**

**YUSUF AZIS HENNY TRI YUDHANTORO**

**NRP 5112100086**

**Dosen Pembimbing I**

**Dr.Eng. Chastine Fatichah, S.Kom, M.Kom**

**Dosen Pembimbing II**

**Dr.Eng. Nanik Suciati, S.Kom, M.Kom**

**JURUSAN TEKNIK INFORMATIKA**

**Fakultas Teknologi Informasi**

**Institut Teknologi Sepuluh Nopember**

**Surabaya 2017**

**IMPLEMENTASI FITUR SIFT DAN FISHER VECTOR ENCODING UNTUK APLIKASI PENGENALAN WAJAH**

TUGAS AKHIR - KI141502

***[Halaman ini sengaja dikosongkan]***

**

**YUSUF AZIS HENNY TRI YUDHANTORO**

**NRP 5112100086**

**Dosen Pembimbing I**

**Dr.Eng. Chastine Fatichah, S.Kom, M.Kom**

**Dosen Pembimbing II**

**Dr.Eng. Nanik Suciati, S.Kom, M.Kom**

**JURUSAN TEKNIK INFORMATIKA**

**Fakultas Teknologi Informasi**

**Institut Teknologi Sepuluh Nopember**

**Surabaya 2017**

**IMPLEMENTASI FITUR SIFT DAN FISHER VECTOR ENCODING UNTUK APLIKASI PENGENALAN WAJAH**

TUGAS AKHIR - KI141502

***[Halaman ini sengaja dikosongkan]***

**

**YUSUF AZIS HENNY TRI YUDHANTORO**

**NRP 5112100086**

**Supervisor I**

**Dr.Eng. Chastine Fatichah, S.Kom, M.Kom.**

**Supervisor II**

**Dr.Eng. Nanik Suciati, S.Kom, M.Kom.**

**DEPARTMENT OF INFORMATICS**

**FACULTY OF INFORMATION TECHNOLOGY**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**SURABAYA 2017**er

Su

**SIFT FEATURE AND FISHER VECTOR ENCODING IMPLEMENTATION FOR FACE RECOGNITION APPLICATION**

**UNDERGRADUATE THESES – KI141502**

***[Halaman ini sengaja dikosongkan]***

# 

# LEMBAR PENGESAHAN

**IMPLEMENTASI FITUR SIFT DAN FISHER VECTOR ENCODING UNTUK APLIKASI PENGENALAN WAJAH**

**TUGAS AKHIR**

Diajukan Guna Memenuhi Salah Satu Syarat   
Memperoleh Gelar Sarjana Komputer pada

Bidang Studi Komputasi Cerdas dan Visualisasi   
Program Studi S-1 Jurusan Teknik Informatika  
Fakultas Teknologi Informasi  
Institut Teknologi Sepuluh Nopember

Oleh :  
**YUSUF AZIS HENNY TRI YUDHANTORO  
NRP : 5112 100 086**

Disetujui oleh Dosen Pembimbing Tugas Akhir :

|  |  |
| --- | --- |
| Dr. Chastine Fatichah, S.Kom, M.Kom. NIP: 19751220 200112 2 002 | ................................ (pembimbing 1) |
|  |  |
| Dr.Eng. Nanik Suciati, S.Kom, M.Kom. NIP: 19710428 199412 2 001 | ................................ (pembimbing 2) |
|  |  |

**Surabaya  
januari 2017**

***[Halaman ini sengaja dikosongkan]***

**IMPLEMENTASI FITUR SIFT DAN FISHER VECTOR ENCODING UNTUK APLIKASI PENGENALAN WAJAH**

|  |  |  |
| --- | --- | --- |
| **Nama Mahasiswa** | **:** | **YUSUF AZIS HENNY TRI Y.** |
| **NRP** | **:** | **5112100086** |
| **Jurusan** | **:** | **Teknik Informatika FTIF-ITS** |
| **Dosen Pembimbing 1** | **:** | **Dr. Chastine Fatichah, S.Kom, M.Kom.** |
| **Dosen Pembimbing 2** | **:** | **Dr.Eng. Nanik Suciati, S.Kom, M.Kom.** |

# ABSTRAK

*Teknik pengenalan individu berbasis fitur biometrik wajah menjadi salah satu teknik yang paling sering digunakan. Hal ini dikarenakan penggunaan wajah untuk pengenalan memiliki beberapa kelebihan, antara lain: pengenalan wajah hanya membutuhkan peralatan kamera yang relatif ekonomis dan algoritma yang baik mampu mengidentifikasi wajah.*

*Dalam Tugas Akhir ini diimplementasikan perangkat lunak pengenalan wajah menggunakan Scalable Invariant Feature Transform (SIFT) mode dense, dan fisher vector encoding, serta mahalanobis metric learning. Hasil dari proses metric learning adalah threshold kesamaan wajah dan bobot jarak fisher vector pasangan gambar wajah yang berguna dalam proses klasifikasi.*

*Dataset yang digunakan dalam proses uji coba berisi empat ratus pasangan gambar kelas positif dan negatif dari Labeled Faces in the Wild (LFW). Akurasi terbaik sebesar 53%.*

***Kata kunci: Biometric, Pengenalan Wajah, SIFT, Fisher Vector, Mahalanobis Metric Learning.***

***[Halaman ini sengaja dikosongkan]***

**SIFT FEATURE AND FISHER VECTOR ENCODING IMPLEMENTATION FOR FACE RECOGNITION APPLICATION**

|  |  |  |
| --- | --- | --- |
| **Student’s Name** | **:** | **YUSUF AZIS HENNY TRI Y.** |
| **Student’s ID** | **:** | **5112100086** |
| **Department** | **:** | **Teknik Informatika FTIF-ITS** |
| **First Advisor** | **:** | **Dr. Chastine Fatichah, S.Kom, M.Kom.** |
| **Second Advisor** | **:** | **Dr.Eng. Nanik Suciati, S.Kom, M.Kom.** |

# ABSTRACT

*Individual recognition techniques based on facial biometric features is one of the most frequently used technique. due to the usage of facial recognition has several advantages such as: facial recognition process only need relatively economic camera equipment and a good algorithm to be able to identify a face.*

*In this undergraduate theses, facial recognition software using dense mode of Scalable Invariant Feature Transform (SIFT) and fisher vector encoding to extract features. Weight of pair-face-image’s fisher vector and distance threshold to classify images learned using mahalanobis metric learning.*

*The dataset used in the testing process contains four hundred pairs of positive and negative class picture from Labeled Faces in the Wild (LFW). The best accuration is 53%.*

**Kata kunci: Biometric, Face Recognition, SIFT, Fisher Vector, Mahalanobis Metric Learning.**

***[Halaman ini sengaja dikosongkan]***

# KATA PENGANTAR



Alhamdulillahirabbil’alamin, segala puji dan syukur bagi Allah SWT, yang telah melimpahkan rahmat, dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul

**IMPLEMENTASI FITUR SIFT DAN FISHER VECTOR ENCODING UNTUK APLIKASI PENGENALAN WAJAH**

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang berharga bagi penulis. Dengan pengerjaan Tugas Akhir, penulis dapat memperdalam, meningkatkan, serta menerapkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS.

Terselesaikannya Tugas Akhir ini tidak lepas dari bantuan dan dukungan dari berbagai pihak. Dan dalam kesempatan ini penulis mengucapkan rasa syukur dan terima kasih kepada:

1. Allah SWT, karena atas izin-Nya lah penulis dapat menyelesaikan Tugas Akhir dengan baik.
2. Kedua orang tua, Dra. Hj. Nur Faizah, M.Pd. serta H. Kholief Arifianto, S.E. dan kedua saudara drg. Dianiza Afika Ningtyas, serta Robby Dwi Indra Kusuma, S.AB. terima kasih atas doa dan bantuan moral dan material selama penulis belajar di Teknik Informatika ITS.
3. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom., selaku ketua jurusan Teknik Informatika ITS
4. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku Koordinator Tugas Akhir di Teknik Informatika ITS.
5. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku koordinator TA.
6. Ibu Dr.Eng. Chastine Fatichah, S.Kom, M.Kom selaku pembimbing I Tugas Akhir yang telah memberikan banyak waktu untuk berdiskusi dan memberi semangat dan motivasi kepada penulis untuk menyelesaikan Tugas Akhir.
7. Ibu Dr.Eng. Nanik Suciati, S.Kom, M.Kom selaku pembimbing II Tugas Akhir yang telah memberikan banyak yang telah memberikan bimbingan dan dukungan selama penulis menyelesaikan Tugas Akhir.
8. Bapak dan Ibu Dosen di Jurusan Teknik Informatika yang telah memberikan ilmu selama penulis kuliah di Teknik Informatika
9. Seluruh Staf dan karyawan Teknik Informatika yang telah memberikan bantuan selama penulis kuliah di Teknik Informatika.
10. Rekan-rekan di laboratorium Komputasi Cerdas dan Visi Jono, Addien, Mustofa, Ghozie, Andre, Dedi, Ihsan, Haqiqi, Ery, dll. yang telah lama saling mendukung dan menyemangati selama pengerjaan Tugas Akhir.
11. Rekan-rekan di laboratorium Dasar Terapan Komputer Ipul, Ichang, Andre, Ode, Ardhana, Mas Hendro,dll. Yang telah mendukung pengerjaan Tugas Akhir di semester kedua.
12. Seluruh rekan-rekan TC 2012 yang saya banggakan.
13. Rekan-rekan GSM Bulan, Burhan, Bagus, Reva, Dicho, Yusron, Fanis, Arif, terimakasih banyak atas perjalanan kemarin, hari ini, dan esok nanti.

Penulis memohon maaf apabila terdapat kekurangan dalam penulisan Tugas Akhir ini. Kritik dan saran penulis harapkan untuk perbaikan dan pembelajaran di kemudian hari. Semoga Tugas Akhir ini dapat memberikan Manfaat yang sebesar besarnya.

Surabaya, Januari 2017

Yusuf Azis Henny Tri Yudhantoro

# DAFTAR ISI

[LEMBAR PENGESAHAN vii](#_Toc473036202)

[ABSTRAK ix](#_Toc473036203)

[ABSTRACT xi](#_Toc473036204)

[KATA PENGANTAR xiii](#_Toc473036205)

[DAFTAR ISI xv](#_Toc473036206)

[DAFTAR GAMBAR xix](#_Toc473036207)

[DAFTAR TABEL xxi](#_Toc473036208)

[DAFTAR KODE SUMBER xxiii](#_Toc473036209)

[BAB I PENDAHULUAN 1](#_Toc473036210)

[1.1. Latar Belakang 1](#_Toc473036211)

[1.2. Rumusan Permasalahan 2](#_Toc473036212)

[1.3. Batasan Masalah 2](#_Toc473036213)

[1.4. Tujuan 2](#_Toc473036214)

[1.5. Manfaat 3](#_Toc473036215)

[1.6. Metodologi 3](#_Toc473036216)

[1.7. Sistematika Penulisan Laporan Tugas Akhir 4](#_Toc473036217)

[BAB II DASAR TEORI 7](#_Toc473036218)

[2.1. Viola-Jones Face Detection 7](#_Toc473036219)

[2.1.1. Haar Feature Detection 7](#_Toc473036220)

[2.1.2. Pembuatan Integral Image 8](#_Toc473036221)

[2.1.3. Adaboost Training 9](#_Toc473036222)

[2.1.4. Cascading Classifier 9](#_Toc473036223)

[2.2. dense Scalable Invariant Feature Transform (d-SIFT) 10](#_Toc473036224)

[2.2.1. Gaussian Blur 10](#_Toc473036225)

[2.2.2. Perhitungan Orientasi Vektor pada Piksel Citra 11](#_Toc473036226)

[2.2.3. Perhitungan Magnitude Vektor pada Piksel Citra 12](#_Toc473036227)

[2.3. Singular Value Decomposition (SVD) 12](#_Toc473036228)

[2.4. PCA-Whitening Transformation 13](#_Toc473036229)

[2.5. Gaussian Mixture Model (GMM) 14](#_Toc473036230)

[2.5.1. Penentuan Jumlah K Gaussian 15](#_Toc473036231)

[2.5.2. Inisialisasi Means Serta Covariance 15](#_Toc473036232)

[2.5.3. Kalkulasi Karakteristik Sebaran Data 16](#_Toc473036233)

[2.5.4. Kalkulasi Probabilitas Data Anggota Klaster 16](#_Toc473036234)

[2.5.5. Kalkulasi Bobot Masing-masing Data 17](#_Toc473036235)

[2.5.6. Kalkulasi Means Baru 17](#_Toc473036236)

[2.5.7. Kalkulasi Covariance Baru 17](#_Toc473036237)

[2.5.8. Kalkulasi Banyaknya Anggota Klaster 18](#_Toc473036238)

[2.5.9. Repetisi Proses 18](#_Toc473036239)

[2.6. Fisher Vector Encoding 18](#_Toc473036240)

[2.7. Mahalanobis Metric Learning 19](#_Toc473036241)

[2.8. Euclidean Distance 20](#_Toc473036242)

[2.9. K-Fold Cross Validation 21](#_Toc473036243)

[2.10. Stochastic Gradient Descent 21](#_Toc473036244)

[BAB III PERANCANGAN PERANGKAT LUNAK 23](#_Toc473036245)

[3.1. Data 23](#_Toc473036246)

[3.1.1. Data Masukan 24](#_Toc473036247)

[3.1.2. Data Pembelajaran 24](#_Toc473036248)

[3.1.3. Data Keluaran 24](#_Toc473036249)

[3.2. Desain Sistem Secara Umum 24](#_Toc473036250)

[3.3. Pre-processing 25](#_Toc473036251)

[3.3.1. Deteksi Wajah dan Pemotongan Gambar 28](#_Toc473036252)

[3.3.2. Resize Gambar 28](#_Toc473036253)

[3.4. Ekstraksi Fitur dense SIFT 28](#_Toc473036254)

[3.4.1. Bluring dan Penentuan Letak Keypoint 29](#_Toc473036255)

[3.4.2. Perhitungan Orientasi dan Magnitude Vektor Lokal 30](#_Toc473036256)

[3.4.3. Gaussian Weighting pada Magnitude 31](#_Toc473036257)

[3.4.4. Perhitungan Histogram pada Local Keypoint Window 31](#_Toc473036258)

[3.4.5. Penyusunan Nilai Histogram 31](#_Toc473036259)

[3.4.6. Penyiapan Fitur 32](#_Toc473036260)

[3.4.7. Kompleksitas Ekstraksi Fitur 32](#_Toc473036261)

[3.5. Gaussian Mixture Model 34](#_Toc473036262)

[3.6. Fisher Vector Encoding 35](#_Toc473036263)

[3.7. Proses Metric Learning 36](#_Toc473036264)

[3.8. Evaluasi Proses Training 39](#_Toc473036265)

[3.9. Pengukuran Jarak dan Klasifikasi 40](#_Toc473036266)

[3.10. Perhitungan Akurasi 41](#_Toc473036267)

[BAB IV IMPLEMENTASI 43](#_Toc473036268)

[4.1. Lingkungan implementasi 43](#_Toc473036269)

[4.2. Implementasi 43](#_Toc473036270)

[4.2.1. Implementasi Tahap Pre-processing 43](#_Toc473036271)

[4.2.2. Implementasi Ekstraksi Fitur dense SIFT 44](#_Toc473036272)

[4.2.3. Implementasi Preparasi Dataset 46](#_Toc473036273)

[4.2.4. Implementasi Pengumpulan Fitur dense SIFT 48](#_Toc473036274)

[4.2.5. Implementasi Pembagian Fold Dataset 49](#_Toc473036275)

[4.2.6. Implementasi Pengambilan Fitur 51](#_Toc473036276)

[4.2.7. Implementasi Learning Gaussian Mixture Model 52](#_Toc473036277)

[4.2.8. Implementasi Fisher Vector Encoding 54](#_Toc473036278)

[4.2.9. Implementasi Pengumpulan Fisher Vector Pada Data Test dan Data Train 55](#_Toc473036279)

[4.2.10. Implementasi inisialisasi matrix PSD, PCA, dan Whitening 56](#_Toc473036280)

[4.2.11. Implementasi Pengulangan Fitur Training 57](#_Toc473036281)

[4.2.12. Implementasi Tester Kesamaan Nama 59](#_Toc473036282)

[4.2.13. Implementasi Perhitungan Jarak 59](#_Toc473036283)

[4.2.14. Implementasi Update b dan w 60](#_Toc473036284)

[4.2.15. Implementasi Learning w dan b Serta Penyimpanan Hasil 60](#_Toc473036285)

[4.2.16. Implementasi Testing 65](#_Toc473036286)

[4.2.17. Implementasi Plot Fungsi Objektif 67](#_Toc473036287)

[BAB V PENGUJIAN DAN EVALUASI 69](#_Toc473036288)

[5.1. Lingkungan Pengujian 69](#_Toc473036289)

[5.2. Data Uji Coba 69](#_Toc473036290)

[5.3. Alur Uji Coba 70](#_Toc473036291)

[5.3.1. Preprocessing 70](#_Toc473036292)

[5.3.2. Ekstraksi Fitur dense SIFT 71](#_Toc473036293)

[5.3.3. Fisher Vector Encoding 71](#_Toc473036294)

[5.3.4. Pengukuran Jarak Fisher Vector 71](#_Toc473036295)

[5.3.5. Perhitungan Performa 71](#_Toc473036296)

[5.4. Skenario Uji Coba K-Gaussian 71](#_Toc473036297)

[5.4.1. Performa Pada K=128, Fold 1 72](#_Toc473036298)

[5.4.2. Performa Pada K=128, Fold 2 73](#_Toc473036299)

[5.4.3. Performa Pada K=128, Fold 3 74](#_Toc473036300)

[5.4.4. Performa Pada K=128, Fold 4 75](#_Toc473036301)

[5.4.5. Performa Pada K=192, Fold 1 76](#_Toc473036302)

[5.4.6. Performa Pada K=192, Fold 2 77](#_Toc473036303)

[5.4.7. Performa Pada K=192, Fold 3 78](#_Toc473036304)

[5.4.8. Performa Pada K=192, Fold 4 79](#_Toc473036305)

[5.4.9. Rekapitulasi Performa Uji Coba K-Gaussian 80](#_Toc473036306)

[5.5. Analisis Hasil Uji Coba 81](#_Toc473036307)

[BAB VI KESIMPULAN DAN SARAN 83](#_Toc473036308)

[6.1. Kesimpulan 83](#_Toc473036309)

[6.2. Saran 83](#_Toc473036310)

[DAFTAR PUSTAKA 85](#_Toc473036311)

[LAMPIRAN 87](#_Toc473036312)

# DAFTAR GAMBAR

[Gambar 2.1 Tipe Fitur Haar pada Viola-Jones Algotithm 7](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036313)

[Gambar 2.2 Penggunaan Fitur Haar pada wajah 8](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036314)

[Gambar 2.3 Ilustrasi perhitungan Integral Image 9](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036315)

[Gambar 2.4 Ilustrasi pengaruh filter Gaussian blur 11](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036316)

[Gambar 2.5 Ilustrasi cross validation menggunakan tiga fold 21](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036317)

[Gambar 3.1 Contoh gambar masukan 24](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036318)

[Gambar 3.2 Ilustrasi proses pre-processing 25](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036319)

[Gambar 3.3 Diagram alir desain sistem pada proses training 26](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036320)

[Gambar 3.4 Diagram alir desain sistem pada proses testing 27](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036321)

[Gambar 3.5 Diagram Alir Algoritma Viola-Jones 28](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036322)

[Gambar 3.6 Diagram Alir Ekstraksi Fitur dense SIFT 29](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036323)

[Gambar 3.7 Ilustrasi Penentuan Lokasi Keypoint 30](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036324)

[Gambar 3.8 Ilustrasi Patch dan Sub-window 31](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036325)

[Gambar 3.9 Penyusunan Nilai Histogram Pada toolbox vl\_feat 32](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036326)

[Gambar 3.10 Ilustrasi Penyiapan Fitur 33](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036327)

[Gambar 3.11 Kompleksitas Memori Ekstraksi Fitur 34](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036328)

[Gambar 3.12 Diagram Alir Algoritma EM 35](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036329)

[Gambar 3.13 Ilustrasi Perubahan Deskriptor 36](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036330)

[Gambar 3.14 Kompleksitas Memori Proses Inisialisasi PSD Matriks 39](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036331)

[Gambar 3.15 Histori Objective Function Value 40](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036332)

[Gambar 5.1 Contoh Pasangan Gambar Uji 70](#_Toc473036333)

[Gambar 5.2 Pengukuran Performa dengan K=128 pada fold 1 72](file:///E:\5112100086\Dropbox\%5bPENTING%20TIDAK%20URGENT%5d\%5bARSIP%20KULIAH%5d\Semester%209\%23tugas_akhir\Apps\%5bGIT%5d\Buku\2.2%205112100086_Yusuf%20Azis%20Henny%20Tri%20yudhantoro.docx#_Toc473036334)

[Gambar 5.3 Pengukuran Performa dengan K=128 pada fold 2 73](#_Toc473036335)

[Gambar 5.4 Pengukuran Performa dengan K=128 pada fold 3 74](#_Toc473036336)

[Gambar 5.5 Pengukuran Performa dengan K=128 pada fold 4 75](#_Toc473036337)

[Gambar 5.6 Pengukuran Performa dengan K=192 pada fold 1 76](#_Toc473036338)

[Gambar 5.7 Pengukuran Performa dengan K=192 pada fold 2 77](#_Toc473036339)

[Gambar 5.8 Pengukuran Performa dengan K=192 pada fold 3 78](#_Toc473036340)

[Gambar 5.9 Pengukuran Performa dengan K=192 pada fold 4 79](#_Toc473036341)

***[Halaman ini sengaja dikosongkan]***

# 

# DAFTAR TABEL

[Tabel 4.1 Lingkungan implementasi Perangkat Lunak 43](#_Toc473036342)

[Tabel 5.1 Confusion Matrix Hasil Uji Coba 80](#_Toc473036343)

[Tabel 5.2 Akurasi dan Waktu Eksekusi Hasil Uji Coba 80](#_Toc473036344)

***[Halaman ini sengaja dikosongkan]***

# DAFTAR KODE SUMBER

[Kode Sumber 4.1 Preprocessing deteksi dan resize wajah 44](#_Toc473036345)

[Kode Sumber 4.2 Ekstraksi Fitur dense SIFT 45](#_Toc473036346)

[Kode Sumber 4.3 Preparasi Dataset 46](#_Toc473036347)

[Kode Sumber 4.4 Pengumpulan Fitur dense SIFT 48](#_Toc473036348)

[Kode Sumber 4.5 Pembagian Fold Dataset 50](#_Toc473036349)

[Kode Sumber 4.6 Pengambilan Fitur 51](#_Toc473036350)

[Kode Sumber 4.7 Learning Gaussian Mixture Model 52](#_Toc473036351)

[Kode Sumber 4.8 Fisher Vector Encoding 54](#_Toc473036352)

[Kode Sumber 4.9 Pengumpulan Fisher Vector pada Data Test dan Data Train 55](#_Toc473036353)

[Kode Sumber 4.10 Inisialisasi matrix PSD, PCA, dan Whitening 56](#_Toc473036354)

[Kode Sumber 4.11 Pengulangan Fitur Training 57](#_Toc473036355)

[Kode Sumber 4.12 Tester Kesamaan Nama 59](#_Toc473036356)

[Kode Sumber 4.13 Perhitungan Jarak 60](#_Toc473036357)

[Kode Sumber 4.14 Update b dan w 60](#_Toc473036358)

[Kode Sumber 4.15 Learning w dan b serta penyimpana hasil 60](#_Toc473036359)

[Kode Sumber 4.16 Testing 65](#_Toc473036360)

[Kode Sumber 4.17 Plot Fungsi Objektif 67](#_Toc473036361)

***[Halaman ini sengaja dikosongkan]***

# 

# BAB I PENDAHULUAN

Pada bab ini dibahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan tugas akhir. Diharapkan dari penjelasan dalam bab ini gambaran tugas akhir secara umum dapat dipahami.

## Latar Belakang

Teknik pengenalan individu berbasis fitur biometrik wajah menjadi salah satu teknik yang paling sering digunakan. Hal ini dikarenakan penggunaan wajah untuk pengenalan memiliki beberapa kelebihan, antara lain: pengenalan wajah hanya membutuhkan peralatan kamera yang relatif ekonomis dan algoritma yang baik untuk mengidentifikasi. Selain itu pengenalan wajah dapat dilakukan secara pasif tanpa ada partisipasi pengguna apabila sistem ditujukan untuk pengamatan [1].

Terkait metode pengenalan wajah, Simonyan *et al* [2] membahas bahwa beberapa penilitian berfokus pada pengembangan ekstraksi fitur baru dengan representasi khusus yang bertujuan untuk melampaui fitur pengenalan wajah standar seperti *Scalable Invariant Feature Transform* (SIFT) [3]. Dalam jurnalnya Simonyan *et al* menemukan bahwa penggunaan *Fisher Vector Encoding* [4]pada fitur SIFT mode dense dari sebuah gambar dapat meningkatkan performa secara umum, serta penggunaan bobot fitur dense SIFT dapat dipelajari melalui *metric learning* [5]untuk meningkatkan performa dan cocok digunakan pada tugas identifikasi pada skala besar tanpa harus mengembangkan algoritma ekstraksi fitur yang baru dan spesifik.

Dari penemuan yang ada, tujuan dari usulan tugas akhir ini yaitu membuat aplikasi pengenalan wajah dengan metode ekstraksi fitur SIFT mode dense dan *Fisher Vector Encoding* serta pembelajaran bobot fitur dan threshold kesamaan wajah menggunakan *mahalanobis metric learning.*

## Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

* + - 1. Bagaimana melakukan ekstraksi fitur SIFT mode dense.
      2. Bagaimana melakukan ekstraksi fitur SIFT mode dense.
      3. Bagaimana melakukan fisher vector encoding.
      4. Bagaimana melakukan pembelajaran *low-rank Mahalanobis Metric Learning* terhadap bobot dan *threshold* jarak.
      5. Bagaimana melakukan klasifikasi pasangan gambar.

## Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Dataset yang diambil adalah Labeled Face in The Wild, University of Massachusetts Amherst, USA [6].
2. Algoritma ekstraksi fitur yang diimplementasikan adalah SIFT mode *dense* yang diencoding pada Fisher Vectormenggunakan library *vl\_feat* [7].
3. Algoritma untuk pendeteksian area wajah adalah Viola Jones yang diimplementasikan menggunakan library Matlab *CascadeObjectDetector.*
4. *Metric learning* yang digunakan adalah *mahalanobis metric learning*
5. *Deployment* aplikasi dilakukan dalam bentuk *console*
6. Bahasa pemrograman yang dipakai adalah Matlab dengan bantuan library tambahan *vl\_feat*.
7. Evaluasi dilakukan dengan metode K-fold validation menggunakan 4 fold

## Tujuan

Tujuan dari pembuatan tugas akhir ini adalah merancang aplikasi yang dapat melakukan pengenalan wajah menggunakan metode ekstraksi fitur SIFT mode *dense* dan Fisher Vector encoding.

## Manfaat

Pengerjaan tugas akhir ini dilakukan dengan harapan bisa memberikan kontribusi pada sistem yang memerlukan verifikasi wajah secara cepat dan dapat diaplikasikan pada sistem atau aplikasi pengenalan wajah berskala besar.

## Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.

Tahap awal yang dilakukan dalam pengerjaan Tugas Akhir ini adalah penyusunan proposal Tugas Akhir. Di dalam proposal diajukan suatu gagasan pembuatan perangkat lunak untuk melakukan pengenalan wajah menggunakan metode ekstraksi fitur SIFT mode *dense* dan Fisher Vector encoding.

1. Studi literatur

Pada tahap ini dilakukan pencarian, pengumpulan, penyaringan, pemahaman, dan pembelajaran literatur yang berhubungan dengan fitur *Viola Jones Face Detection*, *Fitur SIFT, Fisher Vector, Gaussian Mixture Model, Singular Value Decomposition, Whitening Transformation, Principal Component Analysis, Mahalanobis Metric Learning, k-fold Cross Validation*. Literatur yang digunakan meliputi: buku referensi, jurnal, dan dokumentasi internet.

1. Perancangan perangkat lunak

Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Kemudian dilakukan desain suatu sistem dan desain proses-proses yang ada.

1. Implementasi perangkat lunak

Implementasi merupakan tahap membangun rancangan program yang telah dibuat. Pada tahapan ini merealisasikan rancangan yang terdapat pada tahapan sebelumnya, sehingga menjadi sebuah program yang sesuai dengan apa yang telah direncanakan.

1. Pengujian dan evaluasi

Pada tahap ini dilakukan uji coba terhadap perangkat lunak yang telah dibuat untuk mengetahui kemampuan algoritma yang dipakai, mengamati kinerja sistem, serta mengidentifikasi kendala yang mungkin timbul pada aplikasi yang dibuat.

1. Penyusunan buku Tugas Akhir.

Pada tahapan ini disusun buku yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

## Sistematika Penulisan Laporan Tugas Akhir

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

1. Bab I. Pendahuluan

Bab ini berisi penjelasan mengenai latar belakang masalah, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu rumusan permasalahan, batasan masalah, dan sistematika penulisan juga merupakan bagian dari bab ini.

1. Bab II Tinjauan Pustaka

Bab ini berisi penjelasan tentang citra dengan *noise* beserta jenis-jenis *noise*, metode estimasi robust spasial, segmentasi citra dan metode *Fuzzy Co-clustering For Images* (FCCI).

1. Bab III Perancangan Perangkat Lunak

Bab ini berisi penjelasan mengenai desain, perancangan, bahan, dan pemodelan proses yang digunakan dalam Tugas Akhir ini yang direpresentasikan dengan *pseudocode*.

1. Bab IV. Implementasi

Bab ini merupakan pembangunan aplikasi dengan MATLAB sesuai permasalahan dan batasan yang telah dijabarkan pada Bab I.

1. Bab V. Hasil Uji Coba dan Evaluasi

Bab ini berisi penjelasan mengenai data hasil percobaan, pengukuran, dan pembahasan mengenai hasil percobaan yang telah dilakukan.

1. Bab VI. Kesimpulan dan Saran

Bab ini berupa hasil penelitian yang menjawab permasalahan atau yang berupa konsep, program, dan karya rancangan. Selain itu, pada bab ini diberikan saran-saran yang berisi hal-hal yang masih dapat dikerjakan dengan lebih baik dan dapat dikembangkan lebih lanjut, atau berisi masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir.

***[Halaman ini sengaja dikosongkan]***

# BAB II DASAR TEORI

Bab ini berisi penjelasan teori-teori yang berkaitan dengan pembuatan aplikasi pengenalan wajah menggunakan fitur SIFT mode *dense* dan *fisher vector encoding*. Penjelasan ini bertujuan untuk memberikan dasar teori yang mendasari pengembangan perangkat lunak.

## Viola-Jones Face Detection

*Viola-Jones Face Detection* adalah salah satu metode untuk mendeteksi wajah pada sebuah gambar. Algoritma pengenalan wajah terbagi dalam 4 tahapan [8] yakni:

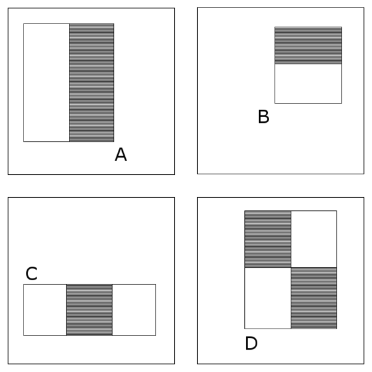
*Haar Feature Selection.*

Pembuatan *Integral Image*.

*Adaboost Training*.

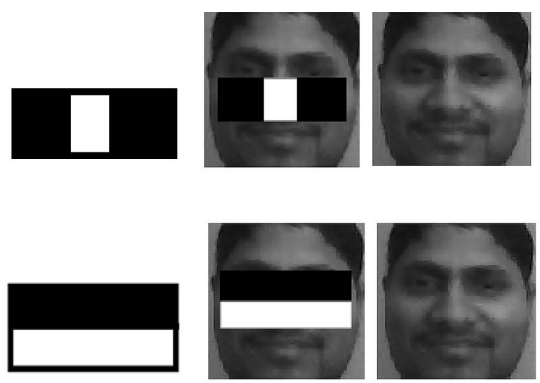
*Cascading Classifiers*

### Haar Feature Detection

*Fitur Haar* adalah pola gelap dan terang yang diatur sedemikian rupa sehingga menyerupai tingkat kecerahan wajah pada area tertentu. *Fitur Haar* yang digunakan pada algoritma ini diberikan pada Gambar 2.1

Sumber : Wikipedia.org

**Gambar 2.1 Tipe Fitur Haar pada Viola-Jones Algotithm**

Sumber : Wikipedia.org

Gambar 2.2 Penggunaan Fitur Haar pada wajah

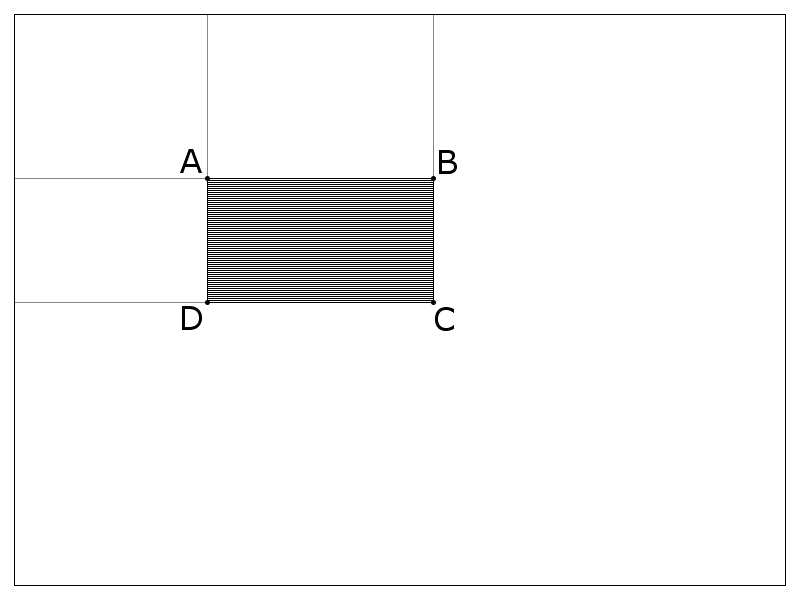
Penggunaan *Fitur Haar* pada Gambar 2.1 dikarenakan kesamaannya dengan pola tingkat kecerahan wajah secara umum. Seperti yang diilustrasikan Gambar 2.2, pola B mirip dengan pola area mata yang lebih gelap daripada pipi, pola C mirip dengan jembatan antara hidung dan pipi, bisa juga digunakan untuk area dahi, mata, dan pipi, dll. Fitur-fitur ini dicocokkan pada lokasi mata, hidung, dan mulut pada sebuah gambar dalam beberapa skala, dan ukuran.

### Pembuatan Integral Image

Penjumlahan area gelap dan terang dapat dengan cepat diperoleh melalui perhitungan *Integral Image*. Perhitungan *Integral* *Image* dilakukan dengan ilustrasi pada Gambar 2.3. Penghitungan area dilakukan dengan menjumlahkan area sesuai dengan persamaan (2.1).

Penentuan sebuah *Fitur Haar* merupakan bagian dari wajah atau tidak diperoleh dari pengurangan integral image pada area gelap dengan area terang. Apabila memberikan nilai negatif maka area tersebut adalah bagian dari wajah, jika positif maka bukan.

|  |  |
| --- | --- |
|  | **(2.1)** |
| **Keterangan:** | |
| : Adalah area yang dihitung dari titik 0,0 sampai ke titik x | |

Sumber: Wikipedia.org

Gambar 2.3 Ilustrasi perhitungan Integral Image

### Adaboost Training

Setiap *Fitur Haar* digunakan untuk mengklasifikasikan sebuah gambar apakah dia wajah atau bukan. Dari situ kita mengetahui mana saja yang merupakan fiturlemah dan kuat dalam tugasnya sebagai *classifier* (yang berikutnya pada pembahasan ini disebut *classifier*).

Kumpulan *classifier* lemah dan kuatkemudian digabung menggunakan Metode *Adaboost Training*, karena metode ini menganggap setiap *classifier* memiliki andil dalam klasifikasi. Kemudian dicari kombinasi *classifier* yang mampu menghasilkan *classifier* baru dengan performa yang lebih baik.

### Cascading Classifier

*Classifier* wajah kemudian disusun menggunakan arsitektur *cascade*. Arsitektur tersebut dibangun dengan cara mengurutkan *classifier* yang ada hasil *Adaboost Training* dari tingkat *error* klasifikasi terrendah sampai yang paling tinggi. Setiap *classifier* bisa jadi terdiri dari beberapa *Fitur Haar* dan disusun *stage-by-stage*.

Setiap *stage* memiliki tugas untuk menentukan apakah sebuah area bisa jadi area wajah atau sama sekali bukan. Proses klasifikasi area wajah dilakukan dengan cara mengetesnya pada setiap stage. Apabila sebuah area gagal dideteksi sebagai area wajah pada sebuah stage, maka stage tersebut dihapus, lalu lanjut ke stage berikutnya. Jika benar maka lanjut pada stage berikutnya.

Dari semua tes yang dilakukan pada *cascade classifier* kemudian dapat dikumpulkan area mana saja yang termasuk bagian dari wajah, lalu ditandailah area akhir yang menunjukkan area wajah secara keseluruhan.

## dense Scalable Invariant Feature Transform (d-SIFT)

Ekstraksi fitur dense SIFT merupakan salah satu proses ekstraksi fitur lokal untuk mendeskripsikan objek pada sebuah gambar. Deskripsi objek diwakili oleh kumpulan *keypoint* yang memiliki *descriptor* unik.

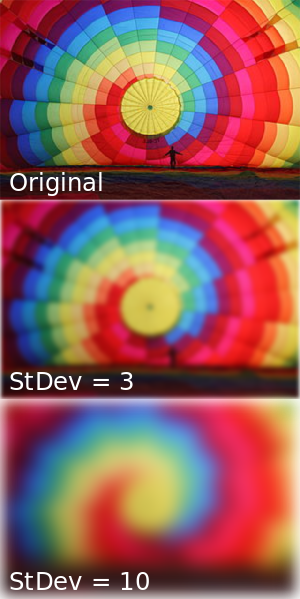
Perbedaan utama antara dense SIFT dan SIFT ada pada proses yang dilalui. SIFT menggunakan algoritma lowe [3] untuk mendeteksi lokasi penting dari sebuah citra, lalu mengesktrak deskripsi lokal pada lokasi tersebut. Sedangkan dense-SIFT mengekstraksi deskripsi lokal pada semua lokasi citra. Hal ini membuat dense SIFT memerlukan waktu proses yang jauh lebih cepat daripada SIFT biasa.

Berikut beberapa teori terkait dengan proses dense SIFT:

* + - * 1. Gaussian Blur
        2. Perhitungan orientasi vektor pada piksel citra.
        3. Perhitungan *magnitude* vektor pada piksel citra.

### Gaussian Blur

Gaussian blur adalah salah satu filter low pass pada citra yang bertujuan mengkaburkan gambar sehingga kontras citra berkurang.Filter ini dioperasikan dengan mengkonvolusi sebuah citra masukan dengan sebuah *Gaussian Kernel*. *Gaussian Kernel* dibentuk dengan parameter ukuran kernel dan σ, pada persamaan (2.2) dan ddilustrasikan pada gambar Gambar 2.4.

Sumber: Wikipedia.org

Gambar 2.4 Ilustrasi pengaruh filter Gaussian blur

|  |  |
| --- | --- |
|  | **(2.2)** |
| **Keterangan:** | |
| adalah jarak titik dari pusat kernel | |
| adalah besar faktor bluring pada distribusi gaussian | |

Pada metode yang diajukan Simonya *et al* [2]*,* Gaussian filter dilakukan pada image dengan 5 skala sigma ( yang berbeda pada satu ukuran gambar yang sama yaitu 160x125. penulis menggunakan skala 1.6 pada skala pertama, dan naik dengan faktor pengali k= pada skala berikutnya.

### Perhitungan Orientasi Vektor pada Piksel Citra

Deskriptor teridiri dari histogram vector yang merupakan besaran (*magnitude)* dana arah (orientasi). Pada proses ini dikumpulkan orientasi lokal pada setiap piksel. Pengumpulan orientasi ini nantinya digunakan untuk mengumpulkan deskripsi pada *keypoint*. Orientasi diperoleh dengan rumus matematis yang dijelaskan pada persamaan (2.3).

|  |  |
| --- | --- |
|  | (2.3) |
| Keterangan: | |
| adalah piksel pada citra | |

### Perhitungan Magnitude Vektor pada Piksel Citra

Pada proses ini dikumpulkan besar gradien lokal pada setiap piksel. Pengumpulan gradien lokal ini nantinya digunakan untuk mengumpulkan deskripsi pada *keypoint*. Gradien lokal diperoleh dengan rumus matematis yang dijelaskan pada persamaan (2.4).

|  |  |
| --- | --- |
|  | (2.4) |
| Keterangan: | |
| adalah piksel pada citra | |

## Singular Value Decomposition (SVD)

*Singular Value Decompotition (SVD)* dapat digunakan untuk mendekomposisi *data point* yang asli menjadi beberapa parameter. Dari parameter yang dihasilkan tersebut dapat merepresentasikan data aslinya menggunakan dimensi yang lebih sedikit. Dengan kata lain, SVD dapat didefinisikan sebagai metode untuk mereduksi data [9]. Secara umum SVD dirumuskan dengan persamaan (2.5)

|  |  |
| --- | --- |
|  | (2.5) |
| Keterangan: | |
| : matriks *othonormal* *eigen vector* dari dengan | |
| : matriks *othonormal eigen vector* dari dengan | |
| : matriks diagonal yang mengandung akar dari dan dengan urutan menurun | |
| merupakan covariance matriks dari data aslinya | |

Ada beberapa persamaan yang dapat digunakan untuk menemukan dekomposisi SVD tersebut.

|  |  |
| --- | --- |
|  | (2.6) |
|  | (2.7) |
|  | (2.8) |
|  | (2.9) |
|  | (2.10) |
| Keterangan: | |
| : *eigen value* | |
| : eigen vector ke-i | |

Pencarian komponen dekomposisi didapatkan awalnya dengan menentukan *eigen value* dari matriks pada *null space*. Pencarian *eigen value* tersebut dirumuskan dengan persamaan (2.9). Kemudian setiap eigen vector dicari dengan persamaan (2.10). Penyusunan eigen vector hasil persamaan (2.10) adalah komponen dekomposisi V.

Komponen dekomposisi didapatkan dari akar dari setiap eigen value, yang disusun menjadi matriks dengan mengalikannya dengan matriks identitas. Setelah komponen dekomposisi S dan V ditemukan, dengan memasukkannya ke persamaan C maka ditemukan juga komponen dekomposisi U.

Komponen dekomposisi telah diurutkan berdasarkan tingkat variasi frequensi yang ditangkap dari sebuah matriks dari yang paling tinggi ke rendah. Kemudian dengan menjumlahkan sebagian komponen tersebut dari komponen tertingginya, maka kita akan mendapatkan matriks asalnya () dengan sedikit penurunan kualitas. Itulah mengapa SVD dapat diaplikasikan untuk reduksi data.

## PCA-Whitening Transformation

*Whitening Transformation* adalah proses yang mentransformasikan sebuah kumpulan data menjadi tidak terkolerasi. Proses disebut whitening karena hasil *whitening* memberikan *covariance* matriks berupa garis diagonal putih[10]*.* Operasi dapat memanfaatkan parameter dekomposisi SVD [11] seperti pada persamaan (2.5). Proses Whitening diberikan pada persamaan (2.11) dan (2.12).

*Principal Component Analysis (PCA)* merupakan metode reduksi dimensi data. Metode PCA dapat dilakukan dengan menggunakan parameter SVD. Proses PCA pada persamaan (2.13) dapat dilakukan karena dekomposisi SVD telah mengurutkan atribut yang menyimpan kepadatan data dari yang tertinggi ke yang terendah.

|  |  |
| --- | --- |
|  | (2.11) |
|  | (2.12) |
|  | (2.13) |
| Keterangan: | |
| : x adalah data asli yang akan dilakukan PCA | |
| : data yang dirotasi | |
| *xWhite* : data yang telah dilakukan proses whitening | |
| : data ter-whitening yang sudah direduksi | |
| *epsilon*: variabel batas bawah nilai S (0.0001) | |

## Gaussian Mixture Model (GMM)

*Gaussian Mixture Model (GMM)* sebuah metode untuk menemukan secara otomatis kluster gaussian dari kumpulan data yang ada. GMM sangat mirip dengan K-Means *clustering* kecuali pada dua hal, yaitu:

* K-Means menghasilkan *hard clustering* (merujuk kelas absolut tertentu, dan tidak saling *overlap*) sedangkan GMM menghasilkan *soft clustering* (merujuk pada probabilitas terhadap kelas tertentu, sehingga dapat *overlap*).
* K-Means menghitung jarak menggunakan Euclidian distance, dimana pada kasus multivariat menganggap setiap atribut memiliki kontribusi yang sama. Sedangkan GMM menggunakan kovarian yang membuat interpretasi terhadap jarak terus berubah karena setiap atribut dapat memberikan kontribusi yang berbeda.

Pada praktiknya GMM seringkali digunakan ketika sebuah data hendak direpresentasikan ke bentuk yang berbeda. Pada kasus penelitian ini GMM berfungsi untuk menemukan cluster fitur dense SIFT. Lebih lanjut parameter kluster tersebut (*weight, means, covariance, priors*) dapat digunakan untuk membentuk representasi yang lebih sederhana menggunakan *fisher vector*.

Parameter GMM diestimasikan dari data training menggunakan *Iterative Expectation-Maximization (EM Algorithm)* [12]*.* GMM dilatih menggunakan algoritma EM dengan tahap sebagai berikut:

Penentuan jumlah K Gaussian.

Inisialisasi means serta covariance.

Kalkulasi karakteristik sebaran data.

Kalkulasi probabilitas data anggota klaster.

Kalkulasi bobot masing-masing data

Kalkulasi means baru

Kalkulasi *covariance* baru.

Kalkulasi banyaknya anggota klaster

Repetisi proses

### Penentuan Jumlah K Gaussian

Penentuan jumlah K-Gaussian dilakukan secara manual, dimana sejumlah data nantinya akan diklaster ulang dengan jumlah klaster sebanyak K sesuai masukan.

### Inisialisasi Means Serta Covariance

Inisialisasi means serta matriks *covariance* dilakukan secara random sebanyak K yang diminta. Untuk *covariance* diinisialisasi menggunakan sebuah matriks identitas (*euclidean*), sedangkan means random.

### Kalkulasi Karakteristik Sebaran Data

Karakteristik sebaran data ke *Xi* terhadap klaster *k* dihitung untuk menentukan bagaimana bentuk sebaran, dan nantinya berguna untuk menentukan probabilitas keanggotaan data terhadap klaster tertentu. Kalkulasi ini dilakukan dengan persamaan (2.14).

|  |  |
| --- | --- |
|  | (2.14) |
|  |
| Keterangan: | |
| : karakteristik sebaran data *Xi* terhadap klaster *k* | |
| : covariance matriks | |
| : vektor data ke *i* | |
| : vector means kluster *k* | |

### Kalkulasi Probabilitas Data Anggota Klaster

Kalkulasi ini digunakan untuk menghitung bagaimana probabilitas data ke *Xi* bersumber dari klaster *k*. Perhitungan probabilitas ini membutuhkan informasi awal probabilitas masing-masing kluster ke-k *P(k)* dimana diinisialisasi dengan nilai dengan n merupakan banyaknya kluster yang kita cari. Perhitungan probabilitas data ke *Xi* bersumber dari klaster *k* dilakukan dengan rumus (2.15).

|  |  |
| --- | --- |
|  | (2.15) |
| Keterangan: | |
| : karakteristik sebaran data *Xi* terhadap klaster *k* | |
| : probabilitas data ke *Xi* bersumber dari klaster *k* | |
| : Probabilitas kluster ke k | |

### Kalkulasi Bobot Masing-masing Data

Kalkulasi ini digunakan untuk menghitung bobot seberapa penting data ke *Xi* terhadap klaster *k*, dihitung dengan persamaan (2.16).

|  |  |
| --- | --- |
|  | (2.16) |
| Keterangan: | |
| : bobot seberapa penting data *Xi* terhadap kluster *k* | |
| : probabilitas data ke *Xi* bersumber dari klaster *k* | |

### Kalkulasi Means Baru

Kalkulasi ini digunakan untuk menghitung means baru dari setiap atribut *a* terhadap kluster *k* dengan rumus (2.17).

|  |  |
| --- | --- |
|  | (2.17) |
| Keterangan | |
| : means dari atribut *a* terhadap kluster k | |
| : bobot seberapa penting data *Xi* terhdap klaster *k* | |
| : data pada atribut ke *a* | |

### Kalkulasi Covariance Baru

Kalkulasi ini berguna untuk menghitung covariance baru secara berpasangan dari fitur *a* dan *b* dari klaster *k*, dan seterusnya untuk setiap pasangan fitur yang digambarkan pada rumus(2.18).

|  |  |
| --- | --- |
|  | (2.18) |
| Keterangan | |
| : covariance dari atribut *a* dab *b* terhadap klaster *k* | |
| : bobot seberapa penting data *Xi* terhdap klaster *k* | |
| : data ke *i* pada klaster a | |

### Kalkulasi Banyaknya Anggota Klaster

Kalkulasi ini berguna untuk menghitung prosentase jumlah anggota pada kelas *k*. Perhitungannya menggambarkan berapa banyak data yang termasuk kedalam kelas *k* dan dilakukan dengan rumus(2.19).

|  |  |
| --- | --- |
|  | (2.19) |
| Keterangan | |
| : probabilitas kelas *k* / banyaknya data pada kelas *k* | |
| : probabilitas data ke *Xi* pada kelas k | |

### Repetisi Proses

Proses 2.5.1 sampai dengan 2.5.8 terus diulang kembali hingga perubahan kepadatan klaster konvergen.

## Fisher Vector Encoding

Fisher vector encoding merupakan metode transformasi representasi data. Fitur dense SIFT yang berbentuk *d*\**k* dengan k adalah banyaknya *keypoint* dan *d* adalah dimensi deskriptor tiap *keypoint* dirubah menjadi bentuk 2\*K\*d, dengan K adalah banyaknya klaster yang dipelajari di GMM. Encoding dilakukan dengan rumus (2.20) untuk komponen vektor pertama dan rumus (2.21) untuk komponen vektor kedua [2].

|  |  |
| --- | --- |
|  | (2.20) |
|  | (2.21) |
| Keterangan | |
| : Jumlah *keypoint* | |
| : Iterasi jumlah *keypoint* | |
| : Iterasi jumlah komponen gaussian sebanyak K gaussian | |
| : Bobot Gaussian pada komponen Gaussian ke - *k* | |
| : Prediksi distribusi Gaussian ke-*k* dari fitur SIFT ke-*p* | |
| : Prediksi distribusi Gaussian dari fitur SIFT ke-*p* | |
| : Varian dari distribusi Gaussian ke-*k* | |

## Mahalanobis Metric Learning

*Mahalanobis Metric Learning* adalah salah satu metode *metric learning* yang berfungsi untuk menentukan bobot jarak antara dua vektor. Dalam *metric learning, mahalanobis distance* seringkali digunakan untuk merujuk pada fungsi jarak *euclidean* pada persamaan (2.22):

|  |  |
| --- | --- |
|  | **(2.22)** |
| **Keterangan:** | |
| : Matriks semi definitif positif | |
| x,y : Vektor yang akan dihitung jaraknya | |

Dimana merupakan sebuah matriks semi definitif positif hasil *metric learning* dari *training set* yang berfungsi sebagai bobot untuk menentukan kontribusi setiap atribut dalam perhitungan jarak*.* Pembelajaran matriks pada dasarnya dapat diaplikasikan pada komputer secara umum apabila dimensinya kurang dari (1000x1000). Namun dengan representasi *fisher vektor* yang berdimensi tinggi (>1000), maka pembelajaran mahalanobis metric tidak dapat dilakukan pada matriks M secara langsung.

Matriks . merupakan *low-rank* *linear projection* dari deskriptor yang mendekomposisikan matriks yang apabila dikalikan dengan transposenya memiliki dimensi yang sama dengan . Hal ini menjelaskan bahwa proses pembelajaran *full-rank* matriks dapat diperoleh melalui melatih *low rank linear projection* . Matriks nantinya akan memproyeksikan *fisher vector* dengan dimensi tinggi menjadi vektorberdimensi rendah . Dengan definisi tersebut, secara matematis fungsi jarak pada persamaan (2.22). dapat ditulis ulang menjadi fungsi jarak pada persamaan (2.23)

|  |  |
| --- | --- |
|  | **(2.23)** |
|  |
| **Keterangan:** | |
| x,y : vektor yang akan dihitung jaraknya. | |
| : *low rank linear projection* dari *M,* hasil *metric learning* | |

## Euclidean Distance

Adalah salah satu metode menghitung jarak antara dua objek (vector atau titik) pada sebuah ruang dimensi. Secara matematis dapat dihitung dengan persamaan (2.24) ataupun (2.25) sesuai dengan kebutuhannya:

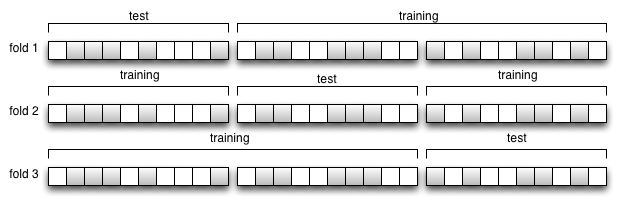
1. Jarak pada ruang dimensi dua:

|  |  |
| --- | --- |
|  | (2.24) |

1. Jarak pada ruang dimens-n:

|  |  |
| --- | --- |
|  | (2.25) |
| Keterangan: | |
| adalah titik pada ruang dimensi satu | |
| adalah vektor pada ruang dimensi 2 atau lebih | |

## K-Fold Cross Validation

Adalah salah satu teknik validasi model terhadap set data independen. Secara umum caranya dengan membagi sebuah *dataset* menjadi K *sub-dataset*. Dari semua sub akan diambil satu sub sebagai *testing data*, dan sisanya menjadi *training data*, begitu seterusnya sampai sub ke K menjadi *testing data*. Lalu performa dihitung dari rata-rata hasil *cross validation*.

**Gambar 2.5 Ilustrasi cross validation menggunakan tiga fold**

Sumber: Wikipedia.org

## Stochastic Gradient Descent

Adalah salah satu teknik pengukuran error yang bekerja secara iteratif. Error pada fungsi objektif tertentu diminimalisir pada setiap iterasi dengan persamaan (2.26).

|  |  |
| --- | --- |
|  | **(2.26)** |
| **Keterangan:** | |
| (w): Fungsi objektif dengan parameter w | |
| : Nilai evaluasi pada fungsi objektif (w), iterasi ke-i | |
| n : jumlah iterasi | |

***[Halaman ini sengaja dikosongkan]***

.

# BAB III PERANCANGAN PERANGKAT LUNAK

Pada bab ini dijelaskan mengenai rancangan sistem perangkat lunak yang akan dibuat. Perancangan yang dijelaskan meliputi data dan proses. Data yang dimaksud adalah data yang akan diolah dalam perangkat lunak baik digunakan sebagai pembelajaran maupun pengujian sehingga tujuan Tugas Akhir ini bisa tercapai. Proses yaitu tahap-tahap yang ada dalam sistem sebagai pengolah data meliputi *pre-processing,* ekstraksi fitur dense SIFT, penerapan *Gaussian Mixture Model* dan *fisher vector encoding,* proses *metric learning,* dan pengukuran jarak.

## Data

Pada sub bab ini akan dijelaskan mengenai data yang digunakan sebagai masukan perangkat lunak untuk selanjutnya diolah dan dilakukan pengujian sehingga menghasilkan data keluaran yang diharapkan. Kumpulan data atau dataset yang digunakan pada penelitian ini didapatkan dari *Labeled Face in the Wild* (LFW) [6]. Dataset terdiri dari 13.233 gambar dari 5.749 orang yang di download melalui internet. Dataset ini merupakan sebuah acuan untuk pengujian algoritma pengenalan wajah secara umum.

Untuk proses pengembangan algoritma, LFW menyediakan 6000 pasangan gambar yang terbagi dalam 10 fold. Masing-masing fold terdiri dari 600 pasangan gambar yang telah didefinisikan sebelumnya secara acak. Dari 600 pasangan gambar tersebut 300 pasang merupakan pasangan positif, dan 300 lainnya merupakan pasangan negatif. Pada analisis kompleksitas di bab 5, diketahui bahwa penggunaan 6000 pasangan gambar ini sangat menyita *resource*, sehingga penulis menurunkannya menjadi 400 pasangan secara keseluruhan. Data ini nantinya akan digunakan baik untuk materi training maupun evaluasi melalui protokol *k-fold* *cross validation* menggunakan empat *fold*.

### Data Masukan

Data masukan adalah data yang digunakan sebagai masukan dari sistem. Data yang digunakan adalah pasangan gambar wajah yang dipilih acak dari kelas positif (orang yang sama) dan kelas negatif (orang yang berbeda). Apabila dua gambar memiliki nama yang sama, maka sistem harus mampu mendeteksi pasangan gambar tersebut merupakan kelas positif, begitu pula sebaliknya. Contoh data masukan dapat dilihat pada Gambar 3.1.

|  |
| --- |
| Sumber: *Labeled Faces in the Wild.*  *University of Massachusetts Amherst, USA* |
| Gambar 3.1 Contoh gambar masukan |

### Data Pembelajaran

Data pembelajaran digunakan pada proses *metric learning*. Data yang digunakan adalah pasangan gambar wajah yang dipilih acak dari kelas positif dan kelas negatif. Proporsi jumlah data diatur 80% untuk pembelajaran, dan 20% sisanya untuk data test atau dengan kata lain data berbanding 4:1.

### Data Keluaran

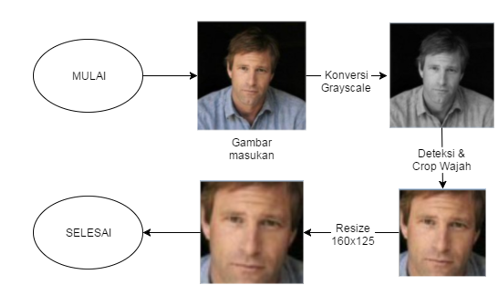
Data keluaran adalah sebuah keputusan yang menentukan apakah pasangan gambar tersebut merupakan kelas positif dengan nilai dan kelas negatif dengan nilai.

## Desain Sistem Secara Umum

Rancangan perangkat lunak pengenalan wajah terbagi dalam dua tahap, yakni tahap pembelajaran dan tahap pegetesan. Pada tahap pembelajaran proses dimulai dengan ekstraksi fitur dense SIFT, lalu pembalajaran *Gaussian Mixture Model, fisher vector encoding*, dan terakhir dilakukan pembelajaran *Mahalanobis Metric learning* untuk mendapatkan metrik berupa matriks model jarak sebagai pengali pada *fisher vector* gambar, serta *threshold* *b* untuk menentukan jarak kelas positif dan negatif. Diagram alir proses pembelajaran ditunjukkan pada Gambar 3.3.

Pada tahap pengetesan proses dimulai dengan membaca masukan berupa pasangan gambar. Proses pengenalan wajah terdiri dari beberapa proses besar, yaitu *preprocessing,* ekstraksi fitur SIFT, *fisher vector encoding*,pengukuran jarak. Diagram alir proses pengetesan perangkat lunak ditunjukkan pada Gambar 3.4.

## Pre-processing

Setiap pasangan gambar yang diproses tidak langsung dilakukan ekstraksi fitur dan pengujian untuk menentukan apakah pasangan gambar tersebut orang yang sama atau bukan. Tahap awal yang dilakukan adalah *pre-processing* yang bertujuan untuk menyeragamkan proses, agar memberikan hasil yang seragam. Secara umum proses pre-processing dijelaskan pada Gambar 3.2.

Sumber : Labeled Faces in the Wild

*University of Massachusetts Amherst, USA*

Gambar 3.2 Ilustrasi proses pre-processing

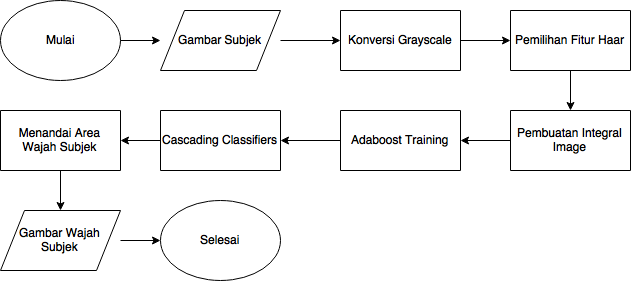


**Gambar 3.3 Diagram alir desain sistem pada proses training**



**Gambar 3.4 Diagram alir desain sistem pada proses testing**

### Deteksi Wajah dan Pemotongan Gambar

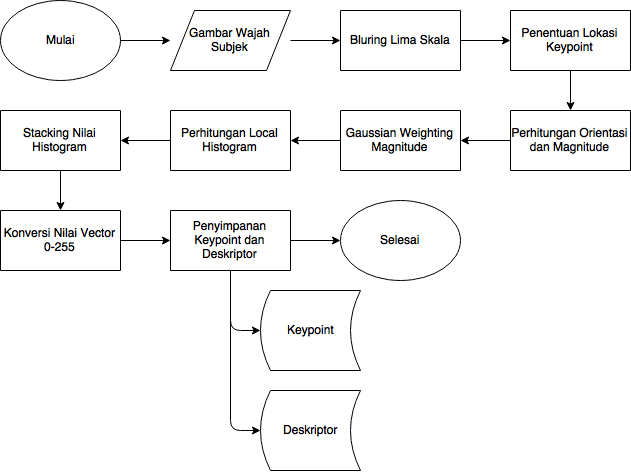
Pre-processing gambar dimulai dengan melakukan konversi *grayscale* dan pengenalan area wajah menggunakan metode *Viola-Jones,* lalu gambar dipotong pada area wajah. Hal ini bertujuan untuk memfokuskan pengambilan fitur dense SIFT hanya pada area wajah agar tidak disalah-artikan untuk mengenali *background* ataupun objek lainnya. Secara detail algoritma deteksi wajah *Viola-Jones* dijelaskan pada gambar Gambar 3.5 dan penjelasan masing-masing tahap pada sub-bab 2.1.

Gambar 3.5 Diagram Alir Algoritma Viola-Jones

### Resize Gambar

Setelah itu gambar wajah yang sudah dipotong akan diseragamkan ukurannya menjadi 160x125 dengan interpolasi. Hal ini ditujukan agar gambar memiliki ukuran yang sama untuk dioperasikan pada algoritma dense SIFT. Baru setelah itu gambar dapat diekstraksi fiturya dan diolah lebih lanjut.

## Ekstraksi Fitur dense SIFT

Pada tahap ekstraksi fitur dense SIFT akan dijelaskan bagaimana alur ekstraksi dari gambar yang ada menjadi sebuah representasi *keypoint* dan *descriptor*. Masukan dari tahap ini adalah hasil akhir dari tahap *preprocessing*. Hasil akhir dari proses ekstraksi fitur dense SIFT adalah kumpulan *keypoint* dan deskriptor yang tersimpan dalam sebuah array. Metode ekstraksi fitur dense SIFT ini dipilih karena memiliki kelebihan dapat mengenali objek melalui fitur lokal*.* Mode dense sendiri diambil untuk memperoleh keypoint di seluruh bagian wajah. Diagram alir proses ekstraksi fitur dense SIFT dapat dilihat pada Gambar 3.6.

Gambar 3.6 Diagram Alir Ekstraksi Fitur dense SIFT

### Bluring dan Penentuan Letak Keypoint

Pada proses ini gambar yang sudah di-preprocessing kemudian dibluring sebanyak 5 skala. Skala bluring pertama adalah 1.6, lalu empat skala berikutnya diperbesar dengan faktor . Setelah itu pada masing-masing skala ditentukan berapa ukuran matriks *patch* untuk mengekstraksi fitur pada citra. Sebuah patch terdiri dari 4x4 blok, yang pada setiap tengahnya merupakkan lokasi ekstraksi descriptor. Simonyan *et al* [2]menggunakan patch berukuran 24 x 24, sehingga setiap blok berukuran 6 piksel. Lokasi ekstraksi dilakukan pada semua piksel citra (menggunakan *sampling step=*1) dengan batas [25,25,100,135]. Keterangan lengkap dapat dilihat pada gambar Gambar 3.7.

|  |
| --- |
|  |
| Sumber: *VLFeat Documentation*  **Gambar 3.7 Ilustrasi Penentuan Lokasi Keypoint**  *Princeton Univesrsity, UK* |

*Library vl\_feat* secara otomatis akan memulai dan mengakhiri ekstraksi fitur tepat pada sehingga akan didapatkan 5394 lokasi *keypoint* yang masing-masing nantinya akan menghasilkan 128x1 dimensi deskriptor. 128 deskriptor ini merupakan histogram dari vector pada 8 bin arah pada masing-masing blok 4x4.

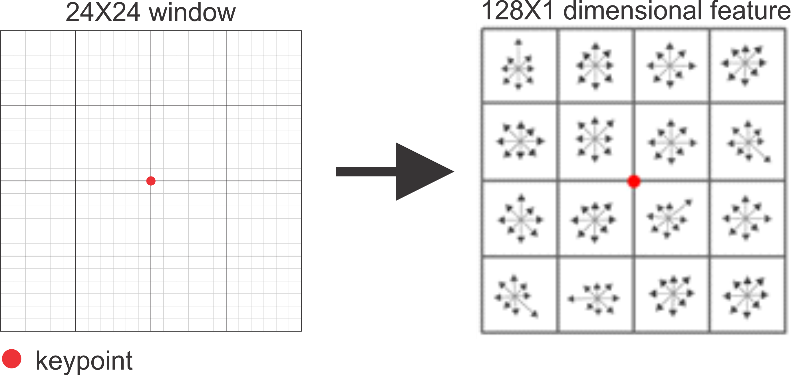
### Perhitungan Orientasi dan Magnitude Vektor Lokal

Pada masing-masing lokasi *keypoint* yang dijelaskan pada sub-bab sebelumnya dibuat *patch*. Setiap block pada *patch* tersebut nantinya akan dicari histogram vektor, dan diambil vektor yang paling dominan. Penjelasan teknis perhitungan orientasi dijelaskan pada sub-bab 2.2.2 dan perhitungan magnitude dijelaskan pada sub-bab 2.2.3.

### Gaussian Weighting pada Magnitude

*Magnitude* pada masing-masing *patch* kemudian dilakukan pembobotan Gaussian dengan mengalikannya dengan kernel Gaussian dengan *variance* sesuai dengan skala blur dan lebar kernel sesuai ukuran patch. Pembobotan ini berguna untuk menskala ulang kontribusi *magnitude*, semakin jauh dari pusat *keypoint* maka semakin kecil kontribusinya dan begitupula sebaliknya.

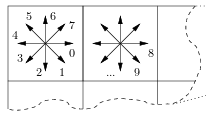
### Perhitungan Histogram pada Local Keypoint Window

Pada orientasi dan magnitude masing-masing blok akan dibentuk histogram 360 derajat arah dengan 360 bin dengan magnitude sebagai nilai yang dimasukkan pada masing-masing bin orientasi. Kemudian histogram dihitung ulang untuk 8 bin orientasi, sehingga pada sebuah patch berukuran 24x24 terdapat 4x4 blok yang masing-masing memiliki deskriptor berupa histogram magnitude pada 8 arah orientasi. Ilustrasi diberikan pada Gambar 3.8.

Sumber : SIFT : Generating a feature, aishack.in

**Gambar 3.8 Ilustrasi Patch dan Sub-window**

### Penyusunan Nilai Histogram

Pada proses ini nilai histogram pada setiap blok kemudian disusun sebagai representasi akhir deskriptor. Terdapat perbedaan penyusunan deskriptor antara *toolbox ­vl\_feat* dan algoritma asalnya milik David G. Lowe. *vl\_feat* menyusun nilai histogram searah jarum jam, sedangkan Lowe sebaliknya [13]. Ilustrasi proses penyusunan diberikan pada Gambar 3.9.

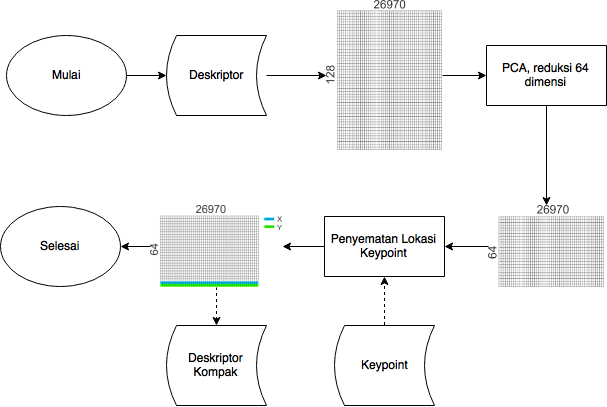
Gambar 3.9 Penyusunan Nilai Histogram Pada toolbox vl\_feat

### Penyiapan Fitur

Setelah melalui proses ekstraksi fitur, maka akan ter-ekstraksi 26,970 keypoint yang masing-masing memiliki deskriptor sepanjang 128 dimensi untuk satu citra. Pada proses ini masing-masing vector deskriptor akan dinormalisasi dengan persamaan (3.1). Kemudian dimensi deskriptor akan dipotong menjadi 64 dan dicantumkan lokasi keypoint (x,y) sebelum nantinya memasuki proses pembelajaran *Gaussian Mixture Model*. Ilustrasi proses ini diberikan pada gambar Gambar 3.10.

|  |  |
| --- | --- |
|  | **(3.1)** |
| **Keterangan:** | |
| : Fungsi normalisasi besaran vektor | |

### Kompleksitas Ekstraksi Fitur

Permasalahan yang muncul pada ekstraksi fitur dense SIFT adalah besarnya memori yang dibutuhkan untuk mengekstraksi fitur seluruh image. Dataset LFW memiliki rekomendasi untuk melakukan pengembangan software dengan mengesktrak fitur dari 10 fold yang masing-masing terdiri dari 600 pasangan yang telah digenerasi secara random. Namun karena besarnya memori yang dibutuhkan maka penulis hanya melakukan ekstraksi pada 100 pasangan dalam 4 folds.

Gambar 3.10 Ilustrasi Penyiapan Fitur

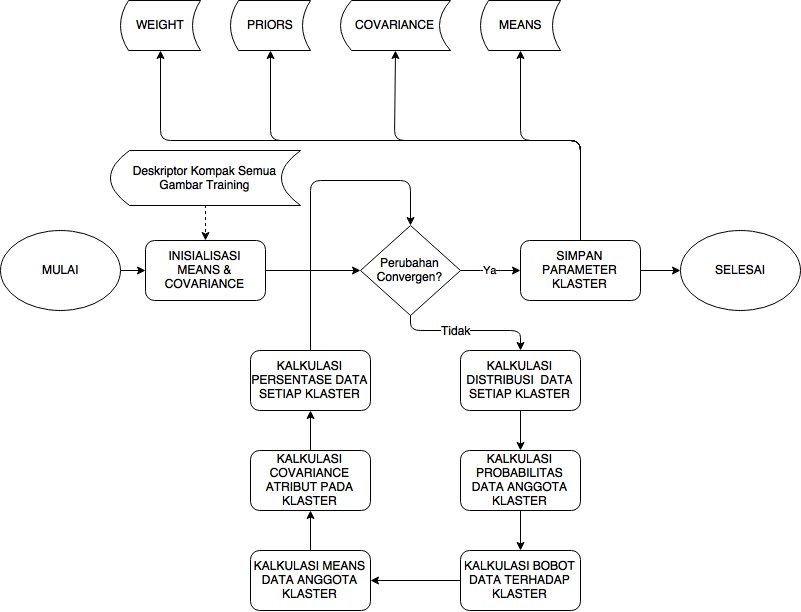
Untuk penjelasan lebih lengkap Gambar 3.11 memberikan informasi kompleksitas memori yang digunakan dalam ekstraksi fitur. Untuk setiap pasang citra (2n) akan diekstrak 5394 keypoint dalam 5 skala yang masing-masing memiliki vektor fitur sepanjang 66, lalu matriks dengan ukuran tersebut dikalikan 4 byte untuk tipe data *single* di matlab.

Grafik tersebut memberikan informasi penggunaan memori ketika sudah berhasil melakukan ekstraksi fitur, pada prosesnya sebenarnya setiap vector descriptor berukuran 128, dan menjadi 64 setelah di PCA, kemudian menjadi 66 dengan informasi lokasi. Ketika melakukan PCA menggunakan matlab, memori yang digunakan bisa naik dua kali lipat.

## Gaussian Mixture Model

Gambar 3.11 Kompleksitas Memori Ekstraksi Fitur

Deskriptor kompakyang didapatkan dari proses sebelumnya akan dipelajari kepadatan gaussiannya dengan *Gaussian Mixture Model* (GMM). GMM berfungsi untuk mengelompokkan kumpulan *descriptor* wajah yang redundan kedalam *k* kelompok distribusi Gaussian. Klaster Gaussian menjelaskan titik-titik pengamatan pada sebuah wajah secara umum, yang dijelaskan oleh *means, covariance, weight, dan prior*. *Weight* menjelaskan seberapa penting sebuah data terhadap klaster tertentu, sedangkan *prior* menjelaskan presentase data yang termasuk kedalam anggota klaster tersebut. GMM dilatih menggunakan algoritma *Expectation Maximization* yang dijelaskan melalui diagram alir pada Gambar 3.12.

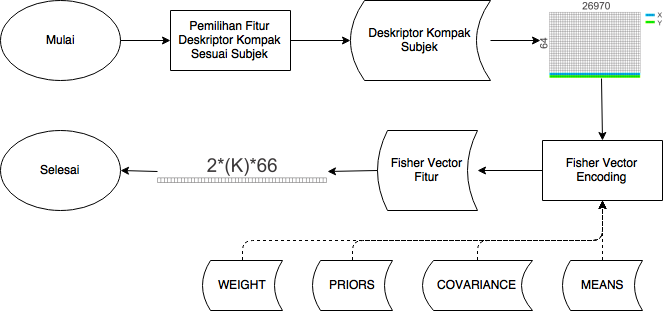
Kalkulasi distribusi data setiap klaster diberikan pada rumus (2.14). Kalkulasi probabilitas sebuah data merupakan anggota dari klaster diberikan pada rumus (2.15). Kalkulasi bobot pentingnya sebuah data terhadap klaster diberikan pada rumus (2.16). Kalkulasi *means* data anggota klaster diberikan pada rumus (2.17). Kalkulasi *covariance* masing-masing atribut pada klaster tertentu diberikan pada rumus (2.18). Sedangkan kalkulasi persentase banyaknya data anggota klaster diberikan pada rumus (2.19).

**Gambar 3.12 Diagram Alir Algoritma EM**

## Fisher Vector Encoding

*Fisher vector* encoding berguna untuk merepresentasikan ulang deskriptor kompak pada proses sebelumnya, sehingga satu gambar akan direpresentasikan dengan satu vector saja. Apabila kita mengacu pada proses GMM pada persamaan (2.14) sampai dengan (2.19), maka descriptor SIFT yang berjumlah *N x* 128 akan dirubah menjadi K distribusi Gaussian yang masing-masing berisi , dan berdimensi . *Fisher Vector* merupakan kumpulan nilai dan dari semua komponen gaussian. *Fisher Vector* inilah yang akan menjadi representasi fitur akhir dari sebuah citra wajah sebagaimana dijelaskan pada persamaan (3.1). Ilustrasi perubahan matriks deskriptor pada Fisher Vector Encoding diberikan pada gambar Gambar 3.13.

|  |  |
| --- | --- |
|  | **(3.1)** |
| **Keterangan:** | |
| : Fisher Vector | |
| : encoded vector pada turunan ke *i* dari komponen Gaussian *j* | |



Gambar 3.13 Ilustrasi Perubahan Deskriptor

## Proses Metric Learning

Pembelajaran bobot dan *threshold* memiliki tujuan utama untuk memaksimalkan fungsi objektif pada persamaan (3.2)

|  |  |
| --- | --- |
|  | (3.2) |
|  |  |
| Keterangan: | |
| : status pasangan gambar (sama = 1 / tidak = -1) | |
| : threshold kesamaan pasangan gambar | |
| : vector fitur pasangan gambar | |

Fungsi objektif tersebut menggambarkan bahwa untuk meminimalkan error bobot maka pada setiap iterasi sistem diharuskan memaksimalkan value sehingga grafik *Stochastic* *Gradient Descent* (SGD) nya selalu naik. Namun, simonyan menyebutkan bahwa optimasi menggunakan metode low-rank *Mahalanobis Metric Learning* merupakan kasus optimasi *Non-Convex* [2], sehingga dalam setiap iterasi nilai tidak selalu mendekati minimum, dan error minimum yang dicapai bisa jadi bukanlah *global minimum*.

Lebih lanjut pembelajaran dilakukan secara iteratif selama 10,000 iterasi dengan memperbarui bobot dan *threshold* secara bersamaan apabila kondisi tertentu tercapai sesuai dengan persamaan (3.3) dan (3.4).

|  |  |
| --- | --- |
|  | (3.3) |
|  |
|  |  |
| Keterangan: | |
| : Bobot baru | |
| : Bobot sebelumnya | |
| : status pasangan gambar +1 jika sama, -1 jika berbeda | |
| : threshold jarak | |
| : learning rate dengan nilai 0.001 | |
| : Fisher vector | |

|  |  |
| --- | --- |
|  | (3.4) |
| Keterangan: | |
| : Threshold jarak baru | |
| : Threshold jarak sebelumnya | |
| : status pasangan gambar +1 jika sama, -1 jika berbeda | |
| : learning rate dengan nilai 0.001 | |
| : Fisher vector | |

Jumlah iterasi tersebut telah dibatasi dari jumlah yang disarankan yakni 1,000,000 iterasi karena keterbatasan waktu. Selain itu penulis juga melakukan pembatasan pada jumlah K klaster yang digunkan untuk memetakan fitur yang telah diekstraksi. Hal ini dikarenakan kompleksitas memori yang diperlukan sangat bergantung pada jumlah K yang digunakan.

Sebagaimana telah dijelaskan pada sub-bab 2.7, maka untuk memulai *low-rank Mahalanobis Metric Learning* diperlukan inisialisasi matriks *M*. Matriks *M* diinisialisasi dengan *Possitive Semi Definite Matriks* (PSD) berukuran (2\*K\*fitur\_vector)2\*8 byte. Karena dalam membuat PSD diperlukan proses SVD dan pengecekan eigen value, maka ukuran tersebut dikali 4. Gambar 3.14 memberikan informasi lebih lengkap terkait variasi K dan dampaknya terhadap besaran memori.

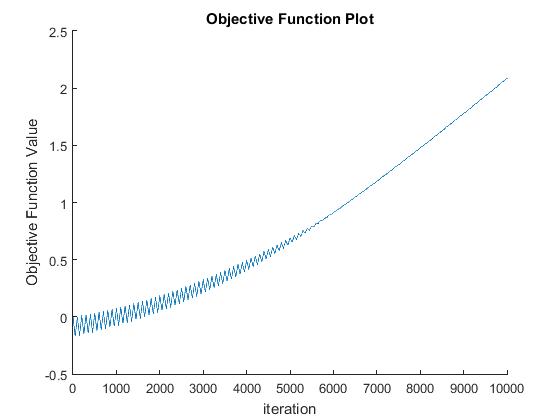
Gambar 3.14 memberikan informasi memori minimum yang dibutuhkan untuk membuat matriks PSD. Namun pada proses yang sebenarnya pengecekan *eigen value* menggunakan matlab setidaknya memerlukan 2.5 kali dari grafik yang sudah ditunjukkan. Oleh Karena itu jumah K dibatasi pada 128 dan 192.

## F:\Dropbox\[PENTING TIDAK URGENT]\[ARSIP KULIAH]\Semester 9\#TUGAS_AKHIR\Apps\[GIT]\Buku\gambar\psd_process.jpgEvaluasi Proses Training

**Gambar 3.14 Kompleksitas Memori Proses Inisialisasi PSD Matriks**

Metode *low-rank Mahalanobis Metric learning* ini ditujukan untuk mencari model bobot terbaik dan *threshold* untuk menyatakan jarak kesamaan sebuah pasangan wajah. Evaluasi proses training diperlukan untuk memastikan apakah proses berjalan dengan baik ataukah tidak. *Stocastic Gradient Descent* dengan fungsi objektif sebagai yang dijelaskan pada persamaan (3.2) dapat digunakan sebagai metode evaluasi proses training. Rumusan SGD sendiri dijelaskan pada persamaan (2.26).

Penelitian simonyan [2] menyebutkan bahwa metode *low-rank Mahalanobis Metric Learning* merupakan kasus optimasi *Non-Convex*. Gambar 3.15 memberikan informasi histori nilai objective function pada salah satu fold untuk setiap iterasi.

Dari grafik tersebut dapat diperoleh informasi bahwa ketika melakukan *low-rank Mahalanobis Metric Learning* fungsi objektif tidak selalu mendapatkan nilai positif. Hal ini mengindikasikan bahwa ****pada awal iterasi (0-5500) sistem salah mengklasifikasikan pasangan gambar. Namun seiring berjalan iterasi, bobot w dan threshold b diperbarui menyebabkan *error* semakin mengecil hingga pada akhir iterasi (>5500) hanya menyisakan sedikit error pada sebagian pasangan gambar training.

**Gambar 3.15 Histori Objective Function Value**

## Pengukuran Jarak dan Klasifikasi

Pada pengujian, *fisher vector* dan yang telah diekstraksi dari pasangan gambar akan dikalikan dengan matriks hasil *mahalanobis metric learning* (yang telah dijelaskan pada sub-bab sebelumnya*.* *Threshold b* juga digunakan untuk proses klasifikasi.

Pada persamaan (3.5) apabila nilai *distance* kurang dari *threshold b*maka kedua gambar adalah orang yang sama (kelas positif), namun jika muncul kondisi yang sebaliknya maka kedua gambar adalah orang yang berbeda (kelas negatif).

|  |  |
| --- | --- |
|  | **(3.5)** |
|  |
| **Keterangan:** | |
| : Jarak dari dua Fisher Vector | |
| : bobot jarak yang telah dipelajari | |
| : fungsi threshold jarak | |
| ***b*** : threshold kesamaan jarak | |

## Perhitungan Akurasi

Pada proses testing akan dilakukan pengecekan keputusan algoritma terhadap pasangan gambar, kemudian hasilnya akan dicatat pada laporan evaluasi. Perhitungan hasil hanya dilakukan pada akurasi dengan persamaan (3.6).

|  |  |
| --- | --- |
|  | (3.6) |
| Keterangan: | |
| : true positive, pasangan benar dan dideteksi benar | |
| : true negative, pasangan salah namun dideteksi benar | |
| : Jumlah semua percobaan yang dilakukan | |

Dikarenakan proses learning yang digunakan cukup lama, maka penulis memutuskan untuk melakukan checkpoint testing, dimana pada sejumlah n iterasi training akan dibagi kedalam 100 checkpoint untuk testing menggunakan parameter w dan threshold b yang didapatkan.

***[Halaman ini sengaja dikosongkan]***

# BAB IV IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Implementasi kode program dilakukan sepenuhnya menggunakan bahasa Matlab.

## Lingkungan implementasi

Spesifikasi perangkat keras dan perangkat lunak yang digunakan ditampilkan pada Tabel 4.1.

Tabel 4.1 Lingkungan implementasi Perangkat Lunak

|  |  |
| --- | --- |
| Perangkat | Spesifikasi |
| Perangkat keras | Prosesor: Intel® Xeon™ E5-2630L CPU @ 2.00 GHz  Memori: 64.00 GB |
| Perangkat lunak | Sistem Operasi: Ubuntu 14.04.4 LTS  Perangkat Pengembang: Matlab  Perangkat Pembantu:  Sublime, Visual Code, vl\_feat |

## Implementasi

Sub-bab implementasi ini menjelaskan tentang implementasi proses yang sudah dijelaskan pada bab desain perangkat lunak.

### Implementasi Tahap Pre-processing

Sub bab ini menjelaskan tentang implementasi tahapan *pre-processing* pada Kode Sumber 4.1. Tahap pre-processing terbagi dalam deteksi wajah menggunakan metode *Viola-Jones, cropping,* serta *resize* Sebelum dilakukan deteksi wajah terlebih dahulu citra dikonversi menjadi citra *grayscale* mengunakan fungsi *rgb2gray()*. *vision.CascadeObjectDetector()* adalah fungsiuntuk mendeteksi wajah dengan metode *Viola-Jones*. Sedangkan *step()* merupakan fungsi untuk mendapatkan *Region of Interest* (ROI) wajah dari detektor diatas. Kedua fungsi tersebut telah disediakan matlab.

Pada baris ke 5 sampai dengan 8, kode melakukan pemilihan pada area wajah yang paling besar apabila wajah yang terdeteksi ada lebih dari satu (fokus objek pada wajah terbesar). Sisanya, program akan otomatis mendeteksi ¼ ukuran gambar dari tengah sebagai wajah. apabila *Viola-Jones* gagal mendeteksi wajah.

Kode Sumber 4.1 Preprocessing deteksi dan resize wajah

1. **function** imageProcessed = preprocess(image)
2. imageBw = rgb2gray(image);
3. faceDetector = vision.CascadeObjectDetector();
4. boxRoi = step(faceDetector, imageBw);
5. **if** size(boxRoi,1)>1
6. [val, pos] = sort(boxRoi(:,4),'descend');
7. boxRoi = boxRoi(pos(1),:);
8. disp(boxRoi);
9. **else** **if** size(boxRoi,1)==0
10. start = round(size(imageBw)/4);
11. big = start+start;
12. boxRoi=[start,big];
13. **end**
14. **end**
15. imageProcessed = imcrop(image, boxRoi);
16. imageProcessed = imresize(imageProcessed,[160,125]);
17. **end**

### Implementasi Ekstraksi Fitur dense SIFT

Sub bab ini menjelaskan tentang implementasi tahapan ekstraksi fitur dense SIFT pada Kode Sumber 4.2. Sebelum dapat diekstrak, image terlebihdahulu ditransformasi menjadi citra *grayscale,* lalu kemudian dilakukan konversi menjadi tipe data float 32-bit (*single())*. kemudian citra di smoothing lima kali lalu diekstrak fitur dense SIFT nya menggunakan fungsi *vl\_dsift()*.

Dari lima level tersebut akan didapatkan fitur dengan dimensi sekitar 26.000x128. setiap vektor descriptor sepanjang 128 kemudian dinormalisasi menggunakan l2 *normalization*. Setelah itu dilakukan PCA untuk mengurangi dimensinya menjadi 64. Kemudian bila opsi menggunakan informasi lokal maka disematkan lokasi x,y disetiap vektor fitur.

Kode Sumber 4.2 Ekstraksi Fitur dense SIFT

1. **function** [dess] = getDsift(image, jenis)
2. I = rgb2gray(image);
3. power = [1:5];
4. sigmas = 1.6\*(2.^(0.5.\*power));
5. dess={};
6. **for** i=1:length(sigmas)
7. height = size(I,1); width = size(I,2);
8. blockSize=6;
9. Is = vl\_imsmooth(single(I), sigmas(i));
10. bound = [25,25,100,135];
11. [kp, des]=vl\_dsift(Is, 'Size', blockSize, 'Step', 1, 'Bounds', bound);
12. **for** j=1:size(des,2)
13. vector = des(:,j);
14. vector = single(vector)/norm(single(vector));
15. vector = vector.^0.5;
16. des(:,j)=vector;
17. **end**
18. des=pca(single(des),'NumComponents',64);
19. **if** jenis==1
20. x=kp(1,:); y=kp(2,:); x=(x/width)-0.5; y=(y/height)-0.5;
21. x=x'; y=y';
22. des=horzcat(des,x); des=horzcat(des,y);
23. **end**
24. dess=vertcat(dess,des);
25. **end**
26. **end**

### Implementasi Preparasi Dataset

Pada Kode Sumber 4.3 dibahas preparasi dataset. Preparasi dilakukan dengan tujuan untuk mengumpulkan list semua lokasi image dari list nama gambar yang digunakan sebagai pasangan positif dan negatif dari dataset. Untuk fungsi zeroPad dipaparkan pada lampiran Lampiran 7.1 zero padding untuk nama gambar

Kode Sumber 4.3 Preparasi Dataset

1. **function** allSets = allSetsPrep(matPath)
2. **if** exist(strcat(matPath,strcat('\','[allSets].mat')))
3. allSets = load(strcat(matPath,strcat('\','[allSets].mat')));
4. allSets = allSets.allSets;
5. **else**
6. pairsList = 'pairs4folds.txt';
7. fileId = fopen(pairsList,'r');
8. line = fgetl(fileId);
9. allSets = {};
10. **while** ischar(line)
11. val = strsplit(line,'\t');
12. **if** length(val)==2
13. disp('skip header');
14. **elseif** length(val)==3
15. name = char(val{1}); numb\_1 = char(val{2}); numb\_2 = char(val{3});
16. len\_numb\_1 = 4-length(numb\_1);len\_numb\_2 = 4-length(numb\_2);
17. numb\_1 = zeroPad(numb\_1, len\_numb\_1);numb\_1=strcat('\_',numb\_1);
18. numb\_2 = zeroPad(numb\_2, len\_numb\_2);numb\_2=strcat('\_',numb\_2);
20. name\_a = strcat(name,strcat(numb\_1,'.jpg'));
21. name\_b = strcat(name,strcat(numb\_2,'.jpg'));
22. set = {name\_a,name\_b}; allSets=vertcat(allSets, set);
23. **elseif** length(val)==4
24. name\_1 = char(val{1});numb\_1 = char(val{2});
25. name\_2 = char(val{3});numb\_2 = char(val{4});
26. len\_numb\_1 = 4-length(numb\_1);len\_numb\_2 = 4-length(numb\_2);
27. numb\_1 = zeroPad(numb\_1, len\_numb\_1);numb\_1=strcat('\_',numb\_1);
28. numb\_2 = zeroPad(numb\_2, len\_numb\_2);numb\_2=strcat('\_',numb\_2);
29. name\_a = strcat(name\_1,strcat(numb\_1,'.jpg'));
30. name\_b = strcat(name\_2,strcat(numb\_2,'.jpg'));
31. set = {name\_a,name\_b}; allSets=vertcat(allSets, set);
32. **end**
33. line=fgetl(fileId);
34. **end**
35. save(strcat(matPath,strcat('\','[allSets].mat')),'allSets','-v7.3');
36. **end**
37. **end**

### Implementasi Pengumpulan Fitur dense SIFT

Pada Kode Sumber 4.4 dibahas pengumpulan fitur dense SIFT dari semua list citra yang akan digunakan sebagai dataset. Penyimpanan dilakukan dalam sebuah Map Key-Value. Map berguna untuk menyimpan hasil ekstraksi fitur secara sistematis berdasarkan nama orang yang citranya diekstraksi fitur.

Kode Sumber 4.4 Pengumpulan Fitur dense SIFT

1. **function** [allSets, allFeatures] = featureExtractPairs(jenis, matPath, datasetPath)
2. allSets = allSetsPrep(matPath);
3. datasetPath = strcat(datasetPath,'/lfw/');
4. keys={}; values={};
5. **for** i=1:length(allSets)
6. file\_a=allSets{i,1};
7. file\_b=allSets{i,2};
8. dir\_a = strsplit(file\_a,'.'); dir\_a = dir\_a{1};
9. dir\_b = strsplit(file\_b,'.'); dir\_b = dir\_b{1};
10. dir\_a = strsplit(dir\_a,'\_'); dir\_a = {dir\_a{1:length(dir\_a)-1}};
11. disp(i);
12. **for** l=1:length(dir\_a)
13. **if** l==1
14. name\_a = dir\_a{l};
15. **else**
16. name\_a = strcat(name\_a, '\_');
17. name\_a = strcat(name\_a, dir\_a{l});
18. **end**
19. **end**
20. dir\_b = strsplit(dir\_b,'\_'); dir\_b = {dir\_b{1:length(dir\_b)-1}};
21. **for** l=1:length(dir\_b)
22. **if** l==1
23. name\_b = dir\_b{l};
24. **else**
25. name\_b = strcat(name\_b, '\_');
26. name\_b = strcat(name\_b, dir\_b{l});
27. **end**
28. **end**
29. get\_a = strcat(datasetPath,strcat(name\_a,strcat('/',file\_a)));
30. get\_b = strcat(datasetPath,strcat(name\_b,strcat('/',file\_b)));
31. image\_a = imread(get\_a); image\_a = preprocess(image\_a); des\_a = getDsift(image\_a, jenis);
32. image\_b = imread(get\_b); image\_b = preprocess(image\_b); des\_b = getDsift(image\_b, jenis);
33. keys=vertcat(keys, file\_a); values=vertcat(values, {des\_a});
34. keys=vertcat(keys, file\_b); values=vertcat(values, {des\_b});
35. **end**
36. allFeatures = containers.Map(keys,values);
37. disp('saving allFeatures...'); save(strcat(matPath,'/[allFeatures].mat'),'allFeatures','-v7.3');
38. **end**

### Implementasi Pembagian Fold Dataset

Pada Kode Sumber 4.5 akan dibuat variabel *folds\_idx* yang berisi nama pasangan citra testing dan training dalam setiap folds. Dengan 400 total dataset dan 4 folds validation, maka setiap fold terdiri dari 100 pasang citra test dan 300 pasang citra train. Nama-nama ini kemudian akan digunakan sebagai parameter untuk memanggil fitur pada key-value map fitur kemudian.

Kode Sumber 4.5 Pembagian Fold Dataset

1. **function** folds\_idx = dataPrep(allSets)
2. bound = zeros(1,4);
3. iter\_bound=1;
4. **for** i=100:100:400
5. bound(iter\_bound)=i; iter\_bound=iter\_bound+1;
6. **end**
7. bounds=cell(1,4);
8. **for** i=1:length(bound)
9. **if** i==1
10. val=[1:bound(i)];
11. **elseif** i==length(bound)
12. val=[bound(length(bound)-1)+1:bound(length(bound))];
13. **else**
14. val=[bound(i-1)+1:bound(i)];
15. **end**
16. bounds{i}=val;
17. **end**
18. pair\_name=cell(4,1);
19. **for** i=1:length(bounds)
20. a={allSets{bounds{i},1}}';
21. b={allSets{bounds{i},2}}';
22. pair\_name{i}=horzcat(a,b);
23. **end**
24. pair\_folds=cell(4,1);
25. **for** i=1:length(pair\_folds)
26. **for** j=1:length(pair\_name)
27. **if** j==1
28. test\_name = pair\_name{j};
29. train\_idx = [j+1:length(pair\_name)];
30. train\_name={};
31. train\_name = vertcat(train\_name,pair\_name{train\_idx});
32. **elseif** j==length(pair\_name)
33. test\_name = pair\_name{j};
34. train\_idx = [1:length(pair\_name)-1];
35. train\_name={};
36. train\_name = vertcat(train\_name, pair\_name{train\_idx});
37. **else**
38. test\_name = pair\_name{j};
39. train\_idx = [1:j-1,j+1:length(pair\_name)];
40. train\_name={};
41. train\_name = vertcat(train\_name, pair\_name{train\_idx});
42. **end**
43. **end**
44. pair\_folds{i}={test\_name, train\_name};
45. **end**
46. folds\_idx = pair\_folds;
47. clear pair\_folds;
48. **end**

### Implementasi Pengambilan Fitur

Proses pengambilan fitur pada Kode Sumber 4.6 berguna untuk memanggil fitur yang besesuaian dengan nama datasetnya. Hal ini digunakan untuk menghemat waktu agar tidak perlu mengulang ekstraksi fitur pada proses-proses berikutnya.

Kode Sumber 4.6 Pengambilan Fitur

1. **function** train\_des = retrieve\_feature(train\_des\_name, matPath)
2. allFeatures = load(strcat(matPath,strcat('\','[allFeatures].mat')));
3. allFeatures = allFeatures.allFeatures;
4. train\_des={};
5. **for** i=1:length(train\_des\_name)
6. disp(i);
7. name\_a=train\_des\_name{i,1};
8. name\_b=train\_des\_name{i,2};
9. des\_a = allFeatures(name\_a);
10. des\_b = allFeatures(name\_b);
11. train\_des = vertcat(train\_des, des\_a);
12. train\_des = vertcat(train\_des, des\_b);
13. **end**
14. **end**

### Implementasi Learning Gaussian Mixture Model

Pada masing-masing folds nantinya akan dilakukan training Gaussian Mixture Model (GMM) sesuai Kode Sumber 4.7. Learning ini berguna untuk secara otomatis meng-kluster fitur hasil ekstraksi menjadi K kelompok. Keluaran dari proses ini adalah mean, covariance matrix, dan priors dari K cluster dari seluruh fitur yang dilearning. Fitur yang dilakukan GMM learning hanya pada fitur training saja (menjadi kluster fitur-fitur penting pada wajah secara umum).

Kode Sumber 4.7 Learning Gaussian Mixture Model

1. **if** exist(strcat(matPath,strcat('\','[gmm\_param\_folds].mat')))
2. mean\_cov\_priors = load(strcat(matPath,strcat('/','[gmm\_param\_folds].mat')));
3. mean\_cov\_priors = mean\_cov\_priors.mean\_cov\_priors;
4. K=input('cluster for gmm? (512/256/128) ');
5. **else**
6. meancovar\_names={};
7. K=input('cluster for gmm? (512/256/128) ');
8. **for** i=1:length(folds\_idx)/2
9. disp(strcat('folds ke ',num2str(i)));
10. train\_des\_name = folds\_idx{i}{2};
11. train\_des=retrieve\_feature(train\_des\_name, matPath);
12. train\_des=cell2mat(train\_des);
13. train\_des = train\_des';
14. disp('learning gmm...');
15. [means, covariances, priors]=vl\_gmm(train\_des,K, 'verbose');
16. mean\_covar\_priors={means, covariances, priors};
17. filename = strcat('[mean\_cov\_priors\_',strcat(num2str(i),'].mat'));
18. disp('temporarily save...');
19. meancovar\_names=vertcat(meancovar\_names,filename);
20. save(strcat(matPath,strcat('/',filename)),'mean\_covar\_priors','-v7.3');
21. **end**
22. **end**
23. disp('saving all gmm parameters as one and clearing the temporal save...');
24. mean\_cov\_priors={};
25. **for** j=1:length(meancovar\_names)
26. name = meancovar\_names{j};
27. meancovar = load(strcat(matPath,strcat('/',name)));
28. meancovar = meancovar.meancovar;
29. mean\_cov\_priors=vertcat(mean\_cov\_priors, meancovar);
30. **end**
31. save(strcat(matPath,strcat('\','[gmm\_param\_folds].mat')),'mean\_cov\_priors','-v7.3');
32. delete(meancovar\_names);

### Implementasi Fisher Vector Encoding

Fisher Vector Encoding berguna untuk merepresentasikan fitur yang telah diekstraksi kedalam bentuk yang lebih sederhana, dari yang mulanya 26.000x66 pada setiap gambar menjadi 2xKx66 pada setiap gambar. Disesuaikan dengan model umum cluster fitur wajah yang penting pada GMM, diberikan pada Kode Sumber 4.8.

Kode Sumber 4.8 Fisher Vector Encoding

1. **function** encoded = fisher\_encode(test\_folds, meancovar)
2. mean = meancovar{1,1};
3. covariance = meancovar{1,2};
4. prior = meancovar{1,3};
5. disp('encoding...');
6. encoded={};
7. **for** i=1:length(test\_folds)
8. dataToEncode = test\_folds{i};
9. dataToEncode = cell2mat(dataToEncode);
10. dataToEncode = dataToEncode';
11. fv\_vector = vl\_fisher(dataToEncode, mean, covariance, prior);
12. encoded = vertcat(encoded, fv\_vector);
13. **end**
14. **end**

### Implementasi Pengumpulan Fisher Vector Pada Data Test dan Data Train

Setiap fold pada data training dan testing kemudian diencoding dan dikumpulkan sesuai dengan urutan fold nya dengan mengimplementasikan Kode Sumber 4.9.

Kode Sumber 4.9 Pengumpulan Fisher Vector pada Data Test dan Data Train

1. TEST\_FOLDS={};
2. meancovar = load(strcat(matPath,'\[gmm\_param\_folds].mat'));
3. meancovar = meancovar.mean\_cov\_priors;
4. **for** i=1:length(folds\_idx)
5. gmmparam = {meancovar{i,:}};
6. test\_folds\_name = folds\_idx{i}{1};
7. disp('retrieving test folds original feature..');
8. test\_folds = retrieve\_feature(test\_folds\_name, matPath);
9. test\_folds = fisher\_encode(test\_folds, gmmparam);
10. TEST\_FOLDS = vertcat(TEST\_FOLDS,{test\_folds});
11. **end**
12. **for** i=1:length(TEST\_FOLDS)
13. target\_size=[100,2];
14. TEST\_FOLDS{i} = reshape\_cell(TEST\_FOLDS{i}, target\_size);
15. **end**
16. TRAIN\_FOLDS={};
17. meancovar = load(strcat(matPath,'\[gmm\_param\_folds].mat'));
18. meancovar = meancovar.mean\_cov\_priors;
19. **for** i=1:length(folds\_idx)
20. gmmparam = {meancovar{i,:}};
21. train\_folds\_name = folds\_idx{i}{2};
22. disp('retrieving train folds original feature..');
23. train\_folds = retrieve\_feature(train\_folds\_name, matPath);
24. train\_folds = fisher\_encode(train\_folds, meancovar);
25. TRAIN\_FOLDS = vertcat(TRAIN\_FOLDS,{train\_folds});
26. **end**
27. **for** i=1:length(TRAIN\_FOLDS)
28. target\_size=[300,2];
29. TRAIN\_FOLDS{i} = reshape\_cell(TRAIN\_FOLDS{i}, target\_size);
30. **end**

### Implementasi inisialisasi matrix PSD, PCA, dan Whitening

Proses ini dilakukan dalam rangka menginisialisasi learning pada weight w. learning diinisialisasi dengan membuat matrix *Possitive Semi Definite (PSD)* dengan ukuran *mxm* dan m=panjang fisher vector sesuai dengan implementasi pada Kode Sumber 4.10. PSD matrix adalah random matrix yang semua *eigen vector* nya lebih dari atau sama dengan nol.

Karena learning dengan menggunakan matriks PSD cukup lama, maka dilakukan proyeksi menjadikannya matriks w dengan dimensi *pxm*, dengan p=128. Proyeksi dapat dilakukan dengan melakukan PCA, kemudian terakhir dilakukan whitening untuk mendekorelasi matrix.

Kode Sumber 4.10 Inisialisasi matrix PSD, PCA, dan Whitening

1. **function** M = psd(n)
2. A = randn(n);
3. [U,ignore] = eig((A+A')/2);
4. M = U\*diag(abs(randn(n,1)))\*U';
5. pass = all(eig((M+M')/2)) >= 0;
6. **if** pass==1
7. disp('psd matrix created');
8. **else**
9. disp('non psd matrix created');
10. **end**
11. **end**
12. **function** xPCAwhite = pca\_whiten(x,k)
13. epsilon = 0.00001;
14. avg = mean(x, 1);
15. x = x - repmat(avg, size(x, 1), 1);
16. sigma = x \* x' / size(x, 2);
17. [U,S,V] = svd(sigma);
18. xRot = U' \* x;
19. xPCAwhite = diag(1./sqrt(diag(S) + epsilon)) \* xRot;
20. xPCAwhite = xPCAwhite(1:k,:);
21. **end**

### Implementasi Pengulangan Fitur Training

Training w dan b tidak dilakukan sampai menyentuh threshold tertentu, melainkan dilakukan sebanyak M kali. Maka data train diulang sehingga mencapai M buah data train seperti pada implementasi pada Kode Sumber 4.11.

Kode Sumber 4.11 Pengulangan Fitur Training

1. **function** [target, names] = itertools (curent\_list, folds\_idx\_names, stoping\_target)
2. stoping\_target = stoping\_target+4;
3. max\_curent\_list = length(curent\_list);
4. times = round(stoping\_target / max\_curent\_list);
5. target\_list = repmat(curent\_list,[times, 1]);
6. names\_list = repmat(folds\_idx\_names, [times,1]);
7. **if** length(target\_list)~=stoping\_target
8. selisih = stoping\_target - length(target\_list);
9. **if** selisih>0
10. tambahan = curent\_list(1:selisih,1:2);
11. tambahan\_names = names\_list(1:selisih,1:2);
12. target\_list = vertcat(target\_list, tambahan);
13. names\_list = vertcat(names\_list, tambahan\_names);
14. names = names\_list; clear names\_list;
15. target = target\_list; clear target\_list;
16. **else**
17. last = length(target\_list); start = (last+selisih);
18. target = cell(start,2); names = cell(start,2);
19. **for** i=1:start
20. target{i,1} = target\_list{i,1};target{i,2} = target\_list{i,2};
21. names{i,1} = names\_list{i,1};names{i,2} = names\_list{i,2};
22. **end**
23. clear target\_list; clear names\_list;
24. **end**
25. disp(size(target)); disp(size(names));
26. **else**
27. target = target\_list; clear target\_list;
28. names = names\_list; clear names\_list;
29. disp(size(target)); disp(size(names));
30. **end**
31. **end**

### Implementasi Tester Kesamaan Nama

Fungsi pada Kode Sumber 4.12 digunakan untuk menentukan apakah sebuah pasangan nama sama atau tidak.

Kode Sumber 4.12 Tester Kesamaan Nama

1. **function** keputusan = yij\_decider(name\_a, name\_b)
2. name\_a = strsplit(name\_a, '.'); name\_b = strsplit(name\_b, '.');
3. name\_a = name\_a{1}; name\_b = name\_b{1};
4. name\_a = strsplit(name\_a, '\_'); name\_b = strsplit(name\_b, '\_');
5. last\_a = length(name\_a); pick\_a = last\_a-1;
6. last\_b = length(name\_b); pick\_b = last\_b-1;
7. name\_a = {name\_a{1:pick\_a}}; name\_b = {name\_b{1:pick\_b}};
8. temp\_a = '';
9. **for** i=1:length(name\_a)
10. temp\_a = strcat(temp\_a, name\_a{i});
11. **end**
12. temp\_b = '';
13. **for** i=1:length(name\_b)
14. temp\_b = strcat(temp\_b, name\_b{i});
15. **end**

### Implementasi Perhitungan Jarak

Perhitungan jarak yang diimplementasikan pada Kode Sumber 4.13 adalah mahalanobis distance, dengan matriks berukuran *mxm* dihasilkan dari pengalian *w’\*w*.

Kode Sumber 4.13 Perhitungan Jarak

1. **function** [dw\_theta, theta\_diff] = dw(theta\_i, theta\_j, w)
2. theta\_diff = theta\_i - theta\_j;
3. dw\_theta = sqrt(theta\_diff'\*(w'\*w)\*theta\_diff);
4. **end**

### Implementasi Update b dan w

Update dilakukan setiapkali *test\_value* <1 seperti pada Kode Sumber 4.14.

Kode Sumber 4.14 Update b dan w

1. **function** [w\_new,b\_new] = bw\_updater(w, b, gamma, yij, psi)
2. w\_new = w - gamma\*yij\*w\*psi;
3. b\_new = b + gamma\*yij\*b;
4. **end**

### Implementasi Learning w dan b Serta Penyimpanan Hasil

Pada potongan Kode Sumber 4.15 dilakukan learning w dan b secara kesuluruhan pada masing-masing folds. Dan dicatat semua hasil performa, waktu, dan histori fungsi objektif.

Kode Sumber 4.15 Learning w dan b serta penyimpana hasil

1. all\_train\_time={};
2. all\_test\_time={};
3. all\_objectivefunc\_history={};
4. all\_grad\_history = {};
5. all\_roc\_score={};
6. all\_b={};
7. **for** i=1:length(TRAIN\_FOLDS)
8. time\_train\_start = tic();
9. **if** jenis==1
   1. fvsize = 2\*K\*66;
10. **else**
    1. fvsize = 2\*K\*64;
11. **end**
12. name\_w=strcat('[w',strcat(num2str(i),'].mat'));
13. w = load(strcat(matPath,strcat('\',name\_w)));
14. w = w.w;
15. **if** size(w,1)==128
16. disp('w ready');
17. **else**
18. disp('trimming...');
19. w=w(1:128,:);
20. disp('w ready');
21. **end**
22. [train\_list, names] = itertools(TRAIN\_FOLDS{i}, folds\_idx{i}{2}, stoping\_target);
23. M\_iter=1;
24. b=1.5;
25. gamma = 0.001;
26. objectivefunc\_history = {};
27. grad\_history={};
28. check\_iter=1;
29. test\_time\_temp={};
30. perform\_result\_temp = {};
31. **while** M\_iter<=stoping\_target
32. disp(['training ' num2str(M\_iter) '...']);
33. theta\_i\_name = names{M\_iter,1}; theta\_j\_name = names{M\_iter,2};
34. yij = yij\_decider(theta\_i\_name, theta\_j\_name);
35. theta\_i = train\_list{M\_iter,1}; theta\_j = train\_list{M\_iter,2};
36. [dw\_theta, theta\_diff] = dw(theta\_i, theta\_j, w);
37. test\_val = yij\*(b-dw\_theta);
38. objectivefunc\_history = vertcat(objectivefunc\_history, test\_val);
39. **if** test\_val<=1
40. psi = theta\_diff\*theta\_diff';
41. [w, b] = bw\_updater(w, b, gamma, yij, psi);
42. disp('test\_val = yij\*(b-dw\_theta)');
43. disp([num2str(test\_val) ' = ' num2str(yij) ' \* (' num2str(b) ' - ' num2str(dw\_theta) ')']);
44. **else**
    1. disp('test\_val = yij\*(b-dw\_theta)');
    2. disp([num2str(test\_val) ' = ' num2str(yij) ' \* (' num2str(b) ' - ' num2str(dw\_theta) ')']);
45. **end**
46. grad = max(1-test\_val,0);
47. grad\_history = vertcat(grad\_history, grad);
48. plot\_obj(objectivefunc\_history, M\_iter);
49. **if** M\_iter==checkpoints{check\_iter}
50. time\_test\_start=tic();
51. [performances, result]=dotest(folds\_idx{i}{1}, TEST\_FOLDS{i}, w, b);
52. perform\_result\_temp = vertcat(perform\_result\_temp, {performances, result});
53. time\_test\_end = toc(time\_test\_start)/60;
54. test\_time\_temp = vertcat(test\_time\_temp, time\_test\_end);
55. test\_val\_ke = M\_iter;
56. check\_iter = check\_iter+1;
57. **end**
58. M\_iter=M\_iter+1;
59. **end**
60. all\_b = vertcat(all\_b, b);
61. all\_objectivefunc\_history = vertcat(all\_objectivefunc\_history,{objectivefunc\_history});
62. all\_grad\_history = vertcat(all\_grad\_history, grad\_history);
63. all\_roc\_score = vertcat(all\_roc\_score, perform\_result\_temp);
64. test\_time\_temp = cell2mat(test\_time\_temp);
65. train\_time\_end = toc(time\_train\_start)/60;
66. elapsed\_train\_time = train\_time\_end - sum(test\_time);
67. all\_train\_time = vertcat(all\_train\_time, elapsed\_train\_time);
68. all\_test\_temp = vertcat(all\_test\_time, mean(test\_time\_temp));
69. disp(strcat('saving w ', num2str(i)));
70. name\_wt = strcat('[wt\_',strcat(num2str(i),'].mat'));
71. save(strcat(matPath,strcat('\',name\_wt)),'w','-v7.3');
72. **end**
73. disp('saving all\_train\_time...');
74. name\_all\_train\_time = strcat('[all\_train\_time\_',strcat(num2str(K),'].mat'));
75. save(strcat(matPath,strcat('\', name\_all\_train\_time)),'w','-v7.3');
76. disp('saving all\_test\_time...');
77. name\_all\_test\_time = strcat('[all\_test\_time\_',strcat(num2str(K),'].mat'));
78. save(strcat(matPath,strcat('\', name\_all\_test\_time)),'w','-v7.3');
79. disp('saving all\_objectivefunc\_history...');
80. name\_all\_objectivefunc\_history = strcat('[all\_objectivefunc\_history\_',strcat(num2str(K),'].mat'));
81. save(strcat(matPath,strcat('\', name\_all\_objectivefunc\_history)),'w','-v7.3');
82. disp('saving all\_objectivefunc\_history...');
83. name\_all\_grad\_history = strcat('[all\_grad\_history\_',strcat(num2str(K),'].mat'));
84. save(strcat(matPath,strcat('\', name\_all\_grad\_history)),'w','-v7.3');
85. disp('saving all\_roc\_score...');
86. name\_all\_roc\_score = strcat('[all\_roc\_score\_',strcat(num2str(K),'].mat'));
87. save(strcat(matPath,strcat('\', name\_all\_roc\_score)),'w','-v7.3');
88. disp('saving all\_b...');
89. name\_all\_b = strcat('[all\_b\_',strcat(num2str(K),'].mat'));
90. save(strcat(matPath,strcat('\',all\_b)),'w','-v7.3');
91. disp('CONGRATS! YOU''RE DONE!');

### Implementasi Testing

Testing dilakukan pada semua data test dengan parameter w dan b hasil training pada Kode Sumber 4.16.

Kode Sumber 4.16 Testing

1. **function** [performances, result] = dotest(names\_test, test\_list, w, b)
2. labels = {};
3. outputs = {};
4. **for** i=1:length(names\_test)
5. disp(num2str(i));
6. theta\_i\_name = names\_test{i,1}; theta\_j\_name = names\_test{i,2};
7. disp(strcat(theta\_i\_name, theta\_j\_name));
8. yij = yij\_decider(theta\_i\_name, theta\_j\_name);
9. **if** yij==1
10. labels = vertcat(labels, 1);
11. **else**
12. labels = vertcat(labels, -1);
13. **end**
14. theta\_i = test\_list{i,1}; theta\_j = test\_list{i,2};
15. distance = dw(theta\_i, theta\_j, w);
16. **if** distance<b
17. outputs = vertcat(outputs, 1);
18. **else**
19. outputs = vertcat(outputs, -1);
20. **end**
21. disp(strcat('outputs ', num2str(outputs{i})));
22. disp(strcat('labels ', num2str(labels{i})));
23. **end**
24. labels\_mat = cell2mat(labels);
25. outputs\_mat = cell2mat(outputs);
26. acc=0;
27. **for** i=1:length(labels\_mat)
28. **if** labels\_mat(i)==outputs\_mat(i)
29. acc = acc+1;
30. **end**
31. **if** labels\_mat(i)==-1
32. labels\_mat(i)=0;
33. **end**
34. **if** outputs\_mat(i)==-1
35. outputs\_mat(i)=0;
36. **end**
37. **end**
38. acc = (acc/length(labels\_mat))\*100;
39. [c,cm,ind, per] = confusion(labels\_mat, outputs\_mat);
40. [tpr, tnr, info] = vl\_roc(labels\_mat, outputs\_mat) ;
41. auc = info.auc;
42. err = info.eer;
43. performances = {tpr, tnr, auc, err, acc, {c,cm,ind,per}};
44. result = {labels\_mat; outputs\_mat};
45. vl\_roc(labels\_mat, outputs\_mat) ;
46. **end**

### Implementasi Plot Fungsi Objektif

Plot fungsi objektif digunakan untuk mengetahui apakah fungsi objektif secara keseluruhan sesuai dengan tujuan atau tidak. Implementasi dilakukan dengan Kode Sumber 4.17

Kode Sumber 4.17 Plot Fungsi Objektif

1. **function** plot\_obj(objectivefunc\_history, M\_iter)
2. objectivefunc\_history = cell2mat(objectivefunc\_history);
3. n = length(objectivefunc\_history);
4. x = [1:n];
5. y = zeros(n,1);
6. **for** i=1:n
7. summation = sum(objectivefunc\_history(1:i));
8. y(i) = summation/n;
9. **end**
10. figure(M\_iter);
11. hold on;
12. plot(x,y);
13. title('Objective Function Plot');
14. xlabel('iteration');
15. ylabel('Objective Function Value');
16. **if** M\_iter==1
17. drawnow;
18. **else**
19. delete(figure(M\_iter-1));
20. drawnow
21. **end**
22. **end**

***[Halaman ini sengaja dikosongkan]***

# BAB V PENGUJIAN DAN EVALUASI

Pada bab ini akan dijelaskan hasil uji coba dan evaluasi program yang telah selesai diimplementasi.

## Lingkungan Pengujian

Lingkungan uji coba yang akan digunakan adalah,

* 1. Perangkat Keras

Prosesor: Intel® Xeon™ E5-2630L CPU @ 2.00 GHz Memori: 64.00 GB.

Sistem Operasi: 64-bit .

* 1. Perangkat Lunak

Sistem Operasi: Ubuntu 14.04.4 LTS.

Perangkat Pengembang: Matlab.

Perangkat pembantu: Sublime, Visual Code, vl\_feat

## Data Uji Coba

Data yang digunakan untuk uji coba implementasi fitur SIFT dan fisher vector *encoding* untuk aplikasi pengenalan wajah adalah kumpulan gambar wajah yang didapatkan dari dataset *Labeled Face in the Wild (LFW)*. Kualitas gambar yang digunakan adalah gambar dengan resolusi 250x250. Data gambar yang digunakan diambil dari gambar wajah orang dengan kondisi yang berbeda (posisi wajah, ekspresi, aksesoris yang terpasang, iluminasi, dll). Data pasangan gambar yang digunakan meliputi dua jenis pasangan. Pasangan gambar dari orang yang sama, dan pasangan gambar dari orang yang berbeda. Jumlah gambar yang diuji sebanyak 50 pasang gambar dari orang yang sama, dan 50 pasang gambar dari orang yang berbeda. Contoh gambar uji dapat dilihat pada Gambar 5.1.

|  |  |  |  |
| --- | --- | --- | --- |
| Aaron P 1 | Aaron P 2 | Adam S 2 | Adam S 4 |
| Abid H 1 | Abid H 3 | Adam Sc 1 | Adam Sc 2 |
| Barbara B 1 | Barbara W 1 | Barbara S 1 | Barret J 1 |
| Angela M 1 | Ann V 1 | Ben C 1 | Ben G 1 |
|  |  |  |  |

Gambar 5.1 Contoh Pasangan Gambar Uji

## Alur Uji Coba

Pada sub bab ini akan dijelaskan mengenai alur kerja dari sistem pengenalan wajah. Dimulai dari *preprocessing* hingga klasifikasi yang telah dijelaskan pada sub-bab 3.2. Penjelasan ringkas masing-masing proses ujicoba dijelaskan pada sub-bab berikut ini

### Preprocessing

Tahap *preprocessing* akan dijelaskan bagaimana alur setiap gambarmasuk hingga menghasilkan gambar yang siap untuk diproses pada tahap berikutnya. Tahap pre-processing dijelaskan lebih lanjut pada sub-bab 3.3.

### Ekstraksi Fitur dense SIFT

Proses ekstraksi fitur dense SIFT dijelaskan pada sub-bab 3.4, dimana ekstraksi diawali dengan penentuan lokasi ekstraksi deskriptor, baru kemudian diekstraksi. Ekstraksi fitur dense SIFT menggunakan *toolbox* *vl\_feat* di matlab.

### Fisher Vector Encoding

Pada proses ini fitur dense SIFT yang diekstrak pada data testing kemudian dilakukan encoding untuk merepresentasikan data menjadi lebih sederhana. Proses ini lebih lengkap dijelaskan pada sub-bab 3.6.

### Pengukuran Jarak Fisher Vector

Pengukuran dilakukan dengan Euclidean distance, namun faktor bobot antara kedua vector adalah bobot dari *metric learning* yang sudah dipelajari pada proses training seperti yang dijelaskan pada sub-bab 3.9

### Perhitungan Performa

Dari pengukuran jarak yang ada, digunakanlah threshold yang telah dipelajari untuk menentukan apakah sebuah pasangan termasuk pasangan positif atau bukan. Setelah itu performa diukur berdasarkan akurasi klasifikasi pasangan gambar tersebut

## Skenario Uji Coba K-Gaussian

Pada sub bab ini akan dijelaskan mengenai skenario uji coba yang telah dilakukan. Skenario uji coba yang dilakukan yaitu memvariasikan banyaknya nilai K gaussians dalam GMM. Variasi yang digunakan adalah 128 dan 192. Pemilihan skenario uji coba yang dilakukan hanya pada satu faktor yaitu K gaussian dikarenakan merupakan faktor paling signifikan, dan juga keterbatasan memori serta waktu yang tersedia untuk melakukan percobaan seperti yang dijelaskan pada sub-bab 3.7. Hasil pengukuran performa diberikan pada Gambar 5.2 sampai dengan Gambar 5.9. Pengukuran performa dilakukan secara *checkpoint* setiap seratus kali iterasi.

### F:\Dropbox\[PENTING TIDAK URGENT]\[ARSIP KULIAH]\Semester 9\#TUGAS_AKHIR\Apps\[GIT]\Buku\gambar\128_result_fold1.jpgPerforma Pada K=128, Fold 1

Gambar 5.2 Pengukuran Performa dengan K=128 pada fold 1

Gambar 5.2 adalah pengukuran performa dengan K=128 pada fold pertama. Grafik *True Positive* ditunjukkan dengan garis berwarna merah, *True Negative* dengan garis berwarna biru, dan akurasi dengan garis berwarna hijau. Dari grafik dapat diketahui bahwaperforma menunjukkan angka 50% dari awal proses sampai dengan iterasi sekitar 5,500. Setelah itu terjadi peningkatan *True Positive* dan akurasi dibarengi dengan berkurangnya *True Negative*. Pada iterasi terakhir menghasilkan 3 pair *True Positive,* dan 47 pair *True Negative* dan 50% akurasi

### Performa Pada K=128, Fold 2

Gambar 5.3 Pengukuran Performa dengan K=128 pada fold 2

Gambar 5.3 adalah pengukuran performa dengan K=128 pada fold kedua. Grafik *True Positive* ditunjukkan dengan garis berwarna merah, *True Negative* dengan garis berwarna biru, dan akurasi dengan garis berwarna hijau. Dari grafik dapat diketahui bahwaperforma menunjukkan angka 50% dari awal proses sampai dengan iterasi sekitar 5,500. Setelah itu terjadi peningkatan *True Positive* dan akurasi dibarengi dengan berkurangnya *True Negative*. Pada iterasi terakhir menghasilkan 8 pair *True Positive,* dan 40 pair *True Negative* dan 48% akurasi.

### Performa Pada K=128, Fold 3

Gambar 5.4 Pengukuran Performa dengan K=128 pada fold 3

Gambar 5.4 adalah pengukuran performa dengan K=128 pada fold ketiga. Grafik *True Positive* ditunjukkan dengan garis berwarna merah, *True Negative* dengan garis berwarna biru, dan akurasi dengan garis berwarna hijau. Dari grafik dapat diketahui bahwaperforma menunjukkan angka 50% dari awal proses sampai dengan iterasi sekitar 5,500. Setelah itu terjadi peningkatan *True Positive* dan akurasi dibarengi dengan berkurangnya *True Negative*. Pada iterasi terakhir menghasilkan 3 pair *True Positive,* dan 48 pair *True Negative* dan 51% akurasi.

### Performa Pada K=128, Fold 4

Gambar 5.5 Pengukuran Performa dengan K=128 pada fold 4

Gambar 5.5 adalah pengukuran performa dengan K=128 pada fold keempat. Grafik *True Positive* ditunjukkan dengan garis berwarna merah, *True Negative* dengan garis berwarna biru, dan akurasi dengan garis berwarna hijau. Dari grafik dapat diketahui bahwaperforma menunjukkan angka 50% dari awal proses sampai dengan iterasi sekitar 5,500. Setelah itu terjadi peningkatan *True Positive* dan akurasi dibarengi dengan berkurangnya *True Negative*. Pada iterasi terakhir menghasilkan 3 pair *True Positive,* dan 48 pair *True Negative* dan 51% akurasi.

### Performa Pada K=192, Fold 1

Gambar 5.6 Pengukuran Performa dengan K=192 pada fold 1

Gambar 5.6 adalah pengukuran performa dengan K=192 pada fold pertama. Grafik *True Positive* ditunjukkan dengan garis berwarna merah, *True Negative* dengan garis berwarna biru, dan akurasi dengan garis berwarna hijau. Dari grafik dapat diketahui bahwaperforma menunjukkan angka 50% dari awal proses sampai dengan iterasi sekitar 5,500. Setelah itu terjadi peningkatan *True Positive* dan akurasi dibarengi dengan berkurangnya *True Negative*. Pada iterasi terakhir menghasilkan 10 pair *True Positive,* dan 43 pair *True Negative* dan 53% akurasi.

### Performa Pada K=192, Fold 2

Gambar 5.7 Pengukuran Performa dengan K=192 pada fold 2

Gambar 5.7 adalah pengukuran performa dengan K=192 pada fold kedua. Grafik *True Positive* ditunjukkan dengan garis berwarna merah, *True Negative* dengan garis berwarna biru, dan akurasi dengan garis berwarna hijau. Dari grafik dapat diketahui bahwaperforma menunjukkan angka 50% dari awal proses sampai dengan iterasi sekitar 5,500. Setelah itu terjadi peningkatan *True Positive* dan akurasi dibarengi dengan berkurangnya *True Negative*. Pada iterasi terakhir menghasilkan 2 pair *True Positive,* dan 50 pair *True Negative* dan 52% akurasi.

### Performa Pada K=192, Fold 3

Gambar 5.8 Pengukuran Performa dengan K=192 pada fold 3

Gambar 5.8 adalah pengukuran performa dengan K=192 pada fold ketiga. Grafik *True Positive* ditunjukkan dengan garis berwarna merah, *True Negative* dengan garis berwarna biru, dan akurasi dengan garis berwarna hijau. Dari grafik dapat diketahui bahwaperforma menunjukkan angka 50% dari awal proses sampai dengan iterasi sekitar 5,500. Setelah itu terjadi peningkatan *True Positive* dan akurasi dibarengi dengan berkurangnya *True Negative*. Pada iterasi terakhir menghasilkan 3 pair *True Positive,* dan 49 pair *True Negative* dan 52% akurasi.

### Performa Pada K=192, Fold 4

Gambar 5.9 Pengukuran Performa dengan K=192 pada fold 4

Gambar 5.9 adalah pengukuran performa dengan K=192 pada fold keempat. Grafik *True Positive* ditunjukkan dengan garis berwarna merah, *True Negative* dengan garis berwarna biru, dan akurasi dengan garis berwarna hijau. Dari grafik dapat diketahui bahwaperforma menunjukkan angka 50% dari awal proses sampai dengan iterasi sekitar 5,500. Setelah itu terjadi peningkatan *True Positive* dan akurasi dibarengi dengan berkurangnya *True Negative*. Pada iterasi terakhir menghasilkan 4 pair *True Positive,* dan 48 pair *True Negative* dan 52% akurasi.

### Rekapitulasi Performa Uji Coba K-Gaussian

Pengukuran performa pada semua parameter uji coba dapat dirangkum pada Tabel 5.1dan Tabel 5.2.

Tabel 5.1 Confusion Matrix Hasil Uji Coba

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *K* | *fold* | *Performa Akhir (pair)* | | | |
| *TP* | *FN* | *TN* | *FP* |
| 128 | 1 | 3 | 47 | 47 | 3 |
| 2 | 8 | 42 | 40 | 10 |
| 3 | 3 | 47 | 48 | 2 |
| 4 | 3 | 47 | 48 | 2 |
| *Rata-rata* | | 4.255 | 45.75 | 45.75 | 4.25 |
| 192 | 1 | 10 | 40 | 43 | 7 |
| 2 | 2 | 48 | 50 | 0 |
| 3 | 3 | 47 | 49 | 1 |
| 4 | 4 | 46 | 48 | 2 |
| *Rata-rata* | | 5 | 45 | 47.5 | 2.5 |

Tabel 5.2 Akurasi dan Waktu Eksekusi Hasil Uji Coba

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *K* | *fold* | *Akurasi*  *Akhir (%)* | *Akurasi Puncak*  *(%)* | *Waktu  Eksekusi*  *(min)* |
|
| 128 | 1 | 50 | 51 | 765.293 |
| 2 | 48 | 50 | 849.602 |
| 3 | 51 | 51 | 867.1972 |
| 4 | 51 | 51 | 861.782 |
| *Rata-rata* | | 50 | 50.75 | 835.968 |
| 192 | 1 | 53 | 54 | 2386.1 |
| 2 | 52 | 52 | 2221.7 |
| 3 | 52 | 52 | 2406.84 |
| 4 | 52 | 53 | 2332.88 |
| *Rata-rata* | | 52.5 | 52.75 | 2336.88 |

Dari Tabel 5.1 dapat diketahui bahwa untuk parameter K=128 memberikan hasil *TP* terbaik 8 pair pada fold 2 dan untuk parameter K=192 memberikan hasil *TP* terbaik 11 pair pada fold 1.

Dari Tabel 5.2 dapat diketahui bahwa akurasi terbaik pada fold 1 dan 3 masing-masing menunjukkan akurasi 51% dan waktu eksekusi rata-rata adalah 835.968 menit. akurasi terbaik pada fold 1 yakni sebesar 51%, dan waktu eksekusi rata-rata adalah 2336.88 menit.

## Analisis Hasil Uji Coba

Dari hasil skenario uji coba menunjukkan bahwa secara umum penggunaan K pada GMM yang lebih tinggi memberikan dampak positif, yakni naiknya akurasi dan *true positive* secara umum. Hal ini dikarenakan semakin banyak K yang diambil, semakin banyak titik pengamatan pada wajah. Selain itu dapat diamati pada grafik performa terlihat bahwa naiknya *true positive* bersamaan dengan turunnya *true negative* dikarenakan berubahnya bobot jarak dan threshold.

Terkait hasil pengukuran performa antar fold dapat memberikan hasil yang berbeda meskipun menggunakan parameter K yang sama dapat dijelaskan sebagai berikut:

* Penyebab pertama, konvergensi distribusi Gaussian pada GMM bisa terjebak pada optimum lokal karena sangat sensitif dengan lokasi awal means nya.
* Penyebab kedua, metode low-rank Mahalanobis Metric Learning sendiri merupakan problem optimasi non-convex dimana optimasi parameter bobot dan *threshold* bisa jadi berujung pada optimum lokal.
* Sedangkan optimasi bobot dan *threshold* dilakukan dengan cara fitting terhadap pasangan *fisher vector* pada data training yang ada

Sehingga dapat disimpulkan bahwa performa sangat sensitif dengan pemilihan dataset training, apakah dataset training mampu merepresentasikan varietas wajah secara umum dengan baik atau tidak.

Secara visual hasil *snapshot* hasil uji coba juga dapat dianalisis. Lampiran A.2 memberikan pasangan gambar *True Positive.* Pada pasangan *True Positive* secara umum memiliki arah wajah yang cenderung sama dan tidak banyak distraksi (aksesoris, kacamata, wajah lain,dll). Selain itu bentuk wajah, rambut, jenggot, dll. Cenderung sama. Perkecualian terdapat pada kasus *Christina Aguilera* pada lampiran A.2 nomor 2. Hal ini menunjukkan bahwa pelatihan GMM memang menentukan titik-titik lokal mana saja yang memberikan kontribusi penting, sehingga beberapa gambar yang tidak seragam masih bisa ditangkap sebagai orang yang sama.

Analisis visual *snapshot* untuk pasangan *False Negative* pada Lampiran A.3 secara umum dikarenakan banyaknya distraksi disekitar gambar. Distraksi dapat berupa aksesoris kacamata, wajah orang lain disekitarnya, ekspreksi yang cenderung sangat berbeda.

Analisis visual *snapshot* untuk pasangan *True Negative* pada Lampiran A.4 secara umum dikarenakan secara gender, ekspreksi, dan aksesoris disekitar area wajah memang jauh berbeda.

Analisis visual *snapshot* untuk pasangan *False Positive*  pada Lampiran A.5 secara umum dikarenakan ada kemiripan pada area tertentu seperti mata, mulut, atau ekspreksi dan rotasi wajah. Namun beberapa kasus seperti pasangan nomor 6 dan 7 pada lampiran tersebut tidak memiliki kemiripan secara visual. Keputusan sama dimungkinan karena pengamatan pada titik-titik lokal mungkin sama dan faktor nilai *threshold* yang *overshoot*.

# BAB VI KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan mengenai kesimpulan dari proses dan uji coba dari program dan saran untuk pengembangan dari program itu sendiri.

## Kesimpulan

Dari hasil uji coba yang telah dilakukan, dapat diambil kesimpulan sebagai berikut :

1. Berdasarkan hasil pengujian pada skenario perubahan nilai K memberikan informasi bahwa penentuan banyaknya K-GMM berpengaruh pada peningkatan performa sistem. Pada K=128 diperoleh akurasi rata-rata 50%. Sedangkan pada K=192 diperoleh akurasi rata-rata 52.5% Hal ini dikarenakan semakin banyak klaster Gaussian yang dipelajari, semakin banyak bagian wajah yang diteliti.
2. Berdasarkan hasil pengujian pada skenario perubahan nilai K memberikan informasi bahwa performa pada fold yang berbeda dengan K yang sama tidak selalu memberikan hasil yang seragam. Hal ini menunjukkan performa sangat sensitif dengan kualitas dan kuantitas data training yang dipilih.

## Saran

Saran yang diberikan untuk pengembangan perangkat lunak ini adalah:

Peningkatan kemampuan perangkat keras untuk melakukan komputasi dapat mempercepat proses *learning*, serta memungkinkan penelitian pada banyak parameter.

Penggunaan *metric learning* yang lain kemungkinan mampu memberikan performa yang baik. Beberapa *metric learning* yang mungkin dapat diterapkan antara lain : *join-distance similiarity, weighted euclidean,* atau vaian *metric learning* lainnya.

Pemilihan data training sangat disarankan untuk memiliki varietas wajah yang cukup (beraneka warna, ras, jenis, dll) dengan jumlah yang banyak dan berimbang antar fold.

# DAFTAR PUSTAKA

|  |  |
| --- | --- |
| [1] | J. Rabia dan A. Hamid R., “A Survey of Face Recognition Techniques,” *Journal of Information Processing Systems,* vol. 5, no. 2, p. 41, 2009. |
| [2] | K. Simonyan, O. Parkhi, A. Vedaldi dan A. Zisserman, “Fisher Vector Faces (FVF) in the Wild,” Visual Geometry Group, University of Oxford, Oxford, 2013. |
| [3] | D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision,* pp. 1-20, 2004. |
| [4] | J. Sanchez, F. Perronnin dan T. Mensink, “Image Classification with the Fisher Vector: Theory and,” HAL archives-ouvertes, Paris, 2013. |
| [5] | A. Bellet, “Tutorial on Metric Learning,” Department of Computer Science Viterbi School of Engineering University of Southern California, California, 2013. |
| [6] | U. o. M. Amherst, “Labeled Faces in the Wild,” University of Massachutes Amherst, [Online]. Available: http://vis-www.cs.umass.edu/lfw. [Diakses 14 December 2015]. |
| [7] | A. Vedaldi, B. Fukerson, K. Lenc, D. Perrone, M. Perdoch, M. Sulc dan H. Sarbotova, “vlfeat.org,” VLFeat, 2007. [Online]. Available: http://www.vlfeat.org/about.html. [Diakses 06 12 2016]. |
| [8] | P. Viola dan M. J. Jones, “Robust Real-Time Face Detection,” *International Journal Of Computer Vision ,* vol. 2, p. 57, 2004. |
| [9] | K. Baker, “Singular Value Decomposition Tutorial,” Kirk Baker, 2005. |
| [10] | Fmadd, “Whitening transformation,” Wikipedia, 13 May 2016. [Online]. Available: https://en.wikipedia.org/wiki/Whitening\_transformation. [Diakses 28 June 2016]. |
| [11] | Watsuen, “Whitening,” Stanford University, 04 April 2011. [Online]. Available: http://ufldl.stanford.edu/wiki/index.php/Whitening. [Diakses 28 June 2016]. |
| [12] | D. Reynolds, “Gaussian Mixture Model,” MIT Lincoln Laboratory, Massachusetts. |
| [13] | A. Vedaldi dan B. Fulkerson, “Tutorials - SIFT,” Princeton University, 2007. [Online]. Available: http://vision.princeton.edu/pvt/SiftFu/SiftFu/SIFTransac/vlfeat/doc/overview/sift.html. [Diakses 16 01 2017]. |

* 1. LAMPIRAN

Lampiran 7.1 zero padding untuk nama gambar

* + 1. **function** [numb\_pad] = zeroPad(numb, len\_numb)
    2. **for** i=1:len\_numb
    3. numb = strcat('0',numb);
    4. **end**
    5. numb\_pad = numb;
    6. **end**

Lampiran 7.2 Hasil Testing Pasangan True Positive (K=192,fold=1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No | Nama 1 | Gambar | Nama 2 | Gambar | Label | Output |
| 1. | Carson\_  Palmer\_  0002 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Carson_Palmer_0002.jpg | Carson\_  Palmer\_  0003 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Carson_Palmer_0003.jpg | +1 | +1 |
| 2. | Christina\_  Aguilera\_  0001 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Christina_Aguilera_0001.jpg | Christina\_  Aguilera\_  0002 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Christina_Aguilera_0002.jpg | +1 | +1 |
| 3 | Daniel\_  Day Lewis\_  0002 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Daniel_Day-Lewis_0002.jpg | Daniel\_  Day Lewis\_  0003 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Daniel_Day-Lewis_0003.jpg | +1 | +1 |
| 4 | Edwina\_  Currie\_  0001 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Edwina_Currie_0001.jpg | Edwina\_  Currie\_  0002 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Edwina_Currie_0002.jpg | +1 | +1 |
| 5 | Luciano\_  Pavarotti\_  0001 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Luciano_Pavarotti_0001.jpg | Luciano\_  Pavarotti\_  0002 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Luciano_Pavarotti_0002.jpg | +1 | +1 |
| 6 | Luiz\_Inacio\_  Lula\_da\_  Silva\_0007 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Luiz_Inacio_Lula_da_Silva_0007.jpg | Luiz\_Inacio\_  Lula\_da\_  Silva\_0036 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Luiz_Inacio_Lula_da_Silva_0036.jpg | +1 | +1 |
| 7 | Patti\_  Labelle\_  0002 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Patti_Labelle_0002.jpg | Patti\_  Labelle\_  0003 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Patti_Labelle_0003.jpg | +1 | +1 |
| 8 | Demetrius\_  Ferraciu\_  0001 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Demetrius_Ferraciu_0001.jpg | Demetrius\_  Ferraciu\_  0002 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Demetrius_Ferraciu_0002.jpg | +1 | +1 |
| 9 | Larry\_  Ellison\_  0001 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Larry_Ellison_0001.jpg | Larry\_  Ellison\_  0003 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Larry_Ellison_0003.jpg | +1 | +1 |
| 10 | Robert\_  Redford\_  0003 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Robert_Redford_0003.jpg | Robert\_  Redford\_  0004 | C:\Users\umum\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Robert_Redford_0004.jpg | +1 | +1 |

Lampiran 7.3 Hasil Testing Pasangan False Negative (K=192,fold=1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No | Nama 1 | Gambar | Nama 2 | Gambar | Label | Output |
| 1. | Andrei\_  Mikhnevich\_  0001 |  | Andrei\_  Mikhnevich\_  0002 |  | +1 | -1 |
| 2. | Atal\_Bihari\_  Vajpayee\_  0004 |  | Atal\_Bihari\_  Vajpayee\_  0006 |  | +1 | -1 |
| 3 | Frances\_  Fisher\_  0001 |  | Frances\_  Fisher\_  0002 |  | +1 | -1 |
| 4 | Gwendal\_  Peizerat\_  0001 |  | Gwendal\_  Peizerat\_  0003 |  | +1 | -1 |
| 5 | Holly\_  Hunter\_  0002 |  | Holly\_  Hunter\_  0006 |  | +1 | -1 |
| 6 | James\_  Smith\_  0001 |  | James\_  Smith\_  0002 |  | +1 | -1 |
| 7 | Jennifer\_  Aniston\_  0001 |  | Jennifer\_  Aniston\_  0016 |  | +1 | -1 |
| 8 | Joerg\_  Haider\_  0001 |  | Joerg\_  Haider\_  0002 |  | +1 | -1 |
| 9 | John\_  Negroponte\_  0005 |  | John\_  Negroponte\_  0026 |  | +1 | -1 |
| 10 | Joseph\_  Estrada\_  0002 |  | Joseph\_  Estrada\_  0003 |  | +1 | -1 |
| 11 | Kofi\_  Annan\_  0013 |  | Kofi\_  Annan\_  0025 |  | +1 | -1 |
| 12 | Mark\_  Philippoussis\_  0003 |  | Mark\_  Philippoussis\_  0011 |  | +1 | -1 |
| 13 | Mesut\_  Yilmaz\_  0001 |  | Mesut\_  Yilmaz\_  0002 |  | +1 | -1 |
| 14 | Monica\_  Seles\_  0004 |  | Monica\_  Seles\_  0005 |  | +1 | -1 |
| 15 | Rachel\_  Hunter\_  0003 |  | Rachel\_  Hunter\_  0004 |  | +1 | -1 |
| 16 | Richard\_  Armitage\_  0006 |  | Richard\_  Armitage\_  0009 |  | +1 | -1 |
| 17 | Robert\_  Mugabe\_  0001 |  | Robert\_  Mugabe\_  0002 |  | +1 | -1 |
| 18 | Rosemarie\_  Stack\_  0001 |  | Rosemarie\_  Stack\_  0002 |  | +1 | -1 |
| 19 | Arnold\_  Schwarzenegger\_  0009 |  | Arnold\_  Schwarzenegger\_  0039 |  | +1 | -1 |
| 20 | Bruce\_  Springsteen\_  0001 |  | Bruce\_  Springsteen\_  0002 |  | +1 | -1 |
| 11 | Carrie-Anne\_  Moss\_  0001 |  | Carrie-Anne\_  Moss\_  0005 |  | +1 | -1 |
| 12 | Ciro\_Gomes\_  0002 |  | Ciro\_Gomes\_  0005 |  | +1 | -1 |
| 13 | Ferenc\_  Madl\_  0001 |  | Ferenc\_  Madl\_  0002 |  | +1 | -1 |
| 14 | Franz\_  Muentefering\_  0003 |  | Franz\_  Muentefering\_  0004 |  | +1 | -1 |
| 15 | Gonzalo\_  Sanchez\_de\_  Lozada\_0008 |  | Gonzalo\_  Sanchez\_de\_  Lozada\_0012 |  | +1 | -1 |
| 16 | Horst\_  Koehler\_  0001 |  | Horst\_  Koehler\_  0002 |  | +1 | -1 |
| 17 | Jean-Claude\_  Trichet\_  0001 |  | Jean-Claude\_  Trichet\_  0002 |  | +1 | -1 |
| 18 | John\_  Kerry\_  0004 |  | John\_  Kerry\_  0012 |  | +1 | -1 |
| 19 | Jose\_Maria\_  Aznar\_  0006 |  | Jose\_Maria\_  Aznar\_  0013 |  | +1 | -1 |
| 20 | Justin\_  Timberlake\_  0001 |  | Justin\_  Timberlake\_  0002 |  | +1 | -1 |
| 21 | Kim\_Dae-  jung\_  0002 |  | Kim\_Dae-  jung\_  0003 |  | +1 | -1 |
| 22 | Li\_  Peng\_  0004 |  | Li\_  Peng\_  0007 |  | +1 | -1 |
| 23 | Lucio\_  Gutierrez\_  0006 |  | Lucio\_  Gutierrez\_  0008 |  | +1 | -1 |
| 24 | Martin\_  McGuinness\_  0003 |  | Martin\_  McGuinness\_  0004 |  | +1 | -1 |
| 25 | Michael\_  Douglas\_  0002 |  | Michael\_  Douglas\_  0004 |  | +1 | -1 |
| 26 | Mike\_  Tyson\_  0001 |  | Mike\_  Tyson\_  0003 |  | +1 | -1 |
| 27 | Nadia\_  Petrova\_  0003 |  | Nadia\_  Petrova\_  0004 |  | +1 | -1 |
| 28 | Nikki\_  Reed\_  0001 |  | Nikki\_  Reed\_  0002 |  | +1 | -1 |
| 29 | Rainer\_  Schuettler\_  0001 |  | Rainer\_  Schuettler\_  0004 |  | +1 | -1 |
| 30 | Rick\_  Santorum\_  0001 |  | Rick\_  Santorum\_  0002 |  | +1 | -1 |

Lampiran 7.4 Hasil Testing Pasangan True Negative (K=192,fold=1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No | Nama 1 | Gambar | Nama 2 | Gambar | Label | Output |
| 1. | Almeida\_  Baptista\_  0001 |  | Jan\_Peter\_  Balkenende\_  0001 |  | -1 | -1 |
| 2. | Andy\_  North\_  0001 |  | Phil\_  Bennett\_  0001 |  | -1 | -1 |
| 3 | Baburam\_  Bhattari\_  0001 |  | Muhammad\_  Saeed\_al-Sahhaf\_  0004 |  | -1 | -1 |
| 4 | Boris\_  Trajkovski\_  0001 |  | Laurent\_  Gbagbo\_  0002 |  | -1 | -1 |
| 5 | Calvin\_  Harrison\_  0001 |  | Richard\_  Gephardt\_  0009 |  | -1 | -1 |
| 6 | Charlie\_  Deane\_  0001 |  | Queen\_  Silvia\_  0001 |  | -1 | -1 |
| 7 | Christian\_  Lirette\_  0001 |  | Sargis\_  Sargsian\_  0001 |  | -1 | -1 |
| 8 | David\_  Hilt\_  0001 |  | Hipolito\_  Mejia\_  0003 |  | -1 | -1 |
| 9 | Don\_  Hewitt\_  0001 |  | Guennadi\_  Chipouline\_  0001 |  | -1 | -1 |
| 10 | E\_Clay\_  Shaw\_  0001 |  | Jerry\_  Jones\_  0001 |  | -1 | -1 |
| 11 | Ellen\_  Saracini\_  0001 |  | Ray\_  Romano\_  0006 |  | -1 | -1 |
| 12 | Gary\_  Leon\_  Ridgway\_0001 |  | Yana\_  Klochkova\_  0001 |  | -1 | -1 |
| 13 | Grant\_  Rossenmeyer\_  0001 |  | Trevor\_  Watson\_  0001 |  | -1 | -1 |
| 14 | Ilan\_  Goldfajn\_  0001 |  | Jennifer\_  Aniston\_  0017 |  | -1 | -1 |
| 15 | Jakob\_  Kellenberger\_  0001 |  | Pedro\_  Mahecha\_  0001 |  | -1 | -1 |
| 16 | Jim\_  Hendry\_  0001 |  | Larry\_  Beinfest\_  0001 |  | -1 | -1 |
| 17 | John\_  Reilly\_  0002 |  | Princess\_  Hisako\_  0001 |  | -1 | -1 |
| 18 | Kathryn\_  Grayson\_  0001 |  | Ruben\_  Sierra\_  0001 |  | -1 | -1 |
| 19 | Laszlo\_  Kovacs\_  0001 |  | Luke\_  Ridnour\_  0001 |  | -1 | -1 |
| 20 | Liu\_  Ye\_  0001 |  | Steve\_  Alford\_  0001 |  | -1 | -1 |
| 21 | Mario\_  Alfaro-Lopez\_0001 |  | Tom\_  Lantos\_  0001 |  | -1 | -1 |
| 22 | Adam\_  Rich\_  0001 |  | John\_  Goold\_  0001 |  | -1 | -1 |
| 23 | Albert\_  Brooks\_  0001 |  | Sheila\_  Copps\_  0003 |  | -1 | -1 |
| 24 | Barbara\_  Becker\_  0001 |  | Franz\_  Beckenbauer\_  0002 |  | -1 | -1 |
| 25 | Bill\_  Herrion\_  0001 |  | Fernando\_  Valenzuela\_  0001 |  | -1 | -1 |
| 26 | Bob\_  Eskridge\_  0001 |  | Marco\_  Pantani\_  0001 |  | -1 | -1 |
| 27 | Brandon\_  Lloyd\_  0001 |  | James\_  Coburn\_  0001 |  | -1 | -1 |
| 28 | Carol\_  Williams\_  0001 |  | Gorden\_  Tallis\_  0001 |  | -1 | -1 |
| 29 | Charlton\_  Heston\_  0004 |  | Desiree\_  Lemosi\_ 0001 |  | -1 | -1 |
| 30 | Chuck\_  Hagel\_  0001 |  | Sebastian\_  Porto\_  0001 |  | -1 | -1 |
| 31 | Denise\_  Johnson\_  0002 |  | Filippo\_  Inzaghi\_  0002 |  | -1 | -1 |
| 32 | Edward\_  Seaga\_  0001 |  | George\_  Roy\_Hill\_  0001 |  | -1 | -1 |
| 33 | Eric\_  Taino\_  0001 |  | Michael\_  Goldrich\_  0001 |  | -1 | -1 |
| 34 | Franz\_  Beckenbauer\_  0002 |  | Rick\_  Bragg\_  0001 |  | -1 | -1 |
| 35 | Gonzalo\_Sanchez\_  de\_Lozada\_  0003 |  | Shannyn\_  Sossamon\_  0001 |  | -1 | -1 |
| 36 | Habib\_  Hisham\_  0001 |  | Paul\_  Wilson\_  0001 |  | -1 | -1 |
| 37 | Huang\_  Suey-Sheng\_  0001 |  | Paul\_  Cerjan\_  0001 |  | -1 | -1 |
| 38 | Jamie\_Lee\_  Curtis\_  0001 |  | William\_  Joppy\_  0001 |  | -1 | -1 |
| 39 | Jerry\_  Sloan\_  0001 |  | Nikki\_  Cascone\_  0001 |  | -1 | -1 |
| 40 | John\_  Lisowski\_  0001 |  | Michelle\_  Kwan\_  0001 |  | -1 | -1 |
| 41 | Johnny\_  Unitas\_  0001 |  | Mark\_  Foley\_  0001 |  | -1 | -1 |
| 42 | Katie\_  Couric\_  0001 |  | Sanjay\_  Gupta\_  0001 |  | -1 | -1 |
| 43 | Kyle\_  Shewfelt\_  0001 |  | Larry\_  Coker\_  0002 |  | -1 | -1 |

Lampiran 7.5 Hasil Testing Pasangan False Positive (K=192,fold=1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No | Nama 1 | Gambar | Nama 2 | Gambar | Label | Output |
| 1. | Alessandra\_  Cerna\_  0001 |  | John\_  Darby\_  0001 |  | -1 | +1 |
| 2. | Courtney\_  Love\_  0001 |  | Jennifer\_  Aniston\_  0003 |  | -1 | +1 |
| 3 | Eric\_  Lloyd\_  0001 |  | Jessica\_  Alba\_  0002 |  | -1 | +1 |
| 4 | Harrison\_  Ford\_  0002 |  | Marcus\_  Gronholm\_  0001 |  | -1 | +1 |
| 5 | Ali\_  Naimi\_  0005 |  | Morris\_  Dees\_  0001 |  | -1 | +1 |
| 6 | Danny\_  Glover\_  0001 |  | Morris\_  Dees\_  0001 |  | -1 | +1 |
| 7 | Martin\_Luther\_  King\_III\_  0001 |  | Maryn\_  McKenna\_  0001 |  | -1 | +1 |

***[Halaman ini sengaja dikosongkan]***

**BIODATA PENULIS**

 Yusuf Azis Henny Tri Yudhantoro atau biasa dipanggil Yudha dilahirkan di Jombang pada tanggal 04 Desember 1993 dan dibesarkan di Jombang. Penulis adalah anak terakhir dari tiga bersaudara.

Penulis menempuh pendidikan di SDIT Al-Ummah Jombang (2000-2005), SDN Jomabatan V Jombang (2005-2006) SMP N 2 Jombang (2006-2009), dan SMA N 2 Jombang (2009-2012). Setelah lulus SMA penulis melanjutkan ke jenjang perkuliahan di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Bidang Studi yang diambil oleh penulis pada saat kuliah di Teknik Informatika ITS adalah Komputasi Cerdas dan Visi.

Selama menempuh kuliah penulis aktif sebagai anggota Himpunan Mahasiswa Teknik Computer (HMTC) ITS pada departemen dalam negeri, dan staff ahli departemen dalam negeri . Selain itu penulis juga aktif di organisasi Lembaga Dakwah Jurusan (LDJ) Keluarga Muslim Informatika (KMI) ITS sebagai staff media. Penulis juga aktif dalam organisasi Lembaga Dakwah Kampus (LDK) Jamaah Masjid Manarul Ilmi (JMMI) ITS sebagai Ketua Divisi Media. Selain itu penulis juga aktif menjadi administrator Lab Komputasi Cerdas dan Visi (KCV) Teknik Informatika ITS dari tahun 2013-2015. Penulis dapat dihubungi melalui alamat *email* yusufazishty12@gmail.com.