TUGAS AKHIR – KS09 1336

PEMBUATAN *WEB SERVICE* SEBAGAI LAYANAN PENDETEKSI KONTEN PORNOGRAFI PADA CITRA DIGITAL DENGAN METODE *IMAGE ZONING*

Lourensius Bisma
NRP 5210100155

Dosen Pembimbing I
Dr.Eng. Febriliyan Samopa, S.Kom, M.Kom.

Dosen Pembimbing II
Radityo P. Wibowo, S.Kom, M.Kom.

JURUSAN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2014

FINAL PROJECT – KS09 1336

DEVELOPMENT OF WEB SERVICE AS PORN DETECTION SERVICE IN DIGITAL IMAGE USING IMAGE ZONING METHOD

Lourensius Bisma
NRP 5210100155

Supervisor I
Dr.Eng. Febriliyan Samopa, S.Kom, M.Kom.

Supervisor II
Radityo P. Wibowo, S.Kom, M.Kom.

INFORMATION SYSTEMS DEPARTMENT
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2014

PEMBUATAN *WEB SERVICE* SEBAGAI LAYANAN PENDETEKSI KONTEN PORNOGRAFI PADA CITRA DIGITAL DENGAN METODE *IMAGE ZONING*

Nama Mahasiswa : Lourensius Bisma

NRP : 5210100155
Jurusan : Sistem Informasi FTIf – ITS
Dosen Pembimbing I : Dr.Eng. Febriliyan Samopa, S.Kom, M.Kom

Dosen Pembimbing II : Radityo P. Wibowo, S.Kom, M.Kom

ABSTRAK

Salah satu tantangan terbesar dalam era globalisasi ini adalah bagaimana mengolah data dalam jaringan internet yang jumlahnya sangat tinggi. Terlebih jika data yang tersimpan dalam sebuah situs terindikasi sebagai konten pornografi. Banyak kasus pelecehan seksual terjadi dengan melibatkan anak di bawah umur. Kebanyakan pelakunya sering mengonsumsi konten pornografi dalam internet. Untuk itu, diperlukan sistem yang secara otomatis mampu menyaring data ke dalam kategori pornografi atau non-pornografi. Melalui pembuatan Tugas Akhir ini, sebuah sistem web service diusulkan untuk mengkategorikan sebuah citra memiliki konten pornografi atau tidak. Metode yang digunakan adalah Image Zoning yaitu pembuatan zona pada citra untuk menilai potensi atau tingkat pornografi yang muncul dari setiap zona tersebut.

Penelitian dalam Tugas Akhir ini akan menggunakan training data yang terdiri dari seribu citra yang memiki konten pornografi dan 1000 citra yang tidak memiliki konten pornografi. Kemudian untuk mendeteksi adanya konten pornografi pada citra digunakan

analisis warna, analisis tekstur dan analisis bentuk. Dari analisis warna digunakan fitur 1) jumlah pixel kulit yang terhubung dan 2) persentase pixel kulit dalam citra. Dari analisis tekstur digunakan fitur 1) homogeneity, 2) correlation, 3) energy, dan 4) contrast yang didapatkan setelah dibentuk gray-level co-coccurences matrix. Dan dari analisis bentuk digunakan fitur berupa ketujuh Hu Moment. Setelah fitur didapatkan maka fitur tersebut dilatih dengan menggunakan Support Vector Machine (SVM) sebagai algoritma pembelajarannya. Sistem juga akan mendeteksi adanya wajah pada citra yang bertujuan untuk meningkatkan performanya. Dari metode ini, sistem diharapkan mampu untuk mengenali pola pada citra yang memiliki konten pornografi dan citra yang tidak memiliki konten pornografi.

Penggunaan web service adalah luaran dari Tugas Akhir ini agar situs-situs berbasis forum yang memiliki fitur pengunggahah citra dapat menggunakan layanan ini dalam menjalankan proses bisnisnya.

Kata kunci: *citra, Gray Level Co-Coccurences Matrix, Hu Moment, Image Zoning, pornografi, Support Vector Machine, web service*

DEVELOPMENT OF WEB SERVICE AS PORN DETECTION SERVICE IN DIGITAL IMAGE USING IMAGE ZONING METHOD

Student Name	**: Lourensius Bisma**
NRP	**: 5210100155**
Department	**: Sistem Informasi FTIf – ITS**
Supervisor I	**: Dr.Eng. Febriliyan Samopa, S.Kom, M.Kom**
Supervisor II	**: Radityo P. Wibowo, S.Kom, M.Kom**

ABSTRACT

One of the greates challenge in globalization age is how to process giant data in internet. Moreover if data saved in a site indicates has porn content. There are many sexual harassment cases involving underage children. Mostly, perpetraitor consumes porn content in internet. Therefor, system which automatically filters data into porn category or not is needed. This undergraduate thesis aims to solve this problems by using web service to provide service enabling porn detection in digital image. Image Zoning is used as a method in this research. Image Zoning will make zones in digital image to assess potentiality of porn content in defined zone.

The research will use training data includes 1000 porn images and 1000 neutral images. Color analysis, texture analysis, and shape analysis is used to detect porn content in an image. Some features exrtacted from color analysis are 1) amount of skin pixel connected and 2) percentage of skin pixel in image. Some features extracted from texture analysis are 1) homogeneity, 2) correlation, 3) energy, and 4) contrast which are obtained from Gray Level Co-Occurences Matrix.. Then, some features extracted from shape analysis is seven Hu Moment. To obtain these features, image processing is used. Then, these features are trained by using Support Vector Machine (SVM) as learning

algorithm. System will also detect face inside image whose purpose is to increase the performance. Hopefully, the method being purposed can classify pattern from image which has porn content and image which doesn't have porn content.

Web service is the output from this undergraduate thesis in order to provide porn service in several forum-based sites to filter its images from porn content.

Keywords : *image, Image Zoning, Gray Level Co-Coccurences Matrix, Hu Moment, pornography, Support Vector Machine, web service*

PEMBUATAN *WEB SERVICE* SEBAGAI LAYANAN PENDETEKSI KONTEN PORNOGRAFI PADA CITRA DIGITAL DENGAN METODE *IMAGE ZONING*

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :

LOURENSIUS BISMA
NRP. 5210 100 155

Surabaya, Juli 2014

KETUA
JURUSAN SISTEM INFORMASI

Dr.Eng. Febriliyan Samopa S.Kom., M.Kom.
NIP 19730219 199802 1 001

PEMBUATAN *WEB SERVICE* SEBAGAI LAYANAN PENDETEKSI KONTEN PORNOGRAFI PADA CITRA DIGITAL DENGAN METODE *IMAGE ZONING*

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :

LOURENSIUS BISMA
5210 100 155

Disetujui Tim Penguji : Tanggal Ujian : Juli 2014
Periode Wisuda : September 2014

Dr. Eng. Febriliyan S, S. Kom, M. Kom **(Pembimbing I)**

Radityo P. Wibowo, S.Kom, M.Kom **(Pembimbing II)**

Bambang Setiawan, S. Kom, M.T **(Penguji 1)**

Hatma Suryotrisongko, S.Kom, M.Eng **(Penguji 2)**

KATA PENGANTAR

Assalamualaikum Warohmatullah.

Salam sejahtera.

Terima kasih yang sebesar-besarnya kepada Tuhan Yang Maha Esa karena berkat karunia-Nya, Tugas Akhir berjudul "Pembuatan *Web Service* Sebagai Layanan Pendeteksi Konten Pornografi pada Citra dengan Menggunakan Metode *Image Zoning*" dapat diselesaikan dengan baik.

Terimakasih dan penghargaan setinggi-tingginya penulis sampaikan kepada:

1. Kedua orang tua dan adikku serta sanak saudara tercinta yang tak hentinya memanjatkan doa setiap saat demi kelancaran penulis dalam menyelesaikan masa studi serta mendapatkan ilmu yang berkah dan manfaat.
2. Bpk. Dr.Eng. Febriliyan Samopa, S.Kom, M.Kom dan Bpk Radityo P. Wibowo, S.Kom, M.Kom. selaku dosen pembimbing yang telah memberikan bimbingan dengan penuh kesabaran kepada penulis selama tugas akhir ini dikerjakan.
3. Ibu Mahendrawati, ST., MSc., PhD selaku dosen wali yang selalu mendampingi dan mengarahkan penulis selama masa studi di kampus perjuangan.
4. Jajaran pengajar dan karyawan JSI yang telah banyak memberikan ilmu dan layanannya selama penulis berada di Jurusan Sistem Informasi ITS.
5. Teman-teman keluarga besar FOXIS yang telah memberikan kenangan yang tak pernah terlupakan.
6. Teman-teman seperjuangan Lab E-Business yang selalu menghadirkan atmosfir Wisuda 110.

7. Berbagai pihak yang tidak dapat disebutkan satu persatu.

Penulis berharap semoga keberadaan Tugas Akhir ini bermanfaat dalam pengembangan ilmu pengetahuan dan berbagai pihak. Terimakasih.

Surabaya, Juni 2014

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

Bab pendahuluan ini membahas tentang latar belakang pengerjaan tugas akhir, rumusan permasalahan yang dihadapi dalam pengerjaan tugas akhir, batasan permasalahan pengerjaan tugas akhir, tujuan pengerjaan tugas akhir, dan manfaat dari pengerjaan tugas akhir.

1.1 Latar Belakang

Internet telah menjadi kebutuhan vital bagi sebagian masyarakat dunia. Tidak hanya untuk mendukung pekerjaan mereka namun juga sebagai media untuk berbagi ide, bersosialisasi dan membuat komunitas untuk sebuah tujuan tertentu. Salah satu buktinya adalah jumlah pengguna situs facebook yang mencapai 1.19 miliar akun atau bisa dikatakan sama dengan jumlah penduduk di Cina (Kompas, 2013). Tidak hanya facebook, situs-situs lain seperti youtube, 9gag, kaskus, dan blogspot yang menyediakan layanan serupa juga mampu masuk menjadi 500 situs paling populer di dunia (Alexa, 2014). Bahkan youtube yang menyediakan layanan berbagi video mampu menjadi peringkat ketiga dari seluruh situs di dunia.

Dengan banyaknya pengguna dan frekuensi aktivitas yang mereka lakukan pada situs-situs tersebut maka dapat dipastikan bahwa volume data elektronik yang tersimpan dalam jaringan internet semakin membeludak. Menurut Brian Gentile (2012), seorang CEO dari perusahaan intelegensia bisnis perangkat lunak, lebih dari tiga exabyte data baru terbentuk setiap hari. Tingginya volume data ini tentu menjadi permasalahan terutama dalam hal kapasitas penyimpanan dan pengelolaan data.

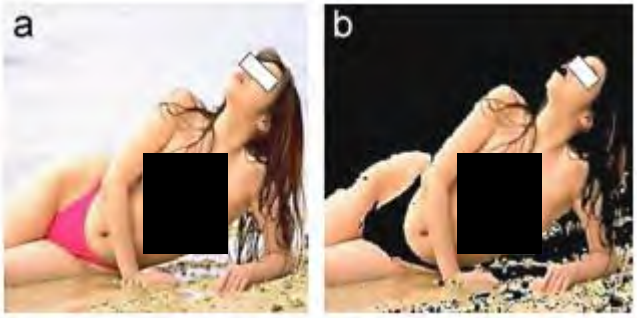
Pornografi adalah penggambaran tingkah laku secara erotis dengan lukisan atau tulisaan untuk membangkitkan nafsu berahi (Kamus Besar Bahasa Indonesia, 2014). Beberapa format

dari pornografi adalah berupa citra atau gambar, file dan *streaming* video, webcam, Mp3, dan file teks. Indonesia merupakan salah satu negara yang menentang keras persebaran pornografi yang kemudian diatur dalam Undang-Undang Pornografi. Hal ini karena pornografi menyebabkan pemirsanya terangsang untuk melakukan hal-hal yang ada dalam media pornografi yang dilihatnya. Salah satu contoh kasus terjadi di Kendari dimana empat pelajar SMA tega memperkosa teman sebayanya akibat sering menonton video porno (Tekno Kompas, 2013). Hal ini membuktikan bahwa pornografi memberikan dampak buruk pada siapa saja yang mengonsumsinya, terutama bagi mereka yang masih di bawah umur.

Salah satu tantangan dalam pengelolaan data bervolume tinggi di internet adalah bagaimana sebuah data tidak mengandung konten-konten berbahaya seperti pornografi. Perlu ada sistem yang secara otomatis mampu mendeteksi apakah sebuah data tergolong ke dalam konten pornografi atau tidak. Sebuah studi menyatakan bahwa sebanyak 30% dari *traffic* internet adalah konten pornografi (Indo-Asian News Service, 2012). Volume tinggi atas konten pornografi juga ditemukan pada situs penyedia file torrent bernama Pirate Bay. Sebanyak 30% dari seluruh file torrent terindikasi sebagai konten pornografi (ALEX CO, 2014). Berdasarkan penjelasan tersebu maka dapat disimpulkan bahwa persebaran data yang mengandung konten pornografi cukup tinggi dan dapat dengan bebas diakses oleh seluruh pengguna internet.

Citra digital adalah representasi diskrit dari atribut pada data, yaitu informasi spasial (*layout*) dan intensitas (warna) (Chris Solomon, 2010). Citra digital adalah istilah lain untuk data elektronik bertipe gambar. Citra digital dapat diolah dengan menerapkan *image processing*. *Image processing* atau pemrosesan citra adalah studi mengenai pengolahan citra sehingga citra mampu dimanipulasi namun tetap memiliki kerangka yang sama dengan aslinya. Salah satu penerapan pemrosesan citra adalah *face recognition* dimana sebuah gambar

dapat dianalisis untuk ditemukan pola atau struktur wajah di dalamnya. Selain untuk mengenali adanya wajah, penerapan pemrosesan data juga dapat digunakan untuk mengenali bentuk lainnya.



Gambar 1.1 *Skin Filtering* dengan (a) citra asal dan (b) citra setelah diproses

(Sumber: Lee J *et. al*, 2006)

Penerapan pemrosesan citra digital telah digunakan untuk mendeteksi kontek pornografi pada sebuah citra sejak tahun 1996. Forsyth dan Fleck (1996) melakukannya dengan cara mengkombinasikan *skin filter* menggunakan warna dan tekstur untuk mendeteksi bagian telanjang pada citra tubuh manusia. Terdapat beberapa metode untuk melakukan *skin filter*, antara lain dengan menggunakan model HSV dan YCbCr. Contoh hasil dari penerapan kedua model tersebut dapat dilihat pada gambar 1.1. Sayangnya metode tersebut hanya memiliki tingkat akurasi sebesar 60% karena hanya mencari keberadaan warna-warna kulit manusia. Padahal perlu ada analisis tambahan selain itu. Lee (2006) menunjukkan algoritma berbasis *learning-based chromatic distribution scheme*. Algoritma ini memiliki fitur tambahan pada *training set* yang digunakan, yaitu bentuk. Hasil dari algoritma ini menunjukkan keakuratan hingga 90%.

Metode pemrosesan citra digital untuk mendeteksi konten pornografi pada citra terus berkembang hingga pada tahun 2012

muncul algoritma bernama *Image Zoning*. Metode ini membagi sebuah citra ke dalam beberapa zona dan mengecek persentase luas citra yang terindikasi sebagai kulit manusia. Penerapan metode ini adalah karena citra digital yang mengandung konten pornografi biasanya terletak di zona tertentu pada citra. Metode ini menunjukkan tingkat akurasi yang lebih tinggi dibanding dengan algoritma yang diusulkan oleh Lee, yakni sebesar 97.4%. Sayangnya metode ini hanya mendeteksi seberapa telanjang manusia saja sehingga untuk beberapa kasus, metode ini dapat gagal diterapkan untuk mendeteksi adanya konten pornografi pada citra.

Indonesia juga memiliki penelitian untuk mendeteksi konten pornografi pada citra. Isa dan Mariana (2009) menggunakan teknik *TSL Color Space* untuk mengecek seberapa telanjang manusia yang terdapat pada sebuah citra dengan tingkat akurasi sebesar 72.19%. Effendy *et. al.* (2010) menggunakan jaringan saraf tiruan untuk mendeteksi area-area sensitif atau yang merepresentasikan adanya pornografi pada citra dengan membagi citra ke dalam zona persegi dengan ukuran tertentu. Kemudian Hidayatullah dan Hapsari (2013) mengusulkan konsep yang hampir sama dengan Effendy namun dengan algoritma yang berbeda, yaitu algoritma Viola dan Jones dengan tingkat akurasi sebesar 83.75%.

Web service adalah layanan yang tersedia dan dapat digunakan oleh dua atau lebih perangkat lunak pada jaringan internet. Bentuk nyata dari *web service* adalah aplikasi web yang didesain untuk menyediakan fungsi tertentu dimana fungsi tersebut dapat digunakan oleh web lain yang membutuhkan. Terdapat tiga komponen utama dari *web service* yaitu: *Web Services Description Language* (WSDL), *Simple Object Access Protocol* (SOAP), dan *Universal Description, Discovery, and Integration* (UDDI). Pengembangan lebih lanjut dari *web service* adalah web API yaitu konsep dari *web service* yang lebih menonjolkan aspek keringanan karena tidak membutuhkan protokol web service (SOAP dan WSDL). Contoh penerapan web

API adalah Facebook API dimana web yang menggunkan layanan tersebut mampu menggunakan beberapa fitur pada facebook seperti menulis komentar dan berbagi status.

Support Vector Machine (SVM) adalah teknik untuk melakukan prediksi, baik dalam kasus klasifikasi maupun regresi yang dipopulerkan pada tahun 1995 (Hsu, Chang, & Lin, 2010). SVM membuat sistem mampu untuk "belajar" dari data yang diberikan. Teknik SVM sangat akurat digunakan terutama untuk klasifikasi citra (Mercier & Lennon, 2003).

Dengan mengkombinasikan metode pemrosesan data dan *web service* maka sebuah sistem diusulkan untuk menyaring citra digital dari konten pornografi diusulkan. Sistem ini akan meminta inputan berupa citra digital kemudian memprosesnya dan memberikan output apakah citra tersebut termasuk ke dalam konten pornografi atau tidak. Teknik SVM akan digunakan untuk melakukan prediksi karena kehandalannya dalam. Sasaran dari sistem ini adalah website yang sering melakukan pengunggahan citra, seperti kaskus dan 9gag. Harapannya dibangunnya sistem ini adalah situs-situs tersebut mau menggunakan layanan ini dan turut berkontribusi dalam mengurangi tingkat pornografi di internet.

1.2 Rumusan Masalah

Perumusan masalah yang akan dibahas pada Tugas Akhir ini, antara lain adalah:

1. Bagaimana mengimplementasikan konsep *Image Zoning* beserta ekstraksi fitur ke dalam bahasa pemrograman Java?
2. Bagaimana mengimplementasikan konsep *Support Vector Machine* (SVM) untuk melakukan klasifikasi citra?
3. Bagaimana menghubungkan konsep *Support Vector Machine* (SVM) dengan bahasa pemrograman Java?

1.3 Batasan Masalah

Batasan masalah pada tugas akhir ini adalah sebagai berikut:

1. Citra digital yang diizinkan untuk diolah adalah jpeg, png, dan bmp dan memiliki resolusi minimal 256 x 256.
2. Citra digital yang digunakan tidak mencakup tidak mengalami proses konversi. Contoh: konversi *grayscale*.
3. Sistem tidak mendeteksi citra yang memiliki konten pornografi dengan lingkungan kartun.
4. Sistem tidak menangani kasus untuk citra yang memiliki konten pornografi namun terdapat *body painting* atau *tattoo*.
5. Sistem tidak menangani kasus untuk citra yang menggambarkan adegan asusila dengan busana.
6. Sistem tidak mendeteksi citra dengan cara menemukan alat vital manusia pada citra.
7. Sistem hany mendeteksi citra dengan ras kaukasoid dan mongoloid.

1.4 Tujuan

Tujuan yang ingin dicapai melalui Tugas Akhir ini adalah membuat *web service* yang menyediakan layanan untuk mendeteksi konten pornografi pada citra digital yang tersimpan pada *cloud server*.

1.5 Manfaat dan Relevansi

Relevansi dan manfaat yang dapat diberikan Tugas Akhir ini adalah sebagai berikut:

1. Tugas Akhir ini memberikan manfaat kepada penulis berupa pengetahuan untuk melakukan pemrosesan citra digital yang dapat digunakan untuk mendeteksi konten pornografi pada citra digital.
2. Tugas Akhir ini memberikan manfaat kepada situs pengunggahan gambar atau yang sejenis untuk melakukan *filtering* citra apakah citra tersebut termasuk ke dalam konten pornografi atau tidak.
3. Tugas Akhir ini memberikan manfaat kepada orang tua dalam hal menjaga anak mereka dari konsumsi konten pornografi di internet.

1.6 Sistematika Pembahasan

Secara garis besar Penulisan dalam Tugas Akhir ini terbagi dalam enam bab, dimana materi dari setiap bab dapat dituliskan sebagai berikut:

BAB I : Pendahuluan

Bab ini berisi uraian mengenai latar belakang permasalahan, tujuan dari Tugas Akhir, manfaat Tugas Akhir, perumusan masalah, batasan masalah serta sistematika yang digunakan dalam pembahasan masalah ini.

BAB II : Tinjauan Pustaka

Pada bab ini akan membahas mengenai teori-teori yang mendukung pembuatan tugas akhir (TA), yaitu tentang citra digital, *RGB Model*, *YcbCr Model*, *skin detection*, *gray level co-occurences matrix*, *texture analysis*, *Hu Moment*, dan *Image Zoning*, *Support Vector Machine* (SVM), libsvm, *web service*, *face detection*, dan *Receiver Operating Characteristic* (ROC).

BAB III : Metode Penelitian

Bab ini membahas langkah-langkah penelitian yang dilakukan selama pengerjaan. Diawali dengan melakukan studi literatur, analisis kebutuhan sistem, pengumpulan data, pembuatan sistem, pengujian dan perbaikan sistem, pengambilan kesimupan dan saran, dan penyusunan Tugas Akhir.

BAB IV : Analisis dan Desain Sistem

Pada bab ini diuraikan hal-hal terkait proses pengumpulan data dan perancangan sistem pada *web service*, antara lain Gambaran Umum Sistem,

Pemilihan Fitur, Pembuatan Flowchart, Pengumpulan Data/Citra, Desain Aplikasi Pembuatan Model Berbasis Desktop, dan Desain *Web Service*.

BAB V : Implementasi dan Uji Coba

Bab ini menjelaskan mengenai lingkungan implementasi sistem, implementasi desain aplikasi berbasis desktop, pengujian aplikasi berbasis desktop, pembuatan file training.txt.model dan range, implementasi desain *web service*, pembuatan aplikasi *client* berbasis PHP, cara penggunaan aplikasi *client*, pengujian aplikasi *client*, analisis keberhasilan dan kegagalan prediksi sistem, analisis ROC, dan penerapan *face detection*.

BAB VI : Penutup

Bab ini berisi kesimpulan dan saran dari seluruh percobaan yang telah dilakukan untuk dibandingkan dengan tujuan dan permasalahan yang ada.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan membahas mengenai teori-teori yang mendukung pembuatan tugas akhir (TA) PEMBUATAN *WEB SERVICE* SEBAGAI LAYANAN PENDETEKSI KONTEN PORNOGRAFI PADA CITRA DIGITAL DENGAN METODE *IMAGE ZONING*, yaitu tentang citra digital, RGB Model, YcbCr Model, skin detection, gray level co-occurences matrix, texture analysis, Hu Moment, dan Image Zoning, Support Vector Machine (SVM), libsvm, dan web service. Hipotesa Sistem dari aplikasi yang akan dibuat. Dengan adanya tinjauan pustaka diharapkan dapat memberikan gambaran secara umum dari penjelasan tugas akhir ini.

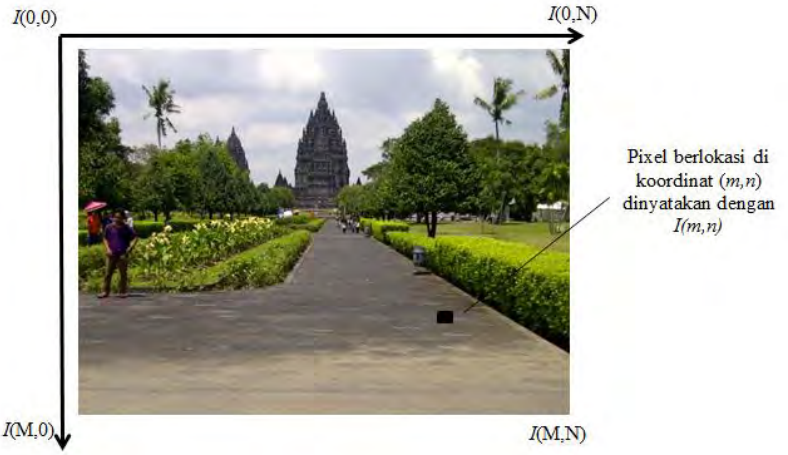
2.1 Citra Digital

Citra Digital dapat dinyatakan sebagai sebuah representasi diskrit dari data yang terdiri dari informasi spasial (*layout)* dan informasi intensitas (warna) (Chris Solomon, 2010). Sebuah citra digital memiliki sekumpulan pixel yang tersebar pada panjang dan lebarnya. Pixel adalah titik yang menyusun sebuah citra digital dalam rentang panjang dan lebar (informasi spasial) dan memiliki satu warna tertentu (informasi warna). Selain informasi spasial dan informasi intensitas, citra digital juga memiliki atribut-atribut lain. Berikut ini adalah atribut-atribut yang dimiliki oleh sebuah citra digital:

1. *Layout*

   Layout menyatakan penyusunan pixel dalam sebuah citra digital dalam rentang panjang dan lebar tertentu. Sebuah citra digital dapat dinyatakan dengan *I(m,n)* yang merepresentasikan respon pixel dalam rentang baris M (*m* = 1,2,..,M) dan rentang kolom N (*n* = 1,2,..,N) dengan *m* menyatakan informasi lokasi baris dan *n* menyatakan informasi lokasi kolom. Perhitungan lokasi *m* dan *n* dimulai

dari pojok kiri atas dari sebuah citra digital sebagai koordinat (0,0). Gambar 2.1 adalah ilustrasi yang menyatakan pernyataan tersebut:



Gambar 2.1 Ilustrasi Citra Digital *I(m,n)* pada rentang M x N
(Sumber: dokumentasi penulis)

2. Warna
   Sebuah citra digital terdiri dari sekumpulan pixel dimana setiap pixel menyimpan satu nilai yang merepresentasikan informasi mengenai warna. Nilai tersebut tersimpan dalam vektor tiga dimensi yang menyatakan sektrum warna dan disebut dengan RGB. RGB merupakan *array* yang menyimpan intensitas warna dasar, yaitu *Red* (R), *Green* (G), dan *Blue* (B).
3. Resolusi
   Resolusi pada citra digital adalah ukuran atau banyaknya pixel. Resolusi dapat dispesifikasikan menjadi tiga jenis, yaitu:
   1) *Spatial Resolution*, adalah jumlah pixel pada sebuah citra digital yang dinyatakan dengan kolom (C) dan baris (R). *Spatial resolution* ditulis dengan format C x

R. Contohnya adalah: 640 x 480, 800 x 600, 1024 x 768 dan sebagainya.

2) *Temporal Resolution*, adalah jumlah citra digital yang tertangkap dalam video dalam satuan periode tertentu. *Temporal resolution* ditulis dengan format *frame per second* atau fps. Contohnya adalah 25 fps, 30 fps, 60fps, dan sebagainya.
3) *Bit Resolution*, adalah jumlah dari nilai intensitas atau warna yang mungkin yang sebuah pixel miliki dan dinyatakan dalam bit. Jumlah warna yang ada pada sebuah citra digital tergantung dari nilai bit-nya. Perhitungannya adalah 2^n dengan *n* adalah jumlah bit. Contohnya adalah citra digital hitam-putih memiliki 2 bit atau 4 warna, citra digital berskala abu-abu (*greyscale*) memiliki 8 bit atau 256 warna, dan citra digital berwarna-warni (*true colour*) memiliki 24 bit atau 16.777.216 warna.

4. Format

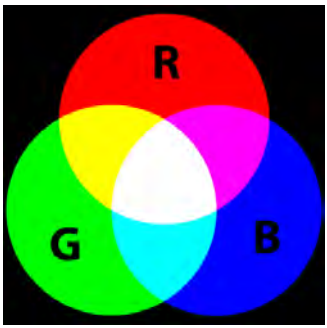
Format adalah informasi mengenai bagaimana data citra tersimpan dan nilai pixel-nya. Tabel 2.1 adalah daftar format untuk citra secara umum berdasarkan Chris (2010).

Tabel 2.1 Daftar Format pada Citra Secara Umum Beserta Keterangannya

Singkatan	Nama	Keterangan
GIF	Graphics interchange format	Terbatas hanya sampai 256 warna.
JPEG	Joint Photographic Experts Group	Paling sering digunakan saat ini dan mampu menyimpan 24 bit – 26 bit warna.
BMP	Bit map picture	Format dasar dari citra.
PNG	Portable network graphics	Format citra terbaru dan didesain untuk menggantikan GIF.

Pada sistem yang akan dibuat, citra adalah sumber daya utama yang akan digunakan sebagai *training data*, data validasi, dan data inputan ketika sistem telah dibuat.

2.2 RGB Model



Gambar 2.2 Ilustrasi *RGB Model*
(Sumber: wikipedia)

RGB adalah model warna dengan melibatkan tiga warna dasar merah (*red*), hijau (*green*) dan biru (*blue*) yang disatukan

untuk membetuk warna lain secara besar (Bazilio *et.al*, 2010). Untuk membentuk warna baru dilakukan pembobotan atau intensitas dari ketiga warna tersebut yang direpresentasikan dengan nilai integer yang tersusun dari nilai 0 untuk hitam dan 1 untuk putih. Jumlah warna yang bisa tercipta pada model RGB tergantung dengan nilai *bit resolution* yang dimiliki sebuah citra. Sebagai contoh untuk citra dengan *bit resolution* 24 bit dengan 16.777.216 warna maka rentang nilai yang dimiliki oleh setiap warna dasar adalah sebesar $R = G = B = \sqrt[3]{16777216}$ atau sebesar 256. Dari sini dapar diketahui bahwa untuk setiap pixel pada citra 24 bit memiliki model RGB mulai (0,0,0) sampai (1,1,1). Ilustrasi dari *RGB Model* dapat dilihat pada gambar 2.2.
Penerapan *RGB Model* dalam Tugas Akhir ini adalah untuk melakukan pemrosesan citra. Sistem yang akan dibuat pertama kali akan mencari nilai RGB dari setiap pixel yang ada pada citra yang diinputkan kemudian nilai RGB tersebut akan disimpan dan digunakan untuk proses analisis.

2.3 YcbCr Model

YCbCr adalah sistem pewarnaan digital yang memisahkan RGB menjadi *luminance* dan *chrominance*. Penerapan YCbCr terdapat pada sistem PAL (pewarnaan *true color*) dan NTSC (pewarnaan kroma) pada televisi. Y dibentuk dengan memberikan bobor pada setiap nilai RGB, selanjutnya nilai Cb dan Cr dihitung dengan menggunakan nilai Y. Hasil ekstraksi nilai Y dapat digunakan untuk mengkonversi gambar menjadi *greyscale*. Berikut ini adalah formula untuk menghitung nilai Y, Cb, dan Cr menurut Singh *et. al* (2003):

$$Y = 0.299\mathrm{R} + 0.587\mathrm{G} + 0.11\mathrm{B} \quad (2.1)$$

$$\mathrm{Cb} = \mathrm{B} - \mathrm{Y} \quad (2.2)$$

$$\mathrm{Cr} = \mathrm{R} - \mathrm{Y} \quad (2.3)$$

Dengan menggabungkan persamaan Cb dan Cr dengan Y maka didapatkan persamaan baru untuk menghitung nilai Cb dan Cr, sebagai berikut:

$$\mathrm{Cb} = 128 - 0.169\mathrm{R} - 0.332\mathrm{G} + 0.500\mathrm{B} \quad (2.4)$$

$$\mathrm{Cr} = 128 + 0.500\mathrm{R} - 0.419\mathrm{G} - 0.081\mathrm{B} \quad (2.5)$$

2.4 *Skin Detection*

Skin detection adalah metode untuk mendeteksi pixel-pixel pada citra yang serupa dengan kulit manusia. *Skin detection* merupakan algoritma utama yang digunakan untuk mendeteksi adanya manusia pada citra. Konsep dari *skin detection* adalah untuk mengetahui seberapa telanjang manusia yang ada pada citra. Fitur yang muncul dari konsep ini adalah persentase luas pixel yang terindikasi sebagai kulit manusia. *Skin detection* dapat dilakukan dengan menggunakan dua pendekatan, yaitu *HSV Model* dan *YCbCr Model*. Namun pada penelitian ini hanya digunakan model YCbCr dikarenakan model ini lebih efektif untuk diterapkan pada *skin detection* (Clayton *et. al.*, 2012). Berikut ini adalah penjelasan dan rentang nilai pixel dengan menggunakan model YCbCr:

- *YCbCr Model*

  Analisis histogram dilakukan pada *YCbCr Model* dari banyak citra untuk mengoptimasi rentang nilai yang sesuai untuk kulit manusia. Chai & Ngan (1999) telah menganalisis dan mendapatkan rentang nilai untuk Cb dan Cr sebagai berikut:

$$77 \leq \mathrm{Cb} \leq 127 \quad (2.6)$$

$$133 \leq \mathrm{Cr} \leq 173 \quad (2.7)$$

Namun Jorge *et. al.* (2011) menyatakan bahwa rentang nilai tersebut hanya berlaku untuk warna kulit ras kaukasoid atau kulit putih padahal manusia tidak hanya berasal dari ras kulit

putih saja. Untuk itu sebuah analisis histogram dilakukan lagi dengan melibatkan warna kulit untuk ras mongoloid dan ras negroid untuk mengoptimasi nilai Cb dan Cr yang sesuai dengan kulit manusia. Berikut ini adalah rentang nilai yang didapatkan setelah melakukan analisis tersebut:

$$80 \leq \mathrm{Cb} \leq 120 \quad (2.8)$$

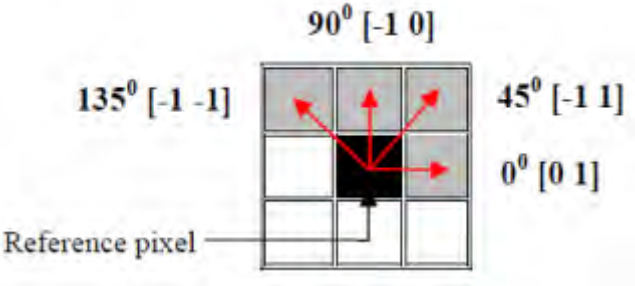
$$133 \leq \mathrm{Cr} \leq 173 \quad (2.9)$$

2.5 *Gray Level Co-Occurences Matrix* (GLCM)

Gray Level Co-Occurences Matrix (GLCM) adalah matriks persegi yang dapat menunjukkan sejumlah informasi mengenai tekstur pada citra (Pathak & Barooah, 2013). GLCM menunjukkan seberapa sering pixel *i* yang disebut dengan *reference pixel* berelasi dengan pixel *j* yang disebut dengan *neighbour pixel* dengan intensitas tertentu muncul. Relasi ini dispesifikasikan lagi dengan atribut lain, yaitu sudut. Secara umum, terdapat empat sudut yang sering digunakan, yaitu 0^o, 90^o, 180^o, dan 270^o. Dari keempat sudut inilah jumlah pixel (*i,j*) yang berkaitan dihitung dan dijumlahkan. Jika diberikan sebuah citra *I* dengan ukuran NxN maka matriks *co-occurence*-nya dapat dicari dengan formula sebagai berikut:

$$\boldsymbol{P}_{(i,j)} = \sum_{x=1}^{N} \sum_{y=1}^{N} \begin{cases} \mathbf{1}, jika\ I(x,y) = i\ and\ I(x + \Delta x, y + \Delta y) = j \\ \mathbf{0}, lainnya \end{cases} \quad (2.10)$$

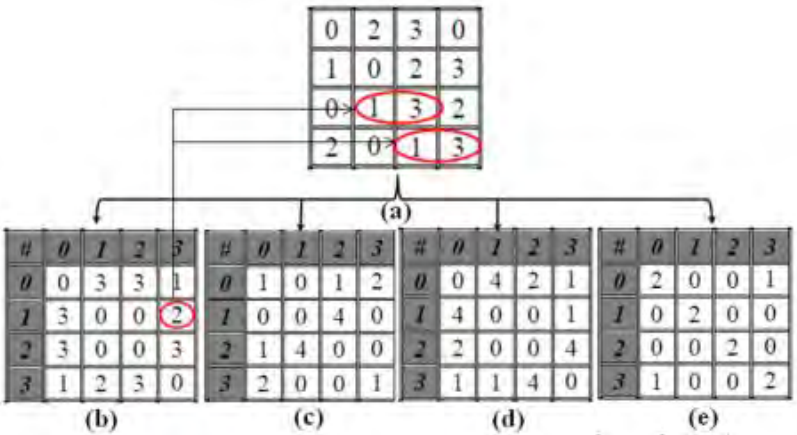
Gambar 2.3 mengilustrasikan detail dari proses untuk memproses *co-occurences matrix* pada citra 4x4. Citra direpresentasikan dengan empat warna *greyscale* dengan nilai 0 – 3 dan satu *neighbour pixel* (d=1). Dengan memakai satu *reference pixel* yang berlokasi di pusat citra maka relasi antara *reference pixel* dan *neighbour pixel* yang diperoleh adalah [-1 0] untuk 90^o, [-1 1] untuk 45^o, [0 1] untuk 0^o, [-1 -1] untuk 135^o.



Gambar 2.3 Arah dari *Co-Occurences Matrix*

(Sumber: Pathak & Barooah, 2013)

Kemudian gambar 2.4 menunjukkan proses pembuatan *co-occurences matrix* dimana (a) adalah citra yang digunakan dengan nilai intensitas per pixel telah dikonversi ke dalam bilangan bulat, (b) adalah *co-occurences matrix* yang terbentuk dari sudut 0^o, (c) adalah adalah *co-occurences matrix* yang terbentuk dari sudut 45^o, (d) adalah *co-occurences matrix* yang terbentuk dari sudut 90^o, (e) adalah *co-occurences matrix* yang terbentuk dari sudut 135^o. Nilai pada setiap *cell* dari matriks yang telah terbentuk dicari dengan menghitung jumlah pixel yang berelasi sama. Semisal untuk relasi [1 3] untuk sudut 0^0 diperoleh jumlahnya adalah sebesar 2. Nilai ini kemudian dimasukkan ke dalam *cell* dengan baris 1 dan kolom 3. Hal ini dilakukan sampai semua *cell* terisi.



Gambar 2.4 Pembuatan Co-Occurences Matrix dengan (a) citra yang digunakan, (b), (c), (d), dan (e) adalah adalah co-occurences matrix yang terbentuk dari sudut 0^o, 45^o, 90^o, dan 135^o
(Sumber: Pathak & Barooah, 2013)

2.6 *Texture-based Analysis*

Texture-based analysis adalah analisis yang dilakukan untuk memberikan nilai pada citra digital dengan menggunakan tekstur-nya. Analisis ini dilakukan dengan menggunakan GLCM yang merepresentasikan level ke-abu-abu-an pada baris *i* dan kolom *j* dalam relasi spasial dalam sebuah citra, dimana mengindikasikan baris dan kolom dari *neighbour matrix* (Clayton, Eulanda, & Eduardo, 2012). Relasi spasial ini kemudian dihubungkan dengan sudut θ. Dari GLCM yang telah dibangun maka sejumlah data statistik diekstrak, antara lain: 1) *homogeneity*, 2) *correlation*, 3) *contrast* dan 4) *energy*. Dari penelitian yang telah dilakukan didapatkan nilai yang optimal adalah dengan $d = 1$, dan dengan sudut delapan arah dengan sudut θ yang besarnya= [0, 45, 90, 135, 180, 225, 270, 315]. Berikut ini adalah formula untuk menghitung *homogeneity*, *correlation*, *contrast* dan *energi*:

1. *Contrast*
   Adalah pengukuran yang digunakan untuk melihat seberapa besar perbedaan tekstur dalam sebuah citra. Pengukuran ini dilakukan untuk setiap pixel dalam satu citra penuh. Untuk citra dengan warna konstan, nilai *contrast* adalah 0. Berikut ini adalah formula untuk menghitung nilai *contrast*:

$$Con = \sum_{i,j}(i-j)^2 P(i,j) \qquad (2.11)$$

2. *Correlation*
   Correlation adalah pengukuran untuk mengetahui bagaimana sebuah pixel berkorelasi dengan pixel lainnya (*neighbour pixel*). Nilai *correlation* adalah 1 untuk pixel yang berkorelasi secara komplit. Sebelum menemukan nilai *correlation*, terdapat variabel *mean* (μ) untuk setiap *i* dan *j*. Setelah itu langkah selanjutnya adalah menghitung nilai standar deviasi (σ) untuk setiap *i* dan *j*. Berikut ini adalah formula untuk menghitung *mean*, *standar deviasi*, dan *correlation*:

$$\mu i = \sum_{1,j=0}^{N-1} i \,.\, P(i,j) \qquad (2.12)$$

$$\mu j = \sum_{1,j=0}^{N-1} j \,.\, P(i,j) \qquad (2.13)$$

$$\sigma i = \sqrt{\sum_{i,j=0}^{N-1} P(i,j)(i-\mu i)^2} \qquad (2.14)$$

$$\sigma j = \sqrt{\sum_{i,j=0}^{N-1} P(i,j)(j-\mu j)^2} \qquad (2.15)$$

$$Corr = \sum_{i,j} P(i,j) \frac{(i-\mu i)(j-\mu j)}{\sigma i \sigma j} \qquad (2.16)$$

3. *Energy*
   Energy adalah pengukuran untuk mengetahui jumlah kuadrat dari GLCM, yang menghitung pasangan pixel dalam *grayscale*. Ketika sebuah citra memiliki warna yang semakin konstan maka nilai *energy* hampir mendekati nilai 1. Dan ketika citra memiliki warna yang heterogen maka nilai *energy* hampir mendekati nilai 0. Berikut ini adalah formula untuk menghitung *energy*:

$$Enrg = \sqrt{\sum_{i,j} P^2(i,j)} \qquad (2.17)$$

4. *Homogeneity*
   Homogeneity adalah sebuah nilai yang merepresentasikan kedekatan dari distribusi elemen pada sisi diagonal GLCM. Berikut ini adalah formula untuk menghitung nilai *homogeneity*:

$$Hmg = \sum_{i,j} \frac{P(i,j)}{1+(i-j)^2} \qquad (2.18)$$

Semisal kita mendapatkan GLCM sebagai berikut:

	0	1	2	3
0	0.1	0	0.4	0.3
1	0	0.2	0.2	0
2	0	0.3	0.2	0.3
3	0	0.1	0	0

Gambar 2.5 Contoh GLCM dengan panjang dan lebar 4 x 4

Maka perhitungan untuk *contrast*, *correlation*, *homogeneity*, dan *energy* adalah sebagai berikut:

$$Con = (0-0)^2 . 0.1 + (0-1)^2 . 0 + (0-2)^2 . 0.4 + (0-3)^2 . 0.3 + (1-0)^2 . 0 + (1-1)^2 . 0.2 + (1-2)^2 . 0.2 + (1-3)^2 . 0 + (2-0)^2 . 0 + (2-1)^2 . 0.3 + (2-2)^2 . 0.3 + (2-3)^2 . 0.3 + (3-0)^2 . 0 + (3-1)^2 . 0.1 + (3-2)^2 . 0 + (3-3)^2 . 0 = \mathbf{9.2}$$

$$\mu i = 0 * 0.1 + 0 * 0 + 0 * 0.4 + 0 * 0.3 + 1 * 0 + 1 * 0.2 + 1 * 0.2 + 1 * 0 + 2 * 0 + 2 * 0.3 + 2 * 0.2 + 2 * 0.3 + 3 * 0 + 3 * 0.1 + 3 * 0 + 3 * 0 = \mathbf{2.3}$$

$$\mu j = 0 * 0.1 + 0 * 0 + 0 * 0 + 0 * 0 + 1 * 0 + 1 * 0.2 + 1 * 0.3 + 1 * 0.1 + 2 * 0.4 + 2 * 0.2 + 2 * 0.2 + 2 * 0 + 3 * 0.3 + 3 * 0 + 3 * 0.3 + 3 * 0 = \mathbf{4.0}$$

$$\sigma i = \sqrt{\begin{matrix} 0.1.(0-2.3)^2 + 0.(0-2.3)^2 + 0.4.(0-2.3)^2 \\ +0.3.(0-2.3)^2 + 0.(1-2.3)^2 + 0.2.(1-2.3)^2 \\ +0.2.(1-2.3)^2 + +0.(1-2.3)^2 + 0.(2-2.3)^2 \\ +0.3.(2-2.3)^2 + +0.2.(2-2.3)^2 + 0.3.(2-2.3)^2 \\ +0.(3-2.3)^2 + 0.1.(3-2.3)^2 + 0.(3-2.3)^2 \\ +0.(3-2.3)^2 \end{matrix}}$$

$$\sigma i = \mathbf{2.242}$$

$$\sigma j = \sqrt{\begin{matrix} 0.1.(0-4)^2 + 0.(1-4)^2 + 0.4.(2-4)^2 \\ +0.3.(3-4)^2 + 0.(0-4)^2 + 0.2.(1-4)^2 \\ +0.2.(2-4)^2 + 0.(3-4)^2 + 0.(0-4)^2 \\ +0.3.(1-4)^2 + 0.2.(2-4)^2 + 0.3.(3-4)^2 \\ +0.(0-4)^2 + 0.1.(1-4)^2 + 0.(2-4)^2 \\ +0.(3-4)^2 \end{matrix}}$$

$$\sigma j = \mathbf{3.203}$$

$$Corr = 0.1\frac{(0-2.3)(0-4)}{2.242*3.203} + 0\frac{(0-2.3)(1-4)}{2.242*3.203} + 0.4\frac{(0-2.3)(2-4)}{2.242*3.203} + 0.3\frac{(0-2.3)(3-4)}{2.242*3.203} + 0\frac{(1-2.3)(0-4)}{2.242*3.203} + 0.2\frac{(1-2.3)(1-4)}{2.242*3.203} + 0.2\frac{(1-2.3)(2-4)}{2.242*3.203} + 0\frac{(1-2.3)(3-4)}{2.242*3.203} + 0\frac{(2-2.3)(0-4)}{2.242*3.203} + 0.3\frac{(2-2.3)(1-4)}{2.242*3.203} + 0.2\frac{(2-2.3)(2-4)}{2.242*3.203} + 0.3\frac{(2-2.3)(3-4)}{2.242*3.203} + 0\frac{(3-2.3)(0-4)}{2.242*3.203} + 0.1\frac{(3-2.3)(1-4)}{2.242*3.203} + 0\frac{(3-2.3)(2-4)}{2.242*3.203} + 0\frac{(3-2.3)(3-4)}{2.242*3.203}$$

$$Corr = \mathbf{0.567}$$

$$Enrg = \sqrt{0.1^2 + 0^2 + 0.4^2 + 0.3^2 + 0^2 + 0.2^2 + 0.2^2 + 0^2 + 0^2 + 0.3^2 + 0.2^2 + 0.3^2 + 0^2 + 0.1^2 + 0^2 + 0^2}$$

$$Enrg = \mathbf{0.755}$$

$$Hmg = \frac{0.1}{1+(0-0)^2} + \frac{0}{1+(0-1)^2} + \frac{0.4}{1+(0-2)^2} + \frac{0.3}{1+(0-3)^2} + \frac{0}{1+(1-0)^2} + \frac{0.2}{1+(1-1)^2} + \frac{0.2}{1+(1-2)^2} + \frac{0}{1+(1-3)^2} + \frac{0}{1+(2-0)^2} + \frac{0.3}{1+(2-1)^2} + \frac{0.2}{1+(2-2)^2} + \frac{0.3}{1+(2-3)^2} + \frac{0}{1+(3-0)^2} + \frac{0.1}{1+(3-1)^2} + \frac{0}{1+(3-2)^2} + \frac{0}{1+(3-3)^2}$$

$$Hmg = \mathbf{1.03}$$

Texture-based analysis akan digunakan sebagai fitur yang akan dipasangkan dengan *training data* untuk mencari tekstur yang sesuai dengan kulit manusia. Kita perlu melakukan analisis ini untuk mendapatkan tekstur yang sesuai dengan kulit manusia. Hal ini dilakukan karena masih banyak benda yang memiliki warna sama dengan kulit manusia, contohnya adalah pakaian. Jika orang menggunakan pakaian yang mirip dengan kulit manusia maka sistem akan tetap menganggap bahwa citra tersebut mengandung unsur telanjang.

2.7 *Hu Moment*

Hu Moment adalah sebuah metode yang dapat digunakan untuk mengenali sebuah bentuk pada citra. *Hu Moment* bekerja dengan memanfaatkan persamaan *moment invariants* dan menurunkannya menjadi tujuh persamaan baru untuk membantu mengenali bentuk pada citra. Jika ada sebuah citra dengan nilai intensitas *f(i,j)* dengan *i* sebagai baris dan *j* sebagai kolom dan *H* adalah tinggi dan *W* adalah lebar citra maka momen invarian yang mentransformasikan fungsi citra *f(i,j)* adalah sebagai berikut:

$$m_{pq} = \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} i^p j^q f(i,j) \quad (2.19)$$

Setelah itu perhitungan dilanjutkan dengan menghitung nilai momen pusat dari sebuah citra dengan menggunakan persamaan sebagai berikut:

$$\mu_{pq} = \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} (i - i')^p (j - j')^q f(i,j) \quad (2.20)$$

Dengan *i'* dan *j'* adalah *centroid* dari objek yang dihitung dengan formula sebagai berikut:

$$i' = \frac{m_{10}}{m_{00}} \quad (2.21)$$

$$j' = \frac{m_{01}}{m_{00}} \quad (2.22)$$

Besaran m_{00} adalah jumlah pixel yang membentuk objek sedangkan m_{10} dan m_{11} adalah pusat massa objek. Momen pusat yang terbentuk sensitif terhadap transformasi rotasi dan penskalaan sehingga perlu dilakukan normalisasi terhadap momen pusat μ_{pq} (Sholahuddin, 2012). Kemudian dilakukan normalisasi pada momen pusat tersebut dengan menggunakan formula sebagai berikut:

$$n_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \; ; \gamma = (p + q + 2)/2 \quad (2.23)$$

Berdasarkan momen pusat yang telah dinormalisasi tersebut, Hu (1986) mengenalkan tujuh momen sebagai berikut:

$$\varphi_1 = n_{20} + n_{02} \quad (2.24)$$

$$\varphi_2 = (n_{20} - n_{02})^2 + 4{n_{11}}^2 \quad (2.25)$$

$$\varphi_3 = (n_{30} - 3n_{12})^2 + (3n_{21} - n_{03})^2 \quad (2.26)$$

$$\varphi_4 = (n_{30} + n_{12})^2 + (n_{21} + n_{03})^2 \quad (2.27)$$

$$\varphi_5 = (n_{30} - 3n_{12})(n_{30} + n_{12})[(n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2] + (3n_{21} - n_{03})(n_{21} + n_{03}[3(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2]) \quad (2.28)$$

$$\varphi_6 = (n_{20} - n_{02})[(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2] + 4n_{11}(n_{30} + n_{12})(n_{21} + n_{03}) \quad (2.29)$$

$$\varphi_7 = (3n_{21} - n_{03})(n_{30} + n_{12})[(n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2] - (n_{30} - 3n_{12})(n_{21} + n_{03})[3(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2]) \quad (2.30)$$

Hu Moment akan dijadikan fitur yang akan digunakan dalam kalkulasi SVM untuk mengklasifikasi citra yang memiliki konten pornografi pada Tugas Akhir ini.

2.8 *Image Zoning*

Dalam sebuah citra, daerah pusat merupakan daerah yang paling merepresentasikan maksud dari citra tersebut (Kalva, Enembreck, & Koerich, 2011). Atau jika tidak maka terdapat daerah lain yang merepresentasikannya. Hal ini berlaku juga dalam konteks pornografi, dimana sebuah citra yang memiliki konten pornografi pasti terlihat pada pusat citra tersebut atau pada daerah-daerah tertentu. Berangkat dari pernyataan tersebut, Clayton *et. al.* (2012) mengusulkan sebuah metode bernama *Image Zoning*. Metode ini membagi citra menjadi beberapa zona dan melakukan ekstraksi fitur dari setiap zona tersebut.

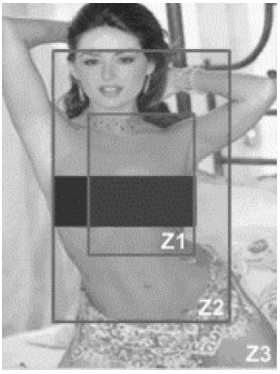
Dengan menganggap *I* adalah sebuah citra, *NZ(I)* adalah jumlah zona pada citra, *Z* menjadi salah satu zona, *FE(Z)* adalah proses ekstraksi fitur dari Z, *FE(I)* adalah vektor dari fitur yang telah diekstrak, *CP* adalah proses klasifikasi maka algoritma *Image Zoning* yang terbentuk adalah sebagai berikut:

```
NZ(I) = k;
  For all image I do;
     divide I into NZ;
     For all zone(Z) do;
```

```
            FE(Z);

            FE(I) = FE(I) + FE(Z);
        End;
    End;
CP(FE(I));
```

Hasil dari algoritma di atas adalah vektor total yang terdiri dari vektor ekstraksi fitur dari setiap zona. Sebagai contoh jika citra akan dibagi menjadi tiga zona maka ilustrasinya adalah seperti yang ditunjukkan pada gambar 2.6 di bawah ini:



Gambar 2.6 Ilutrasi Penggunaan Metode *Image Zoning* dengan zona = 3
(Sumber: Clayton *et. al.*, 2012)

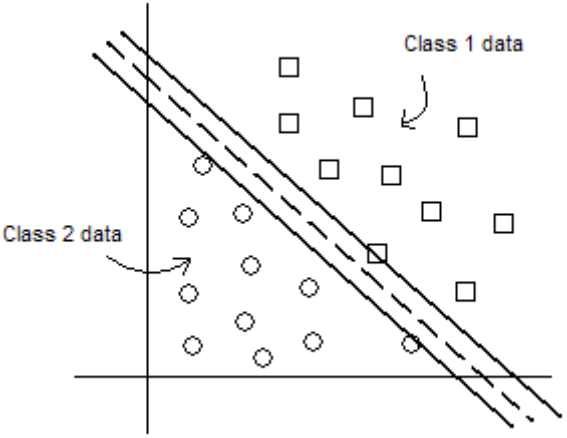
Penerapan *Image Zoning* ternyata juga dilakukan pada kasus pengenalan bentuk pada citra. Contohnya adalah Pirlo dan Impedovo (2012) yang menggunakan metode *Image Zoning* untuk melakukan pengenalan karakter latin. Namun metode *Image Zoning* yang dilakukan pada penelitian ini membagi citra menjadi petak-petak kecil yang sama besarnya. Semisal terdapat citra dengan resolusi 100 x 100 akan diterapkan metode *Image*

Zoning dengan cara Pirlo dan Impeduvo maka akan terbentuk petak-petak kecil dengan ukuran $\frac{1}{n}100$ x $\frac{1}{n}100$.

2.9 *Support Vector Machine* (SVM)

Support Vector Machine (SVM) adalah teknik untuk melakukan prediksi, baik dalam kasus klasifikasi maupun regresi yang dipopulerkan pada tahun 1995 (Hsu, Chang, & Lin, 2010). SVM berada dalam satu kelas dengan *Artificial Neural Network* (ANN) dalam hal fungsi dan kondisi permasalahan. Dalam penerapannya, performa SVM lebih baik dibandingkan dengan ANN karena ANN hanya menemukan solusi *local optimal* sedangkan SVM menemukan solusi *global optimal*. Solusi *local optimal* berarti ketika terdapat perbedaan pada *training data* yang diberikan maka solusinya juga akan berubah, sedangkan *global optimal* berarti solusi yang dihasilkan sama untuk seluruh *training data*. SVM telah diuji oleh banyak ilmuwan dan mampu menyelesaikan solusi terbaik dalam menangani bidang *gene expression analysis*, finansial, cuaca, dan bidang kedokteran.

Gambar 2.7 mengilustrasikan metode SVM untuk mengklasifikasikan beberapa data ke dalam dua kelas. SVM mampu membentuk ambang untuk membatasi data melalui data-data yang berada pada ambang kelas.



Gambar 2.7 Ilustrasi Metode SVM

(Sumber: cnx.org)

Radial Basis Function (RBF) Kernel adalah salah satu jenis kernel yang digunakan pada SVM. Kernel ini mampu membuat batasan yang rumit sehingga hasil prediksi tinggi. Kernel ini juga efektif digunakan untuk mengklasifikasi citra jika dibandingkan jenis kernel lainnya (Yang & Moghaddam, 2000).

Selain pemilihan jenis kernel, hal lain yang perlu diperhatikan untuk menggunakan metode SVM adalah pemilihan parameter yang paling tepat. Hal ini dilakukan untuk mengoptimasi hasil prediksi. *Grid Search* adalah metode yang digunakan untuk melakukan pemilihan fitur. *Grid Search* bekerja dengan cara mencari semua kemungkinan nilai C dan γ yang menghasilkan tingkat akurasi paling tinggi.

Dalam Tugas Akhir ini, SVM dengan kernel RBF akan digunakan untuk mengklasifikasikan citra ke dalam kelas pornografi dan non-pornografi. Setelah didapatkan model dari *training data* maka dilakukan *Grid Search* untuk mengoptimasi model sehingga menghasilkan tingkat akurasi yang tinggi.

2.10 LibSVM

LibSVM adalah sebuah *framework* yang mampu mendukung metode SVM dalam berbagai lingkungan pemnrograman. LibSVM juga mampu mendukung bahasa pemrograman Java. LibSVM ini yang akan digunakan untuk membantuk proses klasifikasi dengan metode SVM pada lingkungan Java. Berikut ini adalah isi dari libsvm untuk lingkungan Java beserta penjelasannya:

- Svm.java: berfungsi untuk menjalankan seluruh fungsi SVM pada *framework* LibSVM.
- Svm_model.java: berfungsi untuk menginisiasi model atau data yang telah diolah melalui proses *training*.
- Svm_node.java: berfungi untuk menginisiasi seluruh fitur yang akan digunakan untuk proses klasifikasi pada LibSVM.
- Svm_parameter.java: berfungsi untuk menginisiasi seluruh parameter yang dibutuhkan.
- Svm_print_interface.java: berfungsi menampilkan hasil pengolahan dari LibSVM baik pada proses *training* maupu *testing*.

Selain kelima file tersebut LibSVM juga menyediakan fungsi untuk melakukan *scaling*, *training*, dan *predicting*. Ketiga file tersebut ditangani oleh file svm_scale.java, svm_train.java, dan svm_predict.java. *Scaling* dilakukan untuk mengkonversi nilai dari fitur atau atribut yang akan digunakan ke dalam rentang nilai antara -1 sampai 1. Hal ini penting karena teknik SVM cenderung tidak akurat jika menangani nilai dengan bilangan bulat yang lebih dari 1.

Untuk menggunakan LibSVM, sebuah model yang berisi hasil olahan *training data* perlu dibuat. Model ini dibentuk setelah melakukan proses *scaling* dan *training* pada LibSVM. Selain itu, sebuah file yang berisi jangkauan nilai minimum dan maksimum dari *training data* juga akan terbentuk setelah melakukan proses *scaling*. Setelah kedua file ini dibentuk, barulah LibSVM akan

dapat melakukan prediksi dari data yang diinputkan. LibSVM juga menyediakan fitur untuk melakukan *Grid Search* yang berfungsi untuk mengoptimasi model sehingga meningkatkan tingkat akurasi.

2.11 *Web Service*

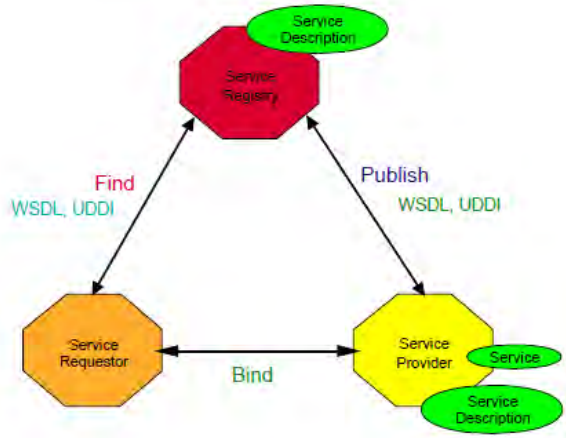
Web service adalah sebuah antarmuka yang menggambarkan sekumpulan fungsi yang dapat diakses dalam jaringan tertentu (pada umumnya internet) pada pesan XML standar (Kreger, 2001). Sebuah *web service* digambarkan menggunakan sebuah standar yang disebut *service description* yang melingkupi seluruh detail yang dibutuhkan untuk berinteraksi dengan layanan, termasuk format, protokol transportasi, dan lokasi.

Arsitektur *web service* berdasarkan interaksi antara tiga peran, yaitu: *service provider*, *service registry,* dan *service requestor*. *Service provider* mendefinisikan *service description* dan mempublikasikannya kepada *service requestor* atau *service registry*. Berikut ini adalah peran dari ketiga elemen tersebut dalam sebuah arsitektur *web service*:

- *Service provider*. Dari perspektif bisnis merupakan pemilik layanan. Sedangkan dari perspepktif arsitektur, merupakan *platform* yang menyediakan akses pada layanan.
- *Service requestor*. Dari perspektif bisnis, merupakan pihak yang meminta beberapa layanan untuk dipuaskan. Dari perspektif arsitektur, merupakan aplikasi yang mencari dan menginisiasi interaksi dengan sebuah layanan.
- *Service Registry*. Merupakan *registry* layanan yang dapat dicari dimana *service provider* mempublikasikan *service description* yang dimilikinya. Secara fungsi, *service registry* mengikat layanan yang disediakan oleh *service provider* sehingga *service requestory* dapat menggunakan layanan yang sama dengan *service provider*. Tetapi *service registry*

bersifat opsional karena *service provider* dapat langsung mengirimkan layanannya kepada *service requestory*.

Ilustrasi dari ketiga elemen ini dapat dilihat pada gambar 2.8 di bawah ini:



Gambar 2.8 Ilustrasi Peran *Service Provider*, *Service Requestory*, dan *Service Registry*

(Sumber: Kreger, 2001)

2.12 *Face Detection*

Untuk meningkatkan performa system yang akan dibangung, digunakan algoritma *face detection*, yaitu algoritma yang akan mendeteksi adanya bagian wajah pada sebuah citra. Penerapan *face detection* dilakukan karena terdapat kemungkinan bahwa citra memiliki mode wajah penuh (*full face*) yang dapat dianggap sebagai konten pornografi oleh sistem. *Face detection* akan dikerjakan ketika citra terindikasi memiliki konten pornografi. Luasan wajah kemudian dibandingkan dengan luasan citra. Jika nilainya lebih besar dari 0.5 maka citra memiliki mode wajah penuh dan dikategorikan tidak memiliki konten pornografi.

Pada Tugas Akhir ini, sebuah *library* bernama *Jon's Java Imaging Library* (JJIL). JJIL merupakan salah satu *open source*

yang mampu untuk menangani pemrosesan *face detection*. Terdapat dua file *.jar* yang harus ditempelkan pada system jika ingin menggunakan JJIL, yaitu *JJILCore.jar* dan *JJIL-J2SE.jar*. Gambar 2.9 mengilustrasikan penerapan JJIL pada sebuah citra untuk kasus *face detection*.

Gambar 2.9 Ilustrasi Hasil Pemrosesan JJIL untuk Kasus *Face Detection* (kiri=input; kanan=output)
(Sumber: dokumentasi penulis)

2.13 *Receiver Operating Characteristic* (ROC)

Receiver Operating Characteristic (ROC) adalah sebuah metode untuk menguji kemampuan sistem prediksi berdasarkan tabel kontigensi (Kadarsah, 2010). Tabel kontigensi adalah table yang dapat merepresentasikan hasil pengujian system atau model dengan cara memberikan label dari setiap data ke dalam nilai aktual dan nilai prediksi. Masing-masing nilai ini kemudian dibagi menjadi positif dan negative. Tabel 2.2 adalah contoh dari tabel kontigensi.

Tabel 2.2 Contoh Tabel Kontigensi dalam ROC

		Prediksi		Total
		Positif	**Negatif**	
Aktual	**Positif**	a	b	a + b
	Negatif	c	d	c + d
Total		a + c	b + d	a + b + c + d

Pengujian model dilakukan dengan menggunakan nilai *accuracy*, *specificity* dan *sensitivity* sesuai dengan nilai yang terbentuk pada tabel kontigensi. *Accuracy* adalah tingkat akurasi model untuk nilai positif dan negatif, *Specificity* adalah tingkat akurasi model untuk memprediksi nilai negatif, Model yang baik akan memiliki nilai *accuracy*, *specificity*, dan *sensitivity* mendekati nilai 1. sedangkan *sensitivity* adalah tingkat akurasi model untuk nilai positif. *False Positive Rate* (FPR) adalah variabel yang digunakan untuk menilai tingkat *error* untuk nilai positif dan *False Negative Rate* (FNR) adalah variabel yang digunakan untuk menilai tingkat *error* negatif dari model. *Positive Predictive Value* (PPV) adalah variabel yang digunakan untuk menilai tingkat prediksi nilai positif yang muncul dan *Negative Predictive Value* (NPV) adalah variable untuk menilai tingkat prediksi nilai negatif yang muncul. Tabel 2.3 menunjukkan formula untuk menghitung variabel-variabel tersebut.

Tabel 2.3 Formula *Accuracy, Sensitivity, Specificity*, FPR, FNR, PPV, dan NPV

Variabel	Formula
Accuracy	(a + d) / (a + b + c + d)
Sensitivity	a / (a + c)
Specificity	d / (b + d)
FPR	1 - *Specificity*
FNR	1 - *Sensitivity*
PPV	a / (a + b)
NPV	d / (c + d)

2.14 Hipotesis Sistem

Kunci utama untuk menjawab apakah citra memiliki konten pornografi atau tidak adalah dengan keberadaan manusia. Oleh karena itu, sebuah metode untuk mendeteksi kulit manusia dimunculkan. Setelah melakukan ekstraksi terhadap seluruh pixel barulah dilakukan ekstraksi fitur berupa persentase pixel yang merupakan kulit manusia dengan seluruh pixel pada gambar serta jumlah pixel-nya. Sayangnya metode ini masih memiliki akurasi yang rendah, yaitu di bawah 70% (Fleck, Forsyth, & Bregler, 1996). Hal ini dikarenakan manusia tidak dapat dideteksi hanya dengan warna kulit saja.

Untuk meningkatkan akurasi dari algoritma pendeteksi konten pornografi, Zheng *et. al.* (2006) membuat penelitian dengan menggunakan fitur bentuk. Fitur ini didapatkan dengan ketujuh nilai *Hu Moment* yang ada pada seluruh pixel yang terdeteksi sebagai kulit manusia. Akurasi yang dihasilkan lebih tinggi yaitu mencapai 90%. Penelitian lain dilakukan oleh Liang K. *et. al.* (2007) dengan menggunakan fitur *skin blob* atau jumlah tonjolan yang ada pada tubuh manusia. Hal ini mengingat bahwa untuk beberapa kasus, citra dapat dikatakan memiliki konten pornografi jika menampilkan postur bagian vital secara telanjang. Metode ini juga memperoleh nilai yang tinggi, yaitu sebesar 90%. Kemudian Clayton *et. al.* (2012) juga memunculkan metode baru bernama *Image Zoning* dengan membagi citra menjadi beberapa zona dan dilakukan ekstraksi untuk setiap zona. Metode ini berdasar atas representasi citra yang biasanya terletak di beberapa zona tertentu dalam citra. Dengan menggunakan fitur warna kulit dan tekstur, metode ini mampu mencapai akurasi 97.4%. Sayangnya metode ini hanya untuk mendeteksi seberapa telanjang manusia sehingga untuk beberapa kasus, metode ini gagal diterapkan.

Penelitian serupa juga dilakukan untuk mendeteksi adanya konten pornografi pada citra digital di Indonesia. Isa dan Mariana (2009) menggunakan teknik *TSL Color Space* untuk

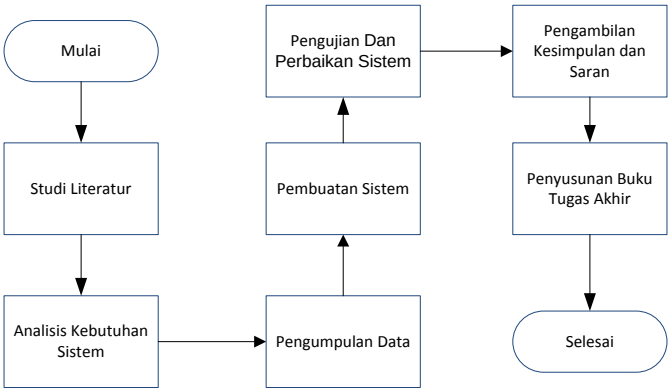
mengecek seberapa telanjang manusia yang terdapat pada sebuah citra. Metode ini sama dengan *skin filtering* dengan tingkat akurasi sebesar 72.19%. Effendy *et. al.* (2010) menggunakan jaringan saraf tiruan untuk mendeteksi area-area sensitif atau yang merepresentasikan adanya pornografi pada citra dengan membagi citra ke dalam zona persegi dengan ukuran tertentu. Kelemahan dari metode ini adalah jika area sensitif tidak masuk ke dalam zona yang telah dibuat maka citra tidak akan diklasifikasikan memiliki konten pornografi. Kemudian Hidayatullah dan Hapsari (2013) mengusulkan konsep yang hampir sama dengan Effendy namun dengan algoritma yang berbeda, yaitu algoritma Viola dan Jones. Mereka berdua mengusulkan metode yang mampu mendeteksi puting pada ebuah citra. Sayangnya kelemahan dari metode ini adalah masih banyak bentuk yang menyerupai puting. Tingkat akurasi yang didapatkan untuk metode ini adalah sebesar 83.75%.

Berdasarkan penelitian yang dilakukan oleh Zheng, Clayton dan Liang K. maka penulis memunculkan hipotesis bahwa sebenarnya tubuh manusia terdiri dari tiga hal, yaitu: 1) warna; 2) tekstur dan 3) bentuk. Penerapan metode Effendy, Hidayatullah, dan Hapsari tidak digunakan untuk menghindari terjadinya *overfitting*. Dari analisis ini, penulis berusaha untuk mengkombinasukan ketiga analisis tersebut ke dalam fitur dan mengoptimalkannya dengan metode *Image Zoning*. Optimasi ini diharapkan dapat berhasil dan diterapkan oleh berbagai situs untuk menyaring seluruh citra dalam server mereka yang memiliki konten pornografi atau tidak.

BAB III
METODOLOGI

Bab ini membahas langkah-langkah penelitian yang dilakukan selama pengerjaan. Diawali dengan melakukan studi literatur, analisis kebutuhan sistem, pengumpulan data, pembuatan sistem, pengujian dan perbaikan sistem, pengambilan kesimupan dan saran, dan penyusunan Tugas Akhir.

Secara umum diagram alur metodologi pengerjaan tugas akhir yang digunakan dapat dilihat pada gambar 3.1:



Gambar 3.1 Diagram Alur Pengerjaan Tugas Akhir

3.1 Studi Literatur

Studi literatur merupakan tahapan untuk mendapatkan referensi mengenai informasi dan data yang dibutuhkan dalam pengerjaan Tugas Akhir. Tahapan ini dilakukan dengan cara menncari penelitian-penelitian yang dapat digunakan unruk mendeteksi konten pornografi pada citra digital. Penulis melakukan pencarian referensi melalui sumber bacaan di Internet, jurnal online, e-book, serta pencarian referensi dari buku cetak. Selain itu, dilakukan

pencarian fitur yang sesuai untuk diterapkan untuk mengklasifikasi citra yang memiliki konten pornografi. Melalui tahapan ini, penulis memiliki pemahaman dasar yang akan digunakan untuk mengerjakan tahapan selanjutnya.

3.2 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem merupakan langkah untuk mengetahui kebutuhan sistem yang akan dibangun, yaitu: 1) jumlah dan jenis data yang akan digunakan dan 2) fungsi yang akan dibangun. Apabila kebutuhan untuk pembuatan sistem telah terpenuhi, maka proses pembuatan sistem akan lebih mudah dibuat.

3.3 Pengumpulan Data

Pengumpulan Data merupakan tahapan untuk mendapatkan kumpulan citra yang terdiri dari citra yang memiliki konten pornografi dan citra biasa yang akan digunakan sebagai *training data*. Jumlah citra yang digunakan adalah sebanyak 2000 buah dengan perbandingan citra dengan konten pornografi dan citra biasa adalah 1:1, artinya 1000 citra yang memiliki konten pornografi dan 1000 citra biasa. Citra biasa terdiri dari wajah, manusia, pemandangan, alat transportasi, dan lain-lain. Sedangkan citra yang memiliki konten pornografi melibatkan ras kaukasoid, mongoloid, dan negroid dengan pencahayaan yang berbeda dan dengan format JPG. Kemudian untuk menguji sistem yang telah dibuat, dikumpulkan juga citra lain yang berdiri sendiri dan tidak dijadikan *training data* sejumlah 1098 buah, dengan 547 citra sebagai citra yang memiliki konten pornografi dan 551 citra sebagai citra biasa.

3.4 Pembuatan Sistem

Pada tahap ini, kegiatan yang dilakukan adalah melakukan konversi spesifikasi desain telah terindentifikasi pada tahap

perancangan sistem menjadi program yang dapat dijalankan. Algoritma *Image Zoning* akan digunakan dengan mengkombinasikannya dengan fitur yang telah dipilih pada tahap sebelumnya. Adapun kegiatan yang dilakukan dalam tahap ini adalah:

1. **Pembuatan fungsi sistem**. Pada tahap ini dibuat fungsi untuk menangani proses input, ekstraksi fitur, mengklasifikasi fitur dengan SVM, dan *output* yang merupakan hasil klasifikasi citra. Pembuatan fungsi masih dibuat dengan kondisi *desktop-based* dengan bantuan Netbeans. Tujuan dari tahap ini adalah meningkatkan efisiensi waktu penulis untuk mengecek keberhasilan fungsi.
2. **Melakukan *training* pada data uji**. Pada tahap ini, seluruh citra yang akan diujikan dimasukkan ke dalam satu folder. Kemudian sistem *desktop-based* yang telah dibangun akan mengekstrak seluruh fitur dari seluruh citra yang ada pada folder tersebut dan menyimpannya sebagai *training set*. *Training set* ini yang akan digunakan untuk mengklasifikasi citra ke dalam pornografi maupun non-pornografi.
3. **Uji coba dan penilaian keberhasilan sistem**. Pada tahap ini, citra selain *training data* yang akan dijadikan bahan uji coba dimasukkan ke dalam satu folder. Kemudian sistem *desktop-based* yang telah dibuat akan mengklasifikasi seluruh citra dan mengeluarkan *output* berupa pesan String yang berisi kata "Porn" jika citra terkategori pornografi atau "Not Porn" jika citra terkategori non-pornografi. Selain itu sistem juga akan mengembalikan inputan berupa pesan *error* jika inputan yang diberikan tidak sesuai dengan fungsi pada sistem. Analisis terhadap waktu rata-rata pemrosesan citra dan akurasi sistem yang telah dibangun juga akan dilakukan.
4. **Pembuatan *web service***. Pada tahap ini dilakukan penerjemahan fungsi yang telah dibuat dengan kondisi *desktop-based* ke dalam bahasa *web service* sehingga sistem mampu berjalan dalam kondisi *web based*.

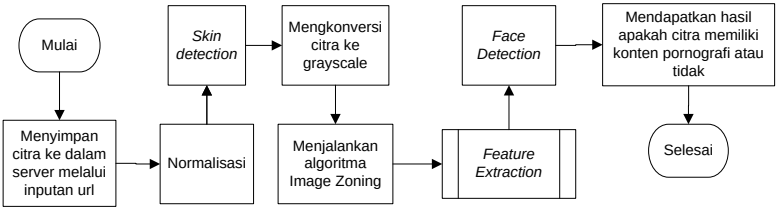
Sedangkan alur pemrosesan yang akan diterapkan pada sistem untuk mengklasifikasi citra adalah sebagai berikut:

1. *Service requestor* memanggil fungsi yang telah disediakan dengan memberikan inputan berupa *url* dari citra yang akan dicek. Kemudian server menyimpan citra yang terdeteksi melalui *url* yang diberikan ke dalam server.
2. Sistem melakukan normalisasi pada citra. Normalisasi yang dimaksud adalah mengubah citra menjadi format dan ukuran yang sesuai dengan fungsi pada server.
3. Sistem melakukan pengecekan terhadap seluruh pixel yang memiliki warna sama dengan kulit manusia dan menghilangkan pixel lainnya.
4. Sistem mengubah citra menjadi format *grayscale* dan melakukan *texture-based analysis*. Hal ini dilakukan untuk mengambil fitur yang dibutuhkan sistem untuk mengklasifikasi citra. Selain itu, ketujuh nilai *Hu Moment* juga akan diekstrak pada tahap ini.
5. Sistem menjalankan algoritma *Image Zoning* dengan membagi citra menjadi beberapa zona. Pada Tugas Akhir ini, jumlah zona yang dihasilkan adalah sebanyak tiga.
6. Sistem melakukan *feature extraction* yaitu melakukan klasifikasi dengan metode SVM. Fitur yang digunakan pada Tugas Akhir ini berdasarkan tiga analisis, yaitu:
    1) **Analisis warna kulit**. Analisis ini dilakukan untuk mengecek seberapa telanjang manusia pada sebuah citra. Fitur yang akan dipakai, antara lain: 1) jumlah pixel yang terhubung pada zona dan 2) proporsi pixel yang terindikasi sebagai kulit dengan seluruh pixel pada citra.
    2) ***Texture-based analysis***. Analisis ini dilakukan untuk mengecek tekstur yang terbentuk dari kulit manusia. Analisis akan dilakukan dengan menggunakan $d=1$ dan $\theta = [0, 45, 90, 135, 180, 225, 270, 315]$. Fitur yang akan dipakai, antara lain: 1) *contrast,* 2) *correlation*, 3) *energy* dan 4) *homogeneity*.

3) **Analisis bentuk**. Analisis ini dilakukan untuk mengecek bentuk dari citra yang memiliki konten pornografi. Analisis ini dilakukan dengan ketujuh *Hu Moment*.

7. Sistem melakukan *face detection* pada citra yang diprediksi memiliki konten pornografi untuk mengecek adanya mode wajah penuh pada citra. Jika ditemukan maka citra tidak memiliki konten pornografi.
8. Sistem menhasilkan nilai kembalian berupa String dengan isi "Porn" jika citra memiliki konten pornografi, "Not Porn" jika citra tidak memiki konten pornografi, dan pesan *error* jika inputan tidak sesuai dengan kemampuan atau fungsi dari sistem.

Gambar 3.2 menunjukkan alur atau jalannya sistem yang akan dibangun.



Gambar 3.2 Alus Kerja Sistem

3.5 Pengujian dan Perbaikan Sistem

Tahapan ini dilakukan untuk mengetahui apakah sistem dapat dijalankan dengan konsep *web service* dan mampu mengklasifikasi apakah citra memiliki konten pornografi atau tidak. Analisis juga dilakukan pada tahap ini untuk menghitung waktu pemrosesan citra serta akurasi yang dihasilkannya. Jika sistem masih belum bekerja maka pada tahap ini juga dilakukan perbaikan sampai sistem benar-benar bisa bekerja dengan baik.

3.6 Pengambilan Kesimpulan dan Saran

Tahapan ini digunakan untuk memberikan kesimpulan atas hasil penelitian yang dilakukan serta memberikan saran yang berguna untuk pengembangan atau perbaikan penelitian selanjutnya

3.7 Penyusunan Buku Tugas Akhir

Tahap ini adalah tahapan pembuatan laporan. Pembuatan laporan ini ditujukan agar seluruh langkah-langkah pengerjaan didokumentasikan dengan lengkap sehingga dapat memberikan informasi yang berguna. Dokumentasi ini akan ditulis dalam format laporan Tugas Akhir untuk menjadi buku tugas akhir.

BAB IV
ANALISIS & DESAIN SISTEM

Pada bab ini akan dijelaskan mengenai gambaran umum sistem, pemilihan fitur, pembuatan *flowchart*, pengumpulan data, desain aplikasi berbasis desktop, dan desain *web service*.

4.1 Gambaran Umum Sistem

Sistem yang akan dibangun nantinya akan mampu untuk mendeteksi citra yang tersimpan dalam sebuah server dalam internet. Sistem ini berupa *web service* yang akan mampu untuk digunakan oleh berbagai situs di internet. Sistem akan membutuhkan inputan berupa url dari citra yang akan diolah dengan sebuah fungsi standar. Kemudian sistem akan mengklasifikasi seluruh citra dan mengeluarkan *output* berupa "Porn" jika citra terkategori pornografi atau "Not Porn" jika citra terkategori non-pornografi. Selain itu sistem juga akan mengembalikan inputan berupa pesan *error* jika inputan yang diberikan tidak sesuai dengan fungsi pada sistem.

4.2 Pemilihan Fitur

Fitur merupakan hal yang penting dalam menentukan apakah sebuah citra memiliki konten pornografi atau tidak. Setelah nilai dari fitur yang didefinisikan telah diambil maka langkah selanjutnya adalah membuat sistem belajar dengan algoritma *Support Vector Machine* (SVM) bagaimana kombinasi nilai fitur yang termasuk ke dalam kategori pornografi dan non pornografi atau yang dinamakan dengan *training*. Langkah untuk mengambil nilai fitur ini yang dinamakan dengan ekstraksi fitur. Setelah tahap ini barulah sistem akan mampu untuk melakukan prediksi pada citra.

Fitur-fitur dipilih dengan menggunakan tiga analisis, yaitu: 1) analisis warna, 2) analisis tekstur, dan 3) analisis bentuk.

Analisis warna digunakan untuk mengetahui pixel mana saja pada sebuah citra yang termasuk ke dalam pixel kulit manusia. Fitur yang diambil pada analisis ini adalah 1) jumlah pixel manusia yang saling berhubungan dan 2) persentase pixel kulit manusia. Analisis tekstur digunakan untuk mengetahui tekstur yang dimiliki oleh kulit manusia. Hal ini untuk menghindari kesalahan prediksi sistem dari benda-benda yang memiliki warna sama dengan kulit manusia. Sistem diminta untuk belajar bagaimana tekstur dari kulit manusia dan tekstur yang bukan kulit manusia. Fitur untuk analisis tekstur adalah 1) *contrast*, 2) *correlation*, 3) *homogeneity*, dan 4) *energy*. Analisis bentuk dilakukan karena tubuh manusia memiliki bentuk yang unik dimana bentuk ini harus diketahui oleh sistem untuk meningkatkan akurasi. Fitur untuk analisis bentuk dilakukan dengan nilai moment yang didefinisikan oleh Hu dan dikenal dengan tujuh *Hu Moment*.

Berdasarkan penjelasan di atas maka sistem akan mengambil 13 fitur yang diekstrak dari citra. Setelah ketigabelas nilai fitur ini didapatkan maka sistem akan melakukan prediksi dengan menggunakan hasil pada tahap *training*.

4.3 Pembuatan Alur Penyelesaian Kode

Algoritma yang sesuai diperlukan untuk mengambil nilai dari ketigabelas fitur pada citra. Berikut ini adalah sekumpulan alur penyelesaian kode untuk melakukan proses ekstraksi fitur pada citra:

4.3.1. *Skin Detection*

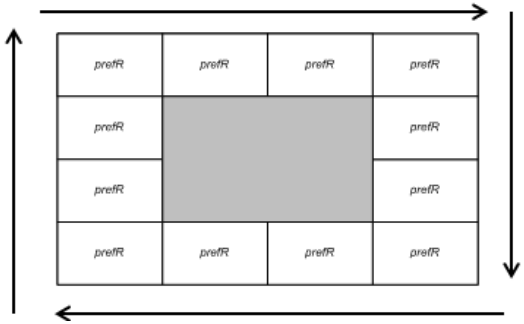
Skin detection merupakan tahap pertama dalam ekstraksi citra. *Skin detection* digunakan untuk mengeliminasi pixel-pixel yang tidak termasuk ke dalam kulit manusia dengan cara mengubah warnanya menjadi hitam. Konversi *grayscale* digunakan kepada pixel kulit manusia untuk mendapatkan *Gray Level Co-Occurences Matrix* (GLCM). GLCM ini yang akan digunakan untuk mendapatkan empat fitur dari analisis tekstur,

yaitu 1) *contrast*, 2) *correlation*, 3) *homogeneity*, dan 4) *energy*. Hasil akhir dari algoritma ini adalah citra yang merepresentasikan kulit manusia saja.

4.3.2. *Image Zoning*

Setelah mendapatkan citra yang hanya merepresentasikan kulit manusia saja maka langkah selanjutnya adalah melakukan *zoning* atau pembuatan zona dari citra. Sejumlah tiga zona akan dibuat pada Tugas Akhit ini. Setiap zona inilah yang akan dilakukan ekstraksi dari ketigabelas fitur yang telah didefinisikan sebelumnya. Kalkulasi dilakukan untuk setiap fitur yang ada pada setiap zona. Hasil kalkulasi dari setiap zona ini yang akan digunakan untuk untuk merepresentasikan karakter dari sebuah citra yang memiliki konten pornografi.

Tahap awal yang dilakukan adalah menginisiasi jumlah dari zona (*maxZone*) dan zona yang akan diproses (*z*, dengan *z* = 1,2,3,...,*maxZone*). Algoritma ini bekerja dengan membuat kotak minimum *prefR* yang dapat terbentuk dari citra yang diinputkan. Kemudian *prefR* akan berjalan menyusuri tepian citra pada zona *z* yang telah diinputkan. Gambar 4.1 menunjukkan proses dari algoritma *Image Zoning* dengan *maxZone* = 2 dan *z* = 2.

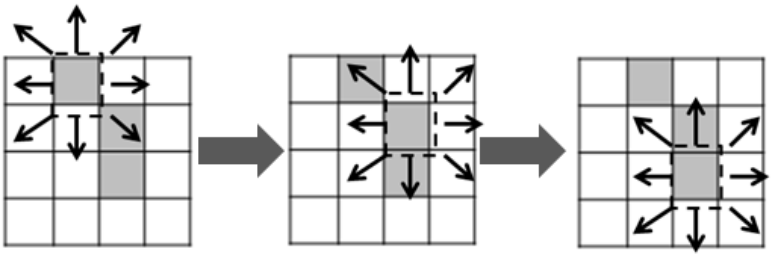


Gambar 4.1 Visualisasi Algoritma *Image Zoning*

Hasil akhir dari algoritma ini adalah sebuah citra yang hanya berisi pixel pada zona yang telah ditentukan.

4.3.2. Jumlah Pixel Kulit yang Terhubung (*Skin Count*) dan Persentase Pixel Kulit (*Skin Percentage*)

Setelah mendapatkan citra yang hanya merepresentasikan kulit manusia maka dilakukan ekstraksi fitur *skin count* atau jumlah pixel kulit manusia yang terhubung. Hal ini dilakukan untuk menilai ada berapa entitas yang ada dalam citra. Entitas ini yang kemudian disangkutpautkan dengan manusia. Untuk menghitung nilai dari *skin count* maka dilakukan proses pengecekan delapan arah mata angin untuk setiap pixel. Gambar 4.2 menunjukkan cara kerja algoritma *skin count* pada sebuah citra dengan *skin count* yang didapatkan adalah 1:



Gambar 4.2 Visualisasi Algoritma *Skin Count* dengan nilai *skin count* = 1

Algoritma bekerja dengan cara meminta citra I dengan panjang W dan lebar H sebagai inputan. Kemudian dilakukan pembuatan matriks W x H dengan nilai (w,h) adalah 0 dengan w = [0,1,2,...,W-1] dan h = [0,1,2,...,H-1]. Setelah itu dilakukan penelusuran pixel p dalam rentang (w,h) untuk mengecek pixel mana saja yang termasuk ke dalam kulit manusia. Jika benar bahwa $p(w,h)$ adalah kulit manusia, maka nilai *skin count* akan bertambah 1. Kemudian nilai matriks W x H dalam koordinat (w,h) akan menjadi 1 yang menandakan bahwa koordinat tersebut tidak perlu diproses lagi jika sewaktu-waktu koordinat tersebut masuk dalam penelusuran.

Persentase pixel kulit manusia atau *skin percentage* yang ada pada citra juga dihitung untuk menunjukkan seberapa besar

atau luas entitas yang ditemukan pada citra. Nilai *skin percentage* dihitung bersamaan dengan *skin count* namun perhitungan *skin percentage* dilakukan dengan menjumlahkan seluruh pixel kulit manusia kemudian membaginya dengan luas citra.

4.3.3. Gray Level Co-Occurences Matrix (GLCM)

GLCM digunakan untuk mendapatkan fitur *contrast, correlation, homogeneity,* dan *energy*. Algortima ini bekerja dengan meminta citra yang telah dikonversi ke dalam skala *grayscale* sebagai inputan. Kemudian intensitas warna setiap pixel diubah menjadi nilai *integer* dan mencari nilai terbesar *N* untuk dijadikan matriks *N* x *N*. GLCM diperoleh dengan menjumlahkan jumlah pasangan nilai pixel *ni* dan *nj* dalam sudut tertentu θ dan menyimpannya ke dalam matriks *N* x *N* pada baris *ni* dan kolom *nj*. Pada Tugas Akhir ini digunakan sudut $\theta = [0^o, 45^o, 90^o, 135^o, 180^o, 225^o, 270^o, 315^o]$.

Algoritma dimulai dengan membuat inisialisasi array tiga dimensi dengan nilai awal -1 untuk menyimpan nilai *integer* dari intensitas warna sesuai dengan model RGB-nya. Dimensi pertama untuk menyimpan nilai R, dimensi kedua untuk menyimpan nilai G, dan dimensi ketiga untuk menyimpan nilai B dengan panjang masing-masing dimensi adalah 256. Karena citra dalam konversi *grayscale* maka nilai *integer* maksimum yang ada pada array adalah sebesar 256. Kemudian langkah selanjutnya adalah membuat array baru dengan nilai awal 0 untuk menyimpan jumlah pasangan pixel untuk setiap intensitas warna dengan berpedoman pada array pertama yang telah dibuat. Kemudian citra kembali ditelusuri untuk mengisi array kedua dengan metode *bucket*. Array kedua ini adalah GLCM yang siap untuk digunakan pada proses selanjutnya. Namun sebelum dikembalikan, setiap nilai pada array tersebut dinormalisasi dengan membaginya dengan nilai maksimum yang ada pada array.

4.3.4. *Contrast, Correlation, Homogeneity,* dan *Energy*

GLCM adalah bahan utama untuk melakukan analisis tekstur yang hasilnya direpresentasikan dengan *contrast*, *correlation*, *homogeneity*, dan *energy*. Keempat fitur ini dapat diproses dengan menelusuri setiap nilai pada GLCM.

4.3.5. *Hu Moment*

Hu Moment adalah fiur yang akan digunakan dalam analisis bentuk. Dengan didapatkannya nilai *Hu Moment* sebuah bentuk dari pornografi dapat diidentifikasi dan dipelajari oleh sistem sehingga sistem mampu untuk mengenali citra mana yang memiliki konten pornografi.

Untuk menjalankan algoritma pencarian nilai *Hu Moment* sebuah citra yang telah mengalami proses *skin detection* digunakan sebagai inputan. Tujuannya adalah agar citra hanya meninggalkan bentuk dari konten pornografi yang ada pada citra. Kemudian sistem akan melakukan penelusuran pada seluruh pixel pada citra. Pixel yang termasuk ke dalam kulit manusia akan diberi nilai 1 dan yang tidak termasuk (pixel berwarna hitam) akan diberi nilai 0.

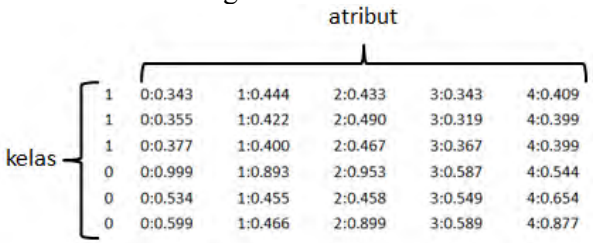
4.4 Pengumpulan Data / Citra

Pada tahap ini dilakukan pengumpulan data atau citra yang akan digunakan sebagai *training data* dan *testing data*. Citra yang dikumpulkan terdiri dari citra yang memiliki konten pornografi dan citra biasa. Jumlah citra yang digunakan adalah sebanyak 2000 buah dengan perbandingan citra dengan konten pornografi dan citra biasa adalah 1:1, artinya 1000 citra yang memiliki konten pornografi dan 1000 citra biasa. Citra biasa terdiri dari wajah, manusia, pemandangan, alat transportasi, dan lain-lain. Sedangkan citra yang memiliki konten pornografi melibatkan ras kaukasoid, mongoloid, dan negroid dengan pencahayaan yang berbeda dan dengan format JPG. Kemudian sejumlah 1098 buah citra, dengan 547 citra sebagai citra yang

memiliki konten pornografi dan 551 citra sebagai citra biasa digunakan sebagai *testing data*. Untuk memudahkan pemrosesan citra maka dibuat folder sesuai dengan kategori dan fungsi seperti di atas dari citra yang dikumpulkan.

4.5 Desain Aplikasi Pembuatan Model Berbasis Desktop

Untuk dapat melakukan prediksi dengan menggunakan LibSVM, sistem perlu memiliki model hasil olahan *training data* dan file jangkauan. Untuk mendapatkan kedua jenis file ini maka sebuah aplikasi berbasis desktop dibuat. Tujuannya adalah untuk mengekstrak fitur-fitur pada citra yang akan digunakan sebagai *training data*. Hasil ekstraksi ini adalah sebuah file bernama training.txt yang akan digunakan untuk proses *training*. Format file ini disesuaikan dengan format file *training* milik LibSVM. Gambar 4.3 adalah contoh file *training* yang mampu untuk diproses oleh LibSVM dengan 2 kelas dan 5 atribut atau fitur.

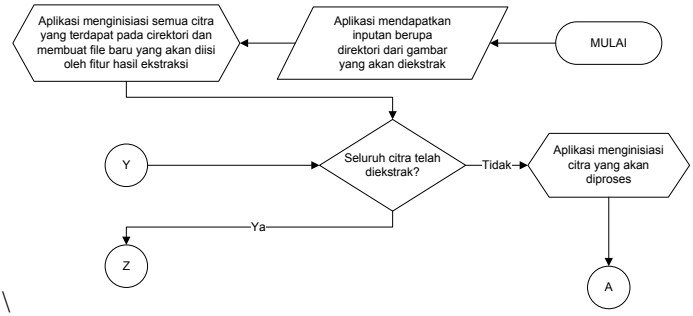


Gambar 4.3 Format Penulisan Training Data pada LibSVM dengan 2 kelas (0 dan 1) dan 5 atribut/fitur (0 sampai 4)

Namun file hasil ekstraksi ini masih belum menjadi model. Untuk mengkonversinya menjadi model digunakan file svm_scale.java dan svm_train.java. Aplikasi ini akan dibuat dengan menggunakan bahasa pemrograman Java dengan jdk 1.7 dan bantuan IDE Netbeans 7.4.

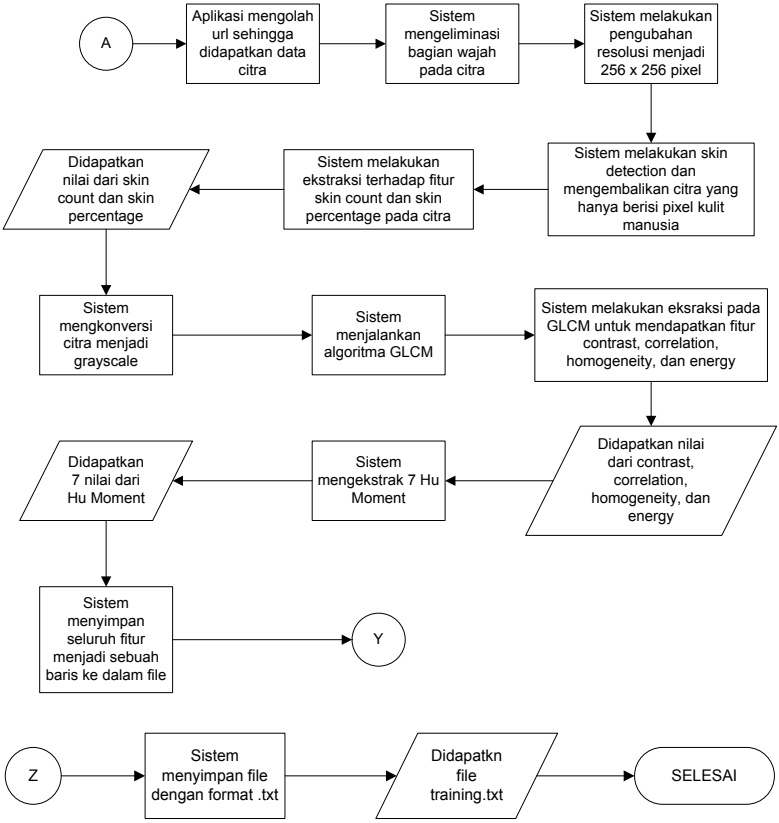
4.5.1. *Application Workflow*

Alur ekstraksi fitur bagian 1 pada aplikasi pembuatan model ini dapat dilihat pada gambar 4.4.



Gambar 4.4 *Application Workflow* (bagian 1)

Gambar 4.5 menunjukkan alur ekstraksi fitur bagian 2 pada aplikasi pembuatan model:



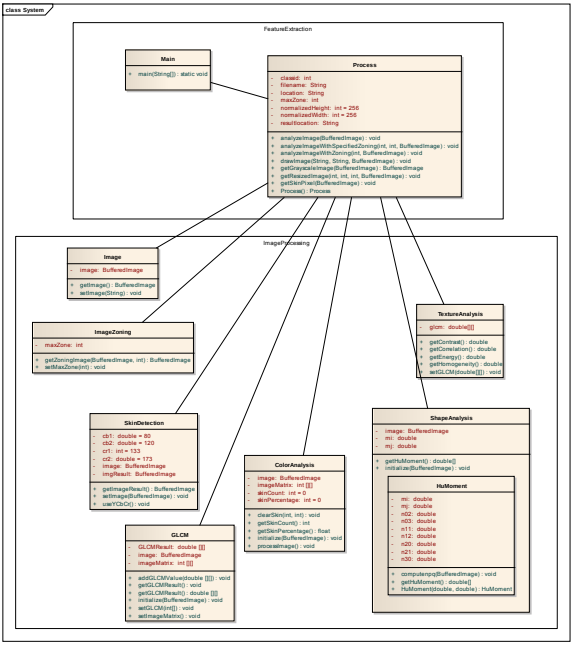
Gambar 4.5 *Application Workflow* (bagian 2)

4.5.2. Antarmuka Aplikasi

Aplikasi desktop ini dibuat dengan tampilan *Command Line Interface* (CLI). Aplikasi hanya di-*compile* kemudian aplikasi mengekstrak seluruh fitur dari seluruh citra yang disimpan dalam sebuah folder. Hal yang perli dilakukan adalah melakukan kustomisasi terhadap direktori penyimpanan citra dan kategori citra tersebut.

4.5.3. *Class Diagram*

Aplikasi ini akan dibangun dengan dua *package*, yaitu *FeatureExtraction* dan *ImageProcessing*. *FeatureExtraction* adalah *package* utama dimana aplikasi akan dijalankan. Sedangkan *ImageProcessing* adalah *package* untuk melakukan pemrosesan citra sesuai dengan *workflow* yang telah dibuat. *Class diagram* dibuat agar aplikasi dapat dbangun dengan terstruktur sesuai dengan fungsi dari setiap *package*. Gambar 4.6 menunjukkan *class diagram* dari aplikasi berdasarkan *package* yang telah didefinisikan.



Gambar 4.6 *Class Diagram* untuk Aplikasi Desktop

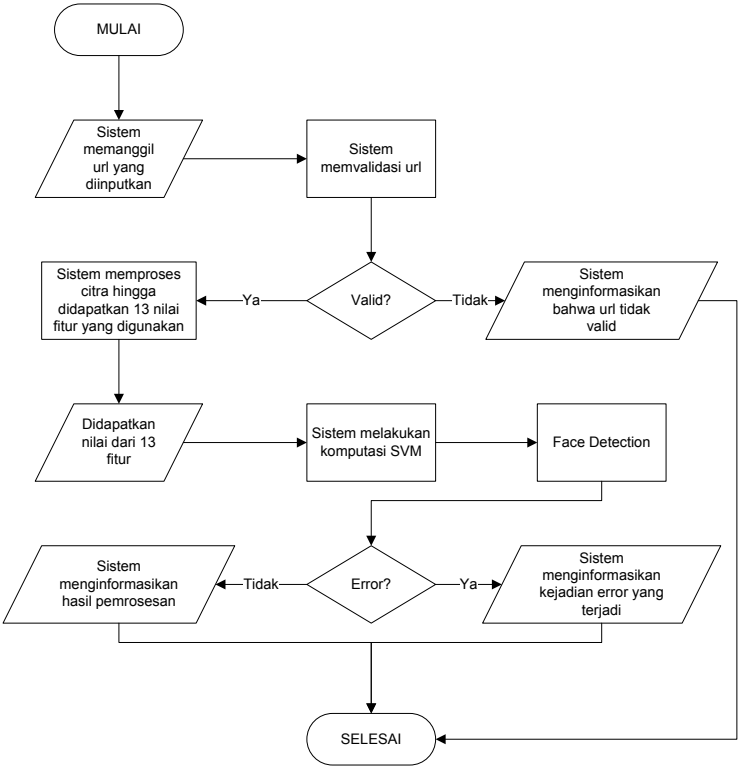
4.6 Desain *Web Service*

Setelah mendapatkan nilai fitur dari *training data*, langkah selanjutnya adalah mengolah hasil tersebut untuk dikombinasikan dengan *web service* yang akan dibangun. Pembangunan *web service* dilakukan dengan bahasa pemrograman Java dengan jdk 1.7 dan bantuan IDE Netbeans 7.4. Berikut ini adalah analisis dan desain yang akan digunakan untuk membangun *web service*:

4.6.1. *Web Service Workflow*

Untuk menggunakan *web service* maka sebuah fungsi umum didefinisikan agar setiap situs pada jaringan internet dapat layanan yang disebarkan. Fungsi ini memiliki parameter berupa url dari citra yang akan dideteksi. Kemudian *web service* akan mengecek apakah url tersebut benar-benar valid atau tidak. Jika tidak valid maka *web service* akan mengembalikan pesan *error* yang menginformasikan bahwa inputan url yang diberikan tidak valid. Jika valid maka *web service* akan mengolah url tersebut sehingga didapatkan citra yang akan diproses. Proses dilanjutkan dengan mendapatkan nilai dari 13 fitur seperti pada penjelasan sebelumnya. Kemudian proses komputasi dengan menggunakan SVM dilakukan dengan inputan berupa 13 nilai fitur yang telah didapatkan. Jika terjadi *error* pada proses ekstraksi maka *web service* akan mengembalikan pesan berupa *error* yang terjadi. Jika tidak terjadi *error* maka *web service* akan mengembalikan pesan berupa *true* atau *false*. *True* jika *web service* mendeteksi adanya konten pornografi pada citra dan *false* jika sebaliknya. Jika *true* maka sistem melakukan *face detection* untuk mencari adanya mode wajah penuh dan jika ditemukan maka sistem akan mengembalikan *false*.

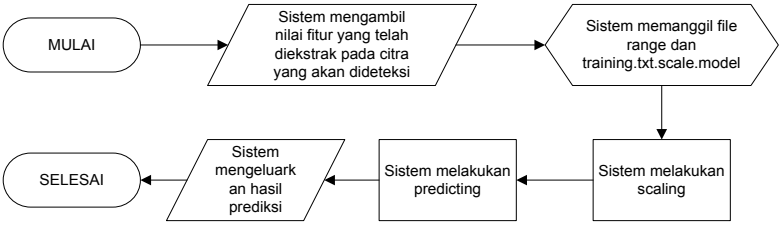
Secara umum *workflow* yang akan terjadi dapat dilihat pada gambar 4.7.



Gambar 4.7 Web Service Workflow

Proses komputasi SVM diawali dengan mengambil file *range* dan *training.txt.scale.model*. File *range* adalah file yang berisi informasi yang dibutuhkan untuk melakukan proses *scaling*, sedangkan file *training.txt.scale.model* adalah file yang merepresentasikan model dari *training data*. Kedua file ini didapatkan dengan menggunakan *lirabry* LibSVM. File *range* didapatkan dengan menggunakan file svm_scale.java, sedangkan *training.txt.scale.model* didapatkan dengan menggunakan file svm_training.java.

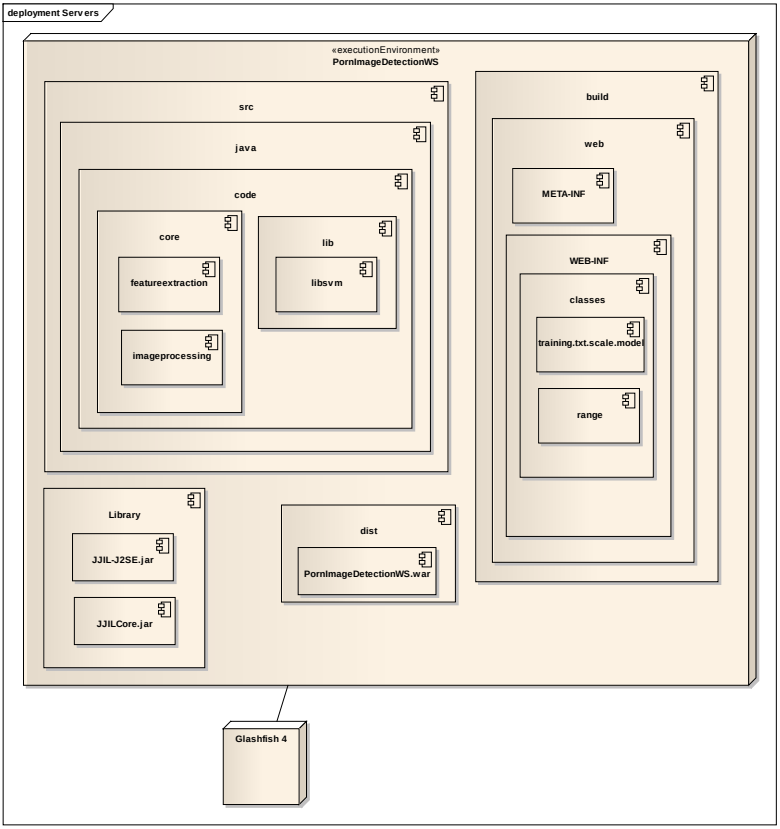
Langkah selanjutnya adalah melakukan *scaling* pada nilai fitur yang telah diekstrak pada citra yang akan dideteksi. Nilai ini kemudian dimasukkan ke dalam fungsi yang mampu untuk memprediksi apakah citra memiliki konten pornografi atau tidak. Fungsi ini dikombinasikan dengan fungsi yang telah disediakan pada file svm_predict.java sehingga mampu mengeluarkan *output* yang diminta. Gambar 4.8 menunjukkan *workflow* pada saat *web service* melakukan proses komputasi SVM:



Gambar 4.8 *Workflow* Komputasi SVM

4.6.2. *Deployment Diagram*

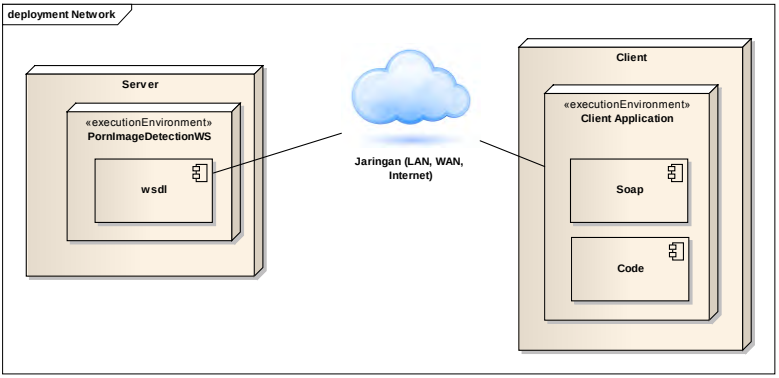
Deployment diagram adalah representasi arsitektur sistem yang akan dibangun. *Deployment diagram* pada Tugas Akhir ini akan dibuat menjadi dua, yaitu *deployement diagram* yang ada pada server dan *deployment diagram* pada jaringan yang akan melibatkan sejumlah *client*. Gambar 4.9 menunjukkan *deployment diagram* yang ada pada server.



Gambar 4.9 *Deployment Diagram* Sistem *Web Service* pada Server

Diagram ini menunjukkan lokasi dari seluruh sumber daya yang akan digunakan pada sistem. Yang perlu diperhatikan adalah bahwa sistem akan dikelola dengan menggunakan Glashfish 4. Glashfish 4 adalah sistem yang mampu untuk menyediakan layanan Java pada web. File *training.txt.scale.model* dan *range* akan diletakkan pada komponen classes pada *web service.*

Berikut ini adalah gambar 4.10 yang menunjukkan *deployment diagram* pada jaringan yang akan melibatkan sejumlah *client*:



Gambar 4.10 *Deployment Diagram Web Service* Sistem pada Jaringan

Diagram ini menunjukkan bagaimana sistem atau *web service* dapat digunakan oleh aplikasi *client* dalam sebuah jaringan (LAN, WAN, internet). Dengan asumsi bahwa *client* telah memiliki aplikasi yang telah dipasang dalam sebuah jaringan maka jika *client* ingin menggunakan *web service* maka aplikasi *client* harus mengetahui link wsdl dari *web service* dan *method* yang digunakan oleh server. Karena sistem dibangun dengan menggunakan lingkungan Java maka jika terdapat aplikasi berbasis PHP, aplikasi tersebut perlu menggunakan fungsi PHP bernama Soap. Fungsi ini yang akan menghubungkan aplikasi *client* dengan fungsi yang ada pada server.

4.6.3. *Class Diagram*

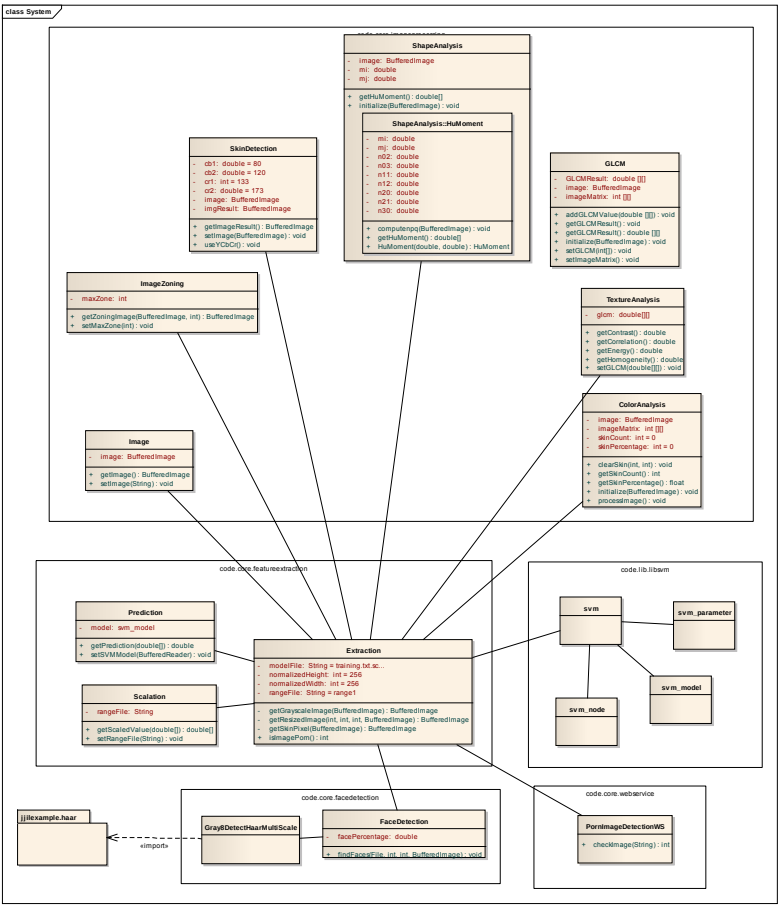
Class diagram merepresentasikan bagaimana susunan *code* yang akan dibangun pada sistem. Sistem akan menggunakan

empat *package* yang bernama code.core.imageprocessing, code.core.featureextraction, code.lib.libsvm, code.core.facedetection, jjilexample.haar dan code.core.webservice. Tabel 4.1 menunjukkan keseluruhan *package* yang ada pada sistem.

Tabel 4.1 Daftar Package Sistem

No	Nama *Package*	Fungsi	*Library*
1	code.core.imageprocessing	Memproses citra	Tidak
2	code.core.featureextraction	Sebagai *package* utama yang akan mengolah code.core.imageproessing, dan code.lib.libsvm sehingga sistem mampu mengeluarkan *output* sesuai dengan input yang diberikan	Tidak
3	code.lib.libsvm	Mengerjakan algoritma SVM	Ya
4	code.core.facedetection	Melakukan *face detection* dan secara aplikatif dapat digunakan untuk mencari adanya mode wajah penuh	Ya, dengan modifikasi
5	Jjexample.haar	Adalah *library* JJIL yang perlu ditanamkan pada aplikasi	Ya
6	code.core.webservice	Membangun *web service*	Tidak

Gambar 4.11 menunjukkan *class diagram* yang akan dibangun pada sistem.



Gambar 4.11 *Class Diagram* Sistem *Web Service*

Halaman ini sengaja dikosongkan

BAB V
IMPLEMENTASI & UJI COBA

Bab ini menjelaskan mengenai proses code pada aplikasi, dan hasil uji coba sistem yang telah dibangun untuk memastikan kesesuaian dengan kebutuhan fungsional

5.1 Lingkungan Implementasi

Tabel 5.1 menunjukkan daftar spesifikasi perangkat keras, perangkat lunak, *library*, dan infrastruktur jaringan yang diterapkan untuk membangun sistem.

Tabel 5.1 Spesifikasi Perangkat Keras, Perangkat Lunak, *Library*, dan Infrastruktur Jaringan Sistem

Spesifikasi	Detail Spesifikasi	
Komputer personal	*Processor*	Intel Core i5 2450M @ 2.50 GHz
	RAM	2048 MB
	Sistem Operasi	Windows 7 Ultimate
Library	Komputasi SVM	LibSVM
	Face detection	Jon's Java Image Library
Perangkat Lunak	Java Development Kit (JDK)	jdk 1.7
	Editor	Netbeans 7.4
	Java Web Service Server	GlassFish 4.0
Infrastruktur Jaringan	Jenis Koneksi	Proxy berautentikasi
	Konfigurasi Proxy	proxy.its.ac.id:8080
	Bandwidth	100 MBps

5.2 Implementasi Desain Aplikasi Berbasis Desktop

Untuk membangun aplikasi berbasis desktop, dibutuhkan komputer personal dan perangkat lunak dengan spesifikasi pada subbab 5.1. Pada tahap ini, spesifikasi untuk *library* dan infrastruktur jaringan belum diperhatikan karena tujuan dari pembuatan aplikasi ini adalah membuat file *tratining.txt.dcale.model* dan *range* yang dibutuhkan untuk proses komputasi SVM. Setelah seluruh spesifikasi terpenuhi maka langkah selanjutnya adalah membuat Java Project baru bernama *PornImageDetector* dengan menggunakan Netbeans 7.4. Sesuai dengan *class diagram* yang telah dibangun maka *package* yang terbentuk adalah:

1) FeatureExtraction, yang berisi *class* sebagai berikut:
   - Main.java: berfungsi sebagai *main class* yang berfungsi untuk membanggil fungsi-fungsi dari *class* Process.java
   - Process.java: berfungsi untuk menjalankan keseluruhan fungsi yang digunakan untuk mengekstrak fitur dari citra
2) Imageprocessing, yang berisi *class* sebagai berikut:
   - ColorAnalysis.java: berfungsi untuk mengekstrak fitur berdasarkan analisis warna, yaitu: 1) *skin count* dan 2) *skin percentage*.
   - GLCM.java: berfungsi untuk membuat *Gray Level Co-Occurences Matrix* (GLCM).
   - Image.java: berfungsi untuk membuat citra dapat diolah pada sistem.
   - ImageZoning.java: berfungsi untuk melakukan membagi citra menjadi beberapa zona (*image zoning*) sehingga dapat dilakukan proses ekstraksi pada masing-masing zona tersebut.
   - ShapeAnalysis.java: berfungsi untuk mengekstrak fitur berdasarkan analisis bentuk, yaitu tujuh *Hu Moment*.

- SkinDetection.java: berfungsi untuk mendeteksi pixel-pixel kulit pada citra dengan menggunakan *YCbCr Model*.
- TextureAnalysis.java: berfungsi untuk mengekstrak fitur berdasarkan analisis tekstur, yaitu: 1) *contrast*, 2) *correlation*, 3) *energy*, dan 4) *homogeneity*.

5.2.1. Implementasi Kode pada *Main.java*

Class ini hanya digunakan untuk memanggil Process.java. Tabel 5.2 menunjukkan kode yang digunakan pada *class* Main.java:

Tabel 5.2 Kode Main.java

No	Kode
1	`package FeatureExtraction;`
2	`public class Main {`
3	
4	`    public static void main(String[] args) {`
5	`        Process process = new Process();`
6	`    }`
7	
8	`}`

5.2.2. Implementasi Kode pada *Process.java*

Serangkaian proses yang ada pada *class* ini antara lain adalah: 1) mengambil seluruh citra yang terletak pada direktori yang telah didefinisikan secara manual, 2) mengubah resolusi citra menjadi 256 x 256, 3) menghilangkan pixel non-kulit, 4) mengkonversi citra menjadi *grayscale*, 5) melakukan ekstraksi dengan metode *image zoning* dengan jumlah zona sebanyak tiga,

dan 6) menyimpan seluruh nilai fitur pada file *training.txt* dengan format LibSVM.

Untuk mendapatkan direktori yang akan digunakan untuk mengambil seluruh citra atau data yang digunakan untuk *training* maka digunakan variabel bertipe String dan bernama *location*.

Sistem ini memiliki dua kelas, yaitu *porn* dan *not porn* yang diidentifikasi dengan nilai 1 dan 0. Untuk mendapatkan file *training* pada setiap kelas maka aplikasi ini akan dijalankan dua kali dengan parameter berbeda. Parameter ini akan diinputkan secara manual pada *class* Process.java ini. Untuk merealisasikan hal ini maka dibuat dua variabel yaitu *classid* dan *filename* yang keduanya bertipe String.

Tabel 5.3 menunjukkan kode yang digunakan untuk membuat *training* untuk kategori pornografi pada *class* ini dengan kondisi parameter *location* adalah "D:\\testing\\porn", *classid* adalah "1", dan *filename* adalah ""D:\\testing\\porn.txt":

Tabel 5.3 Kode Process.java

No	Kode
1	`package FeatureExtraction;`
2	
3	`import ImageProcessing.Image;`
4	`import ImageProcessing.ImageZoning;`
5	`import ImageProcessing.TextureAnalysis;`
6	`import ImageProcessing.ColorAnalysis;`
7	`import ImageProcessing.SkinDetection;`
8	`import ImageProcessing.GLCM;`
9	`import ImageProcessing.ShapeAnalysis;`
10	
11	`import java.awt.Graphics2D;`
12	`import java.awt.image.BufferedImage;`
13	`import java.io.File;`
14	`import java.io.IOException;`
15	`import java.text.DecimalFormat;`
16	`import javax.imageio.ImageIO;`
17	
18	`public final class Process {`
19	
20	`    private final String location;`
21	
22	`    private final int normalizedWidth=256,`

```
normalizedHeight = 256;
    private final int maxZone = 3;

    private String trainedData = "";
    private final String classid = "1";
    private final String filename =
"D:\\testing\\porn.txt";

    public Process(){
        this.location = "D:\\testing\\porn";

        File files = new
File(location);System.out.println("length:
"+files.list().length);
        for(int i=0; i<(files.list().length-1); i++){

            System.out.println("file "+i);
            Image img = new Image();

img.setImage(location+"\\"+files.list()[i]);
            BufferedImage imgused = img.getImage();
            int type = imgused.getType();

            imgused = getResizedImage(imgused,
normalizedWidth, normalizedHeight, type);
            imgused = getSkinPixel(imgused);
            imgused = getGrayscaleImage(imgused);
            analyzeImageWithZoning(imgused, i);
        }

        try{
            File file = new File(filename);
            try (BufferedWriter output = new
BufferedWriter(new FileWriter(file))) {
                output.write(trainedData);
            }
        } catch(IOException e){
            System.out.println(e);
        }

    }

    public void analyzeImageWithZoning(BufferedImage
image, int index){

        String result = classid;

        ImageZoning imgzoning = new ImageZoning();
        imgzoning.setMaxZone(maxZone);

```

```
        float skinCountTotal = 0, skinPercentageTotal
= 0;
        double contrastTotal = 0, correlationTotal =
0, energyTotal = 0, homogeneityTotal = 0;
        double huMomentTotal[];
        huMomentTotal = new double[]{0, 0, 0, 0, 0,
0, 0};

        for(int zone=1; zone<=maxZone; zone++){
            image = imgzoning.getZoningImage(tmp,
zone);

            ColorAnalysis colorAnalysis = new
ColorAnalysis();
            colorAnalysis.initialize(image);
            colorAnalysis.processImage();
            float skinCount =
colorAnalysis.getSkinCount();
            float skinPercentage =
colorAnalysis.getSkinPercentage() / (image.getWidth()
* image.getHeight());

            skinCountTotal += skinCount;
            skinPercentageTotal += skinPercentage;

                      GLCM glcm = new GLCM();
            glcm.initialize(image);
            glcm.setImageMatrix();
            glcm.setGLCM(new int[]{0, 45, 90, 135,
180, 225, 270, 315});
            double [][] glcmResult =
glcm.getGLCMResult();

            TextureAnalysis textureAnalysis = new
TextureAnalysis();
            textureAnalysis.setGLCM(glcmResult);
            double contrast =
textureAnalysis.getContrast();
            double correlation =
textureAnalysis.getCorrelation();
            double energy =
textureAnalysis.getEnergy();
            double homogeneity =
textureAnalysis.getHomogeneity();

            contrastTotal += contrast;
            correlationTotal += correlation;
            energyTotal += energy;
            homogeneityTotal += homogeneity;

```

```
123             ShapeAnalysis shapeAnalysis = new
124 ShapeAnalysis();
125             shapeAnalysis.initialize(image);
126             double [] huMoment =
127 shapeAnalysis.getHuMoment();
128             for(int i=0; i<huMoment.length; i++){
129                 huMomentTotal[i] += huMoment[i];
130             }
131         }
132 
133         result += " 0:"+skinCountTotal+"
134 1:"+skinPercentageTotal;
135         result += " 2:"+contrastTotal+"
136 3:"+correlationTotal+" 4:"+energyTotal+"
137 5:"+homogeneityTotal;
138 
139         for (int i = 0; i < huMomentTotal.length;
140 i++) {
141             result += " "+(i+6)+":"+huMomentTotal[i];
142         }
143 
144         trainedData += result+"\n";
145 
146     }
147 
148     public BufferedImage
149 getResizedImage(BufferedImage img, int width, int
150 height, int type){
151         BufferedImage resizedImage = new
152 BufferedImage(width, height, type);
153         Graphics2D g = resizedImage.createGraphics();
154         g.drawImage(img, 0, 0, width, height, null);
155        g.dispose();
156 
157         return resizedImage;
158     }
159 
160     public BufferedImage getSkinPixel(BufferedImage
161 img){
156 
157         SkinDetection skin = new SkinDetection();
158         skin.setImage(img);
159         skin.setImageResult(img.getWidth(),
160 img.getHeight());
161         skin.useYCbCr();
162 
163         return skin.getImageResult();
164     }
165     public BufferedImage
166 getGrayscaleImage(BufferedImage image){
```

```
167
168         for(int i = 0; i < image.getWidth(); i++){
169             for(int j = 0; j < image.getHeight();
170 j++){
171                 int argb = image.getRGB(i, j);
172                 int rgb[] = new int[] {
173                     (argb >> 16) & 0xff, //red
174                     (argb >>  8) & 0xff, //green
175                     (argb      ) & 0xff  //blue
176                 };
177
178                 double r = (double) rgb[0];
179                 double g = (double) rgb[1];
180                 double b = (double) rgb[2];
181
182                 int grayLevel = (int) (r + g + b) /
183 3;
184                 int gray = (grayLevel << 16) +
185 (grayLevel << 8) + grayLevel;
186
187                 image.setRGB(i, j, gray);
188             }
189         }
190
191         return image;
192     }
193
194 }
```

5.2.3. Implementasi Kode pada *Image.java*

Tabel 5.4 menunjukkan kode yang digunakan pada *class* Image.java.

Tabel 5.4 Kode Image.java

No	Kode
1	`package ImageProcessing;`
2	
3	`import java.awt.image.BufferedImage;`
4	`import java.io.File;`
5	`import javax.imageio.ImageIO;`
6	
7	`public class Image {`
8	`    private BufferedImage image = null;`
9	

```
10      public void setImage(String location){
11          File f = null;
12          try{
13              f = new File(location);
14              image = ImageIO.read(f);
15          } catch(Exception e){
16              System.out.println(e);
17          }
18
19      }
20
21      public BufferedImage getImage(){return image;}
22
23  }
```

5.2.4. Implementasi Kode pada *SkinDetection.java*

Pembuatan kode pada *class* ini berpedoman pada *flowchart* yang telah dibuat pada bab 4. Algoritma *skin detection* akan dijalankan dengan menggunakan *YCbCr Model*. Sebuah parameter bertipe BufferedImage dan bernama *image* dibutuhkan sebagai inputan. Kemudian inputan ini diproses dan menghasilkan BufferedImage baru dengan nama *imgResult* yang hanya berisi pixel kulit. Tabel 5.5 menunjukkan kode pada *class* SkinDetection.java.

Tabel 5.5 Kode SkinDetection.java

No	Kode
1	`package ImageProcessing;`
2	
3	`import java.awt.image.BufferedImage;`
4	
5	`public class SkinDetection {`
6	
7	`    private BufferedImage image = null;`
8	`    private BufferedImage imgResult = null;`
9	
10	`    private double cb1 = 80, cb2 = 120;`
11	`    private double cr1 = 133, cr2 = 173;`
12	
13	`    public void useYCbCr(){`
14	
15	`        for(int i = 0; i < image.getWidth(); i++){`

```
16              for(int j = 0; j < image.getHeight(); j++){
17                  int argb = image.getRGB(i, j);
18                  int rgb[] = new int[] {
19                      (argb >> 16) & 0xff, //red
20                      (argb >>  8) & 0xff, //green
21                      (argb      ) & 0xff  //blue
22                  };
23
24                  double r = (double) rgb[0];
25                  double g = (double) rgb[1];
26                  double b = (double) rgb[2];
27
28                  int grayLevel = (int) (r + g + b) / 3;
29                  int gray = (grayLevel << 16) +
30  (grayLevel << 8) + grayLevel;
31
32                  double cb = 128 - 0.169*r - 0.332*g +
33  0.500*b;
34                  double cr = 128 + 0.500*r - 0.419*g -
35  0.081*b;
36                  if(cb >= cb1 && cb <= cb2 && cr >= cr1
37  && cr <= cr2){
38                      imgResult.setRGB(i, j, argb);
39                      //imgResult.setRGB(i, j, gray);
40                  } else {
41                      imgResult.setRGB(i, j, 0);
42                  }
43              }
44          }
45      }
46
47      public void setImage(BufferedImage img){
48          image = img;
49      }
50
51      public void setImageResult(int width, int height){
52          imgResult = new BufferedImage(width, height,
53  BufferedImage.TYPE_INT_RGB);
54      }
55      public BufferedImage getImageResult(){return
56  imgResult;}
57
58  }
```

5.2.5. Implementasi Kode pada *ImageZoning.java*

Pembuatan kode pada *class* ini berpedoman pada *flowchart* yang telah dibuat pada bab 4. Parameter yang dibutuhkan sebagai inputan pada *class* ini adalah jumlah zona yang direalisasikan dengan nama *maxZone* bertipe integer dan citra yang akan diproses dengan nama *img* dan bertipe BufferedImage. Hasil pemrosesan dari *class* ini berupa citra yang terletak pada masing-masing zona. Tabel 5.6 menunjukkan kode pada *class* ImageZoning.java.

Tabel 5.6 Kode pada ImageZoning.java

No	Kode
1	`package ImageProcessing;`
2	
3	`import java.awt.image.BufferedImage;`
4	
5	`public class ImageZoning {`
6	
7	`    private int maxZone;`
8	
9	`    public void setMaxZone(int zone){maxZone = zone;}`
10	
11	`    public BufferedImage getZoningImage(BufferedImage`
12	`img, int zone){`
13	`        int prefWidth = img.getWidth()/(maxZone*2);`
14	`        int prefHeight = img.getHeight()/(maxZone*2);`
15	
16	`        BufferedImage imageResult = new`
17	`BufferedImage(img.getWidth(), img.getHeight(),`
18	`img.getType());`
19	`        for(int x=0; x<imageResult.getHeight(); x++)`
20	`            for(int y=0; y<imageResult.getHeight();`
21	`y++)`
22	`                imageResult.setRGB(x, y, 0);`
23	
24	`        int maxLoop = 8*(zone-1)+4;`
25	`        int rowLimit = 2*zone;`
26	`        int xstart = prefWidth * (maxZone-zone);`
27	`        int ystart = prefHeight * (maxZone-zone);`
28	
29	`        int index = 0, limitindex = 0, status = 0;`
30	`        while(index<=maxLoop+1){`
31	`            for(int x=xstart; x<(xstart+prefWidth);`
32	`x++)`

```
33                  for(int y=ystart;
34  y<(ystart+prefHeight); y++){
35                      int argb = img.getRGB(x, y);
36                      imageResult.setRGB(x, y, argb);
37                  }
38
39              if(status == 0){
40                  xstart += prefWidth;
41                  index++;
42                  limitindex++;
43                  if(limitindex == rowLimit){
44                      status = 1;
45                      xstart -= prefWidth;
46                      limitindex = 0;
47                  }
48              } else if(status == 1){
49                  ystart += prefHeight;
50                  index++;
51                  limitindex++;
52                  if(limitindex == rowLimit){
53                      status = 2;
54                      ystart -= prefHeight;
55                      limitindex = 0;
56                  }
57              } else if(status == 2){
58                  xstart -= prefWidth;
59                  index++;
60                  limitindex++;
61                  if(limitindex == rowLimit){
62                      status = 3;
63                      xstart += prefWidth;
64                      ystart -= prefHeight;
65                      limitindex = 0;
66                  }
67              } else if(status == 3){
68                  ystart -= prefHeight;
69                  index++;
70                  limitindex++;
71                  if(limitindex == rowLimit){
72                      break;
73                  }
74              }
75          }
76
77          return imageResult;
78      }
79
80  }
```

5.2.6. Implementasi Kode pada *ColorAnalysis.java*

Pembuatan kode pada *class* ini berpedoman pada *flowchart* yang telah dibuat pada bab 4. Tabel 5.7 menunjukkan kode pada *class* ColorAnalysis.java yang digunakan untuk mengekstrak fitur *skin count* dan *skin percentage*.

Tabel 5.7 Kode pada ColorAnalysis.java

No	Kode
1	`package ImageProcessing;`
2	
3	`import java.awt.image.BufferedImage;`
4	
5	`public class ColorAnalysis {`
6	
7	`    private float skinCount = 0;`
8	`    private float skinPercentage = 0;`
9	`    private int [][] imageMatrix;`
10	`    private BufferedImage image = null;`
11	
12	`    private void clearSkin(int xstart, int ystart){`
13	
14	`        for(int status=0; status<5; status++){`
15	
16	`            if(status == 0){`
17	`                imageMatrix[xstart][ystart] = 1;`
18	`                if((xstart+1)<image.getWidth()){`
19	`                    int argb = image.getRGB((xstart+1),`
20	`ystart);`
21	`                    if(argb != -16777216 &&`
22	`imageMatrix[xstart+1][ystart] == 0){`
23	`                        skinPercentage++;`
24	`                        clearSkin((xstart+1), ystart);`
25	`                    }`
26	`                }`
27	`            } else if(status == 1){`
28	`                if((xstart+1)<image.getWidth() &&`
29	`(ystart+1)<image.getHeight()){`
30	`                    int argb = image.getRGB((xstart+1),`
31	`(ystart+1));`
32	`                    if(argb != -16777216 &&`
33	`imageMatrix[xstart+1][ystart+1] == 0){`
34	`                        skinPercentage++;`
35	`                        clearSkin((xstart+1),`
36	`(ystart+1));`
37	`                    }`

```
38                  }
39              } else if(status == 2){
40                  if((ystart+1)<image.getHeight()){
41                      int argb = image.getRGB(xstart,
42  (ystart+1));
43                      if(argb != -16777216 &&
44  imageMatrix[xstart][ystart+1] == 0){
45                          skinPercentage++;
46                          clearSkin(xstart, (ystart+1));
47                      }
48                  }
49              } else if(status == 3){
50                  if((ystart+1)<image.getHeight() &&
51  (xstart-1)>-1){
52                      int argb = image.getRGB((xstart-1),
53  (ystart+1));
54                      if(argb != -16777216 &&
55  imageMatrix[xstart-1][ystart+1] == 0){
56                          skinPercentage++;
57                          clearSkin((xstart-1),
58  (ystart+1));
59                      }
60                  }
61              } else {
62                  if( (xstart-1)>-1){
63                      int argb = image.getRGB((xstart-1),
64  ystart);
65                      if(argb != -16777216 &&
66  imageMatrix[xstart-1][ystart] == 0){
67                          skinPercentage++;
68                          clearSkin((xstart-1), ystart);
69                      }
70                  }
71              }
72          }
73
74      }
75
76      public void processImage(){
77
78          int xindex = 0, yindex = 0;
79          while(xindex<image.getWidth() &&
80  yindex<image.getHeight()){
81
82              if(imageMatrix[xindex][yindex] == 0){
83                  int argb = image.getRGB(xindex,
84  yindex);
85                  if(argb != -16777216){
86                      skinCount++;
87                      skinPercentage++;
```

```
88                          clearSkin(xindex, yindex);
89                      }
90                  }
91
92                  xindex++;
93
94                  if(yindex < image.getHeight() && xindex ==
95  image.getWidth()){
96                      yindex++;
97                      xindex = 0;
98                  }
99
100             }
101
102
103         }
104
105         public void initialize(BufferedImage img){
106             image = img;
107
108             imageMatrix = new
109 int[img.getHeight()][img.getWidth()];
110             for(int i=0; i<imageMatrix.length; i++)
111                 for(int j=0; j<imageMatrix[i].length; j++)
112                     imageMatrix[i][j] = 0;
113         }
114
115         public float getSkinCount(){return skinCount;}
116         public float getSkinPercentage(){return
117 skinPercentage;}
118
119 }
```

5.2.7. Implementasi Kode pada *GLCM.java*

Pembuatan kode pada *class* ini menggunakan *flowchart* yang telah dibuat pada bab 4. GLCM yang dihasilkan pada *class* ini adalah berupa array dua dimensi dengan tipe double. Untuk menjalankan fungsi pada *class* ini dibutuhkan inputan berupa BufferedImage yang telah dikonversi menjadi *grayscale*. Tabel 5.8 menunjukkan kode yang digunakan pada *class* GLCM.java:

Tabel 5.8 Kode pada GLCM.java

No	Kode
1	`package ImageProcessing;`
2	
3	`import java.awt.image.BufferedImage;`
4	`import java.text.DecimalFormat;`
5	
6	`public class GLCM {`
7	
8	`    private BufferedImage image = null;`
9	`    private int [][] imageMatrix;`
10	`    private int maxIntensity = 0, maxGLCMValue = 0;`
11	`    private double [][] GLCMResult;`
12	
13	`    public GLCM(){`
14	
15	`    }`
16	
17	`    public void initialize(BufferedImage img){`
18	`        image = img;`
19	`        imageMatrix = new`
20	`int[img.getHeight()][img.getWidth()];`
21	`    }`
22	
23	`    public GLCM(BufferedImage img, int[] ang){`
24	`        image = img;`
25	`        imageMatrix = new`
26	`int[img.getHeight()][img.getWidth()];`
27	`    }`
28	
29	`    public double[][] getGLCMResult(){`
30	`        for (double[] GLCMResult1 : GLCMResult) {`
31	`            for (int j = 0; j < GLCMResult1.length;`
32	`j++) {`
33	`                GLCMResult1[j] = GLCMResult1[j] /`
34	`maxGLCMValue;`
35	`            }`
36	`        }`
37	
38	`        return GLCMResult;`
39	`    }`
40	
41	`    public void setImageMatrix(){`
42	`        int [][][] cek = new int [256][256][256];`
43	`        for(int i=0; i<256; i++)`
44	`            for(int j=0; j<256; j++)`
45	`                for(int k=0; k<256; k++)`
46	`                    cek[i][j][k] = -1;`

```
47
48
49          for(int i = 0; i < image.getWidth(); i++){
50              for(int j = 0; j < image.getHeight(); j++){
51                  int argb = image.getRGB(i, j);
52                  int rgb[] = new int[] {
53                      (argb >> 16) & 0xff, //red
54                      (argb >>  8) & 0xff, //green
55                      (argb      ) & 0xff  //blue
56                  };
57
58                  int r = rgb[0];
59                  int g = rgb[1];
60                  int b = rgb[2];
61
62                  if(cek[r][g][b] == -1){
63                      cek[r][g][b] = maxIntensity;
64                      maxIntensity++;
65                  }
66
67                  imageMatrix[i][j] = cek[r][g][b];
68
69              }
70          }
71
72
73      }
74
75      public void setGLCM(int [] angle){
76          for(int i=0; i<angle.length; i++){
77              double [][] tmpGLCM = new
78  double[maxIntensity][maxIntensity];
79              for(int xglcm=0; xglcm<tmpGLCM.length;
80  xglcm++)
81                  for(int yglcm=0;
82  yglcm<tmpGLCM[xglcm].length; yglcm++)
83                      tmpGLCM[xglcm][yglcm] = 0;
84
85              if(angle[i] == 0){
86                  for(int y=0; y<imageMatrix.length;
87  y++){
88                      for(int x=0;
89  x<imageMatrix[y].length-1; x++){
90                          int p1 = imageMatrix[x][y];
91                          int p2 = imageMatrix[x+1][y];
92
93                          tmpGLCM[p1][p2]++;
94                          maxGLCMValue++;
95                      }
96                  }
```

```
 97
 98              }
 99
100              if(angle[i] == 45){
101                  for(int y=imageMatrix.length-1; y>=1;
102  y--){
103                      for(int x=0;
104  x<imageMatrix[y].length-1; x++){
105                          int p1 = imageMatrix[x][y];
106                          int p2 = imageMatrix[x+1][y-1];
107
108                          tmpGLCM[p1][p2]++;
109                          maxGLCMValue++;
110                      }
111                  }
112              }
113
114              if(angle[i] == 90){
115                  for(int x=0; x<imageMatrix.length;
116  x++){
117                      for(int y=imageMatrix[x].length-1;
118  y>=1; y--){
119                          int p1 = imageMatrix[x][y];
120                          int p2 = imageMatrix[x][y-1];
121
122                          tmpGLCM[p1][p2]++;
123                          maxGLCMValue++;
124                      }
125                  }
126
127              }
128
129              if(angle[i] == 135){
130                  for(int y=imageMatrix.length-1; y>=1;
131  y--){
132                      for(int x=imageMatrix[y].length-1;
133  x>=1; x--){
134                          int p1 = imageMatrix[x][y];
135                          int p2 = imageMatrix[x-1][y-1];
136
137                          tmpGLCM[p1][p2]++;
138                          maxGLCMValue++;
139                      }
140                  }
141
142              }
143
144              if(angle[i] == 180){
145                  for(int y=0; y<imageMatrix.length;
146  y++){
```

```
147                         for(int x=imageMatrix[y].length-1;
148 x>=1; x--){
149                             int p1 = imageMatrix[x][y];
150                             int p2 = imageMatrix[x-1][y];
151 
152                             tmpGLCM[p1][p2]++;
153                             maxGLCMValue++;
154                         }
155                     }
156 
157                 }
158 
159                 if(angle[i] == 225){
160                     for(int y=0; y<imageMatrix.length-1;
161 y++){
162                         for(int x=imageMatrix[y].length-1;
163 x>=1; x--){
164                             int p1 = imageMatrix[x][y];
165                             int p2 = imageMatrix[x-1][y+1];
166 
167                             tmpGLCM[p1][p2]++;
168                             maxGLCMValue++;
169                         }
170                     }
171 
172                 }
173 
174                 if(angle[i] == 270){
175                     for(int x=imageMatrix.length-1; x>=0;
176 x--){
177                         for(int y=0;
178 y<imageMatrix[x].length-1; y++){
179                             int p1 = imageMatrix[x][y];
180                             int p2 = imageMatrix[x][y+1];
181 
182                             tmpGLCM[p1][p2]++;
183                             maxGLCMValue++;
184                         }
185                     }
186 
187                 }
188 
189                 if(angle[i] == 315){
190                     for(int y=0; y<imageMatrix.length-1;
191 y++){
192                         for(int x=0;
193 x<imageMatrix[y].length-1; x++){
194                             int p1 = imageMatrix[x][y];
195                             int p2 = imageMatrix[x+1][y+1];
196 
```

```
197                             tmpGLCM[p1][p2]++;
198                             maxGLCMValue++;
199                         }
200                     }
201
202                 }
203
204
205                 addGLCMValue(tmpGLCM);
206             }
207
208             int max = 0;
209             for(int i=0; i<GLCMResult.length; i++){
210                 for(int j=0; j<GLCMResult[i].length; j++)
211                     max += GLCMResult[i][j];
212             }
213             for(int i=0; i<GLCMResult.length; i++){
214                 for(int j=0; j<GLCMResult[i].length; j++){
215                     DecimalFormat f = new
216 DecimalFormat("##.000000");
217                     GLCMResult[i][j] = GLCMResult[i][j];
218
219                 }
220             }
221
222
223         }
224
225         private void addGLCMValue(double [][] tmpGLCM){
226             if(GLCMResult == null) GLCMResult = tmpGLCM;
227             else{
228                 for(int i=0; i<GLCMResult.length; i++)
229                     for(int j=0; j<GLCMResult[i].length;
230 j++){
231                         GLCMResult[i][j] += tmpGLCM[i][j];
232                     }
233             }
234         }
235
236 }
```

5.2.8. Implementasi Kode pada *TextureAnalysis.java*

Pembuatan kode pada *class* ini berpedoman pada *flowchart* yang telah dibuat pada bab 4. Analisis tekstur akan diimplementasikan dengan menggunakan fitur *contrast*,

correlation, *energy*, dan *homogeneity*. Inputan untuk *class* ini adalah GLCM dan hasil dari pemrosesannya adalah berupa empat nilai double yang masing-masing merupakan nilai dari keempat fitur di atas. Tabel 5.9 menunjukkan kode pada *class* TextureAnalysis.java.

Tabel 5.9 Kode pada TextureAnalysis.java

No	Kode
1	`package ImageProcessing;`
2	
3	`public class TextureAnalysis {`
4	
5	`    private double [][] glcm;`
6	
7	`    public void setGLCM(double [][] glcmTmp){`
8	`        glcm = glcmTmp;`
9	`    }`
10	
11	`    public double getContrast(){`
12	`        double result = 0;`
13	
14	`        for(int i=0; i<glcm.length; i++)`
15	`            for(int j=0; j<glcm[i].length; j++){`
16	`                result += Math.pow((i-j), 2) *`
17	`glcm[i][j];`
18	`            }`
19	
20	`        return result;`
21	`    }`
22	
23	`    public double getCorrelation(){`
24	`        double result = 0;`
25	
26	`        double mi = 0, mj = 0;`
27	`        double sdi = 0, sdj = 0;`
28	
29	`        for(int i=0; i<glcm.length; i++){`
30	`            for(int j=0; j<glcm[i].length; j++){`
31	`                mi += i * glcm[i][j];`
32	`                mj += j * glcm[i][j];`
33	`            }`
34	`        }`
35	
36	`        for(int i=0; i<glcm.length; i++){`
37	`            for(int j=0; j<glcm[i].length; j++){`
38	`                sdi += glcm[i][j] * Math.pow((i - mi),`

```
39  2);
40                  sdj += glcm[i][j] * Math.pow((j - mj),
41  2);
42              }
43          }
44
45          sdi = Math.sqrt(sdi);
46          sdj = Math.sqrt(sdj);
47
48          for(int i=0; i<glcm.length; i++){
49              for(int j=0; j<glcm[i].length; j++){
50                  result += (glcm[i][j] * ((i-mi)*(j-mj))
51  / (sdi*sdj));
52              }
53          }
54
55          return result;
56      }
57
58      public double getEnergy(){
59          double result = 0;
60
61          for(int i=0; i<glcm.length; i++){
62              for(int j=0; j<glcm[i].length; j++){
63                  double enrgTmp = Math.pow(glcm[i][j],
64  2);
65                  result += enrgTmp;
66              }
67          }
68
69          result = Math.sqrt(result);
70
71          return result;
72      }
73
74      public double getHomogeneity(){
75          double result = 0;
76
77          for(int i=0; i<glcm.length; i++){
78              for(int j=0; j<glcm[i].length; j++){
79                  result += glcm[i][j] / (1+Math.pow((i-
80  j), 2));
81              }
82          }
83
84          return result;
85      }
86
87  }
```

5.2.9. Implementasi Kode pada *ShapeAnalysis.java*

Pembuatan kode pada *class* ini berpedoman pada *flowchart* yang telah dibuat pada bab 4. Analisis bentuk akan dilakukan dengan menggunakan tujuh *Hu Moment* sehingga dalam *class* ini akan terdapat *subclass* bernama HuMoment. Inputan yang dibutuhkan pada *class* ini adalah citra yang telah mengalami proses *skin detection*. Tabel 5.10 menunjukkan kode pada *class* ShapeAnalysis.java.

Tabel 5.10 Kode pada ShapeAnalysis.java

No	Kode
1	`package ImageProcessing;`
2	
3	`import java.awt.image.BufferedImage;`
4	
5	`public class ShapeAnalysis {`
6	
7	`    private BufferedImage image;`
8	`    private double mi, mj;`
9	
10	`    public void initialize(BufferedImage img){`
11	`        image = img;`
12	
13	`        double m00=0, m10=0, m01=0;`
14	`        for(int i=0; i<img.getHeight(); i++)`
15	`            for(int j=0; j<img.getWidth(); j++){`
16	`                int argb = image.getRGB(i, j);`
17	
18	`                if(argb == -16777216) argb = 0;`
19	`                else argb = 1;`
20	
21	`                m00 += argb;`
22	`                m10 += Math.pow(i, 1)*argb;`
23	`                m01 += Math.pow(j, 1)*argb;`
24	`            }`
25	
26	`        mi = m10 / m00;`
27	`        mj = m01 / m00;`
28	`    }`
29	
30	`    public double[] getHuMoment(){`

```
31          HuMoment hu = new HuMoment(mi, mj);
32          hu.computenpq(image);
33
34          return hu.getHuMoment();
35      }
36
37      private class HuMoment{
38          private int[][] pq;
39          private double mi, mj;
40          private double n20, n02, n11, n30, n12, n21,
41  n03;
42
43          public HuMoment(double i, double j){
44
45              mi = i;
46              mj = j;System.out.println(mi+";"+mj);
47          }
48
49          public void computenpq(BufferedImage img){
50              double mi00=0, mi20=0,mi02=0, mi11=0,
51  mi30=0, mi12=0, mi21=0,mi03=0;
52
53              for(int i=0; i<img.getHeight(); i++){
54                  for(int j=0; j<img.getWidth(); j++){
55                      int argb = img.getRGB(i, j);
56
57                      if(argb == -16777216) argb = 0;
58                      else argb = 1;
59
60                      mi00 += argb;
61                      mi20 += Math.pow((i-mi), 2) * argb;
62                      mi02 += Math.pow((j-mj), 2) * argb;
63                      mi11 += Math.pow((i-mi), 1) *
64  Math.pow((j-mj), 1) * argb;
65                      mi30 += Math.pow((i-mi), 3) * argb;
66                      mi12 += Math.pow((i-mi), 1) *
67  Math.pow((j-mj), 2) * argb;
68                      mi21 += Math.pow((i-mi), 2) *
69  Math.pow((j-mj), 1) * argb;
70                      mi03 += Math.pow((j-mj), 3) * argb;
71                  }
72              }
73
74              n20 = mi20 / Math.pow(mi00, 2);
75              n02 = mi02 / Math.pow(mi00, 2);
76              n11 = mi11 / Math.pow(mi00, 2);
77              n30 = mi30 / Math.pow(mi00, 2.5);
78              n12 = mi12 / Math.pow(mi00, 2.5);
79              n21 = mi21 / Math.pow(mi00, 2.5);
80              n03 = mi20 / Math.pow(mi00, 2.5);
```

```
81          }
82
83          public double [] getHuMoment(){
84              double huMoment[] = new double[7];
85
86              huMoment[0] = n20 + n02;
87              huMoment[1] = Math.pow((n20-n02), 2) *
88  4*Math.pow(n11, 2);
89              huMoment[2] = Math.pow((n30+n12), 2) *
90  Math.pow((n21+n03), 2);
91              huMoment[3] = Math.pow((n20-n02), 2) *
92  4*Math.pow(n11, 2);
93              huMoment[4] =
94  (n30*3*n12)*(n30+n12)*(Math.pow((n30+n12), 2)-
95  3*Math.pow((n21-n03), 2))
96                      +(3*n21-
97  n03)*(n21+n03*(3*Math.pow((n30+n12), 2)-Math.pow((n21-
98  n03), 2)));
99              huMoment[5] = (n20-
100 n02)*(Math.pow((n30+n12), 2)-Math.pow((n21+n03), 2))
101                     +4*n11*(n30+n12)*(n21+n03);
102             huMoment[6] = (3*n21-
103 n03)*(n30+n12)*(Math.pow((n30+n12), 2)-
104 3*Math.pow((n21+n03), 2))
105                     -(n30-
106 3*n12)*(n21+n03)*(3*Math.pow((n30+n12), 2)-
107 Math.pow((n21+n03), 2));
108
109             return huMoment;
110
111         }
112     }
113
114 }
```

5.3 Pengujian Desain Aplikasi Berbasis Desktop

Setelah aplikasi berbasis desktop selesai dibangun maka tahap selanjutnya adalah melakukan pengecekan terhadap seluruh fungsi pada aplikasi tersebut. Kemudian hasil ekstraksi, yaitu 13 nilai fitur dari masing-masing citra dicatat. Pengujian aplikasi berbasis desktop dilakukan secara bertahap sesuai dengan *workflow* pada bab 4. Pada pengujian ini, digunakan beberapa citra. Citra pertama adalah citra dengan manusia di dalamnya

namun tidak memiliki konten pornografi, citra kedua adalah citra pemandangan yang tidak memiliki konten pornografi, dan citra ketiga adalah citra yang memiliki konten pornografi. Tabel 5.11 menunjukkan citra yang akan digunakan selama tahap pengujian.

Tabel 5.11 Citra yang Digunakan Selama Tahap Pengujian Aplikasi Berbasis Desktop

Kode Citra	Citra
C1	
C2	

5. 3. 1 Pengujian Kode Untuk Mengubah Resolusi Citra

Pengubahan resolusi citra menjadi 256 x 256 adalah tahap pertama yang akan dilakukan aplikasi. Dengan kode yang telah dibuat citra dengan format bmp, png dan jpg dan dengan berbagai macam resolusi dapat diubah menjadi resolusi 256 x 256.

5. 3. 2 Pengujian Kode *Skin Detection*

Pengujian *skin detection* dilakukan dengan menginputkan citra yang memiliki kulit manusia. Kemudian citra tersebut diproses sehingga menghasilkan citra baru yang hanya memiliki pixel kulit manusia. Untuk merealisasikan hal ini, kode tambahan disisipkan pada *class* Process.java yang berfungsi untuk membuat citra hasil pemrosesan tersebut. Kode ini dibuat ke dalam *method* bernama drawImage() dengan parameter berupa BufferedImage yaitu citra yang akan dilakukan *skin detection*, lokasi untuk menyimpan citra yang baru, dan nama citra hasil pemrosesan *skin detection*. Tabel 5.12 menunjukkan kode tambahan pada Process.java untuk menguji fungsi *skin detection* yang telah dibuat.

Tabel 5.12 Kode Tambahan pada Process.java Untuk Menguji Fungsi *Skin Detection*

No	Kode
1	`public void drawImage(BufferedImage img, String`
2	`savedLocation, String name){`
3	`        try{`
4	`            ImageIO.write(img, "jpg", new`
5	`File(savedLocation+"\\"+name));`
6	`        } catch(IOException e){`
7	`            System.err.println("Unable to output`
8	`results");`
9	`        }`
10	`    }`

Tabel 5.13 menunjukkan sampel citra hasil dari pemrosessan *skin detection*.

Tabel 5.13 Hasil Pengujian Kode *Skin Detection*

Kode Citra	Citra Hasil *Skin Detection*
	
	

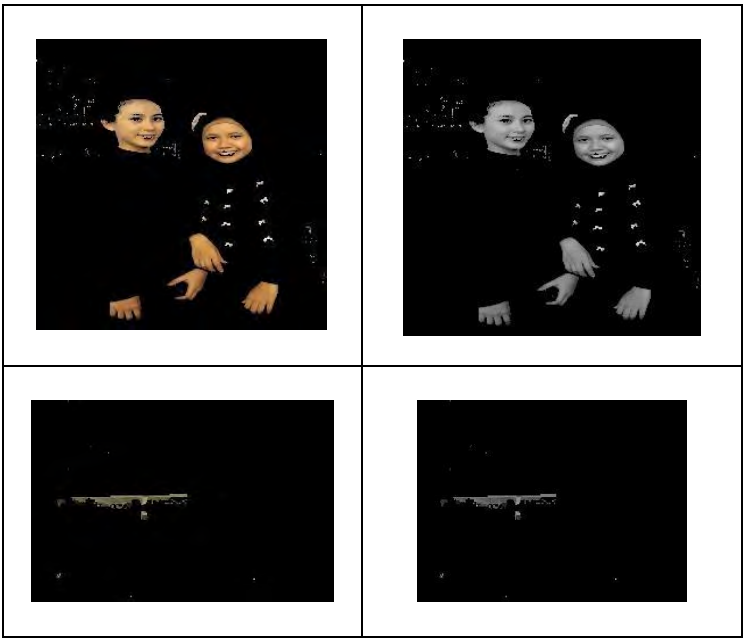
Dari perbandingan yang telah ditunjukkan di atas maka dapat disimpulkan bahwa kode *skin detection* yang telah dibangun telah mampu untuk mendeteksi pixel-pixel yang tergolong ke dalam kulit manusia.

5. 3. 3 Pengujian Kode Untuk Mengkonversi Citra Menjadi *Grayscale*

Kode yang telah dibuat telah mampu untuk mengkonversi citra menjadi *grayscale*. Tabel 5.14 menunjukkan hasil dari konversi citra menjadi *grayscale*.

Tabel 5.14 Hasil Pengujian Kode Konversi Citraa Menjadi *Grayscale*

Citra Asli	Citra Hasil *Skin Detection*

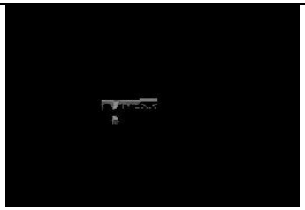


5. 3. 4 Pengujian Kode *Image Zoning*

Kode *Image Zoning* yang telah dibuat akan membagi citra menjadi tiga zona dan menghasilkan citra baru sejumlah zona yang telah didefinisikan, yaitu sejumlah tiga buah. Inputannya adalah citra hasil pemrosesan *skin detection*. Tabel 5.15 menunjukkan hasil dari kode *Image Zoning* yang telah dibuat.

Tabel 5.15 Hasil Pengujian Kode *Image Zoning*

Citra Inputan	**Zona**	**Citra Hasil *Skin Detection***
	1	
	2	
	3	
	1	

		
	2	
	3	

5. 3. 5 Pengujian Kode Ekstraksi *Skin Count* dan *Skin Percentage*

Dari kode yang telah dibuat maka didapatkan nilai dari *skin count* dan *skin percentage* dari ketiga citra yang dapat dilihat pada tabel 5.16.

Tabel 5.16 Hasil Perhitungan *Skin Count* dan *Skin Percentage*

Kode Citra	Zona	*Skin Count*	*Skin Percentage*
C1	1	22	1.237
	2	30	5.07
	3	45	2.208
	Total	97	8.515
C2	1	7	0.596
	2	4	0.654
	3	7	0.181
	Total	18	1.43

5. 3. 6 Pengujian Kode Pembuatan GLCM

GLCM dibuat setelah citra terkonversi ke dalam *grayscale*. Setelah kode GLCM dijlankan didapatkan tiga matriks dengan rentang masing-masing adalah sebesar 127 x 127 dan 189 x 189.

5. 3. 7 Pengujian Kode Ekstraksi *Contrast, Correlation, Energy,* dan *Homogeneity*

Dari kode yang telah dibuat maka didapatkan nilai dari *contrast, correlation, energy* dan *homogeneity* dari ketiga citra yang dapat dilihat pada tabel 5.17

.

Tabel 5.17 Hasil Perhitungan *Contrast*, *Correlation*, *Energy*, dan *Homogeneity*

Kode Citra	Zona	*Contrast*	*Correlation*	*Energy*	*Homogeneity*
C1	1	48.821	0.6005	0.9855	0.9862
	2	183.195	0.6574	0.9439	0.9473
	3	67.639	0.7250	0.9748	0.9766
	Total	299.655	1.9829	2.9042	2.9102
C2	1	10.926	0.6125	0.9926	0.9938
	2	7.156	0.6788	0.9923	0.9934
	3	1.139	0.6405	0.9976	0.9980
	Total	19.221	1.9318	2.9825	2.9852

5. 3. 8 Pengujian Kode Ekstraksi *Hu Moment*

Dari kode yang telah dibuat maka didapatkan nilai dari 7 *Hu Moment* dari ketiga citra yang dapat dilihat pada tabel 5.18.

Tabel 5.18 Hasil Perhitungan 7 *Hu Moment*

Kode Citra	Zona	*Hu1*	*Hu2*	*Hu3*	*Hu4*	*Hu5*	*Hu6*	*Hu7*
C1	1	1.44	0.07	0.00	0.07	0.01	0.10	0.00
	2	1.27	0.02	0.00	0.02	0.05	0.00	0.00
	3	2.79	0.42	3.94	0.42	-26.8	9.30	75.0
	Total	5.5	0.51	3.94	0.51	-26.74	9.4	75
C2	1	0.62	0.00	0.00	0.00	0.00	0.00	0.00
	2	0.43	0.00	0.00	0.00	0.00	0.00	0.00
	3	20.6	1450.4	3136714.6	1450.4	77784054.4	-20142.9	-864356.6
	Total	21.65	1450.4	3136715	1450.4	77784054	-20142.9	-864357

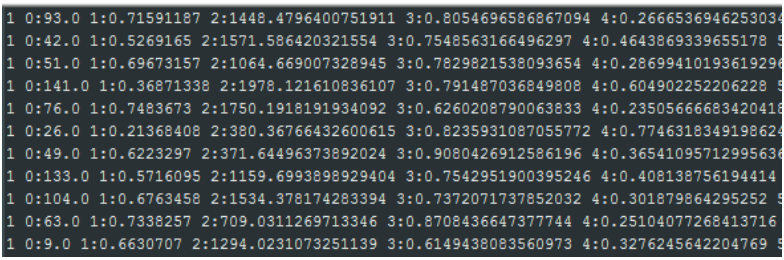
5. 3. 9 Pengujian Kode Pembuatan File *training.txt*

Setelah semua nilai dari fitur didapatkan, langkah selanjutnya adalah menyimpan nilai tersebut ke dalam sebuah file .txt untuk dijadikan sebagai bahan *training*. Kode untuk membuat file ini telah disesuaikan dengan format LibSVM sehingga hasilnya dapat langsung diproses dengan menggunakan LibSVM.

Kode yang telah dibuat untuk membuat file *training.txt* telah diuji dan berfungsi sesuai dengan yang diharapkan, yaitu membuat file *training.txt* dengan format yang digunakan oleh LibSVM.

5.4 Pembuatan File *training.txt.scale.model* dan File *range*

Pada tahap ini, seluruh data *training* yang telah dikumpulkan diekstrak dengan menggunakan aplikasi berbasis desktop. Hasil dari kode tersebut adalah berupa file bernama *training.txt*. Gambar 5.1 menunjukkan isi dari file *training.txt*.



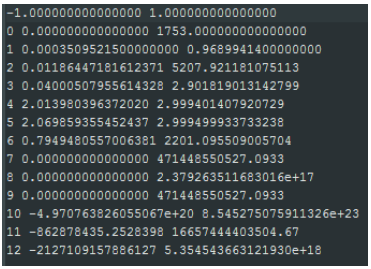
Gambar 5.1 Isi File *training.txt*

Setelah file *training.txt* terebentuk maka langkah selanjutnya adalah membuat file *range*. File ini berfungsi untuk melakukan skalasi terhadap nilai dari setiap fitur. Skalasi dilakukan karena metode SVM akan berkurang akurasinya jika nilai fiturnya tidak berada di antara -1 sampai 1. Pembuatan file *range* dilakukan dengan menggunakan file svm_scale.java milik LibSVM yang dijalankan pada Command Prompt Windows. Untuk menggunakan fungsi skalasi melalui file ini, sebuah perintah dijalankan dengan format sebagai berikut:

```
java svm_scale -s [nama file output] > [nama file scale]
```

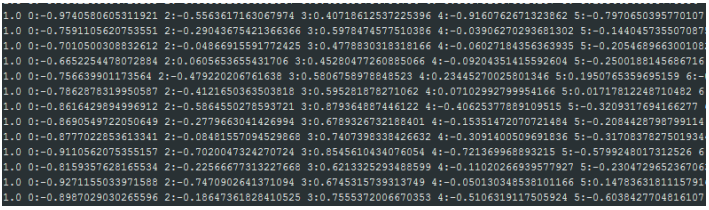
Dengan menggunakan format tersebut maka perintah yang diinputkan adalah `java svm_scale -s range > training.txt.scale`. Perintah tersebut akan menghasilkan dua buah file yaitu file *range* dan file *training.txt.scale*. File *training.txt.scale* adalah file yang berisi hasil dari skalasi (nilai dari setiap fitur hanya terletak pada jangkauan antara -1 sampai

dengan 1) yang telah dilakukan. File ini yang akan digunakan untuk pembuatan file *training.txt.scale.model*. Gambar 5.2 menunjukkan isi dari file *range*.



Gambar 5.2 Isi File *range*

Sedangkan gambar 5.3 menunjukkan isi dari file *training.txt.scale*.



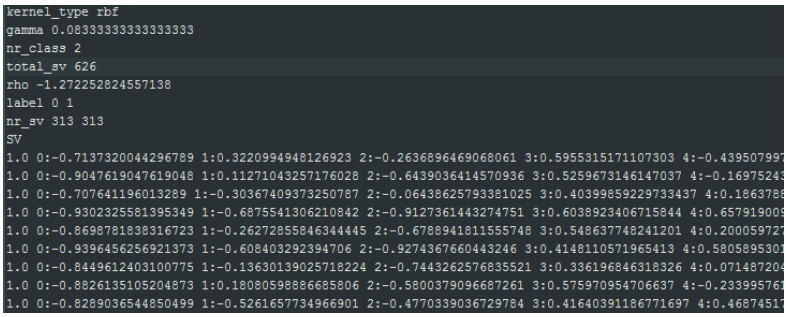
Gambar 5.3 Isi File *training.txt.scale*

Setelah mendapatkan file *training.txt.scale* maka langkah selanjutnya adalah membuat file *training.txt.scale.model* yang akan digunakan selama proses komputasi SVM. Untuk membuat file ini digunakan file svm_predict.java dengan format perintah sebagai berikut:

```
java svm_train [nama file scale]
```

Dengan menggunakan format di atas maka perintah yang digunakan adalah `java svm_train`

`training.txt.scale`. Perintah tersebut akan menghasilkan file *training.txt.scale.model*. Gambar 5.4 menunjukkan isi dari file *training.txt.scale.model*.



Gambar 5.4 Isi File *training.txt.scale.model*

5.5 Pemilihan Parameter SVM

Agar terbentuk model yang paling optimal untuk melakukan prediksi maka pemilihan kombinasi nilai fitur dilakukan. Hal ini dilakukan dengan menggunakan algoritma *Grid Search* dengan bantuan file grid.py dari LibSVM. File ini merupakan file dengan lingkungan Python sehingga harus diawali dengan instalasi lingkungan Python pada system operasi yang digunakan.

Untuk menjalankan grid.py dibutuhkan file *training data* yang telah diskalasi. Berikut ini adalah format perintah untuk menjalankan file grid.py:

```
grid.py [nama file training yang telah diskalasi]
```

Pemrosesan grid.py menghasilkan nilai C = 128.0 dan γ = 2.0. Kedua nilai ini kemudian diimplementasikan pada file bernama svm_parameter.java.

5.6 Implementasi Desain *Web Service*

Untuk membangun *web service* dibutuhkan komputer personal, perangkat lunak, *library*, dan infrastruktur dengan spesifikasi pada subbab 5.1. setelah seluruh spesifikasi terpenuhi maka langkah selanjutnya adalah membuat Java Web Service Project baru bernama *PornImageDetectionWS* dengan menggunakan Netbeans 7.4. Sesuai dengan *class diagram* yang telah dibuat maka *package*_yang ada pada *web service* adalah sebagai berikut:

1) code.core.featureextraction, yang berisi *class* sebagai berikut:
   - Extraction.java: digunakan untuk mengekstrak fitur dari citra yang diinputkan.
   - Prediction.java: digunakan untuk memprediksi hasil dari citra yang diinputkan.
   - Scalation.java: digunakan untuk melakukan skalasi terhadap nilai fiturr yang telah diekstrak.
2) code.core.imageprocessing, yang berisi *class* yang sama dengan yang ada pada *package* Imagerocessing pada aplikasi berbasis desktop.
3) code.code.webservice, yang berisi *class* sebagai berikut:
   - PornImageDetectionWS.java: berfungsi sebagai *web service* yang mampu menerima inputan berupa link dari citra yang akan diuji.
4) code.lib.libsvm, dengan *class* yag digunakan sebagai berikut:
   - svm.java: berfungsi untuk menjalankan komputasi SVM.
   - svm_node.java: digunakan untuk membuat *node* yang berisi nilai fitur sehingga mampu diolah melalui komputasi LibSVM.
5) jjilexample.haar: berfungsi untuk mengaktifkan fungsi *face detection*.

6) code.core.facedetection, dengan *class* yang digunakan sebagai berikut:
   - FaceDetection.java: berfungsi untuk menghitung nilai persentase wajah pada citra.
   - Gray8DetectHaarMultiScale.java: berfungsi untuk melengkapi fitur *face detection*.

Selain pembuatan *package* dan *class*, file *training.txt.scale.model* dan file *range* disimpan pada folder /Web Pages/WEB-INF/classes.

5.5.1. Implementasi Kode *Extraction.java*

Class ini berfungsi untuk mengekstrak nilai fitur dari citra yang diinputkan. Namun inputan yang disediakan sistem tidak berupa file *binary* dari citra melainkan url dari citra. Karena sistem dibangun dengan menggunakan jenis koneksi autentikasi proxy maka diperlukan kode untuk dapat terhubung dengan jaringan tersebut. Gambar 5.5 menunjukkan kode yang diperlukan agar sistem terhubung dengan proxy.its.ac.id:8080 dan dengan username dan password tertentu.

```
Properties sysProps = System.getProperties();
sysProps.put("proxySet", "true");
sysProps.put("proxyHost", "proxy.its.ac.id");
sysProps.put("proxyPort", "8080");
Authenticator authenticator = new Authenticator() {

public PasswordAuthentication getPasswordAuthentication() {
    return (new PasswordAuthentication("[redacted]","[redacted]".toCharArray()));
}
};
Authenticator.setDefault(authenticator);
```

Gambar 5.5 Kode Agar Sistem Terhubung dengan Jaringan Proxy

Pada gambar di atas, bagian yang disamarkan adalah username dan password agar daat terhubung dengan internet. Tabel 5.19 menunjukkan kode lengkap dari *class* Extraction.java.

Tabel 5.19 Kode pada Extraction.java

No	Kode
1	package code.core.featureextraction;
2	
3	import code.core.imageprocessing.*;
4	import code.core.webservice.PornImageDetectionWS;
5	
6	import java.awt.Graphics2D;
7	import java.awt.image.BufferedImage;
8	import java.io.BufferedReader;
9	import java.io.InputStream;
10	import java.io.InputStreamReader;
11	import java.net.Authenticator;
12	import java.net.PasswordAuthentication;
13	import java.net.URL;
14	import java.util.Properties;
15	import javax.imageio.ImageIO;
16	
17	public class Extraction {
18	
19	/* Dynamic variables - start */
20	private final int normalizedWidth=256,
21	normalizedHeight = 256;
22	private final int maxZone = 3;
23	
24	private final String rangeFile = "range1";
25	private final String modelFile =
26	"training.txt.scale.model";
27	/* finish */
28	
29	public boolean isImagePorn(String imageURL){
30	
31	boolean result = false;String error="";
32	
33	Properties sysProps = System.getProperties();
34	sysProps.put("proxySet", "true");
35	sysProps.put("proxyHost", "proxy.its.ac.id");
36	sysProps.put("proxyPort", "8080");
37	Authenticator authenticator = new
38	Authenticator() {
39	
40	public PasswordAuthentication
41	getPasswordAuthentication() {
42	return (new
43	PasswordAuthentication("hani10@mhs.is.its.ac.id","angry
44	bird".toCharArray()));
45	}
46	};

```
47          Authenticator.setDefault(authenticator);
48
49          URL url = null;
50          BufferedImage imgused = null;
51          int type = 0;
52
53          try{
54              url = new URL(imageURL);
55              imgused = ImageIO.read(url.openStream());
56              type = imgused.getType();
57          } catch(Exception e){
58
59          }
60
61
62          imgused = getResizedImage(imgused,
63  normalizedWidth, normalizedHeight, type);
64          imgused = getSkinPixel(imgused);
65          imgused =
66  getGrayscaleImage(imgused);//drawImage(imgused, "D:\\",
67  "test.jpg");
68
69          double [] extractionResult = null;
70
71          try{
72              extractionResult =
73  getExtractedFeatures(imgused);
74
75              Scalation scale = new Scalation();
76              scale.setRangeFile(rangeFile);
77              extractionResult =
78  scale.getScaledValue(extractionResult);
79
80              Prediction prediction = new Prediction();
81              InputStream inputStream =
82  PornImageDetectionWS.class.getClassLoader()
83                      .getResourceAsStream(modelFile);
84              BufferedReader modelBr = new
85  BufferedReader(new InputStreamReader(inputStream));
86              prediction.setSVMModel(modelBr);
87
88              double classResult =
89  prediction.getPrediction(extractionResult);error =
90  String.valueOf(classResult);
91
92              if(classResult == 1) result = true;
93          } catch (Exception e){
94
95          }
96
```

```
 97          return result;
 98
 99
100      }
101
102      private double[] getExtractedFeatures(BufferedImage
103  image){
104
105          ImageZoning imgzoning = new ImageZoning();
106          imgzoning.setMaxZone(maxZone);
107
108          float skinCountTotal = 0, skinPercentageTotal =
109  0;
110          double contrastTotal = 0, correlationTotal = 0,
111  energyTotal = 0, homogeneityTotal = 0;
112          double huMomentTotal[];
113          huMomentTotal = new double[]{0, 0, 0, 0, 0, 0,
114  0};
115
116          BufferedImage tmp = image;
117
118          for(int zone=1; zone<=maxZone; zone++){
119              image = imgzoning.getZoningImage(tmp,
120  zone);
121
122
123              /* Step 1: Lakukan Analisis Warna */
124              ColorAnalysis colorAnalysis = new
125  ColorAnalysis();
126              colorAnalysis.initialize(image);
127              colorAnalysis.processImage();
128              float skinCount =
129  colorAnalysis.getSkinCount();
130              if(skinCount == 0) continue;
131              float skinPercentage =
132  colorAnalysis.getSkinPercentage() / (image.getWidth() *
133  image.getHeight());
134
135              skinCountTotal += skinCount;
136              skinPercentageTotal += skinPercentage;
137
138              /* Step 2: Lakukan Analisis Tekstur */
139
140              GLCM glcm = new GLCM();
141              glcm.initialize(image);
142              glcm.setImageMatrix();
143              glcm.setGLCM(new int[]{0, 45, 90, 135, 180,
144  225, 270, 315});
145              double [][] glcmResult =
146  glcm.getGLCMResult();
```

```
147
148             TextureAnalysis textureAnalysis = new
149 TextureAnalysis();
150             textureAnalysis.setGLCM(glcmResult);
151             double contrast =
152 textureAnalysis.getContrast();
153             double correlation =
154 textureAnalysis.getCorrelation();
155             double energy =
156 textureAnalysis.getEnergy();
157             double homogeneity =
158 textureAnalysis.getHomogeneity();
159
160             contrastTotal += contrast;
161             correlationTotal += correlation;
162             energyTotal += energy;
163             homogeneityTotal += homogeneity;
164
165             /* Step 3: Lakukan Analisis Bentuk */
166             ShapeAnalysis shapeAnalysis = new
167 ShapeAnalysis();
168             shapeAnalysis.initialize(image);
169             double [] huMoment =
170 shapeAnalysis.getHuMoment();
171             String imgname = "";
172             for(int i=0; i<huMoment.length; i++){
173                 huMomentTotal[i] += huMoment[i];
174                 imgname += huMomentTotal[i]+"_";
175             }
176
177             //drawImage(image, "D:\\",
178 imgname+"_"+zone+"hai.jpg");
179
180         }
181
182         double [] result = new double[]{
183             skinCountTotal,
184             skinPercentageTotal,
185             contrastTotal,
186             correlationTotal,
187             energyTotal,
188             homogeneityTotal,
189             huMomentTotal[0],
190             huMomentTotal[1],
191             huMomentTotal[2],
192             huMomentTotal[3],
193             huMomentTotal[4],
194             huMomentTotal[5],
195             huMomentTotal[6]
196         };
```

```
197
198         return result;
199     }
200
201     private BufferedImage
202 getGrayscaleImage(BufferedImage image){
203
204         for(int i = 0; i < image.getWidth(); i++){
205             for(int j = 0; j < image.getHeight(); j++){
206                 int argb = image.getRGB(i, j);
207                 int rgb[] = new int[] {
208                     (argb >> 16) & 0xff, //red
209                     (argb >>  8) & 0xff, //green
210                     (argb      ) & 0xff  //blue
211                 };
212
213                 double r = (double) rgb[0];
214                 double g = (double) rgb[1];
215                 double b = (double) rgb[2];
216
217                 int grayLevel = (int) (r + g + b) / 3;
218                 int gray = (grayLevel << 16) +
219 (grayLevel << 8) + grayLevel;
220
221                 image.setRGB(i, j, gray);
222             }
223         }
224
225         return image;
226     }
227
228     private BufferedImage getResizedImage(BufferedImage
229 img, int width, int height, int type){
230         BufferedImage resizedImage = new
231 BufferedImage(width, height, type);
232         Graphics2D g = resizedImage.createGraphics();
233         g.drawImage(img, 0, 0, width, height, null);
234         g.dispose();
235
236         return resizedImage;
237     }
238
239     private BufferedImage getSkinPixel(BufferedImage
240 img){
241
242         SkinDetection skin = new SkinDetection();
243         skin.setImage(img);
244         skin.setImageResult(img.getWidth(),
245 img.getHeight());
246         skin.useYCbCr();
```

```
247
248             return skin.getImageResult();
249         }
250     }
```

5.5.2. Implementasi Kode *Prediction.java*

Tabel 5.20 menunjukkan kode pada *class* Prediction.java.

Tabel 5.20 Kode pada Prediction.java

No	Kode
1	`package code.core.featureextraction;`
2	
3	`import code.lib.libsvm.*;`
4	`import java.io.BufferedReader;`
5	`import java.io.IOException;`
6	`import java.util.logging.Level;`
7	`import java.util.logging.Logger;`
8	
9	`public class Prediction {`
10	
11	`    private svm_model model;`
12	
13	`    public void setSVMModel(BufferedReader br){`
14	`        try {`
15	`            model = svm.svm_load_model(br);`
16	`        } catch (IOException ex) {`
17	
18	`Logger.getLogger(Prediction.class.getName()).log(Level.`
19	`SEVERE, null, ex);`
20	`        }`
21	`    }`
22	
23	`    public double getPrediction(double [] data){`
24	`        double result = -1;`
25	
26	`        svm_node[] x = new svm_node[13];`
27	`        for(int i=0; i<x.length; i++){`
28	`            x[i] = new svm_node();`
29	`            x[i].index = i;`
30	`            x[i].value = data[i];`
31	`        }`
32	
33	`        result = svm.svm_predict(model, x);`
34	
35	`        return result;`

```
36      }
37
28  }
```

5.5.3. Implementasi Kode *Scalation.java*

Tabel 5.21 menunjukkan kode pada Scalation.java.

Tabel 5.21 Implementasi Kode Scalation.java

No	Kode
1	`package code.core.featureextraction;`
2	
3	`import code.core.webservice.PornImageDetectionWS;`
4	`import java.io.BufferedReader;`
5	`import java.io.BufferedWriter;`
6	`import java.io.File;`
7	`import java.io.FileOutputStream;`
8	`import java.io.InputStream;`
9	`import java.io.InputStreamReader;`
10	`import java.io.OutputStreamWriter;`
11	`import java.io.Writer;`
12	`import java.util.Properties;`
13	`import java.util.StringTokenizer;`
14	`import sun.misc.IOUtils;`
15	
16	`/**`
17	` *`
18	` * @author Bisma`
19	` */`
20	`public class Scalation {`
21	
22	`    private String rangeFile;`
23	
24	`    public void setRangeFile(String file){`
25	`        rangeFile = file;`
26	`    }`
27	
28	`    public double[] getScaledValue(double x[]){`
29	`        String errorMsg = "";`
30	`        double rangeValue[][] = new double[14][2];`
31	
32	`        InputStream inputStream = null;`
33	`        BufferedReader br = null;`
34	`        //StringBuilder out = null;`
35	
36	`        try{`

```
37              inputStream =
38  PornImageDetectionWS.class.getClassLoader()
39                      .getResourceAsStream(rangeFile);
40              br = new BufferedReader(new
41  InputStreamReader(inputStream));
42              int lineIndex = 0;
43              double lower=0, upper=0;
44
45              String line = "";
46              while ((line=br.readLine()) != null) {
47
48                  if(lineIndex >= 14) break;
49
50                  //out.append(line);
51                  errorMsg += "\n"+line+",";
52
53                  String lineArr [] = line.split(" ");
54
55                  if(lineIndex == 0){
56                      lower =
57  Double.parseDouble(lineArr[0]);
58                      upper =
59  Double.parseDouble(lineArr[1]);
60                  } else {
61                      int featureIndex =
62  Integer.parseInt(lineArr[0]);errorMsg +=
63  "@@"+lineArr[0];
64                      rangeValue[featureIndex][0] =
65  Double.parseDouble(lineArr[1]);
66                      rangeValue[featureIndex][1] =
67  Double.parseDouble(lineArr[2]);
68                  }
69                  lineIndex++;
70              }
71              br.close();
72
73              for (int i = 0; i < x.length; i++) {
74                  if (x[i] < rangeValue[i][0])
75  rangeValue[i][0] = x[i];
76                  if (x[i] > rangeValue[i][1])
77  rangeValue[i][1] = x[i];
78                  x[i] = lower + (upper - lower) * (x[i]
79  - rangeValue[i][0])
80                          / (rangeValue[i][1] -
81  rangeValue[i][0]);
82              }
83          } catch (Exception e){
84          }
85          return x;
86      }
```

```
87
88
89  }
```

5.5.4. Implementasi Kode *PornImageDetectionWS.java*

Tabel 5.22 menunjukkan kode pada PornImageDetectionWS.java.

Tabel 5.22 Kode pada PornImageDetectionWS.java

No	Kode
1	`package code.core.webservice;`
2	
3	`import code.core.featureextraction.*;`
4	`import javax.jws.WebMethod;`
5	`import javax.jws.WebParam;`
6	`import javax.jws.WebService;`
7	
8	`@WebService(serviceName =`
9	`"PornImageDetectionWSService")`
10	`public class PornImageDetectionWS {`
11	
12	`    @WebMethod(operationName = "checkImageString")`
13	`    public String checkImageString(@WebParam(name =`
14	`"key") String key, @WebParam(name = "imageURL") String`
15	`imageURL) {`
16	`        String result = "";`
17	`        Extraction extraction = new Extraction();`
18	
19	`        try{`
20	
21	`            boolean isPorn =`
22	`extraction.isImagePorn(imageURL);`
23	`            if(isPorn == true) result = "Porn";`
24	`            else result = "Not Porn";`
25	`        } catch (Exception e){`
26	`            result = "Gambar tidak ditemukan pada url`
27	`atau url tidak valid!";`
28	`        }`
29	
30	`        return result;`
31	`    }`
32	`}`

5.5.5. Implementasi Kode *FaceDetection.java*

Tabel 5.23 menunjukkan kode pada FaceDetection.java.

Tabel 5.23 Kode pada FaceDetection.java

No	Kode
1	`package code.core.facedetection;`
2	
3	
4	`import code.core.imageprocessing.SkinDetection;`
5	`import java.awt.image.BufferedImage;`
6	`import java.io.File;`
7	`import java.io.InputStream;`
8	`import java.util.List;`
9	`import java.util.logging.Level;`
10	`import java.util.logging.Logger;`
11	`import javax.imageio.ImageIO;`
12	`import jjil.algorithm.Gray8Rgb;`
13	`import jjil.algorithm.RgbAvgGray;`
14	`import jjil.core.Error;`
15	`import jjil.core.Image;`
16	`import jjil.core.Rect;`
17	`import jjil.core.RgbImage;`
18	`import jjil.j2se.RgbImageJ2se;`
19	
20	`public class FaceDetection {`
21	
22	`    public double facePercentage=0;`
23	
24	`    public void findFaces(BufferedImage bi, int`
25	`minScale, int maxScale, File output) {`
26	
27	`        try {`
28	
29	`            InputStream is  =`
30	`FaceDetection.class.getResourceAsStream("/jjilexample/h`
31	`aar/HCSB.txt");`
32	`            Gray8DetectHaarMultiScale detectHaar = new`
33	`Gray8DetectHaarMultiScale(is, minScale, maxScale);`
34	`            RgbImage im = RgbImageJ2se.toRgbImage(bi);`
35	`            RgbAvgGray toGray = new RgbAvgGray();`
36	`            toGray.push(im);`
37	`            List<Rect> results =`
38	`detectHaar.pushAndReturn(toGray.getFront());`
39	`            if(results.size() > 0){`
40	`                Image i = detectHaar.getFront();`
41	`                Gray8Rgb g2rgb = new Gray8Rgb();`
42	`                g2rgb.push(i);`

```
43                  RgbImageJ2se conv = new RgbImageJ2se();
44                  conv.toFile((RgbImage)g2rgb.getFront(),
45  output.getCanonicalPath());
46
47                  SkinDetection skin = new
48  SkinDetection();
49                  skin.setImage(bi);
50                  skin.setImageResult(bi.getWidth(),
51  bi.getHeight());
52                  skin.useYCbCr();
53                  double area = 0;
54                  for(int x=0;
55  x<skin.getImageResult().getWidth(); x++){
56                      for(int y=0;
57  y<skin.getImageResult().getHeight(); y++){
58                          int argb =
59  skin.getImageResult().getRGB(x, y);
60
61                          if(argb != -16777216){
62                              area++;
63                          }
64                      }
65                  }
66
67                  File out = new File("tmp_result.jpg");
68                  BufferedImage image =
69  ImageIO.read(out);
70
71                  double faceArea = 0;
72                  for(int x=0; x<image.getWidth(); x++){
73                      for(int y=0; y<image.getHeight();
74  y++){
75                          int argb = image.getRGB(x, y);
76
77                          if(argb != -16777216){
78                              faceArea++;
79                              bi.setRGB(x, y, -16777216);
80                          } else {
81                          }
82                      }
83                  }
84
85                  facePercentage = faceArea / area;
86              }
87
88
89          } catch (Exception e) {
90              System.out.println("exc: "+e);
91          } catch (Error ex) {
92              System.out.println("error: "+ex);
```

```
93              }
94          }
95
96
97  }
```

5.7 Pembuatan Aplikasi *Client* Berbasis PHP

Aplikasi berbasis *client* dibuat untuk menguji keberhasilan *web service* yang telah dibuat. Aplikasi ini dibangun dengan basis PHP karena kebanyakan website menggunakan bahasa pemrograman ini dalam membangun website-nya.

Untuk dapat menggunakan *web service*, aplikasi *client* harus mengetahui link wsdl, *method* dan parameter yang disediakan. Tabel 5.24 menunjukkan informasi terkait link wsdl, *method* dan parameter yang disediakan oleh *web service*.

Tabel 5.24 Informasi Penggunaan *Web Service*

Informasi	Isi	Keterangan
Link wsdl	http://localhost:8080/PornImageDetectionWS/PornImageDetectionWSService?wsdl	Link ini perlu diinputkan dapat menggunakan fungsi SOAP
Method	checkImageString([parameter])	[parameter] diganti dengan variabel bertipe *array* dengan isi sesuai dengan parameter yang didefinisikan
Parameter	imageURL	Adalah parameter yang berfungsi untuk menunjukkan url dari citra yang akan diuji

Hasil dari pemrosesan pada *web service* bertipe data *object* dengan index *return*. Tabel 5.25 menunjukkan kode yang

digunakan pada aplikasi *client* untuk memanggil fungsi yang ada pada *web service* dengan *method* SoapClient.

Tabel 5.25 Kode pada aplikasi *client* untuk memanggil fungsi pada web service dengan SoapClient

No	Kode
1	`<?php`
2	`        $soapclient = new SoapClient(`
3	
4	`'http://localhost:8080/PornImageDetectionWS/PornImageDe`
5	`tectionWSService?wsdl');`
6	
7	`        $image_url = $_POST['url'];`
8	
9	`        $params = array(`
10	`            'imageURL' => $image_url`
11	`        );`
12	
13	`        $result = $soapclient-`
14	`>checkImageString($params);`
15	
16	`?>`

Pada kode di atas, variabel $result adalah hasil pemrosesan *web service* dengan index bernama *return* yang berisi informasi. Hasil Variabel tersebut dapat ditampilkan dengan menggunakan perintah *echo* pada php.

Setelah fungsi untuk memanggil *web service* selesai dibangun maka langkah berikutnya adalah membangun keseluruhan aplikasi *client* beserta antarmukanya. Gambar 5.6 menunjukkan antarmuka yang ada pada aplikasi *client*.

Gambar 5.6 Antarmuka Aplikasi *Client*

Sedangkan tabel 5.26 menunjukkan keseluruhan kode yang digunakan pada aplikasi *client*.

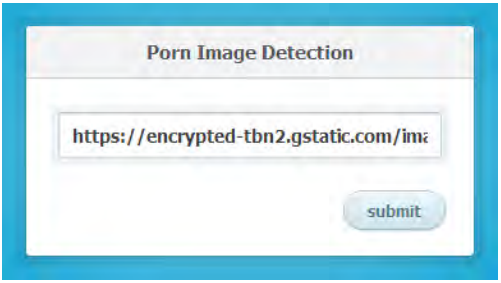
Tabel 5.26 Kode pada Aplikasi *Client*

No	Kode
1	`<!DOCTYPE html>`
2	`<head>`
3	`  <meta charset="utf-8">`
4	`  <meta http-equiv="X-UA-Compatible"`
5	`content="IE=edge,chrome=1">`
6	`  <title>Porn Image Detection</title>`
7	`  <link rel="stylesheet" href="css/style.css">`
8	`  <!--[if lt IE 9]><script`
9	`src="//html5shim.googlecode.com/svn/trunk/html5.js"></s`
10	`cript><![endif]-->`
11	
12	`  <script>`
13	`      function onFormSubmit(e){`
14	`          var url =`
15	`document.getElementById("url").value;`
16	`          if(url == ''){`
17	`              alert("Isikan field url terlebih`
18	`dahulu");`
19	`              e.preventDefault();`
20	`          }`
21	
22	`      }`
23	`  </script>`
24	
25	`</head>`
26	`<body>`
27	`  <section class="container">`
28	`    <div class="login">`
29	`      <h1>Porn Image Detection</h1>`
30	

```
31        <?php if(isset($_POST['submit'])): ?>
32
33          <?php
34          $soapclient = new SoapClient(
35
36  'http://localhost:8080/PornImageDetectionWS/PornImageDe
37  tectionWSService?wsdl');
38
39          $image_url = $_POST['url'];
40
41          $params = array(
42              'imageURL' => $image_url
43          );
44
45          $result = $soapclient-
46  >checkImageString($params);
47
48          ?>
49
50        <center>
51            <?php echo $result->return; ?>
52        </center>
53          <?php else: ?>
54
55        <form onsubmit="onFormSubmit(event)"
56  method="post">
57          <p><input type="text" name="url" id="url"
58  value="" placeholder="Masukkan url gambar"></p>
59          <p class="submit">
60              <input name="submit" value="submit"
61  type="submit">
62          </p>
63        </form>
64
65          <?php endif; ?>
66
67      </div>
68  </body>
69  </html>
```

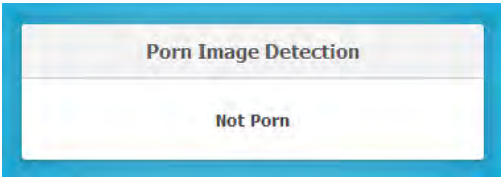
5.8 Cara Penggunaan Aplikasi *Client*

Cara penggunaan aplikasi *client* ini adalah dengan menginputkan url citra yang akan diuji pada field yang tersedia seperti pada gambar 5.7 di bawah ini.



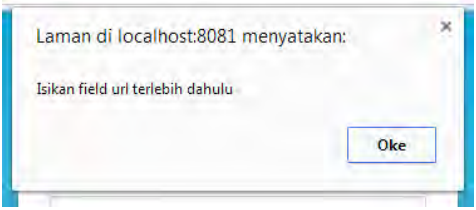
Gambar 5.7 Pengisian Url Citra pada Filed Url Pada Aplikasi *Client*

Kemudian pengguna mengklik tombol Submit sampai aplikasi mengeluarkan hasil prediksinya, yaitu antara "porn" dan "not porn", seperti pada gambar 5.8 di bawah ini.



Gambar 5.8 Hasil Pemrosesan pada Aplikasi *Client*

Jika pengguna belum menginputkan sesuatu namun pengguna telah menekan tombol Submit maka aplikasi akan menampilkan jendela peringatan yang menginformasikan bahwa field url harus diisi terlebih dahulu. Gambar 5.9 menunjukkan antarmuka yang ditampilkan jika field url masih kosong.



Gambar 5.9 Jendela Informasi Jika Pengguna tidak Mengisi Field Url

5.9 Pengujian Aplikasi *Client*

Pengujian dilakukan dengan beberapa tiga kasus, yaitu: 1) citra dengan tubuh manusia namun tidak memiliki konten pornografi; 2) citra tanpa tubuh manusia; dan 3) citra yang memiliki konten pornografi. Berikut ini adalah beberapa citra hasil pengujian dengan hasil prediksi benar dan tidak benar:

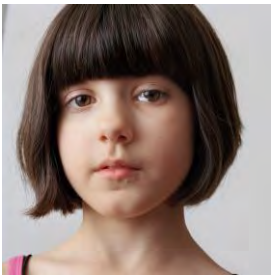
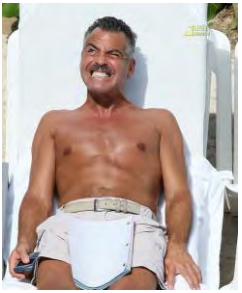
5.8.1. Pengujian Citra dengan Hasil Benar

Berikut ini adalah daftar citra yang diprediksi oleh sistem dengan hasil benar sesuai dengan kasus yang telah didefinisikan di atas:

1) Citra dengan tubuh manusia namun tidak memiliki konten pornografi

Tabel 5.27 Daftar Citra dengan Tubuh Manusia yang Diujikan dengan Hasil Benar

Url	Citra	Prediksi	Kesimpulan
http://www.thehealthyhomeeconomist.com/wp-content/uploads/2013/05/beautiful-skin.jpg		Tidak porno	Benar

			
http://sullydish.files.wordpress.com/2013/04/sage-sohier-about-face-021.jpg		Tidak porno	Benar
http://cdn04.cdn.justjared.com/wp-content/uploads/2008/10/clooney-shirtless/george-clooney-shirtless-02.jpg		Tidak porno	Benar
https://fbcdn-sphotos-c-a.akamaihd.net/hphotos-ak-prn2/t1.0-		Tidak porno	Benar

9/p180x540/1461553_663378730379257_2882658823_n.jpg			

2) Citra tanpa tubuh manusia

Tabel 5.28 Daftar Citra dengan Citra Tanpa Tubuh Manusia dengan Hasil Benar

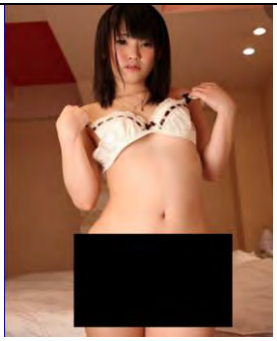
Url	Citra	Prediksi	Kesimpulan
http://upload.wikimedia.org/wikipedia/commons/7/7b/Sand_from_Gobi_Desert.jpg		Tidak porno	Benar
http://upload.wikimedia.org/wikipedia/commons/d/de/Sudan_Envoy_-_Transportation.jpg		Tidak porno	Benar

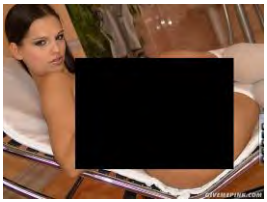
http://thumbs.dreamstime.com/z/chongqing-minsheng-logistics-auto-parts-warehouse-changan-transportation-33254604.jpg		Tidak porno	Benar
http://nsgstone.com/yahoo_site_admin/assets/images/IMGP1154.76123344_std.JPG		Tidak porno	Benar

3) Citra yang memiliki konten pornografi

Tabel 5.29 CItra yang Memiliki Konten Pornografi dengan Hasil Benar

Url	Citra	Prediksi	Kesimpulan
http://4.bp.blogspot.com/-qWJstlzro5Q/U1z0at4oEDI/AAAAAAAAJVc/q2QLtBwr86A/s1600/Sex+porn		Porno	Benar

+Cewe+body+mulus+dan+putih+telanjang+bugil+(1).png			
http://2.bp.blogspot.com/-EpUPotrl7Og/UajPoQoMxNI/AAAAAAAABYs/Im4XWMMQX5Y/s1600/Asian+Porn+6.jpg		Porno	Benar
http://tcdn02.pornex.com/content/97/7f/01/977f01ff7059c988295e52a2c415ff3c143589/big.jpg		Porno	Benar

http://p.im9.eu/givemepink-com-eve-57-hardcore-porn.jpg		Porno	Benar

5.8.2. Pengujian Citra dengan Hasil Salah

Berikut ini adalah daftar citra yang diprediksi oleh sistem dengan hasil tidak benar sesuai dengan kasus yang telah didefinisikan di atas:

1) Citra dengan tubuh manusia namun tidak memiliki konten pornografi

Tabel 5.30 CItra yang Memiliki Konten Pornografi dengan Hasil Salah

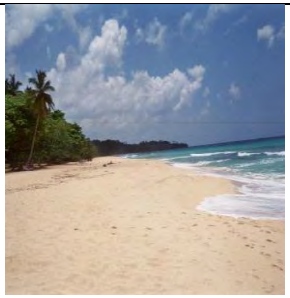
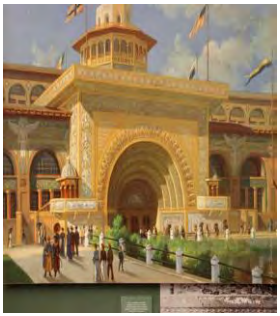
Url	Citra	Prediksi	Kesimpulan
http://www.mentalhealth.org.uk/content/assets/images/Composite/older-people-comp.jpg?view=Standard		Porno	Salah
http://4.bp.blogspot.com/-q5Kamk07pek/T58N9FzBhxI/AAAAAAAAB		Porno	Salah

dw/bQY9wQf_zzA/s1600/El+Valle+Panama+436.JPG			
http://poemsforkush.files.wordpress.com/2012/04/starving-kids-india-child-poverty.jpg		Porno	Salah
http://a.abcnews.com/images/US/ht_alex_reamer_nudist_family_jp_120504_wmain.jpg		Porno	Salah

2) Citra tanpa Tubuh Manusia

Tabel 5.31 Citra Tanpa Tubuh Manusia dengan Hasil Salah

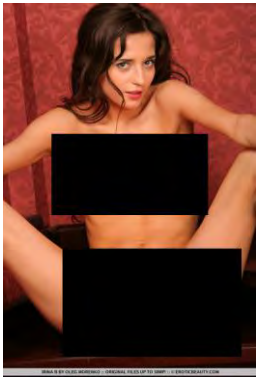
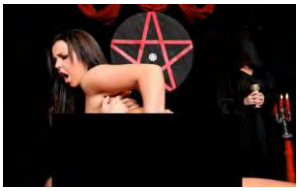
Url	Citra	Prediksi	Kesimpulan
http://2.bp.blogspot.com/_nDNgmK8FIyI/TckG		Porno	Salah

xMxZQuI/AAAAAAAAAAAzg/CizfvUyPZ8o/s1600/most-beach-scenery.jpg			
http://static.freepik.com/free-photo/jersey-beach-scenery---hdr_61-2259.jpg		Porno	Salah
http://3.bp.blogspot.com/_zJpnTu57Koc/TF6y4qngkUI/AAAAAAAAAVyc/ccQyGft7pvM/s1600/LI-archi-sullivan-682b.jpg		Porno	Salah
http://360nourishment.com/wp-content/uploads/2010/12/dry-cracked-		Porno	Salah

skin.jpg			

3) Citra yang Memiliki Konten Pornografi

Tabel 5.32 Citra yang Memiliki Konten Pornografi dengan Hasil Salah

Url	Citra	Prediksi	Kesimpulan
http://www.rabbitsreviews.com/blog/porn/relaxing-with-soft-core-porn.jpg		Tidak Porno	Salah
http://peeperz.com/wp-content/uploads/Porn-Is-Evil-Header.jpg		Tidak porno	Salah

http://is.sankakucomplex.com/data/49/89/4989b33403cf9183f8b45120302e1d02.jpg?228149		Tidak porno	Salah
http://queermenow.net/blog/wp-content/uploads/2009/10/Gridiron-Gang-Bang.jpg		Tidak porno	Salah

5.10.Analisis Keberhasilan dan Kegagalan Prediksi Sistem

Sistem mampu mendeteksi sebagian besar citra ke dalam kategori pornografi dan non-pornografi dengan tepat. Namun sistem juga masih memiliki kekurangan dalam mendeteksi beberapa citra.

Analisis tekstur dan analisis bentuk terbukti sukses untuk membedakan citra yang memiliki konten pornografi dan tidak. Jika hanya analisis warna yang digunakan maka hanya fitur *skin count* dan *skin percentage* yang digunakan. Maka jika nilai kedua jenis fitur ini relatif tinggi maka sudah jelas citra diprediksi memiliki konten pornografi. Namun tabel 5.36 menunjukkan bahwa citra (Citra 1) yang diprediksi secara benar tidak memiliki konten pornogafi justru memiliki nilai fitur *skin count* dan *skin percentage* yang lebih tinggi dibandingkan citra citra yang memiliki konten pornografi (Citra 2).

Tabel 5.33 Perbandingan Nilai Fitur dari Dua Citra Uji (bagian 1)

Citra 1				**Citra 2**			
							
n skin	210	*Hu2*	5.3E-4	*n skin*	150	*Hu2*	3.2E-5
% skin	61.05	*Hu3*	2.1E-5	*% skin*	58.43	*Hu3*	2.2E-7
Cont	2744.5	*Hu4*	5.3E-4	*Cont*	1607.4	*Hu4*	3.2E-5
Corr	2.041	*Hu5*	0.008	*Corr*	2.373	*Hu5*	2.9E-4
Enrg	2.356	*Hu6*	0.005	*Enrg*	2.377	*Hu6*	-8.0E-4
Hmgn	2.429	*Hu7*	-4.1E-5	*Hmgn*	2.482	*Hu7*	1.7E-5
Hu1	1.396			*Hu1*	1.447		

Untuk beberapa kasus ternyata analisis bentuk kurang efektif untuk memprediksi. Hal tersebut dapat dilihat pada tabel 5.37 dimana citra yang memiliki retakan ternyata dianggap memiliki konten pornografi oleh sistem. Citra 1 sukses diprediksi tidak memiliki konten pornografi karena memiliki bentuk yang *plain*. Sedangkan citra 2 memiliki bentuk yang tidak *plain* sehingga sistem memprediksinya memiliki konten pornografi.

Tabel 5.34 Perbandingan Nilai Fitur dari Dua Citra Uji (bagian 2)

Citra 1	**Citra 2**

							
n skin	3	*Hu2*	3.0E-19	*n skin*	4	*Hu2*	1.3E-13
% skin	96.89	*Hu3*	9.2E-16	*% skin*	96.54	*Hu3*	8.1E-14
Cont	1588	*Hu4*	3.0E-19	*Cont*	1158.5	*Hu4*	1.3E-13
Corr	1.708	*Hu5*	1.6E-9	*Corr*	1.923	*Hu5*	-8.1E-8
Enrg	2.013	*Hu6*	2.1E-11	*Enrg*	2.015	*Hu6*	-1.3E-9
Hmgn	2.052	*Hu7*	1.5E-13	*Hmgn*	2.102	*Hu7*	1.3E-12
Hu1	0.878			*Hu1*	0.880		

Kasus kegagalan lainnya adalah sistem masih belum bias untuk memprediksi adanya mode wajah penuh pada citra seperti pada tabel 5.38. Kegagalan ini kemungkinan besar disebabkan oleh fitur *skin count* dan *skin percentage* pada analisis warna. Hal ini bias dilihat dari nilai kedua fitur tersebut yang relatif tinggi.

Tabel 5.35 Nilai Fitur dari Citra Uji yang Gagal Diprediksi (bagian 1)

Citra							
							
n skin	66	*Enrg*	2.270	*Hu3*	5.6E-8	*Hu7*	3.2E-7
% skin	70.13	*Hmgn*	2.343	*Hu4*	1.6E-5		
Cont	1774.7	*Hu1*	1.080	*Hu5*	4.2E-5		
Corr	2.465	*Hu2*	1.6E-5	*Hu6*	-0.002		

Sistem juga belum mampu untuk adanya konten pornorgafi dengan ukuran yang relatif kecil. Contohnya adalah citra 1 yang ditunjukkan pada tabel 5.39. Hal ini disebabkan nilai fitur *skin count* dan *skin percentage* relatif kecil. Selain itu, nilai dari 7 *Hu Moment* pada jenis citra ini relatif tinggi dimana hal ini berbeda dibanding citra lainnya yang memiliki konten pornografi (citra 2).Untuk membuktikan pendapat ini maka sebanyak 30 citra yang memiliki konten pornografi dengan ukuran relatif kecil diuji. Dari pengujian ini didapatkan hasil bahwa sebanyak 100% citra diprediksi salah.

Tabel 5.36 Nilai Fitur dari Citra Uji yang Gagal Diprediksi (bagian 2)

Citra 1				**Citra 2**			
							
n skin	93	*Hu2*	0.157	*n skin*	150	*Hu2*	3.2E-5
% skin	37.52	*Hu3*	5.9E-5	*% skin*	58.43	*Hu3*	2.2E-7
Cont	1910.1	*Hu4*	0.157	*Cont*	1607.4	*Hu4*	3.2E-5
Corr	2.157	*Hu5*	0.063	*Corr*	2.373	*Hu5*	2.9E-4
Enrg	2.594	*Hu6*	-0.030	*Enrg*	2.377	*Hu6*	-8.0E-4
Hmgn	2.617	*Hu7*	-0.002	*Hmgn*	2.482	*Hu7*	1.7E-5
Hu1	2.488			*Hu1*	1.447		

Penyebab kegagalan selanjutnya adalah karena adanya kejanggalan pada citra. Pada beberapa citra terdapat area kulit yang tidak dideteksi sebagai kulit sehingga nilai fitur yang dihasilkan tidak menyerupai *training data* yang normal. Contohnya adalah pada gambar 5.10 di bawah ini:

Gambar 5.10 Contoh Citra yang Gagal Dideteksi Karena Kesalahan Prediksi Pixel Kulit

5.11.Analisis ROC

Kalkulasi total dan analisis ROC dilakukan terhadap cara pemrosesan sistem untuk menilai performanya. Kalkukasi total dilakukan dengan menggunakan 559 citra yang memiliki konten pornografi dan 426 citra yang tidak memiliki konten pornografi. Tabel 5.37 menunjukkan hasil pengujian dari keseluruhan citra tersebut.

Tabel 5.37 Tabel Kontigensi Hasil Pengujian Citra

		Prediksi		**Total**
		Porno	**Tidak porno**	
Aktual	**Porno**	512	35	547
	Tidak porno	171	380	551
Total		683	415	1098

Dari hasil tersebut maka didapatkan beberapa perhitungan terhadap *Accuracy, Sensitivity*, *Specificity*, FPR, FNR, PPV, dan NPV seperti pada tabel 5.38 di bawah ini.

Tabel 5.38 Perhitungan *Accuracy*, *Sensitivity*, *Specificity*, FPR, FNR, PPV, dan NPV

Variabel	Nilai
Accuracy	0.812
Sensitivity	0.749
Specificity	0.916
FPR	0.084
FNR	0.251
PPV	0.936
NPV	0.689

5.12.Penerapan *Face Detection* Untuk Mengoptimasi Performa Sistem

Hasil pengujian menunjukkan bahwa sistem mampu memprediksi kasus dengan akurasi hingga 91.7%. Hal ini menunjukkan bahwa sistem memiliki fungsi yang baik dan telah dapat digunakan di dunia nyata untuk mendeteksi citra yang memiliki konten pornografi dan tidak. Namun sistem juga masih memiliki kekurangan, yang salah satunya adalah kegagalan memprediksi citra yang memiliki mode wajah penuh. Sistem mengeluarkan hasil porno untuk citra dengan jenis tersebut. Untuk itu maka metode *face detection* digunakan untuk mengoptimasi performa sistem.

Jika citra terdeteksi memiliki konten pornografi maka metode *face detection* akan dilakukan. Jika ditemukan bagian wajah pada citra maka dilakukan kalkulasi terhadap nilai luasan dari bagian wajah tersebut. Kemudian nilai luasan tersebut dibandingkan dengan luasan dari bagian kulit sehingga didapatkan nilai persentase bagian wajah. Jika nilainya lebih besar dari 0.5 maka citra tersebut tidak memiliki konten porngorafi melainkan memiliki mode wajah penuh.

Berikut ini adalah tabel 5.39 yang menunjukkan beberapa citra yang awalnya dideteksi memiliki konten pornografi namun setelah *face detection* diterapkan maka hasilnya berubah:

Tabel 5.39 Contoh Citra yang Sukses Diprediksi Melalui Metode *Face Detection*

Url	Citra	Persentase Wajah
http://www.mentalhealth.org.uk/content/assets/images/Composite/older-people-comp.jpg?view=Standard		59.46%
https://asiakoe.files.wordpress.com/2013/01/raffi-ahmad.jpg		65.02%
http://www.mega-wallpaper.com/wallpapers/big/30674_olivia_wilde_face.jpg		66.99%

Dengan adanya *face detection* maka analisis ROC dilakukan kembali. Pada citra yang diujikan, terdapat citra dengan

mode wajah penuh sebanyak 126 buah. Berikut ini adalah tabel 5.40 yaitu tabel kontigensi setelah penerapan *face detection*:

Tabel 5.40 Tabel Kontigensi Hasil Pengujian Citra Setelah Penerapan *Face Detection*

		Prediksi		**Total**
		Porno	**Tidak porno**	
Aktual	**Porno**	512	35	547
	Tidak porno	99	452	551
Total		611	487	1098

Dari hasil tersebut maka didapatkan beberapa perhitungan terhadap *Accuracy, Sensitivity*, *Specificity*, FPR, FNR, PPV, dan NPV seperti pada tabel 5.41 berikut ini:

Tabel 5.41 Perhitungan Accuracy, Sensitivity, Specificity, FPR, FNR, PPV, dan NPV Setelah Penerapan *Face Detection*

Variabel	**Nilai**
Accuracy	0.878
Sensitivity	0.838
Specificity	0.928
FPR	0.072
FNR	0.162
PPV	0.936
NPV	0.820

5.13.Uji Performa *Web Service*

Uji performa dilakukan dengan menggunakan Netbeans Profiler dengan tambahan *tool* bernama JMeter dengan beberapa

skenario. Pengujian ini dilakukan dengan menggunakan *max heap* standar dari Netbeans yaitu 512 MB.

Skenario pertama menggunakan citra dengan resolusi sedang yaitu 320 x 400, dengan *size* 88.5 KB. Berikut ini adalah tabel 5.42 yang menunjukkan performa *web service* dengan skenario pertama:

Tabel 5.42 Hasil Performa *Web Service* dengan Skenario Pertama

No	Resolusi	*Size* (KB)	User	*Sample*	Avg (ms)	Err (%)
1	320 x 400	88.5	1	1	1368	0.00
2	320 x 400	88.5	1	5	1510	0.00
3	320 x 400	88.5	1	10	1717	0.00
4	320 x 400	88.5	1	30	1638	0.00
5	320 x 400	88.5	1	50	1594	0.00
6	320 x 400	88.5	5	1	3896	0.00
7	320 x 400	88.5	5	25	4574	0.00
8	320 x 400	88.5	5	50	4504	0.00
9	320 x 400	88.5	5	150	4555	0.00
10	320 x 400	88.5	5	250	4580	0.00
11	320 x 400	88.5	10	10	7289	0.00
12	320 x 400	88.5	10	50	9025	0.00
13	320 x 400	88.5	10	100	9202	0.00
14	320 x 400	88.5	10	300	9202	0.00
15	320 x 400	88.5	10	500	9403	0.00
16	320 x 400	88.5	30	30	16671	0.00
17	320 x 400	88.5	30	150	26501	0.00
18	320 x 400	88.5	30	300	26881	0.00
19	320 x 400	88.5	30	900	27367	0.00
20	320 x 400	88.5	30	1500	28425	0.00
21	320 x 400	88.5	50	50	25868	0.00
22	320 x 400	88.5	50	250	34008	0.00
23	320 x 400	88.5	50	500	30482	20.00
24	320 x 400	88.5	50	1500	34575	17.82
25	320 x 400	88.5	50	2500	30831	35.41
26	320 x 400	88.5	100	100	41026	19.00
27	320 x 400	88.5	100	500	50113	50.80
28	320 x 400	88.5	200	200	42460	76.26
29	320 x 400	88.5	500	500	10707	100.00

Skenario kedua menggunakan citra dengan resolusi sedang dengan resolusi besar, yaitu 1960 x 1307 dan *size* 276 KB.

Berikut ini adalah tabel 5.43 yang menunjukkan performa *web service* dengan skenario kedua:

Tabel 5.43 Hasil Performa *Web Service* dengan Skenario Kedua

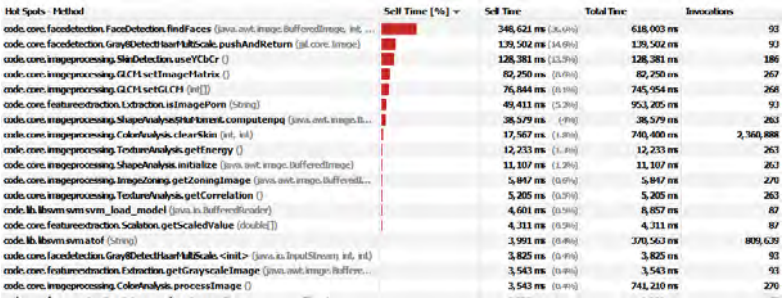
No	Resolusi	*Size* (KB)	User	*Sample*	Avg (ms)	Err (%)
1	1960 x 1307	276	1	1	3.573	0.00
2	1960 x 1307	276	1	5	3507	0.00
3	1960 x 1307	276	1	10	3535	0.00
4	1960 x 1307	276	1	30	3533	0.00
5	1960 x 1307	276	1	50	3841	0.00
6	1960 x 1307	276	5	1	10073	0.00
7	1960 x 1307	276	5	25	9863	0.00
8	1960 x 1307	276	5	50	9080	0.00
9	1960 x 1307	276	5	150	9391	0.00
10	1960 x 1307	276	5	250	9414	0.00
11	1960 x 1307	276	10	10	14255	0.00
12	1960 x 1307	276	10	50	17774	0.00
13	1960 x 1307	276	10	100	18482	0.00
14	1960 x 1307	276	10	300	19382	0.00
15	1960 x 1307	276	10	500	19821	0.00
16	1960 x 1307	276	30	30	30479	0.00
17	1960 x 1307	276	30	150	36849	30.67
18	1960 x 1307	276	30	300	40321	22.67
19	1960 x 1307	276	30	900	36838	13.53
20	1960 x 1307	276	30	1500	37669	10.00
21	1960 x 1307	276	50	50	61144	62.00
22	1960 x 1307	276	50	250	32371	41.60
23	1960 x 1307	276	50	500	37473	38.40
24	1960 x 1307	276	50	1500	32682	39.09
25	1960 x 1307	276	50	2500	37162	28.88
26	1960 x 1307	276	100	100	58215	53.00
27	1960 x 1307	276	100	500	35897	66.20
28	1960 x 1307	276	200	200	21031	74.50
29	1960 x 1307	276	400	400	11645	100.00

Pada skenario pertama diketahui bahwa ketika terdapat 50 *user* dengan *sample* sebanyak 500 buah maka *web service* sudah tidak mampu memenuhi 100% permintaan yang dibuktikan dengan jumlah persentase *error* sebanyak 20%. Selain itu, *web service* hanya mampu memenuhi permintaan dengan jumlah maksimal 500 *user*. Selain itu dilakukan pengujian terhadap kecepatan sistem per *request*. Dari 25 kali pengujian, didapatkan

nilai rata-rata kecepatan sistem adalah 1095 ms, nilai median 1185 ms, nilai minimum 832 ms, dan nilai maksimum 1300 ms.

Pada skenario kedua diketahui bahwa ketika terdapat 30 *user* dengan *sample* sebanyak 150 buah maka *web service* sudah tidak mampu memenuhi 100% permintaan yang dibuktikan dengan jumlah persentase *error* sebanyak 30.67%. Selain itu, *web service* hanya mampu memenuhi permintaan dengan jumlah maksimal 400 *user*. Selain itu dilakukan pengujian terhadap kecepatan sistem per *request*. Dari 25 kali pengujian, didapatkan nilai rata-rata kecepatan sistem adalah 3958 ms, nilai median 3745 ms, nilai minimum 3313 ms, dan nilai maksimum 7371 ms.

Dari pengujian yang dilakukan didapatkan alokasi pemrosesan terbesar adalah dari kode untuk melakukan *face detection*, yaitu *method findFaces* pada *class FaceDetection* dan *method pushAndReturn* dari *class Gray8DetectHaarMultiScale*. Hal ini menunjukkan bahwa *library* yang digunakan justru mengurangi efisiensi waktu. Berikut ini adalah gambar 5.11 yang menunjukkan alokasi waktu respon dari setiap *class* yang ada pada sistem:



Gambar 5.11 Alokasi Waktu Respon per *Class*

BAB VI
PENUTUP

Pada bab ini berisi kesimpulan dari seluruh proses pengerjaan tugas akhir berserta saran untuk proses pengembangan selanjutnya.

6.1 Kesimpulan

Berdasarkan tugas akhir yang telah dilakukan, maka dapat disimpulkan beberapa hal sebagai berikut:

1. Metode *image zoning* dengan fitur *skin count*, *skin percentage*, *contrast*, *correlation*, *energy*, *homogeneity*, dan 7 *Hu Moment* dengan algoritma pembelajaran (*classifier*) *Support Vector Machine* (SVM) dapat digunakan untuk mendeteksi adanya konten pornografi pada citra dengan baik. Terbukti dengan tingkat akurasi sebesar 0.812, *sensitivity* sebesar 0.749, *specificity* sebesar 0.916, FPR sebesar 0.084, FNR sebesar 0.251, PPV sebesar 0.936, dan NPV sebesar 0.689.
2. Sistem masih mengalami kegagalan prediksi dengan kondisi: 1) jika citra memiliki konten pornografi dengan melibatkan banyak orang; 2) terdapat citra yang janggal sehingga pixel kulit tidak dapat dideteksi.
3. Penerapan *face detection* mampu meningkatkan performa *web service* dalam mendeteksi konten pornografi pada citra dengan hasil akhir *accuracy* sebesar 0.876, *sensitivity* sebesar 0.838, *specificity* sebesar 0.928, FPR sebesar 0.072, FNR sebesar 0.162, PPV sebesar 0.936, dan NPV sebesar 0.820.
4. Performa *web service* yang diuji dengan JMeter akan bagus diimplementasikan pada *max heap* 512 MB jika jumlah *user* kurang dari 50 dan dengan jumlah *sample* sebanyak 500 buah untuk resolusi sedang. Sedangkan untuk resolusi tinggi, performa *web service* akan bagus diimplementasikan dengan jumlah *user* kurang dari 30 buah dan *sample* sebanyak150 buah.

5. Performa *web service* yang diuji dengan JMeter memiliki kecepatan waktu respon rata-rata sebesar 1095 ms untuk resolusi rendah dan 3985 ms untuk resolusi sedang.

6.2 Saran

Beberapa hal yang diharapkan dapat dikembangkan pada masa mendatang adalah sebagai berikut:

1. Manusia terdiri dari berbagai macam anggota tubuh, seperti kepala, tangan, badan, kaki, dan lain sebagainya. Biasanya pendekatan pornografi mengarah kepada bagian vital manusia. Untuk itu, sebaiknya metode pendeteksian konten pornografi pada citra seharusnya adalah metode yang mampu untuk mendeteksi adanya bagian vital tersebut pada citra.
2. Pembuatan sistem mandiri yang mampu untuk melakukan pembaruan secara berkala terhadap *database* citra.
3. Pemilihan *library* untuk menangani komputasi SVM dan *face detection* yang lebih efektif dan efisien.
4. Adanya studi khusus yang mampu untuk mengetahui algoritma pembelajaran yang paling efektif dan efisien untuk diterapkan pada sistem pendeteksi konten pornografi pada citra.

DAFTAR PUSTAKA

ALEX CO. (2014, Januari 27). *35 Percent of All The Pirate Bay's Uploads are Porn*. Dipetik Februari 6, 2014, dari www.escapistmagazine.com: http://www.escapistmagazine.com/news/view/131671-35-Percent-of-All-The-Pirate-Bays-Uploads-are-Porn

Alexa. (2014). *The top 500 sites on the web*. Dipetik Februari 6, 2014, dari www.alexa.com: http://www.alexa.com/topsites

Basilio, J. A., Torres, G. A., Sánchez, G., Medina, L. K., Meana, H. M., & Enriquez. (2010). EXPLICIT CONTENT IMAGE DETECTION. *An International Journal(SIPIJ), 2*, 47-55.

Canny, J. (1986). A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions*, 679–698.

Chai, D., & Ngan, K. (1999). Face segmentation using skin-color map in videophone applications. *IEEE Trans. on Circuits and Systems for Video Technology*, 551-564.

Chris Solomon, T. B. (2010). *Fundamentals of Digital Image Processing.* West Sussex: Willy-Blackwell.

Clayton, S., Eulanda, S. M., & Eduardo, S. (2012). *NUDITY DETECTION BASED ON IMAGE ZONING.* Manaus: Institute of Computing (IComp), Federal University of Amazonas (UFAM),.

Effendy, N., Imanto, R., & P.T., A. (2010). *Deteksi Pornografi Pada Citra Digital Menggunakan Pengolahan Citra dan Jaringan Saraf Tiruan.* Yogyakarta: Jurusan Teknik Fisika, Fakultas Teknik Universitas Gadjah Mada.

Fleck, M., Forsyth, D. A., & Bregler, C. (1996). Finding Naked People. *Proc. 4th European Conf. On Computer Vision Vol. 2*, 593-602.

Gentle, B. (2012, Juni 19). *Top 5 Myths About Big Data*. Dipetik Februari 6, 2014, dari http://mashable.com/: http://mashable.com/2012/06/19/big-data-myths/

Hidayattullah, M. F., & Hapsari, Y. (2013). Automatic Nipple Detection Pada Citra Pornografi Menggunakan Algoritma Viola And Jones Berbasis Adaboost Untuk Feature Selection. *SEMINAR NASIONAL TEKNOLOGI INFORMASI & KOMUNIKASI TERAPAN*, (hal. 238-245). Semarang.

Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2010). *A Practical Guide to Support Vector Classi.* Taiwan: Department of Computer Science, National Taiwan University.

Hu, M.-K. (1986). Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions, 8*, 179-187.

Indo-Asian News Service. (2012, April 9). *30 percent of global web traffic is porn - study*. Dipetik Februari 6, 2014, dari gadgets.ndtv.com: http://gadgets.ndtv.com/internet/news/30-percent-of-global-web-traffic-is-porn-study-223878

Isa, S. M., & Mariana, F. (2009). DETEKSI CITRA PORNOGRAFI MENGGUNAKAN TSL COLOR SPACE DAN NUDITY DETECTION ALGORITHM . *Seminar Nasional Informatika*, (hal. A86-A93). Yogyakarta.

K.M., L., S.D., S., & M., W. (2007). *DETECTING PORNOGRAPHIC IMAGES.* Kuala Lumpur: Asia Pacific Institute Of Information Technology.

Kalva, P. R., Enembreck, F., & Koerich, A. L. (2011). Web Image Classification using Combination of Classifiers. *IEEE Latin America Transactions*, 661-671.

Kamus Besar Bahasa Indonesia. (2014, Februari 6). *Kamus Bahasa Indonesia Online* . Dipetik Februari 6, 2014, dari http://kamusbahasaindonesia.org/: http://kamusbahasaindonesia.org/pornografi

Kompas. (2013, Oktober 31). *Facebook Tembus 1,19 Miliar Pengguna Aktif*. Dipetik Februari 6, 2014, dari http://tekno.kompas.com/: http://tekno.kompas.com/read/2013/10/31/1426203/Facebook.Tembus.1.19.Miliar.Pengguna.Aktif

Kreger, H. (2001). *Web Services Conceptual Architecture (WSCA 1.0).* New York: IBM Software Group.

Lee, J. S., Kuo, Y. M., Chung, P. C., & Chen, E. L. (2006). Naked Image Detection Based on Adaptive and Extensible. *Pattern Recognition, 40*, 2261-2270.

Marcial-Basilio, J. A., Aguilar-Torres, G., Sánchez-Pérez, G., Toscano-Medina, L. K., & Pérez-Meana, H. M. (2011). Detection of Pornographic Digital Images. *INTERNATIONAL JOURNAL OF COMPUTERS, 5*, 209-305.

Marcial-Basilio, J. A., Aguilar-Torres, G., Sánchez-Pérez, G., Toscano-Medina, L. K., & Pérez-Meana, H. M. (2011). Detection of Pornographic Digital Images. *INTERNATIONAL JOURNAL OF COMPUTERS, 5*, 298-305.

Mercier, G., & Lennon, M. (2003). *Support vector machines for hyperspectral image classification with spectral-based kernels.* Toulouse: Proc. IGARSS.

Pathak, B., & Barooah, D. (2013). TEXTURE ANALYSIS BASED ON THE GRAY-LEVEL CO-OCCURRENCE MATRIX CONSIDERING POSSIBLE ORIENTATIONS. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2*, 4206-4212.

Powar, V., Jahagirdar, A., & Sirsikar, S. (2011). Skin Detection in YCbCr Color Space. *International Journal of Computer Applications*, 1-4.

Sholahuddin, A. (2012). METODE MOMENT INVARIANT DAN BACKPRORAGATION NEURAL NETWORK

PADA PENGENALAN WAJAH. *Lokakarya Komputasi dalam Sains dan Teknologi Nuklir*, 283-295.

Singh, S. K., Chauhan, D. S., Vatsa, M., & Singh, R. (2003). A Robust Skin Color Based Face Detection Algorithm. *Tamkang Journal of Science and Engineering, 6*, 227-234.

Tekno Kompas. (2013, Juli 16). *Hobi Menonton Film Porno, Empat Siswa Perkosa Temannya*. Dipetik Februari 6, 2014, dari tekno.kompas.com: http://tekno.kompas.com/read/2013/07/16/0258269/hobi.menonton.film.porno.empat.siswa.perkosa.temannya

Zheng, Q.-F., Zeng, W., Wen, G., & Wang, W.-Q. (2006). *Shape-based Adult Images Detection.* Beijing: Institute of Computing Technology.

BIODATA PENULIS

Penulis yang lahir di Surabaya pada tanggal 23 Juli 1991 ini merupakan anak pertama dari dua bersaudara. Penulis telah menempuh pendidikan formal di SDN Percobaan Surabaya, SMPN Negeri 1 Surabaya, SMAN 5 Surabaya, dan akhirnya penulis masuk menjadi mahasiswa Sistem Informasi angkatan 2010 melalui jalur SNMPTN. Penulis sempat bergelut di dunia Palang Merah Remaja (PMR) pada saat menempuh pendidikan SMP dan paduan suara pada saat menempuh pendidikan SMA. Selama menempuh perkuliahan, penulis aktif dalam dunia keilmiahan. Hal ini dibuktikan dengan menjadi staff dan Wakil Kepala Departemen Riset dan Teknologi Himpunan Mahasiswa Sistem Informasi. Selain itu, penulis juga sering mengikuti kompetisi kemahasiswaan pada tingkat institut maupun nasional. Penulis yang memiliki ketertarikan di bidang seni musik dan seni rupa ini bercita-cita untuk menjadi abdi Negara. Hal ini dikarenakan penulis ingin mengembangkan potensi sumber daya manusia Indonesia melalui Teknologi Informasi.

Halaman ini sengaja dikosongkan

LAMPIRAN
ISI FILE *RANGE*

Tabel A.1 Isi File *range*

X
-1.000000000000000 1.000000000000000
0 0.000000000000000 1753.000000000000
1 0.000000000000000 0.9689178500000000
2 0.000000000000000 5207.921181075113
3 0.04000507955614328 2.901819013142799
4 2.014180751134089 3.000000000000000
5 2.213494707622103 3.000000000000000
6 0.7949480557006381 2201.095509005704
7 5.833021259499685e-18 471448550527.0933
8 3.754280902043881e-16 2.379263511683016e+17
9 5.833021259499685e-18 471448550527.0933
10 -4.970763826055067e+20 8.545275075911326e+23
11 -529837801.2513360 16657444403504.67
12 -2141509800276473 5.354543663121930e+18