insialisasi bobot fitur pada setiap kelas.
6. Mthod trainlvq() adalah method untuk proses pelatihan menggunakan Algoritme *Learning Vector Quantization* .

### 5.2.2 Struktur *Class Testing*

Pada Gambar 5.5 merupakan gambar dari struktur c*lass testing*. *class testing* merupakan kelas yang digunakan untuk proses pengujian pada data uji pengenalan tanda tangan. *Class* testing terdiri dari beberapa *method* yaitu :

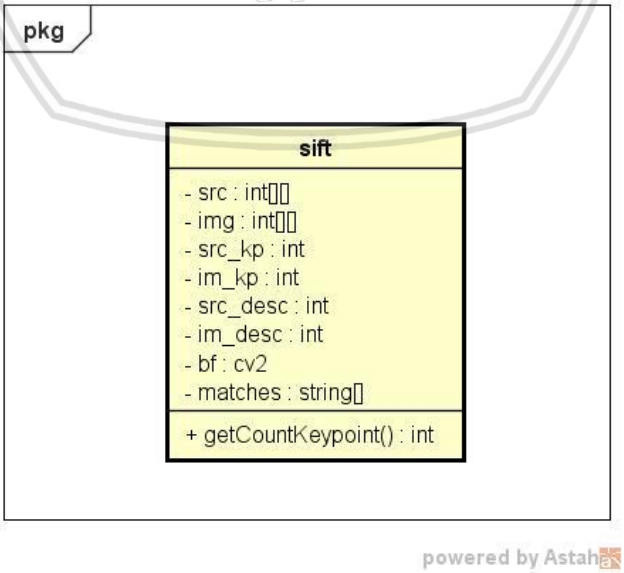
1. Method dataTesting() adalah method untuk inisialisasi dan praproses citra uji.
2. Method fitursift() merupakan method untuk mengolah fitur jumlah *keypoint descriptor* dari citra uji.
3. Method setfitur() merupakan method untuk ekstraksi fitur dari citra uji.

4. Method getCount*Class*() merupakan method untuk mendapat banyaknya kelas yang ada pada *dataset*.
5. Method weight() adalah method inisialisasi bobot fitur menggunakan nilai hasil proses *trainning* yang disimpan pada *database*.
6. Method testinglvq() merupakan *method* yang digunakan untuk proses pengujian citra uji.




Gambar 5.5 Struktur *Class* testing

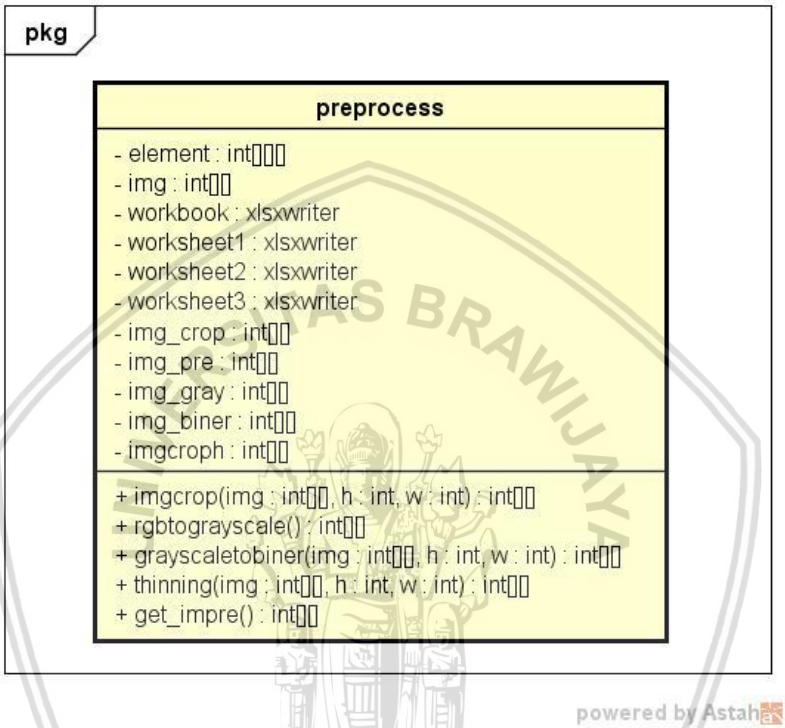
### 5.2.3 Struktur *Class* sift




Gambar 5.6 Struktur *Class* sift

Pada Gambar 5.6 menunjukkan struktur dari *class* sift yang merupakan *class* untuk mengolah data citra dengan hasil *output* berupa jumlah *keypoint descriptor* dalam citra. Terdapat satu *method* yaitu getCount*Keypoint*() yang merupakan *method* berisikan instansiasi *method* deteksi fitur yaitu cv2.xfeatures2d.SIFT_create() yang dipanggil dari c*lass* openCV.

### 5.2.4 Struktur *Class preprocess*



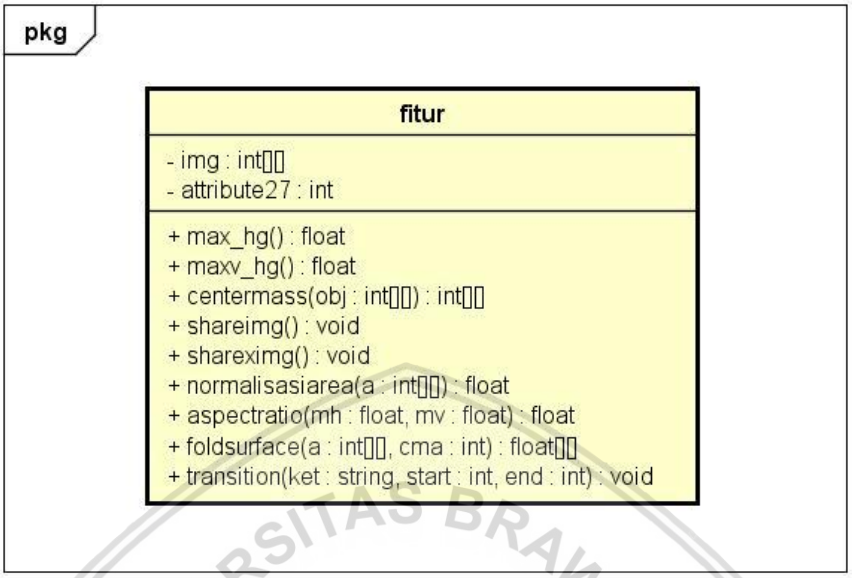

Gambar 5.7 Struktur *Class Preprocess*

Pada Gambar 5.7 merupakan struktur dari *class* preprocess yaitu *class* yang digunakan untuk praproses citra sebelum diolah seperti pengambil ekstraksi fitur citra. Beberapa method yang ada dalam *class* preprocess antara lain adalah imgcrop() untuk pemotongan ukuran citra yang sudah dilakukan proses *thinning*. Method rgbto*grayscale*() merupakan method untuk mengubah citra rgb menjadi citra tingkat keabuan. Method *grayscale*tobiner() merupakan method untuk merubah citra *grayscale* menjadi citra biner. Method *thinning*() merupakan method untuk proses *thinning* menggunakan metode *Zhangsuen*.

### 5.2.5 Struktur *Class* fitur

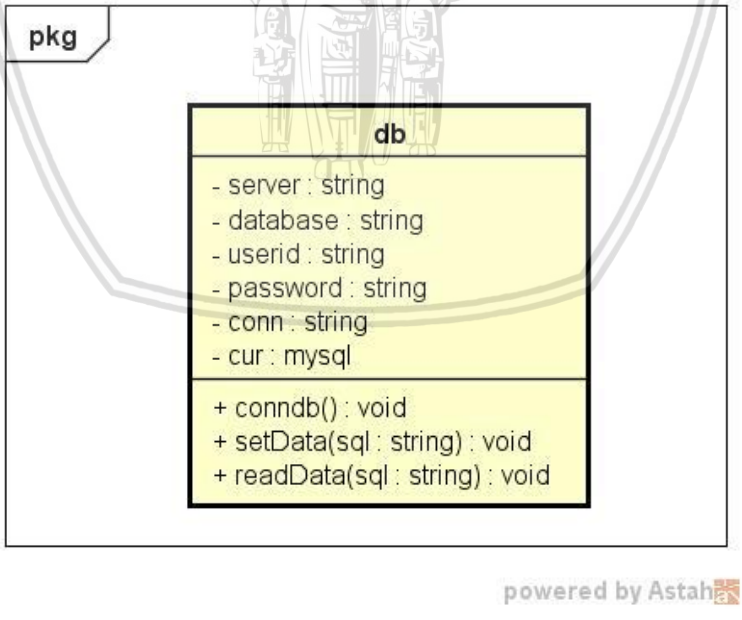
Struktut *class* fitur ditunjukkan pada Gambar 5.8. *Class* fitur digunakan untuk ekstraksi fitur citra. Pada *class* ini terdiri dari method-method seperti max_hg() dan max_vg() yang digunakan untuk menghitung nilai maksimum histogram secara horizontal dan vertikal, *method* untuk menghitung massa pusat, *method* untuk menghitung fitur transisi, *method* untuk menghitung fitur normalisasi area, *method* untuk menghitung rasio panjang dan lebar objek, *method* untuk

menghitung normalisasi 3 area dari satu citra tanda tangan, *method* untuk menghitung area atas dan bawah dari centermass pada 3 area hasil pembagian area citra tanda tangan.




Gambar 5.8 Struktur *Class* fitur

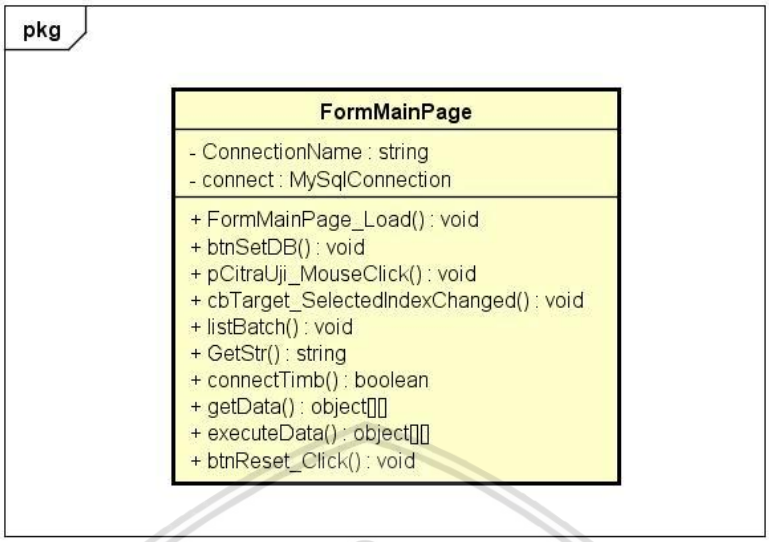
### 5.2.6 Struktur *Class* db




Gambar 5.9 Struktur *Class* db

Pada Gambar 5.9 merupakan struktur dari *Class* db yaitu *Class database* yang digunakan untuk pengaturan koneksi serta proses pemasukkan dan pengeluaran data dalam *database*.

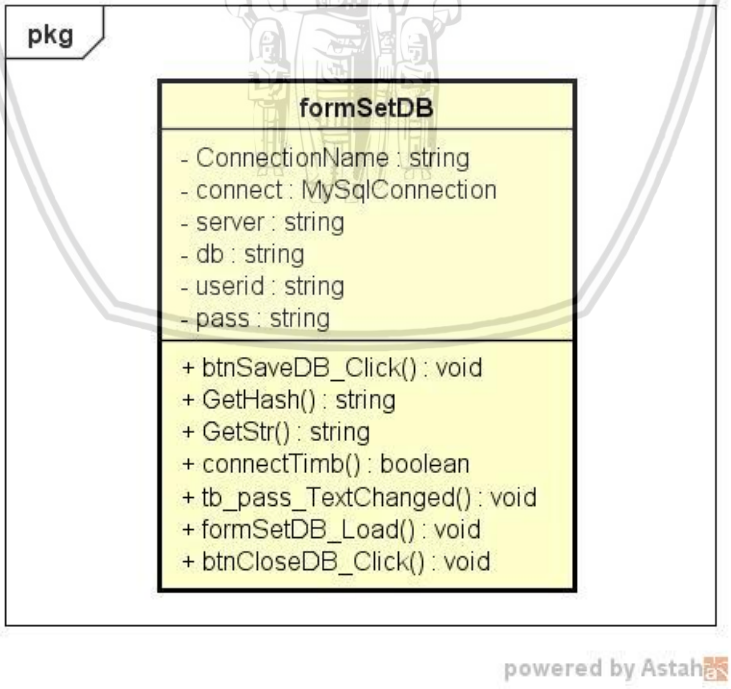
### 5.2.7 Struktur *Class* FormMainPage




Gambar 5.10 Struktur *Class* FormMainPage

Pada Gambar 5.10 merupakan struktur *class* FormMainPage yaitu *class* dari *form* halaman utama sistem pengenalan tanda tangan.

### 5.2.8 Struktur *Class* formSetDB




Gambar 5.11 Struktur *Class* formSetDB

Gambar 5.11 merupakan gambar dari struktur *class* formSetDB yaitu c*lass* pada *user interface* yang digunakan untuk koneksi atau pengaturan *database*.

## 5.3 Implementasi Kode Program

### 5.3.1 Proses Ekstraksi Fitur Fraktal Global

Proses ekstraksi fitur fraktal global meliputi nilai maksimum histogram piksel horizontal dan vertikal, nilai pusat massa objek, normalisasi area objek, rasio aspek, fitur *three surface*, fitur *six fold surface* dan fitur transisi. Kode program untuk ekstraksi fitur fraktal global ditunjukkan pada Gambar 5.12.

```
1  class fitur:
2      def __init__(self, img):
3          self.img = img
4
5      #nilai maximum dari jumlah pixel horizontal
6      def maxh_hg(self):
7          h = 0
8          total = []
9          #print len(self.img)
10         for h in range(len(self.img)):
11             total.append(0)
12         i = 0
13         j = 0
14         max = 0
15         for i in range(len(self.img)):
16             for j in range(len(self.img[i])):
17                 total[i] = total[i]+self.img[i][j]
18             if max < total[i]:
19                 max = total[i]
20         return max, len(self.img[0])
21
22     #nilai maximum dari piksel vertikal
23     def maxv_hg(self):
24         total = []
25         for v in range(len(self.img[0])):
26             total.append(0)
27         max = 0
28         for i in range(len(self.img[0])):
29             for j in range(len(self.img)):
30                 total[i] = total[i]+self.img[j][i]
31             if max < total[i]:
32                 max = total[i]
```

```
33          return max, len(self.img)
34
35      #Nilai massa pusat
36      def centermass(self, obj):
37          cx, cy = 0, 0
38          luas = 0
39          for q in range(len(obj)):
40              for p in range(len(obj[q])):
41                  if obj[q][p] == 1:
42                      luas = luas+1
43                      cx = cx+p+1
44                      cy = cy+q+1
45          cx = cx/luas
46          cy = cy/luas
47          # print(cx, cy)
48          return cx, cy
49
50      #Method pembagian area citra
51      def sharexImg(self, bagx):
52          t = len(self.img)
53          l = (len(self.img[0]))/bagx
54          obj = [[[0 for x in range(l)] for y in range(t)]
    for z in range(bagx)]
55          for i in range(t):
56              a = 0
57              for j in range(len(self.img[i])):
58                  if bagx == 2:
59                      if j < l:
60                          obj[0][i][j] = self.img[i][j]
61                      else:
62                          if a == l :
63                              break
64                          # print(a, i, j)
65                          obj[1][i][a] = self.img[i][j]
66                          a = a + 1
67                  elif bagx == 3:
68                      if j == l+1:
69                          a = 0
70                      if j < l:
```

```
71                          obj[0][i][j] = self.img[i][j]
72                      elif j >= l and j < l+l:
73                          obj[1][i][a] = self.img[i][j]
74                          a = a + 1
75                      else:
76                          #print(a, i, j)
77                          if a == l :
78                              break
79                          obj[2][i][a] = self.img[i][j]
80                          a = a + 1
81          return obj
82
83      # fitur normalisasi area
84      def normalisasiarea(self, a):
85          area = 0
86          for i in range(len(a)):
87              for j in range(len(a[i])):
88                  area = area + a[i][j]
89          na = float(area)/float(len(a)*len(a[0]))
90          #print(na)
91          return na
92
93      # fitur aspek rasio
94      def aspectratio(self, a):
95          mv = len(a)
96          mh = len(a[0])
97          # print(mv, mh)
98          ra = float(mv)/float(mh)
99          return ra
100
101     # fitur fold surface
102     def foldsurface(self, a, cma):
103         fsup, fsbot = 0, 0
104
105         for i in range(0, cma):
106             for j in range(len(a[0])):
107                 fsup = fsup + a[i][j]
108
109         for i in range(cma+1, len(a)):
```

```
110             for j in range(len(a[0])):
111                 fsbot = fsbot + a[i][j]
112 
113         return fsup, fsbot
114 
115     # fitur transisi
116     def transition(self, ket, start, end):
117         ## trt -> total ratio transisi
118         ## ttr -> jumlah total transisi
119         ## ttl -> total kemungkinan transisi
120 
121         trt, ttr = 0, 0
122         cond = 'True'
123         if ket == 'upbottom' and start == 0 and end == 1:
    #up to bottom 0-1
124             for i in range(len(self.img)-1):
125                 for j in range(len(self.img[0])):
126                     if (self.img[i][j] == 0 and
    self.img[i+1][j] == 1):
127                         trt += (float(j+1)/float(i+2))
128                         ttr = ttr + 1
129                         # break
130             # print('ub', trt)
131 
132         elif ket == 'upbottom' and start == 1 and end == 0:
    #up to bottom 1-
133             for i in range(len(self.img)-1):
134                 for j in range(len(self.img[0])):
135                     if (self.img[i][j] == 1 and
    self.img[i+1][j] == 0):
136                         trt += (float(j+1)/float(i+2))
137                         ttr = ttr + 1
138                         # break
139             # print('ub', trt)
140 
141         elif ket == 'bottomup' and start == 0 and end == 1:
    #bottom to up 0-
142             i = len(self.img)-1
143             for k in range(len(self.img)-1):
```

```
144                 for j in range(len(self.img[0])):
145                     if (self.img[i][j] == 0 and self.img[i-
    1][j] == 1):
146                         trt += (float(j+1)/float(k+2))
147                         ttr = ttr + 1
148                         # break
149                 i = i - 1
150             # print('bu',trt)
151         elif ket == 'bottomup' and start == 1 and end == 0:
    #bottom to up 1-0
152             i = len(self.img)-1
153             for k in range(len(self.img)-1):
154                 for j in range(len(self.img[0])):
155                     if (self.img[i][j] == 1 and self.img[i-
    1][j] == 0):
156                         trt += (float(j+1)/float(k+2))
157                         ttr = ttr + 1
158                         # break
159                 i = i - 1
160             # print('bu',trt)
161
162         elif ket == 'leftright' and start == 0 and end ==
    1: #left to right 0-1
163             for j in range(len(self.img[0])-1):
164                 for i in range(len(self.img)):
165                     if (self.img[i][j] == 0 and
    self.img[i][j+1] == 1):
166                         trt += (float(i+1)/float(j+2))
167                         ttr = ttr + 1
168                         # break
169             # print('lr',trt)
170         elif ket == 'leftright' and start == 1 and end ==
    0: #left to right 1-0
171             for j in range(len(self.img[0])-1):
172                 for i in range(len(self.img)):
173                     if (self.img[i][j] == 1 and
    self.img[i][j+1] == 0):
174                         # trt += (float(j+2)/float(i+1))
175                         trt += (float(i+1)/float(j+2))
```

```
176                         ttr = ttr + 1
177                         # break
178             # print('lr',trt)
179         elif ket == 'rightleft' and start == 0 and end ==
    1: #right to left 0-1
180             j = len(self.img[0])-1
181             for k in range(len(self.img[0])-1):
182                 for i in range(len(self.img)):
183                     if (self.img[i][j] == 0 and
    self.img[i][j-1] == 1):
184                         trt += (float(i+1)/float(k+2))
185                         ttr = ttr + 1
186                         # break
187                 j = j - 1
188             # print('rl', trt)
189         elif ket == 'rightleft' and start == 1 and end ==
    0: #right to left 1-0
190             j = len(self.img[0])-1
191             for k in range(len(self.img[0])-1):
192                 for i in range(len(self.img)):
193                     if (self.img[i][j] == 1 and
    self.img[i][j-1] == 0):
194                         trt += (float(i+1)/float(k+2))
195                         ttr = ttr + 1
196                         # break
197                 j = j - 1
198             # print('rl', trt)
199         elif ket == 'total' and start == 0 and end == 1:
    #total transisi 0-1
200             ttl = 0
201             for i in range(len(self.img)-1): #upbottom
202                 for j in range(len(self.img[0])):
203                     if (self.img[i][j] == 0 and
    self.img[i+1][j] == 1):
204                         ttr = ttr + 1
205                     ttl = ttl + 1
206             for j in range(len(self.img[0])-1): #leftright
207                 for i in range(len(self.img)):
208
```

```
209                     if (self.img[i][j] == 0 and
    self.img[i][j+1] == 1):
210                         ttr = ttr + 1
211                     ttl = ttl + 1
212         elif ket == 'total' and start == 1 and end == 0:
    #total transisi 1-0
213             ttl = 0
214             for i in range(len(self.img)-1): #upbottom
215                 for j in range(len(self.img[0])):
216                     if (self.img[i][j] == 1 and
    self.img[i+1][j] == 0):
217                         ttr = ttr + 1
218                     ttl = ttl + 1
219             for j in range(len(self.img[0])-1): #leftright
220                 for i in range(len(self.img)):
221                     if (self.img[i][j] == 1 and
    self.img[i][j+1] == 0):
222                         ttr = ttr + 1
223                     ttl = ttl + 1
224         return trt, ttr
```

Gambar 5.12 Kode Program Ekstraksi Fitur Fraktal Global

Penjelasan:

1. Baris 1 merupakan deklarasi *class* fitur.
2. Baris 2 – 3 merupakan *method constructor* untuk deklarasi dan inisialisasi citra yang diekstraksi fiturnya.
3. Baris 6-20 merupakan *method* untuk menghitung fitur nilai maksimum histogram piksel secara horizontal.
4. Baris 23-33 merupakan *method* untuk menghitung fitur nilai maksimum histogram piksel secara vertikal.
5. Baris 36-48 merupakan *method* fitur pusat massa objek. Parameter dari *method* ini adalah objek.
6. Baris 31-81 merupakan *method* untuk pembagian area sebuah citra. Paremeter masukkan berupa jumlah area yang diinginkan.
7. Baris 84-91 merupakan fungsi untuk menghitung normalisasi area objek. Parameter masukkan berupa objek array 2 dimensi.
8. Baris 94-99 merupakan fungsi untuk menghitung fitur rasio aspek.
9. Baris 101-113 merupakan fungsi untuk menghitung banyak piksel hitam yang berada di atas dan di bawah nilai pusat massa (koordinat *y*).
10. Baris 116 merupakan deklarasi fungsi fitur transisi.
11. Baris 123-130 merupakan proses perhitungan fitur transisi untuk perpindahan nilai piksel 0 ke 1 dengan alur dari atas ke bawah.

12. Baris 132-139 merupakan proses perhitungan fitur transisi untuk perpindahan nilai piksel 1 ke 0 dengan alur dari atas ke bawah.
13. Baris 141-149 merupakan proses perhitungan fitur transisi untuk perpindahan nilai piksel 0 ke 1 dengan alur dari bawah ke atas.
14. Baris 152-160 merupakan proses perhitungan fitur transisi untuk perpindahan nilai piksel 1 ke 0 dengan alur dari bawah ke atas.
15. Baris 162-169 merupakan proses perhitungan fitur transisi untuk perpindahan nilai piksel 0 ke 1 dengan alur dari kiri ke kanan.
16. Baris 170-177 merupakan proses perhitungan fitur transisi untuk perpindahan nilai piksel 1 ke 0 dengan alur dari kiri ke kanan.
17. Baris 179-188 merupakan proses perhitungan fitur transisi untuk perpindahan nilai piksel 0 ke 1 dengan alur dari kanan ke kiri.
18. Baris 189-198 merupakan proses perhitungan fitur transisi untuk perpindahan nilai piksel 1 ke 0 dengan alur dari kanan ke kiri.
19. Baris 199-211 merupakan proses perhitungan fitur transisi untuk jumlah total perpindahan nilai piksel 0 ke 1.
20. Baris 212-223 merupakan proses perhitungan fitur transisi untuk jumlah total perpindahan nilai piksel 1 ke 0.

### 5.3.2 Proses Ekstraksi Fitur Jumlah *Keypoint Descriptor Scale Invariant Features Transform*(SIFT)

Proses ekstraksi fitur pada pengenalan tanda tangan dengan Algoritme *Learning Vector Quantization* (LVQ) salah satunya adalah dengan meggunakan metode *Scale Invariant Features Transform* (SIFT). Dimana ekstraksi fitur dengan *Scale Invariant Features Transform* (SIFT) menghasilkan fitur berupa jumlah *keypoint* yang sama antara satu citra dengan citra yang lain pada dataset. Kode program fungsi ekstraksi fitur menggunakan *Scale Invariant Features Transform* (SIFT) ditunjukkan pada Gambar 5.13.

```
1     def getCountKeypoint(self):
2         self.sift = cv2.xfeatures2d.SIFT_create(2000)
3         self.src_kp, self.src_desc =
  self.sift.detectAndCompute(self.src, None)
4         return len(self.src_desc)
```

Gambar 5.13 Kode Program Ekstraksi Fitur dengan *Scale Invariant Features Transform*(SIFT)

Penjelasan:

1. Baris 1 adalah deklarasi *method* tanpa parameter.
2. Baris 2 merupakan instansiasi atribut sift dari kelas OpenCV sebagai deklarasi penggunaan metode SIFT. Dengan nilai parameter jumlah maksimum *keypoint* sebesar 2000.
3. Baris 3 merupakan inisialisasi nilai banyak *keypoint* pada citra yang ditunjukkan dengan variabel src_kp. Inisialisasi nilai *keypoint descriptor* pada citra yang ditunjukkan dengan atribut src_desc. Nilai src_kp dan src_desc

diperoleh dari pengembalian nilai *method* detectAndCompute dengan parameter citra.

4. Baris 4 merupakan pengembalian nilai jumlah *keypoint descriptor* yang diperoleh dari panjang array variabel src_kp.

### 5.3.3 Proses Pembelajaran Learning Vector Quatization (LVQ)

Kode program proses pembelajaran *Learning Vector Quantization* (LVQ) menggunakan *method* trainlvq yang berada di *class trainning*. Pada proses pembelajaran *Learning Vector Quantization* (LVQ) menggunakan iterasi maksimum sebanyak 10 iterasi (Hamidi, et al., 2017). Data pembelajaran sistem pengenalan tanda tangan memiliki jumlah sebanyak citra latih dikali dengan jumlah citra latih dikurangi 1.

```
1     def trainlvq(self, alf, deca, maxe, em):
2         self.wrsh =
self.workbook.add_worksheet('threshold')
3         sql = "select * from α"
4         al = self.dbconn.readData(sql)
5         self.α = al[0][0]
6         decα = al[0][1]
7         eps = al[0][3]
8         maxepoch = al[0][2]
9         # self.α = alf
10        # decα = deca
11        # eps = em
12        # maxepoch = maxe
13        epoch = 0
14        self.t = [[] for x in range(self.jmlClass)]
#threshold kelasSistem
15
16        while ((epoch < maxepoch) and (self.α > eps)):
17            benar = 0
18            epoch = epoch + 1
19            eucl = [0 for x in range(self.jmlClass)]
20            for x in range(len(self.img)): #data citra
trainning
21                kelasSistem = 0
22                min = 9999
23                for j in range(self.jmlClass): #data Class
24                    jmlhWF = 0
```

```
25                      for k in range(len(self.f[x])): #data
fitur
26                          jmlhWF += math.pow((self.f[x][k] -
self.w[j][k]),2)
27                          # jmlhWF += (self.f[x][k] -
self.w[j][k])*(self.f[x][k] - self.w[j][k])
28                      eucl[j] = math.sqrt(jmlhWF)
29                      if min>eucl[j]:
30                          min = eucl[j]
31                          kelasSistem = j
32                  # if epoch == maxepoch:
33                  #     print(min, kelasSistem)
34
35                  if (kelasSistem+1) == self.cl[x]:
36                      benar = benar + 1
                        for k in range(len(self.f[x])):
37                          self.w[kelasSistem][k] +=
(self.α*(self.f[x][k]-self.w[kelasSistem][k]))
38                      if epoch == maxepoch:
39                          if epoch == maxepoch:
40                          # print(x, (kelasSistem+1), min)
41                          self.t[kelasSistem].append(min)
42                          #
self.t[kelasSistem][a[kelasSistem]] = min
43                          self.t[kelasSistem].sort(reverse =
True)
44                              # sql = "insert into
threshold(id_Class) values("+ str(kelasSistem+1) +")"
45                              # self.dbconn.setData(sql)
46                          if len(self.t[kelasSistem]) > 1:
47                              sql = "UPDATE threshold SET
threshold = "+str(self.t[kelasSistem][0])+" WHERE
id_Class="+str(kelasSistem+1)
48                                  # sql = "UPDATE threshold
SET threshold = "+str(self.t[kelasSistem][1])+" WHERE
id_Class="+str(kelasSistem+1)
49                              self.dbconn.setData(sql)
50                          else:
51
```

```
                                sql = "UPDATE threshold SET
threshold = "+str(self.t[kelasSistem][0])+" WHERE
52  id_Class="+str(kelasSistem+1)
53                              self.dbconn.setData(sql)
54                          # a[kelasSistem] += 1
55                          self.wrsh.write(x, 0, min)
                            # print(x, (kelasSistem+1),
56  eucl[j])
57
58                  else:
59                      for k in range(len(self.f[x])):
                            self.w[kelasSistem][k] -=
60  (self.α*(self.f[x][k]-self.w[kelasSistem][k]))
61              self.α -= (self.α * decα)
62
63          print(alf, decα, eps, epoch, benar)
64          for x in range(self.jml_Class):
65              for y in range(len(self.f[0])):
                    sql = "UPDATE _Classdata SET w"+str(y+1)+" =
66  "+str(self.w[x][y])+" WHERE id_Class="+str(x+1)
67                  self.dbconn.setData(sql)
68          sql = "UPDATE α SET α = "+str(self.α)
            self.dbconn.setData(sql)
```

Gambar 5.14 Kode Program Proses Pembelajaran *Learning Vector Quantization* (LVQ)

Penjelasan :

1. Baris 1 merupakan deklarasi dari *method*.
2. Baris 2 merupakan baris inisialisasi *worksheet Excel* dengan nama *threshold*.
3. Baris 3-4 merupakan baris pengambilan data parameter dari tabel α.
4. Baris 5 adalah Inisialisasi nilai variabel α.
5. Baris 6 merupakan inisialisasi nilai variabel *dec*α.
6. Baris 7 merupakan inisialisasi nilai variabel *eps*.
7. Baris 8 merupakan inisialisasi nilai variabel *maxepoch* untuk maksimum iterasi.
8. Baris 13 adalah inisialisasi awal nilai variabel *epoch*.
9. Baris 14 merupakan instansiasi dan inisialisasi variabel array *threshold* dengan nama t dan indeks berdasarkan banyak kelas.

10. Baris 16 adalah deklarasi pengulangan menggunakan *while* dengan syarat nilai variabel *epoch* harus dibawah nilai maksimum iterasi atau *maxepoch* dan nilai α harus diatas *eps* atau minimum *error*.
11. Baris 18 merupakan proses penambahan nilai *epoch* dengan menambahkan satu angka.
12. Baris 19 merupakan instansiasi dan inisialisasi dari variabel *eucl* dengan indeks berdasarkan banyak kelas. Variabel *eucl* digunakan untuk menyimpam nilai jarak satu citra pada setiap kelas.
13. Baris 20 adalah deklarasi pengulangan *for* untuk pengulangan data latih atau citra yang akan dilatih.
14. Baris 21 dan 22 merupakan inisialisasi variabel kelas sistem dan nilai minimum. Dimana variabel kelas sistem merupakan hasil kelas klasifikasi.
15. Baris 23 merupakan deklarasi pengulangan sebanyak jumlah kelas yang ada didalam pelatihan sistem pengenalan tanda tangan.
16. Baris 25-28 merupakan perhitungan nilai *euclidean distance* untuk setiap citra latih (*x*) pada setiap kelas yang disimbolkan dengan variabel *j*. Perhitungan nilai dilakukan dengan menghitung nilai *euclidean distance* dari setiap fitur yang ada pada setiap data latih.
17. Baris 29-31 merupakan proses mencari nilai minimum dari *euclidean distance* pada data latih (*x*). Nilai minimum *euclidean distance* menunjukkan kelas yang menjadi hasil kelas sistem dari data latih ke-*x*.
18. Baris 35-37 merupakan proses pengubahan nilai bobot pada kelas hasil perhitungan sistem. Dengan ketentuan jika kelas yang dihasilkan sistem sesuai dengan kelas target yang dimiliki data ke *i* maka bobot setiap fitur pada kelas yang ditunjukkan sistem akan semakin didekatkan nilainya.
19. Baris 38-52 merupakan percabangan yang akan dieksekusi setelah nilai epoch mencapai *maxepoch*. Percabangan ini adalah untuk menentukan nilai *threshold* pada kelas sistem yang benar. Nilai jarak minimum dari setiap citra pada klasifikasi yang benar sesuai dengan kelas target akan diurutkan dari nilai yang terbesar ke terkecil pada setiap kelas (ditunjukkan pada baris 41-43). Untuk *threshold* 1 merupakan nilai jarak minimum yang paling besar sehingga ditunjukkan dengan indeks bernilai 0 dari hasil pengurutan nilai jarak minimum perkelas. Dan untuk *threshold* 2 merupakan nilai jarak minimum terbesar kedua dengan ditunjukkan oleh indeks 1 dari hasil pengurutan. Hasil *threshold* disimpan pada *database*.
20. Baris 54 merupakan proses pemasukkan data ke *worksheet Excel*. Data yang dimasukkan adalah nilai-nilai minimum jarak dari klasifikasi data yang sudah benar.
21. Baris 58-59 merupakan proses pengubahan nilai bobot pada kelas hasil perhitungan sistem. Jika nilai kelas data ke *i* tidak sesuai target, maka nilai bobot setiap fitur pada kelas yang dihasilkan sistem akan di jauhkan nilainya.

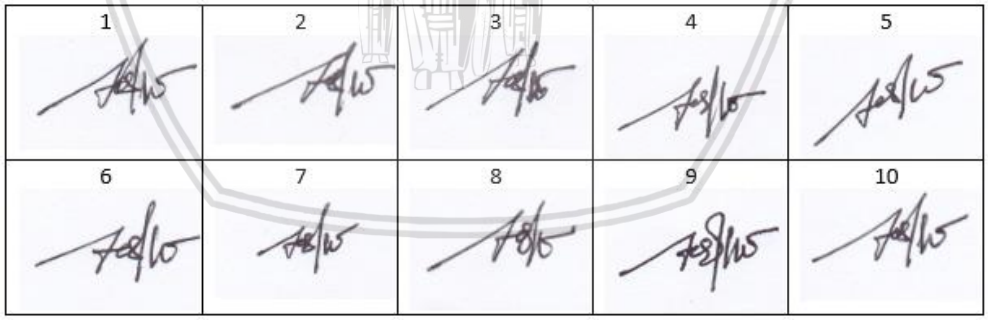
22. Baris 60 merupakan pembaruan nilai dari nilai *α* yaitu dengan mengalikan nilai *α* dengan *decα*.
23. Baris 63-66 merupakan baris untuk mengubah nilai bobot fitur pada setiap kelas klasifikasi sistem tanda tangan yang ada pada *database*.
24. Baris 67-68 merupakan proses untuk mengubah nilai α pada *database* tabel *α*.

# BAB 6 PENGUJIAN DAN PEMBAHASAN

Bab pengujian dan pembahasan merupakan bab yang menguraikan tentang hasil pengujian sistem pengenalan tanda tangan dengan algoritme *Learning Vector Quantization* (LVQ) yang telah dirancang dan membahas hasil dari pengujian tersebut. Pengujian sistem pengenalan tanda tangan dengan algoritme *Learning Vector Quantization* (LVQ) berdasarkan pada parameter yang digunakan dalam Algoritme. Sehingga, pengujian yang dilakukan antara lain pengujian nilai *Learning Rate* (α), pengujian nilai pengurang α (*decrement* α), pengujian jumlah data latih dan data uji, pengujian nilai maksimum iterasi dan pengujian nilai *error* minimum.

## 6.1 Hasil dan Pembahasan Pengujian Variasi Fitur

Pengujian variasi fitur digunakan untuk menguji variasi nilai fitur yang digunakan untuk pengenalan tanda tangan. Pengujian variasi fitur dilakukan dengan menggunakan perhitungan standar deviasi dari nilai-nilai semua fitur dalam kelas yang sama. Tujuan dari pengujian fitur dengan menggunakan nilai standar deviasi adalah untuk melakukan pengurangan fitur sehingga hanya fitur dengan perseberan variasi paling dekat atau sedikit yang akan dipilih. Fitur-fitur yang diuji antara lain nilai maksimum dari histogram horizontal dan vertikal, massa pusat, rasio aspek antara nilai maksimum histogram horizontal dan vertikal, nilai normalisasi area tanda tangan, fitur *three surface*, fitur *six fold surface*, Fitur transisi dan fitur banyak *keypoint descriptor* suatu citra yang diperoleh dari proses *Scale Invariant Features Transform* (SIFT). Gambar 6.1 merupakan citra tanda tangan yang digunakan untuk pengujian fitur.




Gambar 6.1 10 Citra Tanda Tangan untuk Pengujian Variasi Fitur

Hasil nilai ekstraksi fitur dari citra tanda tangan untuk pengujian fitur yang sudah ternormalisasi beserta hasil perhitungan standar deviasi setiap fitur ditunjukkan pada Tabel 6.1.

**Tabel 6.1 Nilai fitur dan Hasil Standar Deviasi Fitur dalam 10 Citra**

| Fitur | Citra | | | | | | | | | | STD |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
| 1 | 0.183 | 0.400 | 0.233 | 0.233 | 0.217 | 0.167 | 0.183 | 0.217 | 0.217 | 0.150 | 0.069 |
| 2 | 0.217 | 0.317 | 0.300 | 0.333 | 0.417 | 0.333 | 0.400 | 0.317 | 0.250 | 0.317 | 0.060 |

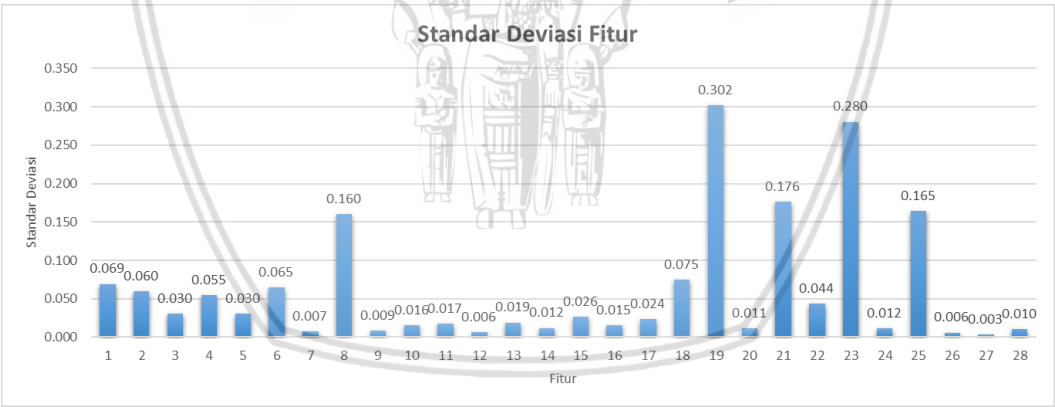
| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 0.586 | 0.621 | 0.552 | 0.621 | 0.552 | 0.621 | 0.586 | 0.552 | 0.586 | 0.621 | 0.030 |
| 4 | 0.542 | 0.593 | 0.695 | 0.593 | 0.644 | 0.627 | 0.593 | 0.627 | 0.712 | 0.695 | 0.055 |
| 5 | 0.345 | 0.345 | 0.414 | 0.345 | 0.310 | 0.345 | 0.345 | 0.379 | 0.310 | 0.345 | 0.030 |
| 6 | 0.441 | 0.508 | 0.508 | 0.542 | 0.305 | 0.492 | 0.441 | 0.458 | 0.475 | 0.475 | 0.065 |
| 7 | 0.057 | 0.080 | 0.072 | 0.065 | 0.061 | 0.057 | 0.066 | 0.059 | 0.061 | 0.065 | 0.007 |
| 8 | 0.846 | 0.792 | 0.778 | 0.700 | 0.520 | 0.500 | 0.458 | 0.684 | 0.867 | 0.474 | 0.160 |
| 9 | 0.027 | 0.053 | 0.037 | 0.029 | 0.042 | 0.026 | 0.032 | 0.028 | 0.028 | 0.028 | 0.009 |
| 10 | 0.088 | 0.131 | 0.076 | 0.102 | 0.102 | 0.086 | 0.107 | 0.087 | 0.108 | 0.106 | 0.016 |
| 11 | 0.055 | 0.056 | 0.104 | 0.064 | 0.038 | 0.058 | 0.060 | 0.061 | 0.046 | 0.061 | 0.017 |
| 12 | 0.021 | 0.036 | 0.021 | 0.018 | 0.028 | 0.017 | 0.021 | 0.018 | 0.016 | 0.016 | 0.006 |
| 13 | 0.033 | 0.073 | 0.096 | 0.045 | 0.077 | 0.044 | 0.050 | 0.050 | 0.065 | 0.065 | 0.019 |
| 14 | 0.089 | 0.106 | 0.067 | 0.076 | 0.104 | 0.085 | 0.092 | 0.082 | 0.090 | 0.091 | 0.012 |
| 15 | 0.086 | 0.157 | 0.073 | 0.129 | 0.097 | 0.079 | 0.113 | 0.090 | 0.123 | 0.117 | 0.026 |
| 16 | 0.052 | 0.059 | 0.090 | 0.048 | 0.059 | 0.054 | 0.071 | 0.085 | 0.045 | 0.072 | 0.015 |
| 17 | 0.056 | 0.053 | 0.118 | 0.075 | 0.031 | 0.058 | 0.050 | 0.044 | 0.046 | 0.051 | 0.024 |
| 18 | 0.054 | 0.073 | 0.178 | 0.073 | 0.033 | 0.268 | 0.068 | 0.022 | 0.075 | 0.070 | 0.075 |
| 19 | 0.083 | 0.389 | 0.333 | 0.661 | 0.050 | 0.018 | 0.704 | 0.741 | 0.600 | 0.017 | 0.302 |
| 20 | 0.021 | 0.047 | 0.034 | 0.022 | 0.017 | 0.019 | 0.040 | 0.040 | 0.018 | 0.018 | 0.011 |
| 21 | 0.071 | 0.095 | 0.038 | 0.466 | 0.083 | 0.439 | 0.386 | 0.415 | 0.383 | 0.350 | 0.176 |
| 22 | 0.053 | 0.071 | 0.174 | 0.048 | 0.077 | 0.143 | 0.067 | 0.089 | 0.049 | 0.045 | 0.044 |
| 23 | 0.194 | 0.029 | 0.500 | 0.423 | 0.017 | 0.019 | 0.673 | 0.661 | 0.017 | 0.018 | 0.280 |
| 24 | 0.021 | 0.044 | 0.033 | 0.043 | 0.017 | 0.019 | 0.038 | 0.038 | 0.017 | 0.018 | 0.012 |
| 25 | 0.077 | 0.105 | 0.042 | 0.448 | 0.050 | 0.421 | 0.351 | 0.377 | 0.350 | 0.317 | 0.165 |
| 26 | 0.024 | 0.016 | 0.031 | 0.017 | 0.030 | 0.017 | 0.017 | 0.016 | 0.026 | 0.023 | 0.006 |
| 27 | 0.024 | 0.020 | 0.022 | 0.017 | 0.026 | 0.017 | 0.017 | 0.016 | 0.022 | 0.023 | 0.003 |
| 28 | 0.011 | 0.046 | 0.024 | 0.025 | 0.025 | 0.009 | 0.026 | 0.021 | 0.030 | 0.018 | 0.010 |

**Tabel 6.2 Keterangan Nama Fitur**

| No. | Nama Fitur | Keterangan |
|---|---|---|
| 1 | Max HH | Nilai Maksimum Histogram Vertikal |
| 2 | Max HV | Nilai Maksimum Histogram Horizontal |
| 3 | massa pusat 1 | Massa Pusat Area 1 pada x |
| 4 | massa pusat 1 | Massa Pusat Area 1 pada y |
| 5 | massa pusat 2 | Massa Pusat Area 2 pada x |
| 6 | massa pusat 2 | Massa Pusat Area 1 pada y |
| 7 | N. Area | Normalisasi Area |
| 8 | Aspek Rasio | Nilai Rasio lebar dan panjang objek dalam citra |
| 9 | N. Area 1 | Normalisasi Area bagian 1 |
| 10 | N. Area 2 | Normalisasi Area bagian 2 |
| 11 | N. Area 3 | Normalisasi Area bagian 3 |
| 12 | fs1a | Fitur *Six Fold Surface* Atas Massa Pusat bagian 1 |
| 13 | fs1b | Fitur *Six Fold Surface* Bawah Massa Pusat bagian 1 |
| 14 | fs2a | Fitur *Six Fold Surface* Atas Massa Pusat bagian 2 |

| | | |
|---|---|---|
| 15 | fs2b | Fitur *Six Fold Surface* Bawah Massa Pusat bagian 2 |
| 16 | fs3a | Fitur *Six Fold Surface* Atas Massa Pusat bagian 3 |
| 17 | fs3b | Fitur *Six Fold Surface* Bawah Massa Pusat bagian 3 |
| 18 | tbu01 | Fitur Transisi Piksel 0 ke 1 dari Bawah ke Atas |
| 19 | tub01 | Fitur Transisi Piksel 0 ke 1 dari Atas ke Bawah |
| 20 | tlr01 | Fitur Transisi Piksel 0 ke 1 dari Kiri ke Kanan |
| 21 | trl01 | Fitur Transisi Piksel 0 ke 1 dari Kanan Ke Kiri |
| 22 | tbu10 | Fitur Transisi Piksel 1 ke 0 dari Bawah ke Atas |
| 23 | tub10 | Fitur Transisi Piksel 1 ke 0 dari Atas ke Bawah |
| 24 | tlr10 | Fitur Transisi Piksel 1 ke 0 dari Kiri ke Kanan |
| 25 | trl10 | Fitur Transisi Piksel 1 ke 0 dari Kanan Ke Kiri |
| 26 | total01 | Fitur Total Transisi Piksel 0 ke 1 |
| 27 | total10 | Fitur Total Transisi Piksel 0 ke 1 |
| 28 | sift | Fitur Banyak *Keypoint* |

Pada Tabel 6.1 standar deviasi setaip fitur dapat direpresentasikan dalam bentuk grafik yang ditunjukkan pada Gambar 6.2. Dari Tabel 6.1 nilai standar deviasi fitur tertinggi adalah 0,328 yaitu pada fitur aspek ratio. Sedangkan nilai standar deviasi fitur terendah adalah 0,006 yang dimiliki oleh fitur total transisi piksel bernilai 1 ke 0 dan 0 ke 1 serta transisi dari piksel 1 ke 0 dengan arah dari bawah citra ke atas citra.

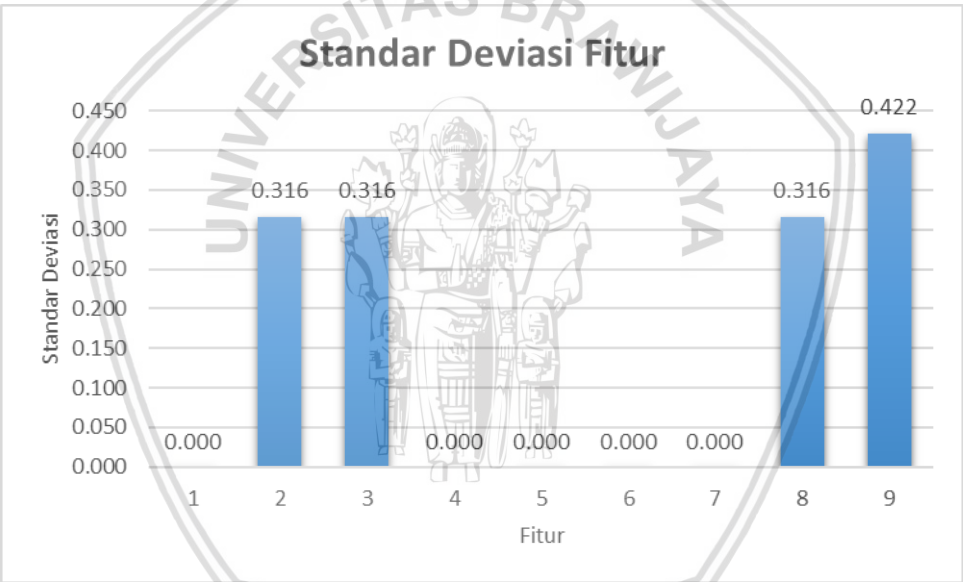



Gambar 6.2 Gambar Nilai Standar Deviasi Fitur

Pada penelitian pengenalan tanda tangan ada beberapa fitur yang lain yang dapat digunakan yaitu fitur dengan pembagian citra menjadi beberapa zona. Ekstraksi dengan metode pembagian zona ini adalah satu citra dibagi menjadi 9 zona. Dimana piksel pada citra harus memiliki nilai 0 yang merepresentasikan putih dan 1 yang merepresentasikan hitam. Selanjutnya setiap piksel pada setiap zona dijumlahkan. Jika jumlah nilai piksel pada zona lebih dari 1 maka fitur bernilai 1 dan jika jumlah nilai piksel pada zona sama dengan 0 maka fitur bernilai 0. Sehingga dalam ektraksi fitur pengenalan tanda tangan menggunakan jumlah nilai piksel setiap zona mempunyai fitur sebanyak jumlah zona yang dibuat (Hidayatno, et al., 2008).

**Tabel 6.3 Standar Deviasi Fitur dengan Menggunakan Ekstraksi Ciri Pembagian Zona**

| Fitur | Citra | | | | | | | | | | STD |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
| Fitur 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Fitur 2 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.316228 |
| Fitur 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0.316228 |
| Fitur 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| Fitur 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| Fitur 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| Fitur 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| Fitur 8 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0.316228 |
| Fitur 9 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0.421637 |




Gambar 6.3 Gambar Nilai Standar Deviasi Fitur dengan Ekstraksi Ciri Pembagian Zona

Dari Tabel 6.3 dapat disimpulkan bahwa fitur 1, 4, 5, 6, dan 7 memiliki nilai standar deviasi 0. Sedangkan pada fitur 2, 3, 8 memiliki nilai standar deviasi 0,316228 dan fitur 9 nilai standar deviasinya adalah 0,4216. Pada penggunaan ekstraksi fitur menggunakan pembagian zona yaitu dengan 9 fitur diperoleh akurasi 95% pada data latih pengenalan tanda tangan menggunakan Algoritme *backpropagation* (Hidayatno, et al., 2008). Nilai standar deviasi dari fitur dengan ekstraksi ciri pembagian zona memiliki nilai tertinggi yaitu 0,422 dan lebih besar nilainya daripada keseluruhan nilai standar deviasi fitur fraktal global dan jumlah *keypoint descriptor*. Kesimpulan yang diperoleh dari pengenalan tanda tangan menggunakan ekstraksi fitur fraktal global dan jumlah *keypoint descriptor* memiliki variasi yang dapat digunakan sebagai fitur dalam pengenalan tanda tangan, karena

nilai standar deviasi fitur-fitur dibawah nilai terbesar dari standar deviasi fitur pengenalan tanda tangan menggunakan pembagian zona pada citra yaitu sebesar 0,422.

## 6.2 Hasil dan Pembahasan Pengujian Nilai *Learning Rate* (α)

Pada pengujian nilai *Learning Rate* atau α merupakan pengujian terhadap nilai α awal yang digunakan untuk proses pelatihan dataset pengenalan tanda tangan. Sehingga dataset yang digunakan dalam pengujian nilai *Learning Rate* atau α adalah data yang dikelompokkan pada data latih. Beberapa parameter dalam pengujian nilai *Learning Rate* (α) yaitu:

Jumlah kelas = 20

Jumlah data uji = 100

Nilai *decrement* α = 0,1

Nilai maksimum iterasi = 100

Nilai *error* minimum = $1 \times 10^{-10}$

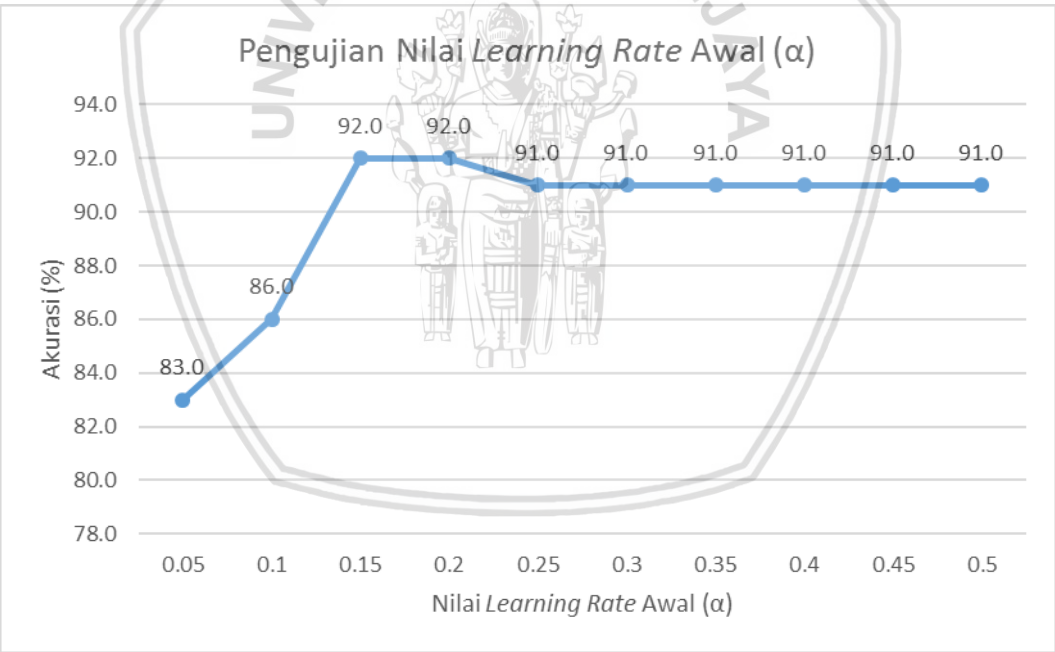
Untuk nilai pengurang *Learning Rate* atau *decrement* α dan nilai maksimum iterasi ditentukan berdasarkan pada penelitian jaringan syaraf tiruan *Learning Vector Quantization* untuk pengenalan tanda tangan (Qur'ani & Rosmalinda, 2010). Untuk nilai *error* minimum ditentukan berdasarkan penelitian implementasi *Learning Vector Quantization* untuk klasifikasi kualitas air sungai (Hamidi, et al., 2017). Rentang nilai *Learning Rate* atau α yang digunakan untuk pengujian adalah 0,025 sampai 0,25 dengan kelipatan 0,025.

Tabel 6.4 menunjukkan hasil dari pengujian nilai *Learning Rate* atau α awal pada data latih dengan setiap nilai α dilakukan 10 kali percobaan.

**Tabel 6.4 Hasil Pengujian Nilai *Learning Rate* (α) dengan Menggunakan Data Latih**

| A | Percobaan (Akurasi %) | | | | | | | | | | Rata-rata Akurasi (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
| 0.05 | 83 | 83 | 83 | 83 | 83 | 83 | 83 | 83 | 83 | 83 | 83,0 |
| 0.1 | 86 | 86 | 86 | 86 | 86 | 86 | 86 | 86 | 86 | 86 | 86,0 |
| 0.15 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92,0 |
| 0.2 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92,0 |
| 0.25 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91,0 |
| 0.3 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91,0 |
| 0.35 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91,0 |
| 0.4 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91,0 |
| 0.45 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91,0 |
| 0.5 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91,0 |

Tabel 6.4 menjadi dasar dalam membuat grafik yang merepresentasikan perubahan rata-rata akurasi pada setiap nilai *Learning Rate* awal yang diuji pada proses pelatihan yang ditunjukkan pada Gambar 6.4. Pada Gambar 6.4 rata-rata akurasi terbesar adalah pada nilai *Learning Rate* = 0,15 dengan besar akurasi 92%. Sedangkan nilai akurasi terkecil berada pada nilai *Learning Rate* = 0,05. Hasil akurasi pada nilai *Learning Rate* dibawah 0,15 yaitu mulai dari nilai *Learning Rate* 0,05 sampai 0,1 mengalami perubahan nilai akurasi yang semakin meningkat dengan ditunjukkan adanya peningkatan yang dapat dilihat di grafik pada Gambar 6.1. Untuk nilai *Learning Rate* diatas 0,15 nilai akurasi mulai mengalami penurunan. Pada nilai *Learning Rate* 0,05 nilai akurasi sangat rendah yaitu 83% dengan rata-rata kebenaran hanya 83 citra dari 100 citra latih. Maka, pengujian *Learning Rate* mendapatkan nilai *Learning Rate* atau α yang optimal yaitu 0,15. Dari pembahasan pengujian nilai *Learning Rate* disimpulkan bahwa jika nilai *Learning Rate* terlalu kecil maka akurasi akan semakin kecil karena perubahan bobot fitur citra latih yang lambat, sedangkan jika nilai *Learning Rate* terlalu besar dapat membuat nilai akurasi juga semakin kecil karena proses pelatihan yang cepat yang disebabkan oleh proses perubahan bobot fitur terlalu besar nilainya. Sehingga, diketahui bahwa nilai α atau *Learning Rate* mempengaruhi perubahan bobot dalam proses pelatihan (Hamidi, et al., 2017).




Gambar 6.4 Grafik Pengujian Nilai *Learning Rate* Awal pada Data Latih

## 6.3 Hasil dan Pembahasan Pengujian Nilai Pengurang *Learning Rate* (α) atau *Decrement* α

Pengujian nilai *decrement* α adalah pengujian terhadap besar nilai pengurang *Learning Rate* atau α. Pada pengujian *decrement* α variabel yang diperlukan antara lain :

Jumlah kelas = 20

Jumlah data latih = 100

Nilai *Learning Rate* atau α = 0,15

Nilai maksimum iterasi = 100

Nilai *error* minimum = $1 \times 10^{-10}$

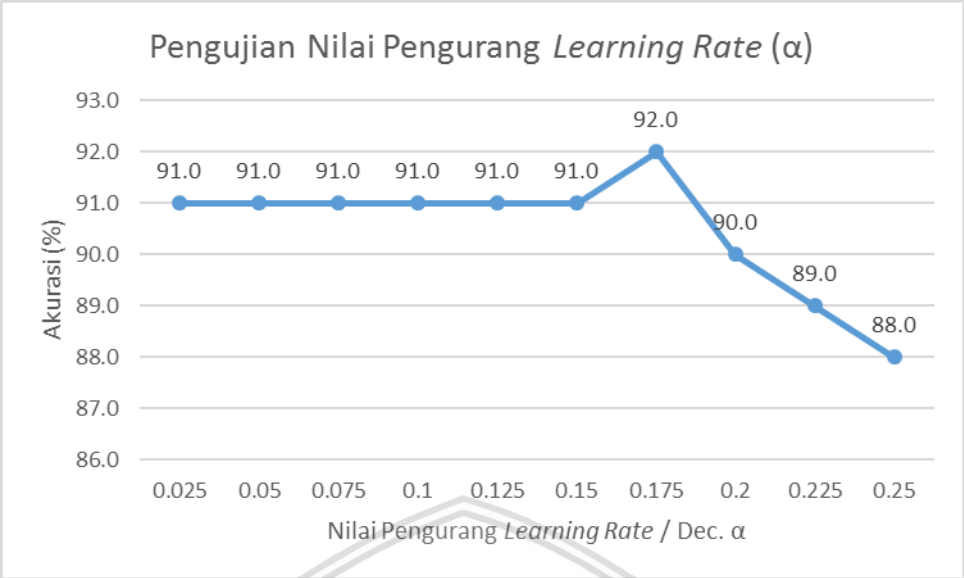
Nilai *Learning Rate* atau α didapatkan dari pengujian sebelumnya yaitu pengujian nilai *Learning Rate* (α) awal dengan nilai akurasi yang terbaik. Nilai *decrement* α yang digunakan memiliki nilai antara 0,025 sampai 0,25 dengan kelipatan 0,025.

Pada Tabel 6.5 merepresentasikan hasil pengujian nilai pengurang *Learning Rate* (α) atau *decrement* α dengan 10 percobaan setiap nilai *decrement* α.

**Tabel 6.5 Hasil Pengujian Nilai Pengurang *Learning Rate* (α) atau *Decrement* α**

| Dec. A | Percobaan (Akurasi %) | | | | | | | | | | Rata-rata Akurasi (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
| 0.025 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91,0 |
| 0.05 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91,0 |
| 0.075 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91,0 |
| 0.1 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91,0 |
| 0.125 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91,0 |
| 0.15 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91 | 91,0 |
| 0.175 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92,0 |
| 0.2 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90,0 |
| 0.225 | 89 | 89 | 89 | 89 | 89 | 89 | 89 | 89 | 89 | 89 | 89,0 |
| 0.25 | 88 | 88 | 88 | 88 | 88 | 88 | 88 | 88 | 88 | 88 | 88,0 |

Berdasarkan Tabel 6.5 maka digambarkan grafik yang menunjukkan perubahan rata-rata akurasi pada setiap nilai *decrement* α pada data latih yang ditunjukkan pada Gambar 6.5. Nilai *decrement* α dengan akurasi terbesar adalah 0,175 dimana akurasi mencapai 92%. Pada nilai *decrement* α dibawah 0,175 yaitu dari nilai *decrement* α 0,025 sampai dengan 0,15 akurasi stabil dengan besar 91%. Lalu mengalami penurunan setelah 0,175. Karena nilai *decrement* α merupakan nilai yang digunakan untuk pengurangan *Learning Rate*, maka analisis nilai *decrement* α sama dengan *Learning Rate* atau α. Jika nilai *decrement* α kecil, maka akurasi akan kecil disebabkan perubahan nilai α semakin kecil disetiap proses pembelajaran. Dan jika nilai *decrement* α terlalu besar maka nilai α akan terlalu besar berubahnya sehingga mempengaruhi kecepatan perubahan bobot fitur pada proses pelatihan (Hamidi, et al., 2017).




Gambar 6.5 Grafik Pengujian Nilai Pengurang *Learning Rate* (α ) atau *Decrement* α

## 6.4 Hasil dan Pembahasan Pengujian Nilai Maksimum Iterasi

Pengujian nilai maksimum iterasi adalah pengujian terhadap maksimum iterasi yang dilakukan dalam proses pembelajaran. Beberapa parameter dalam pengujian nilai maksimum iterasi yaitu:

Jumlah kelas = 20

Jumlah data latih = 100

Nilai *Learning Rate* atau α = 0.15

Nilai *decrement* α = 0.175

Nilai *error* minimum = $1 \times 10^{-10}$

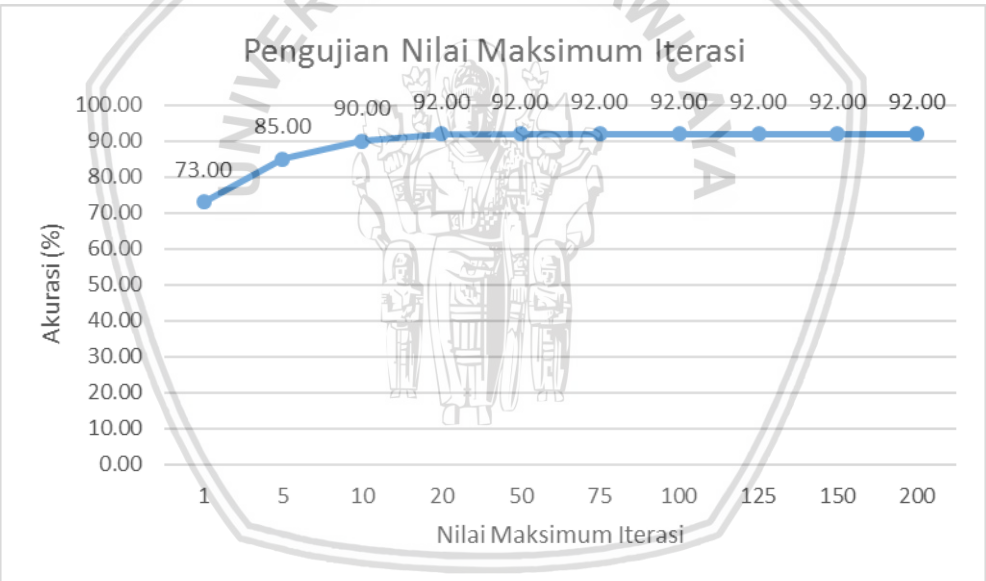
Untuk nilai *decrement* α didapatkan dari hasil pengujian nilai pengurangan *Learning Rate* (α) atau *decrement* α dan nilai α didapatkan dari pengujian nilai *Learning Rate* atau α awal. Nilai maksimum iterasi pada pengujian antara lain adalah 1, 5, 10, 20, 50, 75, 100, 125, 150, 200.

Pada Tabel 6.6 merupakan hasil pengujian nilai maksimum iterasi dengan setiap nilai maksimum iterasinya terdiri dari 10 percobaan. Tabel 6.6 merupakan data untuk membuat representasi data nilai akurasi dari pengujian nilai maksimum iterasi pada data latih berupa grafik yang ditunjukkan pada Gambar 6.6. Pada Gambar 6.6 dapat dilihat bahwa akurasi mengalami kenaikan dari nilai maksimum iterasi 1 sampai dengan 20. Nilai maksimum iterasi sebesar 20 iterasi sampai dengan 200 iterasi memiliki besar akurasi yang stabil dengan nilai akurasi sebersar 92%. Sehingga nilai maksimum akurasi yang digunakan adalah nilai maksimum iterasi awal dalam kondisi stabil yaitu pada 20 iterasi. Hasil yang pada pengujian nilai maksimum iterasi ini berkesimpulan bahwa semakin meningkat

nilai maksimum iterasi maka perubahan bobot fitur akan semakin kecil sehingga nilai akurasi cenderung stabil (Hamidi, et al., 2017).

**Tabel 6.6 Hasil Pengujian Nilai Maksimum Iterasi**

| Maksimum Iterasi | Percobaan (Akurasi %) | | | | | | | | | | Rata-rata Akurasi (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
| 1 | 73 | 73 | 73 | 73 | 73 | 73 | 73 | 73 | 73 | 73 | 73,00 |
| 5 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85 | 85,00 |
| 10 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90,00 |
| 20 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92,00 |
| 50 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92,00 |
| 75 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92,00 |
| 100 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92,00 |
| 125 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92,00 |
| 150 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92,00 |
| 200 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92 | 92,00 |




Gambar 6.6 Grafik Pengujian Nilai Maksimum Iterasi

## 6.5 Hasil dan Pembahasan Pengujian Data Uji dengan Nilai Parameter Optimal

Pengujian data uji menggunakan citra yang berbeda dengan citra latih. Nilai parameter dalam pengujian ini didapatkan dari pengujian-pengujian sebelumnya seperti pengujian nilai *Learning Rate* awal, pengujian nilai *decrement* α, pengujian nilai maksimum iterasi dan pengujian *error* minimum. Parameter-parameter pengujian beserta nilai optimalnya yaitu:

Jumlah kelas = 20

Nilai *Learning Rate* atau α = 0,15

Nilai *decrement* α = 0,175

Nilai Maksimum Iterasi = 20

Nilai *error* minimum = 1 x $10^{-10}$

Nilai *Threshold* yang digunakan dalam pengujian ini ada nilai *threshold* yang berasal dari nilai terbesar dari nilai jarak minimum. Nilai *threshold* setiap kelas yang digunakan dalam pengujian data uji ini ditunjukkan pada Tabel 6.7.

**Tabel 6.7 Nilai *Threshold***

| Kelas | *Threshold* | | Kelas | *Threshold* |
|---|---|---|---|---|
| 1 | 0,526076 | | 11 | 0,787148 |
| 2 | 0,747745 | | 12 | 0,647276 |
| 3 | 0,756082 | | 13 | 0,606513 |
| 4 | 0,641419 | | 14 | 0,560318 |
| 5 | 1,08431 | | 15 | 0,458215 |
| 6 | 0,761411 | | 16 | 0,507071 |
| 7 | 0,625381 | | 17 | 0,57291 |
| 8 | 0,57099 | | 18 | 0,795093 |
| 9 | 0,546792 | | 19 | 0,588162 |
| 10 | 0,480182 | | 20 | 0,493065 |

### 6.5.1 Data Uji dengan Kelas Sudah Dilatih

Data uji yang digunakan pada pengujian data uji ini menggunakan citra sebanyak 100 citra.

**Tabel 6.8 Hasil Pengujian Data Uji dengan Nilai Parameter Optimal**

| Percobaan Ke - | Akurasi |
|---|---|
| 1 | 51 |
| 2 | 51 |
| 3 | 51 |
| 4 | 51 |
| 5 | 51 |
| 6 | 51 |
| 7 | 51 |
| 8 | 51 |
| 9 | 51 |
| 10 | 51 |
| Minimum | 51 |
| Maksimum | 51 |
| Rata-rata | 51 |

Pada Tabel 6.7 ditunjukkan hasil dari pengujian data uji dengan menggunakan nilai parameter optimal. Pengujian data uji dilakukan melalui 10 kali percobaan. Dari hasil pengujian data uji diperoleh nilai akurasi stabil dari 10 kali pecobaan yaitu sebesar 51% dengan jumlah benar dalam pengenalan citra uji sebesar 51 citra dari 100 citra uji. Hasil akurasi dari pengujian ini rendah dikarenakan jarak citra dengan bobot kelas melebihi nilai *threshold* yang sudah ditentukan dalam pelatihan. Sehingga, banyak citra yang masuk kedalam data yang tidak dikenali.

### 6.5.2 Data Uji dengan Kelas Belum Dilatih

Jumlah data uji yang digunakan dalam pengujian data uji dengan kelas yang belum pernah dilatih ini adalah sebanyak 25 citra.

**Tabel 6.9 Hasil Pengujian Data Uji dengan Kelas yang Belum Dilatih**

| Percobaan Ke - | Akurasi (%) |
|---|---|
| 1 | 76 |
| 2 | 76 |
| 3 | 76 |
| 4 | 76 |
| 5 | 76 |
| Minimum | 76 |
| Maksimum | 76 |
| Rata-rata | 76 |

Dari pengujian data uji yang kelasnya belum pernah dilatih diperoleh akurasi sebesar 76%. Sebanyak 19 citra dinyatakan data tidak dikenali dari 25 citra uji. Dan sebanyak 6 citra dikenali dengan masuk kekelas yang sudah dikenali. Dari hasil pengujian ini dapat disimpulkan bahwa nilai *threshold* dengan menggunakan nilai maksimum dari jarak terpendek masih memiliki kekurangan yaitu nilai *threshold* kurang kecil dalam pengenalan tanda tangan pada citra yang datanya belum dikenali.

## 6.6 Hasil dan Pembahasan Pengujian *Threshold*

Pengujian nilai *threshold* untuk data uji baik yang kelasnya sudah dilatih maupun yang belum pernah dilatih adalah pengujian batas nilai jarak suatu data untuk masuk ke kelas klasifikasi dari data yang sudah dilatih. Hasil output dari pengujian ini berupa pemberitahuan bahwa data tidak dikenali. Sehingga data yang belum pernah dilatih tidak akan masuk klasifikasi ke kelas klasifikasi sistem pengenalan tanda tangan. Nilai *threshold* didapatkan pada proses pelatihan. Nilai *threshold* pertama diperoleh dari nilai maksimum dari jarak minimum seluruh citra yang sesuai dengan kelas target. Dan nilai *threshold* kedua diperoleh dari nilai maksimum kedua dari nilai jarak minimum kelas sistem yang sesuai dengan kelas target. Nilai *threshold* pertama ditunjukkan pada Tabel 6.10 dan nilai *threshold* kedua ditunjukkan pada Tabel 6.11.

**Tabel 6.10 Nilai *Threshold* Pertama**

| Kelas | *Threshold* | | Kelas | *Threshold* |
|---|---|---|---|---|
| 1 | 0,526076 | | 11 | 0,787148 |
| 2 | 0,747745 | | 12 | 0,647276 |
| 3 | 0,756082 | | 13 | 0,606513 |
| 4 | 0,641419 | | 14 | 0,560318 |
| 5 | 1,08431 | | 15 | 0,458215 |
| 6 | 0,761411 | | 16 | 0,507071 |
| 7 | 0,625381 | | 17 | 0,57291 |
| 8 | 0,57099 | | 18 | 0,795093 |
| 9 | 0,546792 | | 19 | 0,588162 |
| 10 | 0,480182 | | 20 | 0,493065 |

**Tabel 6.11 Nilai *Threshold* Kedua**

| Kelas | *Threshold* | | Kelas | *Threshold* |
|---|---|---|---|---|
| 1 | 0,516611 | | 11 | 0,479105 |
| 2 | 0,608821 | | 12 | 0,458096 |
| 3 | 0,753699 | | 13 | 0,485717 |
| 4 | 0,621192 | | 14 | 0,528446 |
| 5 | 0,401505 | | 15 | 0,399413 |
| 6 | 0,665985 | | 16 | 0,433302 |
| 7 | 0,530464 | | 17 | 0,507436 |
| 8 | 0,540166 | | 18 | 0,789869 |
| 9 | 0,407666 | | 19 | 0,58784 |
| 10 | 0,462462 | | 20 | 0,468675 |

Hasil Pengujian dari nilai *threshold* dapat dilihat pada Tabel 6.12.

**Tabel 6.12 Hasil Pengujian Nilai *Threshold***

| Citra Uji ke- | Kelas Target | Hasil Klasifikasi | | | Keterangan Hasil | | |
|---|---|---|---|---|---|---|---|
| | | non-*Threshold* | *Threshold*1 | *Threshold*2 | non-*Threshold* | *Threshold* 1 | *Threshold* 2 |
| 1 | 1 | 1 | - | - | TRUE | FALSE | FALSE |
| 2 | 1 | 1 | 1 | 1 | TRUE | TRUE | TRUE |
| 3 | 1 | 1 | 1 | 1 | TRUE | TRUE | TRUE |
| 4 | 1 | 1 | - | - | TRUE | FALSE | FALSE |
| 5 | 1 | 1 | 1 | 1 | TRUE | TRUE | TRUE |
| 6 | 2 | 2 | 2 | 2 | TRUE | TRUE | TRUE |
| 7 | 2 | 2 | 2 | - | TRUE | TRUE | FALSE |
| 8 | 2 | 2 | 2 | 2 | TRUE | TRUE | TRUE |
| 9 | 2 | 2 | - | - | TRUE | FALSE | FALSE |
| 10 | 2 | 2 | - | - | TRUE | FALSE | FALSE |
| 11 | 3 | 3 | 3 | 3 | TRUE | TRUE | TRUE |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 12 | 3 | 3 | - | - | TRUE | FALSE | FALSE |
| 13 | 3 | 3 | 3 | 3 | TRUE | TRUE | TRUE |
| 14 | 3 | 3 | - | - | TRUE | FALSE | FALSE |
| 15 | 3 | 3 | 3 | 3 | TRUE | TRUE | TRUE |
| 16 | 4 | 4 | 4 | 4 | TRUE | TRUE | TRUE |
| 17 | 4 | 4 | - | - | TRUE | FALSE | FALSE |
| 18 | 4 | 4 | - | - | TRUE | FALSE | FALSE |
| 19 | 4 | 4 | 4 | 4 | TRUE | TRUE | TRUE |
| 20 | 4 | 4 | - | - | TRUE | FALSE | FALSE |
| 21 | 5 | 5 | 5 | - | TRUE | TRUE | FALSE |
| 22 | 5 | 20 | - | - | FALSE | FALSE | FALSE |
| 23 | 5 | 5 | 5 | - | TRUE | TRUE | FALSE |
| 24 | 5 | 5 | 5 | - | TRUE | TRUE | FALSE |
| 25 | 5 | 5 | 5 | - | TRUE | TRUE | FALSE |
| 26 | 6 | 6 | 6 | 6 | TRUE | TRUE | TRUE |
| 27 | 6 | 6 | 6 | 6 | TRUE | TRUE | TRUE |
| 28 | 6 | 9 | - | - | FALSE | FALSE | FALSE |
| 29 | 6 | 6 | 6 | - | TRUE | TRUE | FALSE |
| 30 | 6 | 6 | 6 | 6 | TRUE | TRUE | TRUE |
| 31 | 7 | 7 | - | - | TRUE | FALSE | FALSE |
| 32 | 7 | 7 | - | - | TRUE | FALSE | FALSE |
| 33 | 7 | 7 | - | - | TRUE | FALSE | FALSE |
| 34 | 7 | 7 | - | - | TRUE | FALSE | FALSE |
| 35 | 7 | 5 | 5 | - | FALSE | FALSE | FALSE |
| 36 | 8 | 8 | 8 | 8 | TRUE | TRUE | TRUE |
| 37 | 8 | 8 | 8 | 8 | TRUE | TRUE | TRUE |
| 38 | 8 | 8 | - | - | TRUE | FALSE | FALSE |
| 39 | 8 | 8 | 8 | 8 | TRUE | TRUE | TRUE |
| 40 | 8 | 8 | 8 | 8 | TRUE | TRUE | TRUE |
| 41 | 9 | 9 | 9 | - | TRUE | TRUE | FALSE |
| 42 | 9 | 9 | 9 | - | TRUE | TRUE | FALSE |
| 43 | 9 | 9 | 9 | - | TRUE | TRUE | FALSE |
| 44 | 9 | 9 | - | - | TRUE | FALSE | FALSE |
| 45 | 9 | 9 | 9 | - | TRUE | TRUE | FALSE |
| 46 | 10 | 10 | 10 | 10 | TRUE | TRUE | TRUE |
| 47 | 10 | 10 | - | - | TRUE | FALSE | FALSE |
| 48 | 10 | 10 | - | - | TRUE | FALSE | FALSE |
| 49 | 10 | 10 | - | - | TRUE | FALSE | FALSE |
| 50 | 10 | 2 | 2 | 2 | FALSE | FALSE | FALSE |
| 51 | 11 | 11 | 11 | - | TRUE | TRUE | FALSE |
| 52 | 11 | 11 | 11 | - | TRUE | TRUE | FALSE |
| 53 | 11 | 11 | 11 | - | TRUE | TRUE | FALSE |
| 54 | 11 | 11 | 11 | - | TRUE | TRUE | FALSE |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 55 | 11 | 11 | 11 | - | TRUE | TRUE | FALSE |
| 56 | 12 | 12 | - | - | TRUE | FALSE | FALSE |
| 57 | 12 | 12 | - | - | TRUE | FALSE | FALSE |
| 58 | 12 | 12 | - | - | TRUE | FALSE | FALSE |
| 59 | 12 | 12 | - | - | TRUE | FALSE | FALSE |
| 60 | 12 | 12 | - | - | TRUE | FALSE | FALSE |
| 61 | 13 | 13 | 13 | 13 | TRUE | TRUE | TRUE |
| 62 | 13 | 13 | - | - | TRUE | FALSE | FALSE |
| 63 | 13 | 13 | 13 | - | TRUE | TRUE | FALSE |
| 64 | 13 | 13 | 13 | 13 | TRUE | TRUE | TRUE |
| 65 | 13 | 15 | 15 | 15 | FALSE | FALSE | FALSE |
| 66 | 14 | 14 | - | - | TRUE | FALSE | FALSE |
| 67 | 14 | 14 | 14 | 14 | TRUE | TRUE | TRUE |
| 68 | 14 | 14 | 14 | 14 | TRUE | TRUE | TRUE |
| 69 | 14 | 14 | - | - | TRUE | FALSE | FALSE |
| 70 | 14 | 14 | - | - | TRUE | FALSE | FALSE |
| 71 | 15 | 13 | 13 | 13 | FALSE | FALSE | FALSE |
| 72 | 15 | 15 | - | - | TRUE | FALSE | FALSE |
| 73 | 15 | 15 | 15 | - | TRUE | TRUE | FALSE |
| 74 | 15 | 15 | - | - | TRUE | FALSE | FALSE |
| 75 | 15 | 15 | - | - | TRUE | FALSE | FALSE |
| 76 | 16 | 16 | - | - | TRUE | FALSE | FALSE |
| 77 | 16 | 16 | 16 | - | TRUE | TRUE | FALSE |
| 78 | 16 | 1 | - | - | FALSE | FALSE | FALSE |
| 79 | 16 | 16 | 16 | - | TRUE | TRUE | FALSE |
| 80 | 16 | 1 | - | - | FALSE | FALSE | FALSE |
| 81 | 17 | 17 | - | - | TRUE | FALSE | FALSE |
| 82 | 17 | 17 | - | - | TRUE | FALSE | FALSE |
| 83 | 17 | 2 | 2 | - | FALSE | FALSE | FALSE |
| 84 | 17 | 17 | - | - | TRUE | FALSE | FALSE |
| 85 | 17 | 2 | 2 | - | FALSE | FALSE | FALSE |
| 86 | 18 | 18 | 18 | 18 | TRUE | TRUE | TRUE |
| 87 | 18 | 18 | 18 | 18 | TRUE | TRUE | TRUE |
| 88 | 18 | 18 | 18 | 18 | TRUE | TRUE | TRUE |
| 89 | 18 | 18 | 18 | 18 | TRUE | TRUE | TRUE |
| 90 | 18 | 18 | 18 | 18 | TRUE | TRUE | TRUE |
| 91 | 19 | 19 | - | - | TRUE | FALSE | FALSE |
| 92 | 19 | 19 | - | - | TRUE | FALSE | FALSE |
| 93 | 19 | 19 | 19 | 19 | TRUE | TRUE | TRUE |
| 94 | 19 | 19 | 19 | 19 | TRUE | TRUE | TRUE |
| 95 | 19 | 19 | 19 | 19 | TRUE | TRUE | TRUE |
| 96 | 20 | 20 | 20 | 20 | TRUE | TRUE | TRUE |
| 97 | 20 | 20 | - | - | TRUE | FALSE | FALSE |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 98 | 20 | 20 | 20 | 20 | TRUE | TRUE | TRUE |
| 99 | 20 | 20 | - | - | TRUE | FALSE | FALSE |
| 100 | 20 | 20 | - | - | TRUE | FALSE | FALSE |
| 101 | - | 4 | - | - | FALSE | TRUE | TRUE |
| 102 | - | 4 | - | - | FALSE | TRUE | TRUE |
| 103 | - | 4 | - | - | FALSE | TRUE | TRUE |
| 104 | - | 4 | 4 | - | FALSE | FALSE | TRUE |
| 105 | - | 3 | - | - | FALSE | TRUE | TRUE |
| 106 | - | 1 | 2 | - | FALSE | FALSE | TRUE |
| 107 | - | 1 | - | - | FALSE | TRUE | TRUE |
| 108 | - | 1 | 2 | - | FALSE | FALSE | TRUE |
| 109 | - | 1 | - | - | FALSE | TRUE | TRUE |
| 110 | - | 11 | - | - | FALSE | TRUE | TRUE |
| 111 | - | 1 | - | - | FALSE | TRUE | TRUE |
| 112 | - | 1 | - | - | FALSE | TRUE | TRUE |
| 113 | - | 11 | - | - | FALSE | TRUE | TRUE |
| 114 | - | 11 | - | - | FALSE | TRUE | TRUE |
| 115 | - | 10 | - | - | FALSE | TRUE | TRUE |
| 116 | - | 3 | 4 | - | FALSE | FALSE | TRUE |
| 117 | - | 9 | - | - | FALSE | TRUE | TRUE |
| 118 | - | 9 | - | - | FALSE | TRUE | TRUE |
| 119 | - | 8 | - | - | FALSE | TRUE | TRUE |
| 120 | - | 7 | - | - | FALSE | TRUE | TRUE |
| 121 | - | 9 | 9 | - | FALSE | FALSE | TRUE |
| 122 | - | 9 | 9 | - | FALSE | FALSE | TRUE |
| 123 | - | 9 | - | - | FALSE | TRUE | TRUE |
| 124 | - | 9 | - | - | FALSE | TRUE | TRUE |
| 125 | - | 9 | - | - | FALSE | TRUE | TRUE |
| Akurasi Data Uji (Kelas Sudah dilatih) | | | | | 89.0 | 51.0 | 32.0 |
| Akurasi Data Uji (Kelas Belum dilatih) | | | | | 0.0 | 76.0 | 100.0 |
| Akurasi Keseluruhan Data Uji (%) | | | | | 71.2 | 56.0 | 45.6 |

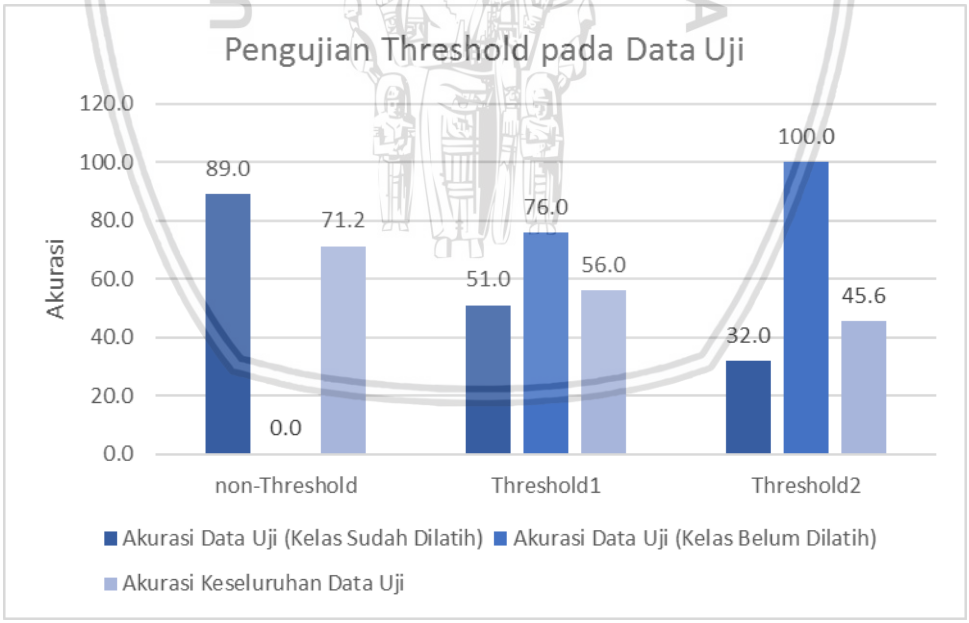
Dari hasil pengujian nilai *threshold* pada Tabel 6.12, nilai akurasi terbesar adalah 71,2% pada pengujian tanpa menggunakan *threshold*. Pada pengujian tanpa menggunakan *threshold* diketahui bahwa data uji dengan kelas yang sudah dilatih memperoleh 89 citra berhasil dikenali dan 25 citra dari data tidak dikenali masuk kedalam kelas-kelas yang sudah dilatih. Sehingga, akurasi pada pengujian tanpa menggunakan *threshold* yaitu 89% pada data uji dengan kelas sudah dilatih dan 0% pada kelas data yang tidak dikenali. Sehingga dari pengujian tanpa *threshold* ini disimpulkan bahwa sistem belum dapat mengenali data yang seharusnya tidak dikenali atau tidak masuk kelas manapun.

Untuk pengujian nilai *threshold* menggunakan nilai *threshold* pertama, nilai akurasi yang diperoleh adalah sebesar 56% dari 125 data uji. Pada data uji dengan kelas yang sudah dilatih memperoleh akurasi sebesar 51% dari 100 citra.

Sedangkan pada data uji dengan kelas yang belum dilatihkan memperoleh jumlah benar sebanyak 19 citra dari 25 citra. Sehingga dapat disimpulkan bahwa, dengan nilai *threshold* dari nilai terbesar jarak minimum, sistem masih dapat memasukkan data yang seharusnya tidak dikenali kedalam kelas yang sudah dilatih.

Pengujian nilai *threshold* yang terakhir adalah nilai *threshold* dengan menggunakan nilai terbesar kedua dari jarak minimum. Nilai akurasi yang diperoleh adalah sebesar 45,6% dari 125 citra. Jumlah benar pada data uji dengan kelas yang sudah dilatih sebesar 32 citra benar dari 100 citra. Dan untuk citra dengan data uji kelas yang tidak dikenali, diperoleh jumlah benar 25 citra dari 25 citra atau nilai akurasinya mencapai 100%. Dari pengujian dengan nilai *threshold* kedua, sistem berhasil tidak mengenali data yang seharusnya tidak masuk kekelas latih.

Diagram hasil pengujian *threshold* pada setiap data uji ditunjukkan pada Gambar 6.7. Pada Gambar 6.7, hasil pengujian data uji pada kelas yang dilatih adalah semakin menurun nilai akurasinya jika nilai *threshold* semakin kecil. Hasil pengujian data uji pada kelas yang belum dilatih adalah akurasi terhadap data yang tidak dikenali akan semakin meningkan jika nilai *threshold* semakin rendah. Untuk pengujian menggunakan keseluruhan data uji baik kelas yang sudah dilatih maupun yang belum dilatih menghasilkan akurasi yang terus menurun karena data uji dengan kelas yang sudah dilatih lebih banyak jumlahnya daripada data uji dengan kelas yang belum dilatih.




Gambar 6.7 Grafik Pengujian Nilai *Threshold*

## *6.7* Hasil dan Pembahasan Pengujian Implementasi Fitur Jumlah *Keypoint Descriptor*

Pengujian implementasi fitur jumlah *keypoint descriptor* adalah menguji perbandingan akurasi pengenalan tanda tangan antara sistem dengan tambahan fitur jumlah *keypoint descriptor* dan tanpa menggunakan fitur jumlah *keypoint descriptor*. Langkah awal pengujian dimulai dari proses pelatihan sampai dengan proses pengujian. Pengujian dilakukan pada 200 dataset dengan pembagian data latih sebanyak 100 data dan data uji sebanyak 100 data. Nilai *threshold* dalam pengujian implementasi fitur jumlah *keypoint* tidak digunakan. Parameter optimal dalam pengujian pengenalan tanda tangan dengan tambahan fitur jumlah *keypoint descriptor* antara lain:

Nilai *learning rate* awal = 0,15

Nilai pengurang *learning rate* = 0,175

Maksimum iterasi = 20

Nilai minimum *error* = $1 \times 10^{-10}$

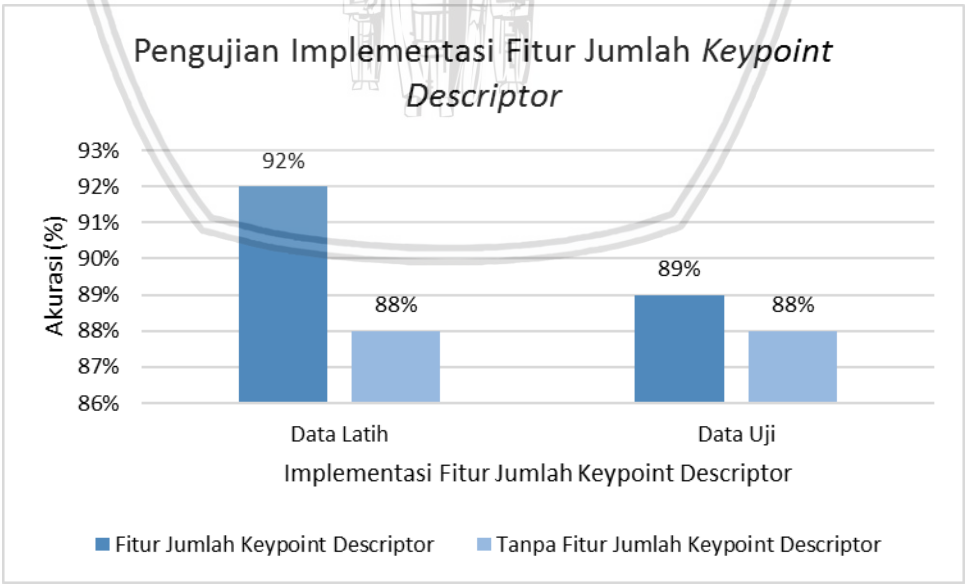
Parameter optimal pengujian sistem pengenalan tanda tangan tanpa menggunakan fitur jumlah *keypoint descriptor* antara lain:

Nilai *learning rate* awal = 0,1

Nilai pengurang *learning rate* = 0,125

Maksimum iterasi = 20

Nilai minimum *error* = $1 \times 10^{-10}$




Gambar 6.8 Grafik Pengujian Implementasi Fitur Jumlah *Keypoint Descriptor*

Pada Gambar 6.8 menunjukkan hasil akurasi dari pengujian implementasi fitur jumlah *keypoint descriptor* pada data uji dan data latih. Hasil akurasi tertinggi dari data latih diperoleh dari penggunaan fitur jumlah *keypoint descriptor* yaitu sebesar 92%, sedangkan akurasi untuk pengenalan tanda tangan tanpa menggunakan fitur jumlah *keypoint descriptor* adalah 88%. Pada pengujian menggunakan data uji diperoleh akurasi tertinggi sebesar 89% dari implementasi fitur jumlah *keypoint descriptor* pada pengenalan tanda tangan. Akurasi terendah dari pengujian menggunakan data uji adalah pada pengenalan tanda tangan tanpa menggunakan fitur jumlah *keypoint descriptor* yaitu sebesar 88%. Kesimpulan yang diperoleh dari pengujian ini adalah penggunaan fitur jumlah *keypoint descriptor* pada sistem pengenalan tanda tangan menggunakan algoritme *Learning Vector Quantization* (LVQ) dapat meningkatkan akurasi dari proses pengenalan tanda tangan. Meningkatnya akurasi dikarenakan *keypoint descriptor* merupakan fitur yang unik karena merupakan deskripsi dari sebuah objek dan fitur yang baik untuk pengenalan suatu objek (Hilman, 2015).

# BAB 7 PENUTUP

Bab penutup adalah bab yang menerangkan tentang uraian kesimpulan dan saran yang didapatkan dari hasil pembahasan penelitian mengenai ektraksi fitur menggunakan metode *Scale Invariant Features Transform* (SIFT) pada pengenalan tanda tangan menggunakan Algoritme *Learning Vector Quantization* (LVQ).

## 7.1 Kesimpulan

Kesimpulan pada penelitian skripsi ini antara lain :

1. Dalam penelitian ini proses pengenalan tanda tangan menggunakan algoritme *Learning Vector Quantization* (LVQ). Fitur-fitur yang digunakan antara lain nilai maksimum piksel horizontal dan vertikal, nilai pusat massa objek tanda tangan pada citra, nilai normalisasi luas objek tanda tangan, nilai rasio antara nilai maksimum horizontal piksel dengan vertikal piksel, nilai massa pusat pada tiga area, jumlah area objek tanda tangan yang dilewati garis atas dan bawah massa pusat dari tiga area pembagian citra tanda tangan, dan sepuluh fitur transisi dari nilai piksel 0 ke 1 atau 1 ke 0. Proses pelatihan dilakukan dengan menggunakan parameter-parameter yaitu bobot fitur dengan nilai random, nilai optimal *Learning Rate* awal, nilai optimal pengurang nilai *Learning Rate*, jumlah iterasi optimal. Nilai optimal beberapa parameter didapatkan dari hasil pengujian secara bertahap dari 100 citra latih dari 10 kelas. Hasil dari proses pelatihan adalah bobot fitur setiap kelas yang akan digunakan dalam proses pengujian pengenalan tanda tangan pada 100 citra uji dari 10 kelas dan 25 citra uji dengan kelas yang belum pernah dilatih.
2. Rata-rata akurasi citra yang diperoleh dalam pengujian adalah 92% untuk data latih, 71,2% untuk data uji tanpa menggunakan *threshold*, 56% dari data uji dengan menggunakan *threshold* 1 dan 45,6% untuk data uji dengan menggunakan *threshold* 2. Dengan parameter awal pelatihan yaitu nilai *Learning Rate* atau α sebesar 0.15, nilai pengurang α sebersar 0.175, banyak iterasi awal atau nilai maksimum iterasi pelatihan sebesar 20 kali iterasi dan nilai *error* minimum sebesar $1 \times 10^{-10}$. Parameter pelatihan didapatkan dari hasil pengujian secara berurutan.
3. Implementasi penggunaan fitur jumlah *keypoint descriptor* yaitu dengan cara menggunakan metode *Scale Invariant Features Transform* (SIFT). Kode program untuk memperoleh nilai jumlah *keypoint descriptor* dengan menggunakan *library* di OpenCV 3 yaitu detectAndCompute yang menghasilkan beberapa *keypoint descriptor*. Fitur jumlah *keypoint descriptor* selanjutnya diproses bersamaan dengan fitur fraktal global untuk pengenalan tanda tangan menggunakan algoritme *Learning Vector Quantization* (LVQ).
4. Implementasi ektraksi fitur jumlah *keypoint descriptor* pada pengenalan tanda tangan menggunakan algoritme *Learning Vector Quantization* lebih

baik daripada tanpa menggunakan ekstraksi fitur jumlah *keypoint descriptor*. Akurasi yang diperoleh dari penambahan ektraksi fitur jumlah *keypoint descriptor* pada pengenalan tanda tangan menggunakan algoritme *Learning Vector Quantization* adalah sebesar 92% pada data latih dan 89% pada data uji. Pengenalan tanda tangan tanpa menggunakan fitur jumlah *keypoint descriptor* diperoleh akurasi sebesar 88% pada data latih dan 88% pada data uji.

## 7.2 Saran

Saran yang diperoleh dari hasil penelitian skripsi ini yang dapat digunakan pada penelitian yang lebih lanjut antara lain :

1. Proses pengambilan fitur dapat menggunakan cara pembagian area atau grid. Pembagian area ini digunakan untuk lebih mengambil nilai fitur yang unik yang berada didalam objek.
2. Proses normalisasi fitur dengan menggunakan beberapa metode sehingga memungkikan untuk mengoptimalkan tingkat akurasi dari pengenalan tanda tangan.
3. Metode dalam pencarian *keypoint* dapat menggunakan metode-metode lain seperti SURF.
4. Untuk menghasilkan bobot fitur yang baik dapat dikembangkan lebih lanjut dengan cara optimasi bobot fitur menggunakan algoritme-Algoritme optimasi yang nantinya dapat menambah akurasi dari penelitian ini.

# DAFTAR PUSTAKA


Agustina, S. E. & Mukhlash, I., 2012. Implementasi Metode Scale *Invariant* Feature Transform (SIFT) dan Metode Continuoly Adaptive Mean-Shift (Camshift) Pada Penjejakan Objek Bergerak. *Jurnal Sains dan Seni,* 1(1), pp. 1-6.

Arniantya, R., Setiawan, B. D. & Adikara, P. P., 2018. Optimasi Vector Bobot Pada *Learning Vector Quantization* Menggunakan Algoritme genetika Untuk Identifikasi Jenis Attention Deficit Hyperactivity Disorder Pada Anak. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 2(2), pp. 679-687.

Bashyal, S. & Venayagamoorthy, G. K., 2008. Recognition of Facial Expression using Gabor Wavelets and *Learning Vector Quantization* . *Engineering Application of Artificial Intelligence,* Volume 21, pp. 1056-1064.

Choudhary, N. Y., Patil, R., Bhadade, U. & Chaudhari, B. M., 2013. Signature Recognition & Verification System Using Back Propagation Neural Network. *International Journal of IT, Engineering and Applied Sciences Research (IJIEASR),* January, 2(1), pp. 1-8.

Fisher, R. B. et al., 2004. *Dictionary of Computer Vision and Image Processing.* New Jersey: John Wiley & Sons, Inc..

Ginting, E., Zarlis, M. & Situmorang, Z., 2014. Kombinasi Algoritme Jaringan Syaraf Tiruan *Learning Vector Quantization* (LVQ) Dan Self Organizing Kohonen Pada Kecepatan Pengenalan Pola Tanda Tangan. *Techsi,* 4(1), pp. 97-110.

Gurney, K., 1997. *An Introduction to Neural Networks.* London: UCL Press.

Hamidi, R., Furqon, M. T. & Rahayudi, B., 2017. Implementasi *Learning Vector Quantization* (LVQ) untuk Klasifikasi Kualitas Air Sungai. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 1(12), pp. 1758-1763.

Haralick, R. M. & Sharpio, L. G., 1991. Glossary of Computer Vision Term. *Pattern Recognition,* 24(1), pp. 69-93.

Hidayatno, A., Isnanto, R. R. & Buana, D. K. W., 2008. Identifikasi Tanda-Tangan Menggunakan Jaringan Syaraf Tiruan Perambatan Balik (*Backpropagation*). *Jurnal Teknologi,* 1(2), pp. 100-106.

Hilman, F. P., 2015. Perbandingan Metode SURF dan SIFT dalam Sistem Identifikasi Tanda Tangan. *e-Proceeding of Engineering,* 2(2), p. 2467.

Kadir, A. & Susanto, A., 2013. *Teori dan Aplikasi Pengolahan Citra.* 1 ed. Yogyakarta: ANDI.

Kohonen, T., Hynninen, J., Laaksonen, J. K. J. & Torkkola, K., 1996. *LVQ_PAK: The Learning Vector Quantization Program Package.* Finland: SF-02150 Espoo.

Kumar, M., 2012. Signature Verification Using Neural Network. *International Journal on Computer Science and Engineering (IJCSE),* 4(09), pp. 1498-1504.

Kumar, P., Singh, S., Garg, A. & Prabbat, N., 2013. Hand Written Signature Recognition & Verification using Neural Network. *International Journal of Advance Research in Computer Science and Software Engineering,* 3(3), p. 558.

Lowe, D. G., 2004. Distinctive Image Features from Scale-*Invariant Keypoint*s. *International Journal of Computer,* 60(2), pp. 91-110.

Prabowo, A., Sarwoko, E. A. & Riyanto, D. E., 2006. Perbandingan Antara Metode Kohonen Neural Network dengan *Learning Vector Quantization* Pada Pengenalan Pola Tanda Tangan. *Jurnal Sains & Matematika (JSM),* 14(4), p. 147.

Qur'ani, D. Y. & Rosmalinda, S., 2010. Jaringan Syaraf Tiruan *Learning Vector Quantization* untuk Aplikasi Pengenalan Tanda Tangan. *Seminar Nasional Aplikasi teknologi 2010 (SNATI 2010),* p. 8.

Rahmi, A., Wijayaningrum, V. N., Mahmudy, W. F. & Parewe, A. M. A. K., 2016. Offline Signature Recognition Using *Backpropagation* Neural Network. *Indonesian Journal of Electrical Engineering and Computer Science,* 4(3), pp. 678 - 683.

Rojas, R., 1996. *Neural Networks.* Berlin: Springer-Verlag.

Sudarma, M. & Sutramiani, N. P., 2014. The *Thinning* Zhang-Suen Application Method in the Image of Balinese Scripts on the Papyrus. *International Journal of Computer Applications,* 91(1).

Sutramiani, N. P., Putra, I. K. G. D. & Sudarma, M., 2015. Local Adptive *Threshold*ing Pada Preprocessing Citra Lontar Aksara Bali. *Jurnal Teknologi Elektro,* 14(1), p. 29.

Wibowo, A. M., 2001. *Naskah akademik rancangan undang-undang tentang tanda tangan elektronik dan transaksi elektronik: laporan penelitian tahap pertama versi 1.04.* s.l.:Republik Indonesia.