TUGAS AKHIR – KI141502

IMPLEMENTASI METODE HYBRID SALIENCY EXTREME LEARNING MACHINE UNTUK MELAKUKAN SALIENCY DETECTION DALAM SEGMENTASI CITRA

GIAN SEBASTIAN ANJASMARA
5113100132

Dosen Pembimbing
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.
Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

IMPLEMENTASI METODE HYBRID SALIENCY EXTREME LEARNING MACHINE UNTUK MELAKUKAN SALIENCY DETECTION DALAM SEGMENTASI CITRA

GIAN SEBASTIAN ANJASMARA
5113100132

Dosen Pembimbing I
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Dosen Pembimbing II
Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2017

[*Halaman ini sengaja dikosongkan*]

FINAL PROJECT - KI141502

IMPLEMENTATION OF SALIENCY EXTREME LEARNING MACHINE HYBRID METHOD TO PERFORM SALIENCY DETECTION IN IMAGE SEGMENTATION

GIAN SEBASTIAN ANJASMARA
5113100132

Supervisor I
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Supervisor II
Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
Sepuluh Nopember Institute of Technology
Surabaya, 2017

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

IMPLEMENTASI METODE HYBRID SALIENCY EXTREME LEARNING MACHINE UNTUK MELAKUKAN SALIENCY DETECTION DALAM SEGMENTASI CITRA

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Komputasi Cerdas dan Visi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
GIAN SEBASTIAN ANJASMARA
NRP: 5113 100 132

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.
NIP. 197512202001122002 ..... (Pembimbing 1)

Dr.Eng. Nanik Suciati, S.Kom., M.Kom.
NIP. 197104281994122001 ..... (Pembimbing 2)

SURABAYA
MEI, 2017

[*Halaman ini sengaja dikosongkan*]

IMPLEMENTASI METODE HYBRID SALIENCY EXTREME LEARNING MACHINE UNTUK MELAKUKAN SALIENCY DETECTION DALAM SEGMENTASI CITRA

Nama Mahasiswa	: Gian Sebastian Anjasmara
NRP	: 5113 100 132
Jurusan	: Teknik Informatika, FTIf ITS
Dosen Pembimbing 1	: Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.
Dosen Pembimbing 2	: Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

Abstrak

Saliency dari suatu objek, baik itu benda, manusia atau piksel adalah keadaan dari objek tersebut yang terlihat kontras dibandingkan dengan sekitarnya atau tetangganya. Dibutuhkan metode saliency detection yang tepat dalam segmentasi citra untuk mengidentifikasi dan memisahkan daerah yang paling menonjol atau area salient object dari suatu citra.

Tugas akhir ini mengusulkan metode Saliency Extreme Learning Machine yang memadukan antara model Spectral Residual sebagai metode bottom-up dan Extreme Learning Machine (ELM) classifier sebagai metode top-down untuk melakukan segmentasi citra area salient object. Untuk menghindari pelabelan training samples secara manual, digunakan metode thresholding untuk menentukan training samples positif dan negatif dari prior saliency map yang dihasilkan oleh model Spectral Residual. Setelah training samples terbentuk dilakukan ekstraksi fitur dan saliency detection dengan ELM classifier. ELM classifier menghasilkan trained saliency map dalam empat skala superpixels yang digabungkan menjadi satu sebagai acuan dalam pembentukan object map.

Uji coba yang dilakukan terhadap 50 natural images menunjukkan bahwa metode ini dapat memberikan hasil segmentasi area salient object yang akurat dengan rata-rata

presisi, recall dan F1 score masing-masing sebesar 90,26%, 91,52%, dan 90,39%.

Kata kunci: Extreme learning machine, pelabelan training samples, salient object, spectral residual.

IMPLEMENTATION OF SALIENCY EXTREME LEARNING MACHINE HYBRID METHOD TO PERFORM SALIENCY DETECTION IN IMAGE SEGMENTATION

Student Name	: Gian Sebastian Anjasmara
Registration Number	: 5113 100 132
Department	: Informatics Engineering, FTIf ITS
First Supervisor	: Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.
Second Supervisor	: Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

Abstract

Saliency of an object, be it an item, a person or a pixel is the state or quality by which it stands out relative to its neighbors. It takes the right saliency detection method in image segmentation to identify and separate the most prominent area or salient object area of an image.

In this research we propose Saliency Extreme Learning Machine method which combines Spectral Residual model as bottom-up method and Extreme Learning Machine (ELM) classifier as top-down method to segment salient object area of an image. To avoid manual labeling of training samples, thresholding method is used to determine positive and negative samples from prior saliency map produced by Spectral Residual model. After training samples are formed, feature extraction and saliency detection with ELM classifier are performed. The ELM classifier generates trained saliency map in four superpixels scales that will be combined into one as a reference in forming object map.

Testing conducted on 50 natural images show that this method can provide accurate segmentation of salient object area with an average of precision, recall and F1 score of 90.26%, 91.52% and 90.39% respectively.

Keywords: Extreme learning machine, labeling of training samples, salient object, spectral residual.

KATA PENGANTAR

Segala puji bagi Tuhan yang Maha Esa yang telah melimpahkan rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul **"Implementasi Metode Hybrid Saliency Extreme Learning Machine untuk Melakukan Saliency Detection dalam Segmentasi Citra"**.

Dalam perancangan, pengerjaan, dan penyusunan tugas akhir ini, penulis banyak mendapatkan bantuan dari berbagai pihak. Penulis ingin mengucapkan terima kasih kepada:

1. Orang tua penulis Bapak Renold Anjasmara dan Ibu Lies Tjandra yang telah memberikan dukungan moral, spiritual dan material serta senantiasa memberikan doa demi kelancaran dan kemudahan penulis dalam mengerjakan tugas akhir.
2. Ibu Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. dan Ibu Dr.Eng. Nanik Suciati, S.Kom., M.Kom. selaku dosen pembimbing penulis yang telah memberi ide, nasihat dan arahan sehingga penulis dapat menyelesaikan tugas akhir dengan tepat waktu.
3. Seluruh saudara kandung: dua kakak (kak Maya dan kak Vania) dan keluarga (kak Yuli dan kak Nita) serta seluruh keluarga besar yang telah memberikan dukungan yang besar baik secara langsung maupun secara implisit.
4. Teman-teman di Lab KCV dan Lab MI: para admin dan teman-teman yang telah banyak membantu memfasilitasi dan menemani penulis dalam pengerjaan tugas akhir.
5. Teman-teman seperti Hariyanto, Lino, Rifqi, Nindy, Zaza, Dhita, Nyoman, Naufal dan anggota Burgator yang sama-sama mengarungi empat tahun masa perkuliahan bersama penulis, yang saat berinteraksi dan diam-diam mengamati tingkah laku mereka, telah memberi penulis banyak

pelajaran hidup secara tersirat maupun tersurat dengan gratis.

6. Pihak-pihak lain yang tidak bisa penulis sebutkan satu-persatu.

Penulis menyadari masih ada kekurangan dalam penyusunan tugas akhir ini. Penulis mohon maaf atas kesalahan, kelalaian maupun kekurangan dalam penyusunan tugas akhir ini. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan ke depan.

Surabaya, Mei 2017

Penulis

DAFTAR ISI

[*Halaman ini sengaja dikosongkan*]

DAFTAR GAMBAR

DAFTAR TABEL

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE SUMBER

[*Halaman ini sengaja dikosongkan*]

BAB I
PENDAHULUAN

Pada bab ini dibahas hal-hal yang mendasari tugas akhir. Bahasan meliputi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan tugas akhir.

1.1 Latar Belakang

Segmentasi citra [1] adalah salah satu tahapan penting dalam visi komputer dan pengolahan citra, yang nantinya akan digunakan untuk temu kembali citra, pengenalan objek dan klasifikasi data. Segmentasi citra dapat dilihat sebagai masalah klasifikasi, yaitu dengan menandai masing-masing piksel menurut ciri-ciri penting tertentu. Beberapa metode-metode klasifikasi sudah berhasil melakukan segmentasi citra, di antaranya metode berbasis *Support Vector Machine* (SVM) dan *Extreme Learning Machine* (ELM).

ELM, metode yang pertama kali diperkenalkan oleh Huang et al. [2,3], merupakan algoritma *machine learning* sederhana dengan karakteristik *learning speed* yang cepat dan kemampuan generalisasi yang baik. ELM secara sederhana adalah *Feedforward Neural Network* dengan *single layer hidden nodes*, dengan bobot antara *input* dan *hidden nodes* ditetapkan menggunakan *random feature mapping* atau fungsi *kernel* dan tidak pernah diperbaharui bobotnya. Jika dibandingkan dengan SVM, ELM membutuhkan optimasi *constraints* yang lebih sedikit sehingga menghasilkan implementasi yang lebih sederhana, *learning speed* yang lebih cepat dan kemampuan generalisasi yang lebih baik [4].

Saliency detection yang bertujuan untuk mengidentifikasi daerah yang paling menonjol dari suatu objek telah menarik banyak perhatian dalam berbagai disiplin ilmu. Sebagai salah satu masalah yang paling mendasar dalam visi komputer, *saliency detection* telah diterapkan dalam banyak pekerjaan yang

berhubungan dengan visual, di antaranya ada segmentasi citra, kompresi citra, deteksi dan pengenalan objek [5]. *Saliency detection* diterapkan dalam segmentasi citra untuk mengidentifikasi daerah yang paling menonjol atau area *salient object* sebagai *foreground* dari suatu citra. *Saliency map* merupakan hasil keluaran dari *saliency detection* yang berfungsi sebagai peta yang merepresentasikan nilai *saliency* dari suatu citra.

Pada umumnya, metode *saliency detection* dapat dikelompokkan menjadi dua kategori, yaitu metode *top-down* dan metode *bottom-up*. Metode *bottom-up* merupakan metode yang tergolong cepat, *data-driven* dan *stimulus-driven*, yang membangun *saliency map* berdasarkan informasi *low-level* seperti warna, fitur dan jarak spatial. Sedangkan metode *top-down* yang memanfaatkan *classifier* merupakan metode yang lebih lambat, *task-driven* dan membutuhkan *supervised learning* berdasarkan *training samples*. Metode *bottom-up* lebih efektif dalam mendeteksi bentuk detil dari *salient object*, namun seringkali hasil identifikasi mengandung banyak *noise*. Berbeda dengan metode *top-down* yang lebih efektif dalam mendeteksi bentuk *salient object* secara global dengan berbagai ukuran dan kategori, namun sebagian besar dari metode *top-down* yang ada banyak memakan waktu pada proses *off-line training* atau proses pemberian label pada *training samples* secara manual [6].

Maka dari itu dalam tugas akhir ini akan mengimplementasikan pendekatan baru yang memadukan antara model *Spectral Residual* sebagai metode *bottom-up* dan ELM *classifier* sebagai metode *top-down* untuk melakukan segmentasi citra area *salient object*. Untuk menghindari proses *labeling training samples* secara manual, digunakan metode *thresholding* untuk menentukan *training samples* positif dan negatif dari *prior saliency map* yang dihasilkan oleh model *Spectral Residual*. Setelah *training samples* terbentuk, dilakukan ekstraksi fitur (RGB, CIELab dan *Uniform Local Binary Pattern*) dan *saliency detection* dengan ELM *classifier*. *Training samples dan testing*

samples berbasis *multi-scale superpixels* untuk mendeteksi *salient object* pada berbagai skala yang dapat meningkatkan akurasi dari *saliency map*. ELM *classifier* menghasilkan *trained saliency map* dalam empat skala yang akan digabungkan menjadi satu sebagai acuan dalam pembentukan *object map*.

Hasil yang diharapkan pada implementasi metode *Saliency Extreme Learning Machine* adalah dapat melakukan segmentasi citra area *salient object* dengan cepat dan benar, serta menghindari pelabelan *training samples* secara manual.

1.2 Rumusan Masalah

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

1. Bagaimana menentukan label *training samples* secara otomatis dengan menggunakan metode *bottom-up* dan metode *thresholding*?
2. Bagaimana mengimplementasikan model *Spectral Residual* sebagai metode *bottom-up* dan ELM *classifier* sebagai metode *top-down* untuk melakukan segmentasi citra area *salient object*?
3. Bagaimana mengevaluasi kinerja metode *Saliency Extreme Learning Machine*?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan antara lain:

1. Aplikasi yang dibuat adalah aplikasi berbasis *desktop*.
2. Metode yang digabungkan menggunakan model *Spectral Residual* sebagai metode *bottom-up* dan ELM *classifier* sebagai metode *top-down*.
3. Dataset yang digunakan diambil dari *MSRA Salient Object Database* [7] dalam berbagai ukuran.

1.4 Tujuan Tugas Akhir

Tujuan tugas akhir ini adalah melakukan implementasi model *Spectral Residual* sebagai metode *bottom-up* dan ELM *classifier* sebagai metode *top-down* untuk melakukan segmentasi citra area *salient object*.

1.5 Manfaat Tugas Akhir

Manfaat dari tugas akhir ini adalah mengeksplorasi kinerja dari ELM *classifier* ketika digabungkan dengan model *Spectral Residual* sebagai metode *bottom-up* dalam melakukan segmentasi citra area *salient object*.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Studi Literatur

   Pada studi literatur, dilakukan pengumpulan data dan studi terhadap sejumlah referensi yang diperlukan dalam pengerjaan tugas akhir. Referensi tersebut didapatkan dari beberapa artikel yang dipublikasikan oleh jurnal. Selain dari artikel, studi literatur juga dilakukan melalui pencarian referensi dari internet yang membahas mengenai informasi yang dibutuhkan. Di antaranya informasi mengenai model *Spectral Residual, Uniform Local Binary Pattern,* metode *Graph Cut,* dan *Extreme Learning Machine classifier*.

2. Analisis dan Desain Perangkat Lunak

   Pada tahap ini disusun rancang bangun dari perangkat lunak yang dibangun. Pengguna dapat memilih citra yang telah disediakan sebagai data masukan. Kemudian, sistem akan memproses citra dengan melakukan *preprocessing*, *saliency detection* dengan model *Spectral Residual*, ekstraksi fitur, *saliency detection* dengan ELM *classifier*, dan segmentasi citra. Setelah proses selesai, sistem akan menampilkan *object*

map yang menandakan area *salient object* pada citra masukan.

3. Implementasi Perangkat Lunak

   Perangkat lunak akan dibangun menggunakan bahasa pemrograman MATLAB menggunakan kakas bantu IDE MATLAB 9.1 (R2016b) pada platform *desktop*. Selain itu, digunakan alat bernama GUIDE untuk membuat GUI (*Graphical User Interface*) aplikasi pada MATLAB. *Toolbox* yang digunakan untuk mendukung pengerjaan adalah *image processing toolbox, neural network toolbox, computer vision system toolbox* dan *statistics and machine learning toolbox.* Kakas bantu pendukung lain di antaranya *draw.io* untuk dokumentasi dan *Microsoft Excel* sebagai pengolah angka.

4. Uji Coba dan Evaluasi

   Dalam tahap ini, dilakukan pengujian parameter-parameter yang dibutuhkan pada proses *saliency detection* dengan model *Spectral Residual*, *saliency detection* dengan ELM *classifier*, dan segmentasi citra.. Citra keluaran yang telah tersegmentasi dibandingkan dengan citra *ground truth* untuk dihitung performanya dengan menghitung nilai presisi, *recall* dan *F1 score.*

1.7 Sistematika Laporan

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

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Dasar Teori

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan

Bab ini membahas tahap analisis permasalahan dan perancangan dari sistem yang akan dibangun. Analisis permasalahan membahas permasalahan yang diangkat dalam pengerjaan tugas akhir.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode yang digunakan untuk proses implementasi.

Bab V Uji Coba dan Evaluasi

Bab ini membahas tahap-tahap uji coba. Kemudian hasil uji coba dievaluasi untuk kinerja dari aplikasi yang dibangun.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan aplikasi ke depannya.

BAB II
DASAR TEORI

Pada bab ini diuraikan mengenai dasar-dasar teori yang digunakan dalam pengerjaan tugas akhir dengan tujuan untuk memberikan gambaran secara umum terhadap penelitian yang dikerjakan. Bab ini berisi penjelasan mengenai segmentasi citra, *saliency detection*, model *Spectral Residual*, jenis-jenis filter citra, *multi-scale superpixels*, pelabelan *training samples* secara otomatis, RGB, CIELAB dan *Uniform Local Binary Pattern* untuk ekstraksi fitur, *Extreme Learning Machine classifier,* metode *Graph Cut* untuk *smoothing saliency map* serta pembentukan *object map* dengan metode Otsu.

2.1 Segmentasi Citra

Segmentasi citra [1] adalah proses yang membagi citra digital menjadi beberapa segmen dengan memberikan label pada setiap piksel. Setiap piksel dalam suatu segmen atau daerah yang sama memiliki hubungan dengan beberapa properti karakteristik, seperti warna, intensitas dan tekstur. Tujuan dari segmentasi citra adalah untuk menyederhanakan atau mengubah representasi citra menjadi sesuatu yang lebih bermakna dan mudah dianalisis. Segmentasi citra biasanya digunakan untuk menemukan objek (*foreground*) dan batas-batas dalam citra.

2.2 *Saliency Detection*

Saliency dari suatu objek, baik itu benda, manusia atau piksel adalah keadaan dari objek tersebut yang terlihat kontras dibandingkan dengan sekitarnya atau tetangganya. *Saliency detection* diterapkan dalam segmentasi citra untuk mengidentifikasi daerah yang paling menonjol atau area *salient object* sebagai *foreground* dari suatu citra. *Saliency map* merupakan hasil keluaran dari *saliency detection* yang berfungsi sebagai peta yang merepresentasikan nilai *saliency* dari suatu citra [8].

Pada umumnya, metode *saliency detection* dapat dikelompokkan menjadi dua kategori, yaitu metode *top-down* dan metode *bottom-up*. Metode *bottom-up* merupakan metode yang tergolong cepat, *data-driven* dan *stimulus-driven*, yang membangun *saliency map* berdasarkan informasi *low-level* seperti warna, fitur dan jarak spasial. Metode *bottom-up* lebih efektif dalam mendeteksi bentuk detil dari *salient object*, namun seringkali hasil identifikasi mengandung banyak *noise* karena sifatnya yang bergantung pada data masukan. Contoh metode *bottom-up* yang digunakan dalam tugas akhir ini adalah model *Spectral Residual* [5]. Sedangkan metode *top-down* yang memanfaatkan *classifier* merupakan metode yang lebih lambat, *task-driven* dan membutuhkan *supervised learning* berdasarkan *training samples*. Metode *top-down* lebih efektif dalam mendeteksi bentuk *salient object* secara global dengan berbagai ukuran dan kategori, namun sebagian besar dari metode *top-down* yang ada banyak memakan waktu pada proses *off-line training* atau proses pemberian label pada *training samples* secara manual [6]. *Saliency map* yang dihasilkan oleh metode *bottom-up* dapat dimanfaatkan sebagai salah satu solusi untuk menghindari proses *labeling training samples* secara manual.

2.3 *Saliency Detection* dengan Model *Spectral Residual*

Saliency detection dengan pendekatan model *Spectral Residual* [5] adalah metode *bottom-up* yang digunakan dalam tugas akhir ini untuk membuat *prior saliency map*. Seperti yang sudah dijelaskan pada sub-bab sebelumnya, metode *bottom-up* memanfaatkan informasi *low-level* seperti warna, tepi dan intensitas untuk mendeteksi kandidat objek atau *proto object*. Dengan memanfaatkan karakteristik dari *natural image*, *log spectrum* dari citra dianalisis untuk mendapatkan *spectral residual*. Kemudian *spectral residual* yang direpresentasikan dalam domain frekuensi diubah ke dalam domain spasial untuk mendapatkan *saliency map*, yang merepresentasikan letak dari *proto object*.

2.3.1 *Fourier Transform*

Sebuah citra yang direpresentasikan dalam domain spasial terdiri dari sekumpulan piksel yang nilainya ditentukan oleh serangkaian *channel* yaitu *channel* merah, hijau, biru. Setiap *channel* tersebut mengandung seperangkat nilai intensitas atau nilai *grayscale*. Dengan demikian, dalam domain spasial citra didefinisikan oleh nilai intensitas dalam setiap *channel* milik piksel.

Citra juga dapat direpresentasikan dalam domain frekuensi, dengan setiap *channel*-nya direpresentasikan dalam bentuk gelombang sinusoidal. Setiap *channel* memiliki nilai amplitudo yang disimpan di lokasi yang bukan berbasis koordinat spasial, melainkan berbasis frekuensi. Jumlah frekuensi sesuai dengan jumlah piksel dalam citra pada domain spasial. Semua pola gelombang sinusoidal dapat dibentuk dengan menggabungkan sejumlah gelombang sinus yang memiliki frekuensi dengan nilai amplitudo yang tepat. Oleh karena itu, representasi domain frekuensi hanyalah cara lain untuk menyimpan dan merekonstruksi citra dalam domain spasial.

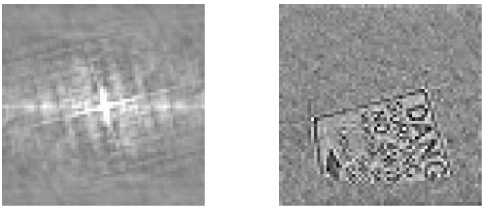
Fourier transform [9] adalah alat pengolahan cira yang menguraikan sebuah citra menjadi komponen gelombang sinusoidal. Dengan kata lain, *Fourier transform* mengubah representasi citra ke dalam domain frekuensi. *Fourier transform* menawarkan cara baru untuk melakukan pengolahan citra yang biasa dilakukan dalam domain spasial seperti meningkatkan kecerahan dan kontras, *blurring*, *sharpening* dan *noise removal*. Selain itu, *Fourier transform* juga memberikan cara pengolahan citra yang tidak dapat dilakukan dalam domain spasial, seperti *deblurring* dari distorsi kamera.

Fourier transform $F(I(x))$ menghasilkan citra keluaran bernilai bilangan kompleks yang dapat ditampilkan dengan dua komponen, yaitu *magnitude spectrum* $A(f)$ dan *phase spectrum* $P(f)$. Kedua *spectrum* tersebut didefinisikan pada persamaan (2.1) dan (2.2).

$$A(f) = absolute(F(I(x))) \quad (2.1)$$

$$P(f) = angle(F(I(x))) \quad (2.2)$$

Magnitude spectrum mendeskripsikan tingkat intensitas dari frekuensi pada citra dan banyak menyimpan informasi warna dari citra. Sedangkan *phase spectrum* menentukan pergeseran gelombang sinusoid pada citra dan banyak menyimpan informasi posisi dari citra. Kedua komponen tersebut digunakan untuk pengolahan citra dalam domain frekuensi dan digunakan dalam *Inverse Fourier transform* untuk mengubah kembali citra *Fourier* ke dalam domain spasial yang benar [10]. Perbedaan citra yang direkonstruksi hanya dari *magnitude spectrum* atau *phase spectrum* ditunjukkan pada **Gambar 2.1**.



(a) (b)

Gambar 2.1 Rekonstruksi Citra Hanya dari (a) *Magnitude Spectrum* dan (b) *Phase Spectrum*

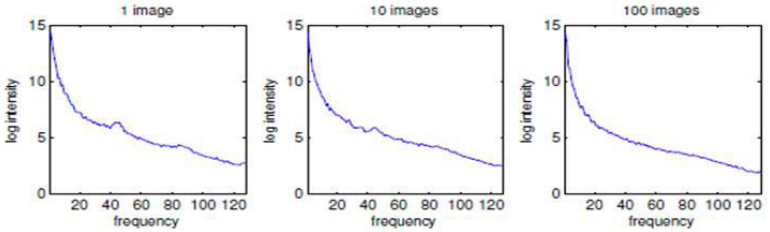
2.3.2 *Log Spectrum*

Rentang dinamis dari nilai intensitas piksel dalam citra dapat dikompres dengan mengganti setiap nilai piksel dengan logaritmanya. Efek yang diberikan oleh transformasi logaritmik tersebut adalah nilai piksel berintensitas rendah ditingkatkan, sedangkan nilai piksel berintensitas tinggi dikompres ke dalam rentang nilai yang relatif kecil [11]. Penerapan transformasi logaritmik ke citra dapat berguna dalam aplikasi dengan rentang dinamis dari nilai intensitas piksel dalam citra yang terlalu besar untuk ditampilkan di layar, contohnya seperti *magnitude*

spectrum dari citra *Fourier* yang terlihat gelap total di layar. Dengan menerapkan transformasi logaritmik pada *magnitude spectrum*, didapatkan *log spectrum* dari citra *Fourier* yang digunakan untuk proses analisis pola dari citra *spectrum* [10]. *Log spectrum* $L(f)$ didefinisikan pada persamaan (2.3).

$$L(f) = log(A(f)) \qquad (2.3)$$

Natural image telah diterima secara luas memiliki distribusi informasi statistik yang dapat diprediksi. *Log spectrum* dari *natural image* yang berbeda memiliki kecenderungan statistik yang serupa, meskipun setiap citra tetap memiliki statistik singularitas. Pada **Gambar 2.2** ditunjukkan kurva statistik rata-rata *log spectrum* dari 1, 10 dan 100 citra. Hasil kurva tersebut menunjukkan adanya *local linearity* dalam rata-rata *log spectrum* [5].



Gambar 2.2 Kurva Statistik Rata-Rata *Log Spectrum* dari 1, 10 dan 100 Citra [5]

2.3.3 *Spectral Residual*

Dalam *log spectrum*, citra yang berbeda dengan kemiripan statistik yang besar dapat diamati, yang harus diperhatikan adalah singularitas yang tampak menonjol dari kurva yang telah di-*smoothing*. *Spectral residual* [5] atau statistik singularitas dalam *log spectrum* bertanggung jawab atas daerah anomali pada citra, di mana *proto object* biasanya tampak. *Spectral residual* dari sebuah citra didapatkan dengan mengurangi *log spectrum* citra

tersebut dengan rata-rata *log spectrum*. Umumnya untuk mengambil rata-rata *log spectrum* dibutuhkan banyak citra, namun pada model ini hanya digunakan satu citra masukan yang dikonvolusi menggunakan *local average filter* atau *mean filter* $h_n(f)$ untuk mendapatkan aproksimasi. *Spectral residual* $R(f)$ didefinisikan pada persamaan (2.4).

$$R(f) = L(f) - h_n(f) \times L(f) \quad (2.4)$$

Dari *spectral residual* yang dihasilkan, *saliency map* $S(f)$ dapat dibentuk sesuai dengan persamaan (2.5).

$$S(f) = g(x) \times \|F^{-1}[\exp(R(f) + i \times P(f))\|^2 \quad (2.5)$$

Di mana F^{-1} adalah *Inverse Fourier transform*, $P(f)$ adalah *phase spectrum*, i adalah unit *imaginary* dan $g(x)$ adalah *guided filter* untuk efek visual yang lebih baik.

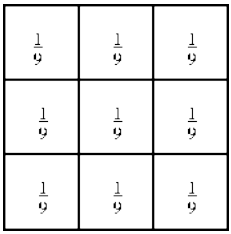
2.4 Filter Citra

Filter citra digunakan untuk mereduksi frekuensi tinggi pada citra seperti *smoothing* atau frekuensi rendah seperti deteksi tepi. Beberapa contoh dari filter citra adalah *mean filter, disk filter, Gaussian filter* dan *guided filter*.

Sebuah citra dapat diterapkan filter baik dalam domain frekuensi maupun domain spasial. Untuk melakukan *filtering* dalam domain frekuensi, citra diubah ke dalam domain frekuensi dan dikalikan dengan filter frekuensi sebelum dikembalikan ke domain spasial. Sedangkan untuk melakukan *filtering* dalam domain spasial, citra masukan dikonvolusi dengan sebuah *kernel.* Semua filter frekuensi dapat diterapkan dalam domain spasial jika terdapat *kernel* untuk efek filter yang diinginkan. *Filtering* citra dalam domain spasial memiliki tingkat komputasi yang lebih rendah [12].

2.4.1 *Mean Filter*

Mean filter digunakan untuk *smoothing* citra, mereduksi variasi intensitas piksel antara satu piksel dengan yang lain. Filter ini banyak digunakan untuk mereduksi *noise* pada citra. Inti dari *mean filtering* adalah mengganti setiap nilai piksel pada citra dengan nilai rata-rata piksel tetangganya [13]. *Kernel* dari *mean filter* berukuran 3x3 ditunjukkan pada **Gambar 2.3**.



Gambar 2.3 *Kernel Mean Filter* Berukuran 3x3 [13]

Cara kerja *mean filter* yang mengganti nilai piksel dengan nilai rata-rata piksel tetangganya mengakibatkan hilangnya nilai piksel yang terlalu rendah atau tinggi dibanding tetangganya. Piksel yang terlalu rendah atau tinggi ini bisa jadi merupakan sebuah *noise* karena nilainya yang jauh berbeda dengan piksel sekitarnya.

2.4.2 *Disk Filter*

Disk filter [14] digunakan untuk mereduksi *noise* pada citra seperti halnya *mean filter*, namun memiliki perbedaan pada bentuk dan bobot dalam *kernel* yang digunakan. Inti dari *disk filter* adalah mengganti setiap nilai piksel pada citra dengan nilai rata-rata piksel tetangganya, dengan bentuk *kernel* berupa *pillbox* dalam matriks persegi berukuran $2 \times radius + 1$. Setiap elemen pada matriks *kernel* diberi bobot berdasarkan seberapa besar daerah pada elemen tersebut yang terselimuti oleh *pillbox*, jika semakin tertutup maka bobot yang diberikan semakin besar. *Kernel* dari *disk filter* dengan radius bernilai 1 ditunjukkan pada **Gambar 2.4**.

0	1/7	0
1/7	1/3	1/7
0	1/7	0

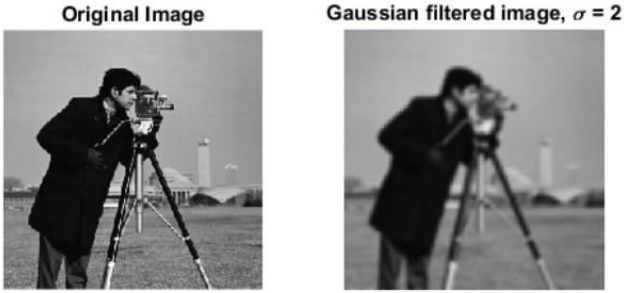
Gambar 2.4 *Kernel* dari *Disk Filter* dengan radius = 1

2.4.3 *Gaussian Filter*

Gaussian filter [15] digunakan untuk memberikan efek *blur* sekaligus mereduksi *noise* dan detil pada citra. *Kernel* dari *Gaussian filter* memiliki bentuk seperti lonceng atau distribusi *Gaussian*. Matriks *kernel* dibentuk berdasarkan aproksimasi dari persamaan (2.6), di mana σ adalah standar deviasi dari distribusi.

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

Nilai σ yang besar menghasilkan puncak distribusi yang lebih lebar dan efek *blur* yang lebih besar pada citra. Bobot elemen dalam *kernel* semakin berkurang nilainya jika letaknya semakin jauh dari pusat *kernel*. Seiring meningkatnya nilai σ, ukuran *kernel* harus disesuaikan dengan formula $2 \times ceil(2 \times \sigma) + 1$ untuk mempertahankan sifat *Gaussian filter*. Hasil dari citra yang diterapkan *Gaussian filter* dengan σ bernilai 2 ditunjukkan pada **Gambar 2.5**.



Gambar 2.5 Perbedaan Citra Asli dengan Citra yang Diterapkan *Gaussian Filter dengan* $\sigma = 2$ [15]

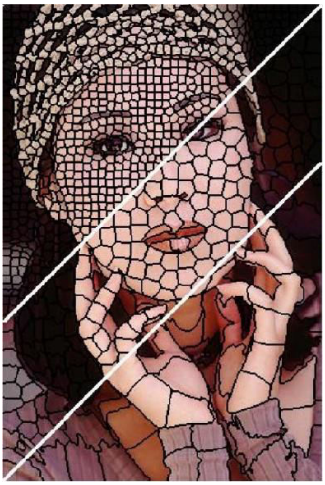
2.4.4 *Guided Filter*

Guided filter [16] digunakan untuk melakukan *edge-preserving smoothing* pada citra dengan bantuan citra kedua yang disebut *guidance image. Guidance image* dapat berupa citra masukan itu sendiri, versi berbeda dari citra masukan, atau bahkan citra yang sama sekali berbeda. *Guidance image* digunakan sebagai panduan dalam melakukan *filtering*. Inti dari *guided filter* adalah mengganti setiap nilai piksel pada citra dengan nilai rata-rata piksel tetangganya, namun dengan memperhitungkan statistik seperti nilai varians pada wilayah spasial yang sesuai pada *guidance image*. *Guided filter* terbukti efektif dan efisien dalam berbagai aplikasi pengolahan citra seperti reduksi *noise, smoothing* detil pada citra, kompresi HDR dan *feathering* citra. Hasil dari citra yang diterapkan *guided filter* ditunjukkan pada **Gambar 2.6**.

Gambar 2.6 Perbedaan Citra Asli dengan Citra yang Diterapkan *Guided Filter* [16]

2.5 *Multi-Scale Superpixels*

Tingkat *saliency* pada citra umumnya diidentifikasi pada tiga tingkat, yaitu piksel, *superpixels* dan daerah. Metode *Saliency Extreme Learning Machine* dibangun berbasis *superpixels*, yang merupakan sekelompok piksel dengan fitur atau informasi yang serupa. *Superpixels* berukuran homogen dan tidak merusak informasi batas dari *salient object* [4]. *Superpixels* dapat menangkap redundansi pada citra, representasi yang lebih efisien dan mengurangi kompleksitas pengolahan citra pada tahap selanjutnya, selain itu *superpixels* terbukti semakin berguna dalam aplikasi estimasi kedalaman, segmentasi citra, skeletonisasi, estimasi model tubuh dan lokalisasi objek [17]. Contoh penerapan *superpixels* pada citra ditunjukkan pada **Gambar 2.7**.



Gambar 2.7 Citra yang Tersegmentasi dalam *Superpixels* Berukuran 64, 256 dan 1024 dari Atas ke Bawah [17]

Model *multi-scale superpixels*, yang bertujuan untuk mendeteksi *salient object* pada berbagai skala, telah terbukti efektif dalam meningkatkan akurasi *saliency map* [6]. Skala yang dimaksud adalah ukuran *superpixels* atau jumlah piksel dalam setiap *superpixel.* Dalam metode *Saliency Extreme Learning Machine*, digunakan empat skala *superpixels* bernilai 100, 150, 200 dan 250 ketika membentuk *training samples dan testing samples*. Sehingga untuk satu citra masukan dihasilkan empat *saliency map* masing-masing untuk skala yang berbeda.

2.6 Pelabelan *Training Samples* Secara Otomatis

Prior saliency map yang dihasilkan oleh *saliency detection* dengan pendekatan model *Spectral Residual* (metode *bottom-up)* dapat dimanfaatkan sebagai solusi untuk menghindari proses *labeling training samples* secara manual [3]. Setiap piksel dari *prior saliency map* mempunyai nilai *saliency,* sehingga nilai *saliency* dari setiap *superpixel* didapatkan dari rata-rata nilai *saliency* semua piksel yang dikandung *superpixel* tersebut.

Training samples untuk mempelajari *classifier* mencakup *training samples* positif dari *salient object* dan *training samples* negatif dari *background*. Untuk menentukan *training samples* positif dan negatif digunakan dua *threshold*, yaitu *threshold* positif $Thre_P$ dan *threshold* negatif $Thre_N$. *Threshold* negatif $Thre_N$ ditetapkan dengan nilai 0,05, sedangkan *threshold* positif $Thre_P$ didefinisikan pada persamaan (2.7).

$$Thre_P = 1.5 \times \overline{S} \qquad (2.7)$$

Di mana $\overline{S}$ adalah rata-rata nilai *saliency* semua piksel di *prior saliency map*. *Superpixels* dengan nilai *saliency* yang lebih besar dari *threshold* positif akan diberi label +1 sebagai *training samples* positif, sebaliknya *superpixels* dengan nilai *saliency* yang lebih kecil dari *threshold* negatif akan diberi label -1 sebagai *training samples* negatif. *Superpixels* yang memiliki nilai *saliency* di antara *threshold* positif dan negatif tidak digunakan sebagai *training samples*. Dengan asumsi bahwa *salient object* biasanya tidak tampak pada batas-batas citra, *superpixels* di sepanjang batas citra diberi label -1 sebagai *training samples* negatif.

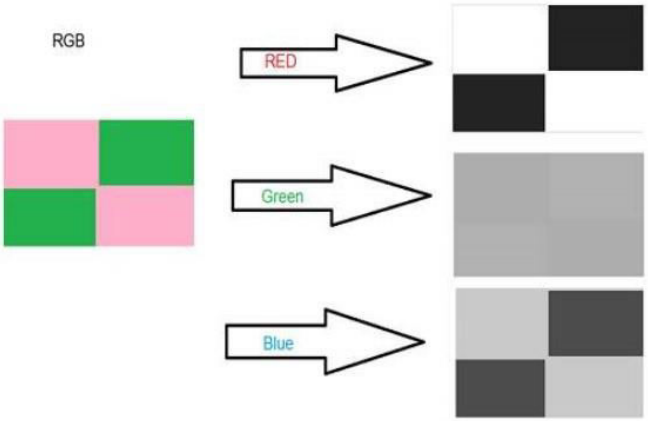
2.7 Ekstraksi Fitur

Banyak fitur seperti warna, tekstur dan lokasi diterapkan untuk menentukan nilai *saliency.* Dalam metode *Saliency Extreme Learning Machine*, digunakan dua jenis fitur yaitu warna dan tekstur untuk mendeskripsikan setiap *superpixel.* Pada fitur warna akan digunakan RGB dan CIELAB, sedangkan pada fitur tekstur akan digunakan *Uniform Local Binary Pattern*. Vektor fitur berukuran 65 dimensi terbentuk dari gabungan RGB (3 fitur), CIELAB (3 fitur) dan *Uniform Local Binary Pattern* (59 fitur).

2.7.1 Fitur Warna RGB

RGB [18] yang merupakan akronim dari *red, green* dan *blue*, merupakan ruang warna yang paling sering digunakan

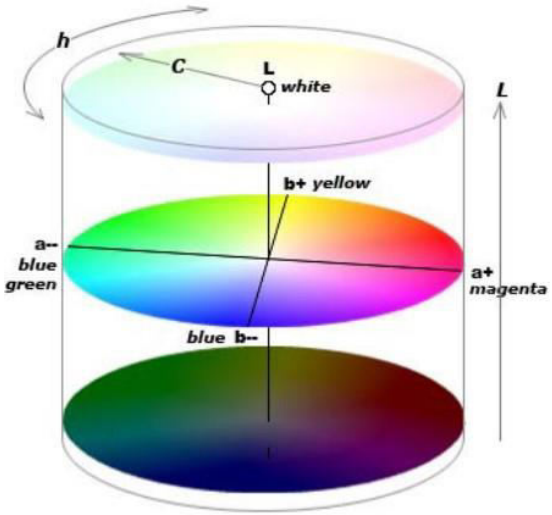
dalam pengolahan citra. Setiap citra yang berwarna terbentuk dari tiga citra yang berbeda, satu untuk setiap *channel red, green* dan *blue* seperti yang ditunjukkan pada **Gambar 2.8**. Nilai dari setiap *channel* tersebut disimpan dalam tiga matriks yang digabungkan untuk membentuk citra berwarna yang utuh.



Gambar 2.8 Citra Terdiri dari *Channel Red, Green dan Blue* [18]

2.7.2 Fitur Warna CIELAB

CIELAB [19] adalah ruang warna yang didefinisikan oleh *International Commission on Illumination* (disingkat CIE untuk nama Prancis-nya) pada tahun 1976. CIELAB menggunakan model tiga dimensi yang disebut dengan L*a*b*. Dimensi L* untuk mendeskripsikan kecerahan warna, 0 untuk hitam dan 100 untuk putih. Dimensi a* mendeskripsikan jenis warna hijau – merah, dengan nilai negatif mengindikasikan warna hijau dan sebaliknya nilai positif mengindikasikan warna merah. Dimensi b* mendeskripsikan jenis warna biru – kuning, dengan nilai negatif mengindikasikan warna biru dan sebaliknya nilai positif mengindikasikan warna kuning. Tanda * setelah L, a dan b digunakan untuk membedakannya dari versi lain. Model dari CIELAB ditunjukkan pada **Gambar 2.9**.



Gambar 2.9 Model dari Ruang Warna CIELAB [19]

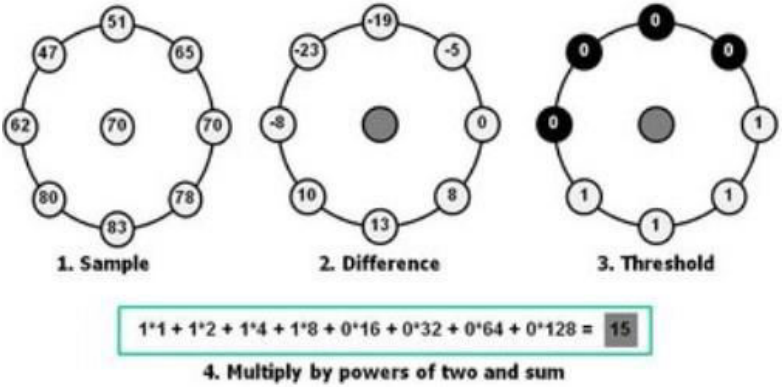
2.7.3 Fitur Tekstur *Uniform Local Binary Pattern*

Local Binary Pattern (LBP) [20] adalah operator tekstur sederhana namun efisien yang bertujuan untuk mendapatkan tekstur lokal dari setiap piksel pada citra. LBP memberikan label pada setiap piksel dengan melakukan *thresholding* terhadap piksel tetangganya. LBP yang pertama kali diajukan (Ojala et al. 1996) menggunakan tetangga berjumlah 8 dengan radius bernilai 1, kemudian pada tahun 2002 dikembangkan kembali sehingga dapat memakai piksel tetangga dengan berbagai ukuran dan radius. LBP dengan tetangga berjumlah 8 dan radius bernilai 1 memiliki 2^8 atau 256 kemungkinan kombinasi pola untuk setiap piksel. Nilai LBP dari setiap piksel dapat dihitung menggunakan persamaan (2.8) dan (2.9).

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \qquad (2.8)$$

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & otherwise \end{cases} \qquad (2.9)$$

Di mana *P* adalah jumlah tetangga, *R* adalah radius, *s(x)* adalah fungsi *threshold*, g_c adalah nilai *grayscale* dari piksel dan $g_p$adalah nilai *grayscale* dari piksel tetangga. Contoh komputasi nilai LBP dengan *P* bernilai 8 dan *R* bernilai 1 ditunjukkan pada **Gambar 2.10**.



Gambar 2.10 Komputasi Nilai LBP dengan *P* = 8 dan *R* = 1 [20]

Uniform Local Binary Pattern (*Uniform* LBP) [21] merupakan perkembangan dari LBP yang menggunakan pola *uniform*. *Uniform* LBP berguna untuk mengurangi panjang vektor fitur dan bersifat invarian terhadap rotasi. Perkembangan *Uniform* LBP terinspirasi oleh fakta bahwa beberapa pola (khususnya pola *uniform*) lebih sering tampak pada tekstur citra, sedangkan pola yang jarang tampak atau pola non-*uniform* memiliki jumlah yang sangat sedikit sehingga sulit untuk dilakukan estimasi. LBP disebut *uniform* jika pola binernya maksimal mengandung dua transisi *bitwise* dari 0 ke 1 atau sebaliknya. Semua LBP yang memiliki pola non-*uniform* akan diberi satu label yang tetap. Pada LBP dengan tetangga berjumlah 8 dan radius bernilai 1 yang memiliki 256 kombinasi pola, 58 di antaranya adalah pola *uniform*, sehingga panjang vektor fitur dapat berkurang dari 256 ke 59. Sebagai contoh, LBP dengan nilai 5 selalu memiliki pola

biner [1 0 1 0 0 0 0 0], karena mengandung lebih dari dua transisi *bitwise* maka pola tersebut termasuk non-*uniform* dan diberikan label khusus (misalkan 58) untuk semua pola non-*uniform*. Sedangkan LBP dengan nilai 254 selalu memiliki pola [0 1 1 1 1 1 1 1], karena hanya mengandung dua transisi *bitwise* maka pola tersebut termasuk *uniform* dan diberikan label sesuai urutan untuk pola *uniform* (nilai 254 termasuk urutan ke-56 dalam *range* 0 sampai 57 pola *uniform*).

2.8 *Extreme Learning Machine Classifier*

Dalam beberapa dekade terakhir, telah banyak metode yang dikembangkan untuk melatih *Single Layer Feedforward Neural Network* (SLFN), di antaranya adalah metode *gradient based* dengan algoritma *back-propagation*, metode optimisasi seperti *Support Vector Machine*, dan metode *Least Mean Square* seperti fungsi *Radial Basis*. Perlu diketahui bahwa pada hampir semua algoritma SLFN, bobot masukan yang menghubungkan *input layer* ke *hidden layer* dan nilai bias pada *hidden layer* harus dikonfigurasi lebih lanjut selama jalannya proses *training* [3]. *Extreme Learning Machine* (ELM), metode yang pertama kali diperkenalkan oleh Huang et al. [2,3], merupakan algoritma *machine learning* sederhana dengan karakteristik *learning speed* yang cepat dan kemampuan generalisasi yang baik. ELM secara sederhana adalah SLFN dengan bobot antara *input* dan *hidden nodes* ditetapkan menggunakan *random feature mapping* atau *kernel* dan tidak pernah diperbaharui bobotnya. Fungsi keluaran dari ELM $f_L(x)$ ditunjukkan pada persamaan (2.10) berikut.

$$f_L(x) = h(x)\beta \qquad (2.10)$$

Di mana $\beta = [\beta_1, \dots, \beta_L]^T$ adalah vektor bobot keluaran antara *hidden layer* dengan L *nodes* dan *output nodes*, $h(x) = [h_1(x), \dots, h_L(x)]$ adalah vektor keluaran dari *hidden layer*. $h(x)$ merupakan fungsi *feature mapping* untuk memetakan data

masukan yang berdimensi *d* ke *hidden layer feature space* yang berdimensi *L*.

Huang et al. [3] telah membuktikan bahwa bobot masukan yang menghubungkan *input layer* ke *hidden layer* dari SLFN dapat ditetapkan secara acak jika fungsi aktivasi pada *hidden layer* memiliki sifat *infinitely differentiable*. Fungsi aktivasi tersebut mencakup hampir semua fungsi kontinu bersifat linier dan nonlinier, seperti fungsi *Sigmoid*, fungsi *Gaussian* dan fungsi *Radial Basis*. ELM menerapkan hal tersebut dalam *feature mapping* sehingga ELM memiliki kemampuan untuk melakukan aproksimasi fungsi aktivasi secara universal [22]. Dengan kemampuan tersebut, nilai bias pada *hidden layer* tidak perlu digunakan, yang menyebabkan ELM memiliki *optimization constraint* yang lebih ringan. *Optimization constraint* yang ringan menghasilkan kemampuan generalisasi yang lebih baik dan kompleksitas komputasional yang lebih rendah.

Setelah *random feature mapping* dilakukan, proses lain dari ELM dapat dianggap sebagai sistem linier. Proses tersebut adalah mencari solusi *least-squares* dari persamaan (2.11).

$$H\beta = T \qquad (2.11)$$

Di mana H adalah matriks *training samples* dan T adalah matriks label dari *training samples*. Solusi *least-squares* yang memiliki *norm* terkecil dari sistem linier tersebut ditunjukkan pada persamaan (2.12) dan (2.13).

$$\beta = H^{+}T \qquad (2.12)$$

$$H^{+} = H^{T}(HH^{T})^{-1} \qquad (2.13)$$

Di mana H^{+} adalah *Moore-Penrose generalized inverse* atau *pseudoinverse* [3] dari matriks H. *Pseudoinverse* digunakan karena cenderung mencapai *training error* terkecil dan nilai bobot keluaran yang paling minimum jika dibandingkan dengan solusi

least-squares yang lain, kedua karakteristik tersebut penting untuk mencapai kemampuan generalisasi yang lebih baik.

Dalam metode *Saliency Extreme Learning Machine*, ELM *classifier* digunakan untuk melakukan *saliency detection* secara *top-down* dengan memanfaatkan *training samples* yang didapatkan dari *prior saliency map* dan ekstraksi fitur. Proses yang dilakukan pertama kali adalah melakukan *feature mapping* menggunakan *linear kernel*. *Linear kernel* tidak membutuhkan parameter tambahan dan lebih cepat secara komputasional jika dibandingkan dengan *kernel* atau fungsi *feature mapping* lainnya. *Linear kernel* K yang digunakan didefinisikan pada persamaan (2.14).

$$K = DD^T \quad (2.14)$$

Linear kernel juga diterapkan dalam *kernel* ELM Ω_{ELM} yang didefinisikan pada persamaan (2.15).

$$\Omega_{ELM} = HH^T \quad (2.15)$$

Di mana H adalah matriks dari *training samples* $\{r_i, t_i\}_{i=1}^{N_t}$ dan D adalah matriks *testing samples*. r_i adalah *training sample* ke i, t_i adalah label yang sesuai, N_t adalah jumlah *training samples*. Perlu diingat bahwa *training samples* dikumpulkan dari keempat skala *superpixels* terlebih dahulu, sedangkan *testing samples* terdiri dari skala *superpixels* yang sedang diuji. Nilai bobot keluaran β antara *hidden layer* dan *output layer* pada persamaan (2.12) dapat dihitung dengan menerapkan persamaan (2.13) dan (2.15) menjadi persamaan (2.16).

$$\beta = H^T(\Omega_{ELM} + \frac{I}{C})^{-1}T \quad (2.16)$$

Di mana I adalah N_t x N_t unit matriks, C adalah parameter yang ditentukan oleh pengguna, yang menyediakan *tradeoff*

antara jarak dari *separating margin* dan *training error*. Pada tugas akhir ini ditentukan nilai $C = 1$. $T = \{t_i\}_{i=1}^{N_t}$ adalah vektor label dari *training samples*. Sehingga fungsi keluaran *S* dari ELM pada persamaan (2.10) dapat ditulis secara detil dengan menerapkan persamaan (2.14) dan (2.16) menjadi persamaan (2.17).

$$S = DD^T(\Omega_{ELM} + \frac{I}{C})^{-1}T \qquad (2.17)$$

Di mana fungsi keluaran *S* merupakan vektor yang berisi nilai *saliency* dari setiap *superpixel*. *Trained saliency map* dihasilkan dengan memetakan nilai *saliency* dari setiap *superpixel* ke piksel yang dikandungnya. Hasil keluaran dari ELM *classifier* terdiri dari empat skala *trained saliency map* yang akan di-*smoothing* menggunakan metode *Graph Cut*, setelah itu digabungkan dengan mengambil rata-rata nilai *saliency* setiap piksel dari empat skala yang berbeda.

2.9 Metode *Graph Cut*

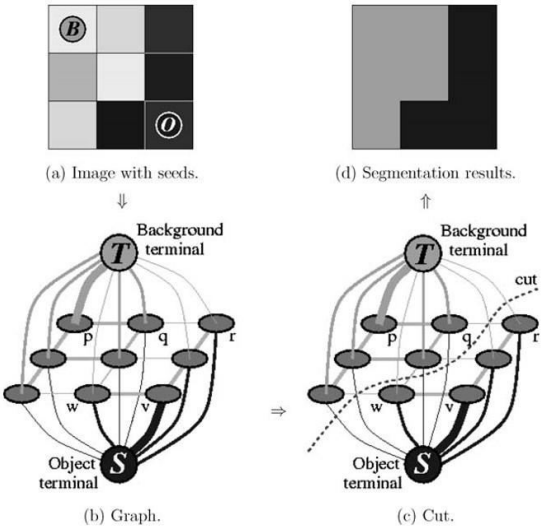
Sebuah *s-t graph* adalah *graph* berbobot dan berarah yang memiliki dua *node* penting yaitu *source s* dan *sink t*. Sedangkan *s-t cut* adalah sekumpulan *edge* E_{cut} yang menyebabkan terputusnya atau hilangnya jalur dari *source* ke *sink*, sehingga *source* dan *sink* berada di *subset* yang berbeda. Biaya dari *s-t cut* adalah jumlah bobot dari *edge* dalam E_{cut}. Permasalahan minimum *cut* pada *s-t graph* adalah mencari E_{cut} dengan jumlah bobot terkecil, sedangkan permasalahan maksimum *flow* adalah memaksimalkan jumlah *flow* yang mengalir dari *source* ke *sink*. Teori *max-flow*/*min-cut* menyatakan bahwa maksimum *flow* dari *source* ke *sink* sama dengan jumlah bobot dari minimum *cut* [23].

Graph Cut [24] adalah algoritma *energy minimization* yang banyak digunakan dalam bidang visi komputer untuk menyelesaikan masalah seperti segmentasi citra dan *smoothing* citra. *Graph Cut* menggunakan model *max-flow*/*min-cut* dengan merancang *graph* dari piksel-piksel pada citra, kemudian

minimum *cut* dari *graph* tersebut memotong *edge* yang menghubungkan objek atau label yang berbeda sehingga citra dapat tersegmentasi. Tujuan utama dari *Graph Cut* adalah meminimalisir energi yang digunakan untuk melakukan proses labeling setiap piksel, energi $E(f)$ untuk melakukan labeling f didefinisikan pada persamaan (2.18).

$$E(f) = E_{data}(f) + E_{smooth}(f) \qquad (2.18)$$

Di mana *data cost* $E_{data}(f)$ merupakan biaya yang ditetapkan untuk menentukan label piksel, sedangkan *smoothness cost* $E_{smooth}(f)$ merupakan biaya untuk menentukan label dari piksel tetangga. Jika label dari piksel tetangga sama dengan label piksel terpilih maka *smoothness cost* bernilai 0, untuk kondisi sebaliknya maka *smoothness cost* bernilai lebih besar dari 0. Contoh segmentasi citra berukuran 3x3 dengan menggunakan metode *Graph Cut* ditunjukkan pada **Gambar 2.11**.



Gambar 2.11 Segmentasi Citra Berukuran 3x3 Menggunakan Metode *Graph Cut* [24]

Dalam metode *Saliency Extreme Learning Machine*, metode *Graph Cut* digunakan untuk melakukan *smoothing* terhadap *trained saliency map* M_0 di setiap skala. Pertama akan dibangun *graph* $G = (V, E, T)$, di mana E adalah kumpulan dari *edges* yang menghubungkan *nodes V* (piksel), sedangkan T adalah *data cost* untuk *foreground* dan *background*. Untuk setiap piksel p, T terbagi menjadi dua komponen, yaitu $T^f(p)$ untuk *foreground* dan $T^b(p)$ untuk *background* yang didefinisikan pada persamaan (2.19).

$$T^f(p) = M_0(p),\ T^b(p) = 1 - M_0(p) \qquad (2.19)$$

Metode *Graph Cut* menghasilkan *binary map* M_1 yang akan digabungkan dengan *trained saliency map* sesuai dengan persamaan (2.20).

$$M_C = \frac{M_0 + M_1}{2} \qquad (2.20)$$

Di mana M_C adalah *smoothed trained saliency map*, dengan *noise* yang sudah tereduksi dan lebih menonjolkan nilai *saliency* dari *salient object*.

2.10 Pembentukan *Object Map* dengan Metode Otsu

Object map merupakan citra biner yang didapatkan dari *saliency map* $S(x)$ yang disegmentasi berdasarkan nilai *threshold*. Pembentukan *object map* $O(x)$ didefinisikan pada persamaan (2.21).

$$O(x) = \begin{cases} 1, & S(x) > threshold \\ 0, & otherwise \end{cases} \qquad (2.21)$$

Untuk pemilihan nilai *threshold* digunakan metode Otsu [25]. Pada metode Otsu didefinisikan bahwa sebuah citra memiliki dua kelas piksel yaitu piksel *foreground* dan piksel *background*, kemudian *threshold* optimum yang meminimalkan

varians intra kelas dari dua kelas tersebut dicari. Pencarian *threshold* optimum didefinisikan pada persamaan (2.22).

$$\sigma_w^2(t) = w_0(t)\sigma_0^2(t) + w_1(t)\sigma_1^2(t) \qquad (2.22)$$

Di mana σ_w^2 adalah jumlah varians dari kedua kelas, w_0 dan w_1 adalah probabilitas kedua kelas dipisahkan oleh *threshold* t yang dihitung dari histogram, σ_0^2 dan σ_1^2 adalah varians dari kedua kelas tersebut.

Dalam tugas akhir ini, *object map* dibentuk dari *trained saliency map* yang disegmentasi menggunakan metode Otsu. *Object map* merupakan citra keluaran yang menandakan area *salient object* (sebagai piksel berwarna putih) dan area non-*salient object* (sebagai piksel berwarna hitam) pada citra masukan.

2.11 *Confusion Matrix*

Confusion matrix [26] adalah sebuah statistik klasifikasi yang menyimpan informasi mengenai prediksi kelas dan kelas asli. *Confusion matrix* banyak digunakan untuk menguji performa dari suatu metode klasifikasi. Pada tugas akhir ini, *confusion matrix* mencatat informasi klasifikasi piksel menjadi area *salient object* (*foreground*) atau area non-*salient object* (*background*). Struktur *confusion matrix* ditunjukkan pada **Gambar 2.12**.

Dalam *confusion matrix* yang digunakan, terdapat statistik *true positive* (TP) yaitu piksel area *salient object* yang terklasifikasi sebagai area *salient object, true negative* (TN) yaitu piksel area non-*salient object* yang terklasifikasi sebagai area non-*salient object, false positive* (FP) yaitu piksel area non-*salient object* yang terklasifikasi sebagai area *salient object*, *false negative* (FN) yaitu piksel area *salient object* yang terklasifikasi sebagai area non-*salient object*.

		Prediksi	
Aktual		*Salient Object*	Non-*Salient Object*
	Salient Object	TP	FN
	Non-*Salient Object*	FP	TN

Gambar 2.12 Confusion Matrix Berdasarkan Area

Dari *confusion matrix* bisa didapatkan berbagai informasi mengenai performa *classifier,* di antaranya *recall* dan presisi. *Recall* adalah jumlah data benar yang terklasifikasi dari keseluruhan data aktual yang benar pada *ground truth*. Rumus perhitungan *recall* ditunjukkan pada persamaan (2.23).

$$Recall = \frac{TP}{TP+FN} \qquad (2.23)$$

Presisi adalah jumlah data benar yang terklasifikasi dari keseluruhan data yang diprediksi benar oleh sistem. Rumus perhitungan presisi ditunjukkan pada persamaan (2.24).

$$Presisi = \frac{TP}{TP+FP} \qquad (2.24)$$

F1 score atau *F-measure* adalah pengukuran performa *classifier* dalam mendapatkan informasi yang diinginkan dengan memperhatikan keseimbangan antara nilai *recall* dan presisi. Rumus perhitungan *F1 score* ditunjukkan pada persamaan (2.25).

$$F1\ score = \frac{2TP}{2TP+FP+FN} \qquad (2.25)$$

[*Halaman ini sengaja dikosongkan*]

BAB III
ANALISIS DAN PERANCANGAN

Bab analisis dan perancangan berisi analisis kebutuhan dan perancangan aplikasi yang akan dibangun. Tahap analisis membahas mengenai analisis kebutuhan yang menjadi dasar dari tahap perancangan.

3.1 Tahap Analisis

Tahap analisis mendefinisikan kebutuhan yang akan dipenuhi dalam pembangunan aplikasi segmentasi citra area *salient object* menggunakan metode *Saliency Extreme Learning Machine*. Selain itu dijelaskan pula alasan pengerjaan masing-masing tahap pada tugas akhir ini.

3.1.1 Deskripsi Umum

Pada tugas akhir ini dibangun aplikasi untuk melakukan segmentasi area *salient object* pada citra. Data masukan yang digunakan adalah *natural image* dalam berbagai ukuran yang bersumber dari *MSRA Salient Object Database*. Data keluaran dari aplikasi merupakan *object map* atau citra biner (hitam putih) yang merepresentasikan hasil segmentasi area *salient object*.

Aplikasi ini diharapkan dapat digunakan untuk mengidentifikasi *salient object* pada citra secara cepat dan benar. Secara teori, *Extreme Learning Machine* (ELM) merupakan algoritma *machine learning* sederhana dengan karakteristik *learning speed* yang cepat dan kemampuan generalisasi yang baik. Oleh karena itu, manfaat lain dari aplikasi ini adalah mengeksplorasi kinerja dari ELM *classifier* ketika digabungkan dengan metode *bottom-up* dalam melakukan segmentasi citra area *salient object*.

3.1.2 Spesifikasi Kebutuhan Sistem

Pada aplikasi segmentasi area *salient object* dibutuhkan beberapa proses untuk dapat memenuhi kebutuhan sistem dalam menghasilkan segmentasi yang akurat. Proses tersebut antara lain:

1. *Preprocessing*

   Preprocessing dilakukan untuk membagi citra masukan ke dalam empat skala *superpixels* dan me-*resize* citra masukan.
2. *Saliency detection* dengan model *Spectral Residual*

   Saliency detection dengan model *Spectral Residual* dilakukan untuk membentuk *prior saliency map* yang menjadi acuan dalam pelabelan *training samples*.
3. Ekstraksi fitur

   Ekstraksi fitur dilakukan untuk mendapatkan data yang representatif terhadap karakteristik objek dalam citra.
4. *Saliency detection* dengan ELM *classifier*

   Saliency detection dengan ELM *classifier* dilakukan untuk membentuk *trained saliency map* dengan memanfaatkan kemampuan ELM *classifier* yang dilatih menggunakan *training samples*.
5. Segmentasi citra

   Segmentasi citra dilakukan untuk membentuk *object map* dari *trained saliency map* yang disegmentasi menggunakan metode Otsu.

3.1.3 Analisis Permasalahan

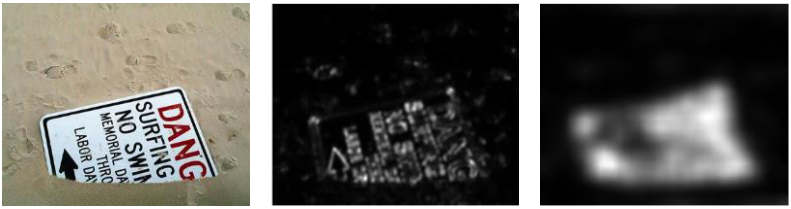
Untuk mencapai hasil segmentasi yang akurat, terdapat beberapa permasalahan yang dapat menurunkan akurasi segmentasi. Permasalahan-permasalahan tersebut dijelaskan pada sub-bab berikutnya.

3.1.3.1 Analisis Permasalahan *Preprocessing*

Pemilihan skala pada citra masukan mempengaruhi *saliency map* yang dihasilkan. Untuk citra dengan skala kecil dapat menghasilkan *saliency map* untuk *salient object* yang relatif besar dalam citra tersebut, sedangkan citra pada skala besar hanya menghasilkan *saliency map* untuk *salient object* yang relatif kecil dalam citra. Dalam tugas akhir ini, *prior saliency map* dibentuk dari citra masukan dengan skala yang kecil untuk mendeteksi *proto object* yang menonjol jika dilihat sekilas secara visual.

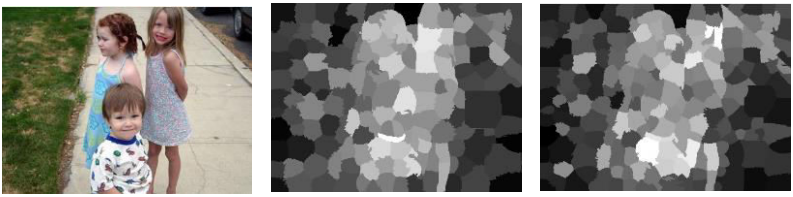
Maka dari itu sebelum *prior saliency map* dibentuk, citra masukan di-*resize* ke skala 64x64 yang terbukti merupakan skala yang baik untuk kondisi visual secara normal. Perbedaan *prior saliency map* dengan skala kecil dan skala besar ditunjukkan pada **Gambar 3.1**.

Training samples dan testing samples dalam metode *Saliency Extreme Learning Machine* dibangun berbasis *multi-scale superpixels*. Hal ini bertujuan untuk mendeteksi *salient object* pada berbagai skala yang dapat meningkatkan akurasi dari *saliency map* dan mengurangi kompleksitas saat pengolahan citra. Citra masukan dibagi ke dalam empat skala *superpixels* yang masing-masing berukuran 100, 150, 200 dan 250 piksel. Sehingga untuk satu citra masukan dihasilkan empat *saliency map* masing-masing untuk skala yang berbeda. Hasil *saliency map* dengan skala *superpixels* yang berbeda ditunjukkan pada **Gambar 3.2**.



(a) (b) (c)

Gambar 3.1 Perbedaan *Prior Saliency Map*, (a) Citra Masukan, (b) Skala Besar 400x300, (c) Skala Kecil 64x64



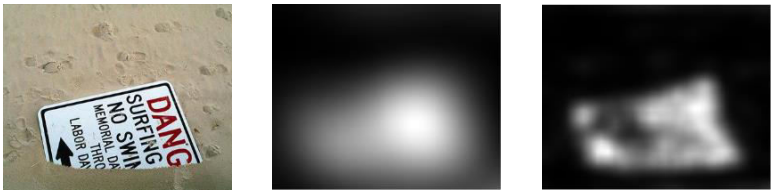
(a) (b) (c)

Gambar 3.2 Perbedaan Hasil *Saliency Map*, (a) Citra Masukan, (b) *Superpixels* Berukuran 150, (c) *Superpixels* Berukuran 200

3.1.3.2 Analisis Permasalahan *Saliency Detection* dengan Model *Spectral Residual*

Saliency detection dengan model *Spectral Residual* menghasilkan *prior saliency map* yang digunakan sebagai acuan dalam pelabelan *training samples*. Nilai *saliency* setiap piksel dari *prior saliency map* yang dibentuk mempengaruhi label dari *training samples*. Pemilihan filter dalam melakukan *smoothing* menjadi faktor penting terhadap kualitas *prior saliency map*. Pada tugas akhir ini, *prior saliency map* di-*smoothing* menggunakan *guided filter*. *Guided filter* digunakan untuk melakukan *edge-preserving smoothing* pada citra dengan bantuan citra kedua yang disebut *guidance image*. *Guided filter* dipilih karena kemampuannya untuk menjaga tepi-tepi yang tajam ketika proses *smoothing saliency map*, sehingga bentuk detil dari *salient object* tetap terjaga. Hasil *saliency map* yang di-*smoothing* menggunakan filter yang berbeda ditunjukkan pada **Gambar 3.3**.

Training samples untuk mempelajari *classifier* mencakup *training samples* positif dari *salient object* dan *training samples* negatif dari *background*. Untuk menentukan *training samples* positif dan negatif digunakan dua *threshold*, yaitu *threshold* positif dan *threshold* negatif. Pemilihan nilai *threshold* mempengaruhi kualitas dari *training samples*, sehingga nilai *threshold* yang tepat dicari berdasarkan uji coba. Dengan asumsi bahwa *salient object* tidak tampak pada batas citra, *superpixels* di sepanjang batas citra diberi label *training samples* negatif.



(a) (b) (c)

Gambar 3.3 Perbedaan Hasil *Saliency Map*, (a) Citra Masukan, (b) *Gaussian Filter*, (c) *Guided Filter*

3.1.3.3 Analisis Permasalahan Ekstraksi Fitur

Ekstraksi fitur perlu dilakukan dengan baik, karena data fitur yang sesuai dan representatif terhadap *salient object* akan menghasilkan hasil segmentasi yang akurat. Banyak fitur seperti warna, tekstur dan lokasi diterapkan untuk menentukan nilai *saliency*. Dalam metode *Saliency Extreme Learning Machine*, digunakan dua jenis fitur yaitu warna dan tekstur untuk mendeskripsikan area *salient object* dan area non-*salient object.* Pada fitur warna digunakan RGB dan CIELAB, sedangkan pada fitur tekstur digunakan *Uniform Local Binary Pattern*. Fitur warna RGB dipilih untuk mendeskripsikan model warna merah, hijau dan biru, sedangkan fitur warna CIELAB dipilih untuk merepresentasikan ruang warna L*a*b* yang secara statistik berkorelasi dengan persepsi manusia mengenai warna. Fitur tekstur *Uniform Local Binary Pattern* dipilih untuk mendeskripsikan tekstur lokal, mode *uniform* pada *Local Binary Pattern* berguna untuk mengurangi panjang vektor fitur dan bersifat invarian terhadap rotasi.

3.1.3.4 Analisis Permasalahan *Saliency Detection* dengan ELM *Classifier*

Saliency detection dengan ELM *classifier* menghasilkan *trained saliency map* dengan memanfaatkan kemampuan ELM *classifier* yang dilatih menggunakan *training samples*. ELM menggunakan *random feature mapping* atau *kernel* untuk memetakan *samples* yang berdimensi *d* ke *hidden layer feature space* yang berdimensi *L*. ELM memiliki kemampuan untuk melakukan aproksimasi fungsi aktivasi secara universal, namun pemilihan fungsi *feature mapping* atau *kernel* yang tepat dapat meningkatkan akurasi dari *trained saliency map*. Dalam tugas akhir ini, *feature mapping* dilakukan menggunakan *linear kernel*. *Linear kernel* dipilih karena dapat menghasilkan *trained saliency map* dengan akurasi yang baik, selain itu *linear kernel* tidak membutuhkan parameter tambahan dan lebih cepat secara komputasional jika dibandingkan dengan *kernel* atau fungsi *feature mapping* lainnya.

Saat menentukan nilai bobot keluaran antara *hidden layer* dan *output layer*, diperlukan parameter *C* yang ditentukan oleh pengguna, yang menyediakan *tradeoff* antara jarak dari *separating margin* dan *training error*. Nilai *C* yang relatif kecil menyebabkan *classifier* memilih jarak dari *separating margin* dengan kedua kelas relatif besar, meskipun terjadi misklasifikasi pada beberapa data, rentan terjadi *underfitting*. Sedangkan nilai *C* yang relatif besar, menyebabkan *classifier* memilih *separating margin* yang menghasilkan *training error* terkecil, namun jarak dari *separating margin* dengan kedua kelas relatif dekat, rentan terjadi *overfitting*. Parameter *C* mempengaruhi hasil klasifikasi dari ELM, sehingga nilai *C* yang tepat dicari berdasarkan uji coba.

3.1.3.5 Analisis Permasalahan Segmentasi Citra

Object map merupakan citra biner yang didapatkan dari *trained saliency map* yang disegmentasi berdasarkan nilai *threshold*. Nilai *threshold* dihasilkan menggunakan metode Otsu yang sudah cukup baik dalam melakukan segmentasi citra, hal tersebut dibuktikan dengan akurasi yang tinggi dari *object map*. Namun tidak menutup kemungkinan bahwa terdapat nilai *threshold* yang lebih tepat dalam melakukan segmentasi citra. Pemilihan nilai *threshold* yang menghasilkan akurasi *object map* tertinggi dapat dicari menggunakan proses uji coba.

3.2 Tahap Perancangan

Tahap perancangan dilakukan untuk merancang proses secara keseluruhan berdasarkan fungsionalitas dan kebutuhan dari aplikasi segmentasi citra area *salient object* menggunakan metode *Saliency Extreme Learning Machine*.

3.2.1 Perancangan Data

Perancangan data dilakukan untuk memastikan pengoperasian aplikasi berjalan dengan benar. Data masukan (*input*) adalah data yang diperlukan dalam pengoperasian aplikasi

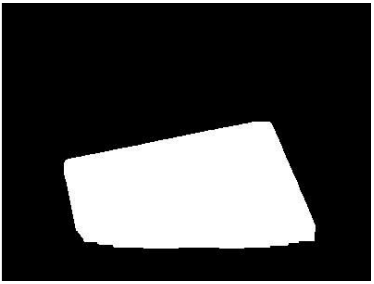
dan data keluaran (*output*) adalah data yang akan digunakan oleh pengguna.

Data masukan untuk aplikasi berupa *natural image* dalam berbagai ukuran yang diperoleh dari *MSRA Salient Object Database*. Citra *ground truth* juga diperoleh dari sumber yang sama. Contoh citra yang digunakan sebagai data masukan ditunjukkan pada **Gambar 3.4**.

(a) (b)

Gambar 3.4 Citra Masukan Berupa (a) *Natural Image* dan (b) *Ground Truth*

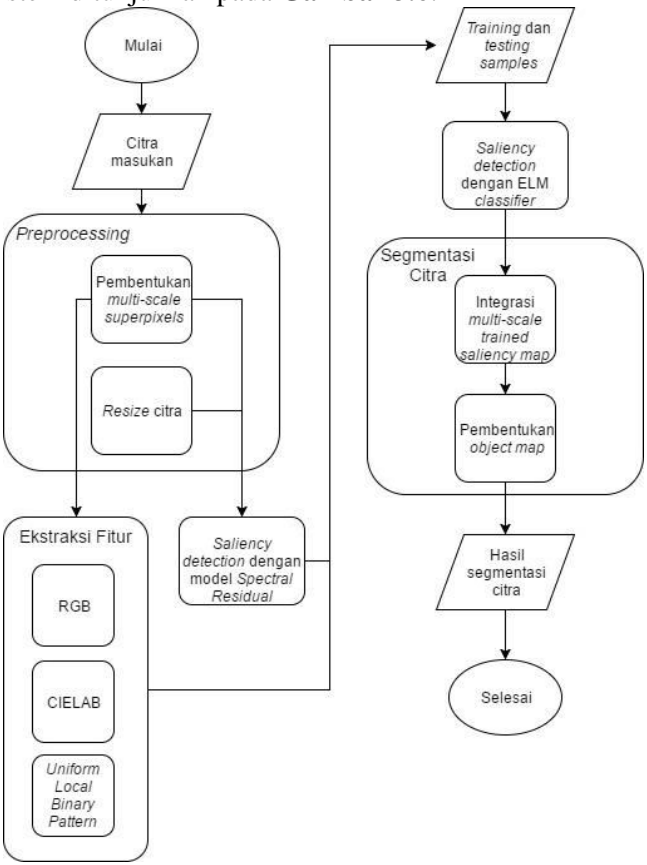
Data keluaran aplikasi segmentasi area *salient object* merupakan *object map* atau citra biner hasil segmentasi dari data masukan. Bagian yang berwarna putih pada citra merupakan area *salient object*, sedangkan bagian yang berwarna hitam merupakan area non-*salient object*. Contoh citra keluaran ditunjukkan pada **Gambar 3.5**.



Gambar 3.5 *Object Map*

3.2.1 Perancangan Sistem

Perancangan sistem dilakukan untuk menggambarkan proses secara keseluruhan dari aplikasi segmentasi citra area *salient object*. Sistem terdiri dari lima proses, proses pertama adalah *preprocessing*, proses kedua adalah *saliency detection* dengan model *Spectral Residual*, proses ketiga adalah ekstraksi fitur, proses keempat adalah *saliency detection* dengan ELM *classifier*, proses kelima adalah segmentasi citra. Diagram alir dari sistem ditunjukkan pada **Gambar 3.6**.



Gambar 3.6 Diagram Alir Keseluruhan Sistem

3.2.1.1 *Preprocessing*

Supaya aplikasi dapat berjalan optimal, diperlukan persiapan data terlebih dahulu pada tahap *preprocessing*. Pada tugas akhir ini, tahap *preprocessing* yang dilakukan terdiri dari proses pembagian citra masukan ke dalam empat skala *superpixels* dan proses pengubahan ukuran citra masukan ke skala yang kecil. Tahap *preprocessing* menghasilkan empat skala *superpixels* dari citra masukan dan citra masukan yang telah di-*resize*.

Pembagian citra masukan ke dalam empat skala *superpixels* bertujuan untuk mendeteksi *salient object* pada berbagai skala yang dapat meningkatkan akurasi dari *saliency map* dan mengurangi kompleksitas saat pengolahan citra. Citra masukan dibagi ke dalam empat skala *superpixels* yang masing-masing berukuran 100, 150, 200 dan 250 piksel. *Prior saliency map* dibentuk dari citra masukan dengan skala yang kecil untuk mendeteksi *proto object* yang menonjol jika dilihat sekilas secara visual. Maka dari itu sebelum *prior saliency map* dibentuk, citra masukan di-*resize* ke skala 64x64 yang terbukti merupakan skala yang baik untuk kondisi visual secara normal.

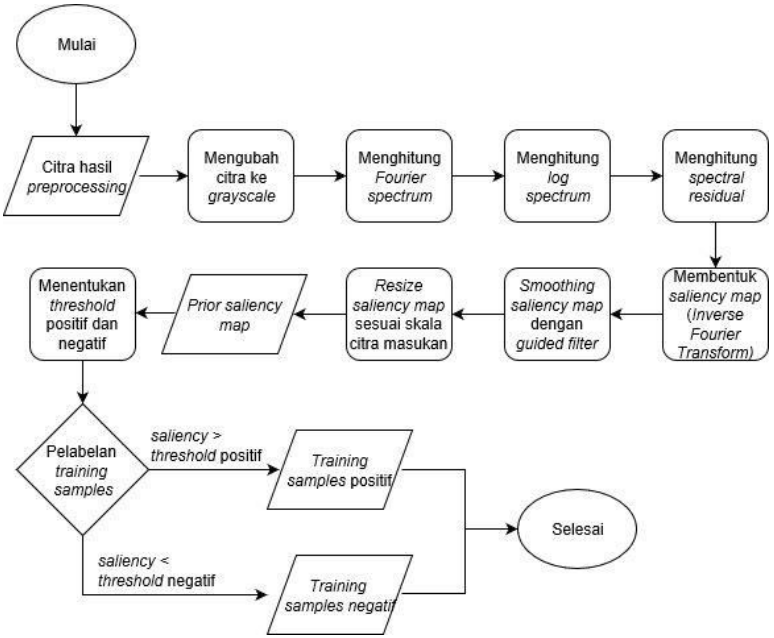
3.2.1.2 *Saliency Detection* dengan Model *Spectral Residual*

Saliency detection dengan pendekatan model *Spectral Residual* adalah metode *bottom-up* yang digunakan dalam tugas akhir ini untuk membuat *prior saliency map.* Data yang digunakan dalam tahap ini adalah data yang dihasilkan dari tahap *preprocessing*. Tahap *saliency detection* dengan pendekatan model *Spectral Residual* menghasilkan *training samples* positif dan negatif untuk setiap skala *superpixels* yang didapatkan dari *prior saliency map* yang dibentuk. Diagram alir proses ditunjukkan pada **Gambar 3.7**.

Pertama, citra hasil *preprocessing* diubah ke bentuk *grayscale*, kemudian dilakukan *Fourier transform* untuk mengubah representasi citra ke dalam domain frekuensi. *Fourier transform* menghasilkan citra keluaran bernilai bilangan kompleks yang dapat ditampilkan dengan dua komponen, yaitu

magnitude spectrum dan *phase spectrum*. *Log spectrum* didapatkan dengan menerapkan transformasi logaritmik pada *magnitude spectrum*. *Spectral residual* dari sebuah citra didapatkan dengan mengurangi *log spectrum* citra tersebut dengan rata-rata *log spectrum*. Umumnya untuk mengambil rata-rata *log spectrum* dibutuhkan banyak citra, namun pada model ini hanya digunakan satu *log spectrum* citra yang dikonvolusi menggunakan *local average filter* atau *mean filter* untuk mendapatkan aproksimasi. *Spectral residual* yang direpresentasikan dalam domain frekuensi diubah ke dalam domain spasial menggunakan *Inverse Fourier transform* untuk mendapatkan *prior saliency map*. *Prior saliency map* di-*smoothing* menggunakan *guided filter* untuk efek visual yang lebih baik. *Prior saliency map* di-*resize* ke skala semula agar dapat digunakan untuk proses selanjutnya.

Setiap piksel dari *prior saliency map* mempunyai nilai *saliency,* sehingga nilai *saliency* dari setiap *superpixel* didapatkan dari rata-rata nilai *saliency* semua piksel yang dikandung *superpixel* tersebut. *Training samples* untuk mempelajari *classifier* mencakup *training samples* positif dari *salient object* dan *training samples* negatif dari *background*. Untuk menentukan *training samples* positif dan negatif digunakan dua *threshold*, yaitu *threshold* positif dan *threshold* negatif. *Superpixels* dengan nilai *saliency* yang lebih besar dari *threshold* positif akan diberi label +1 sebagai *training samples* positif, sebaliknya *superpixels* dengan nilai *saliency* yang lebih kecil dari *threshold* negatif akan diberi label -1 sebagai *training samples* negatif. *Superpixels* yang memiliki nilai *saliency* di antara *threshold* positif dan negatif tidak digunakan sebagai *training samples*. Dengan asumsi bahwa *salient object* biasanya tidak tampak pada batas-batas citra, *superpixels* di sepanjang batas citra diberi label -1 sebagai *training samples* negatif.



Gambar 3.7 Diagram Alir Proses *Saliency Detection* dengan Model *Spectral Residual*

3.2.1.3 Ekstraksi Fitur

Dalam tugas akhir ini, digunakan dua jenis fitur yaitu warna dan tekstur untuk mendeskripsikan area *salient object* dan area non-*salient object.* Pada fitur warna digunakan RGB dan CIELAB, sedangkan pada fitur tekstur digunakan *Uniform Local Binary Pattern* (*Uniform* LBP). Nilai RGB dan CIELAB untuk satu *superpixel* didapatkan dari rata-rata nilai RGB dan CIELAB semua piksel yang dikandung *superpixel* tersebut. *Uniform* LBP menggunakan tetangga berukuran 3x3, yang kemudian setiap pikselnya diberi nilai atau label dengan *range* 0 sampai 58, nilai 58 ditetapkan untuk semua pola non-*uniform*. *Uniform* LBP untuk satu *superpixel* didapatkan dari histogram nilai *Uniform* LBP semua piksel yang dikandung *superpixel* tersebut. Sehingga pada tahap ekstraksi fitur dihasilkan vektor fitur berukuran 65 dimensi

yang terbentuk dari gabungan RGB (3 fitur), CIELAB (3 fitur) dan *Uniform Local Binary Pattern* (59 fitur) untuk setiap *superpixel*.

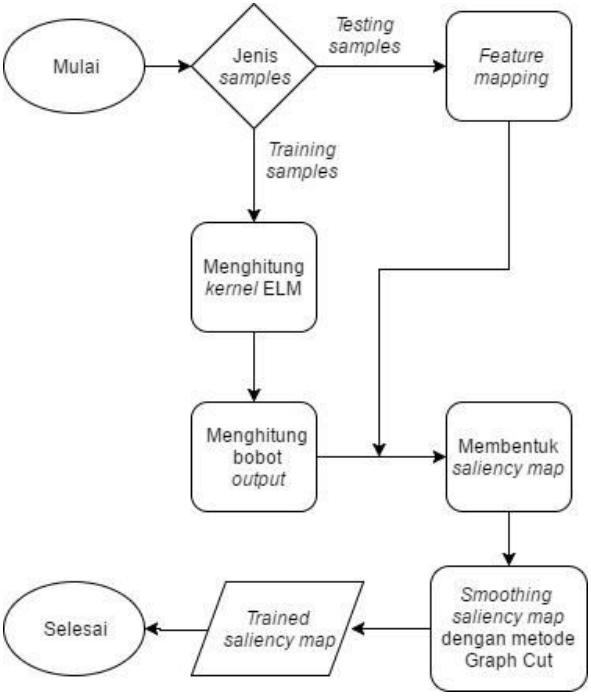
3.2.1.4 *Saliency Detection* dengan ELM *Classifier*

Saliency detection dengan ELM *classifier* adalah metode *saliency detection top-down* yang memanfaatkan ELM *classifier* untuk membuat *trained saliency map.* Data yang digunakan dalam tahap ini adalah *training* dan *testing samples* yang dihasilkan dari tahap *saliency detection* dengan model *Spectral Residual* dan tahap ekstraksi fitur. Matriks label *training samples* berukuran *N*x1 dan matriks fitur berukuran *N*x65 untuk setiap skala *superpixels* digabungkan membentuk matriks *training samples* berukuran *K*x66, di mana *N* adalah jumlah *superpixels* yang terbentuk dari skala 100, 150, 200 atau 250 dan *K* adalah jumlah *superpixels* yang terbentuk dari gabungan seluruh skala. Matriks *testing samples* terdiri dari matriks fitur untuk skala *superpixels* yang akan diujikan. Tahap *saliency detection* dengan ELM *classifier* menghasilkan empat skala *trained saliency map* yang telah di-*smoothing* menggunakan metode *Graph Cut*. Diagram alir proses ditunjukkan pada **Gambar 3.8**.

Tahap *saliency detection* dengan ELM *classifier* pertama kali dapat dibedakan berdasarkan jenis *samples*. *Testing samples* digunakan ketika mencari vektor keluaran dari *hidden layer*, sedangkan *training samples* digunakan ketika mencari vektor bobot keluaran antara *hidden layer* dan *output layer*. *Feature mapping* menggunakan *linear kernel* berfungsi untuk memetakan *testing samples* ke *hidden layer* yang menghasilkan vektor keluaran dari *hidden layer*. *Linear kernel* juga diterapkan dalam menghitung *kernel* ELM dengan memanfaatkan *training samples*. Vektor bobot keluaran terkecil antara *hidden layer* dan *output layer* dihitung dengan memanfaatkan *kernel* ELM dan *pseudoinverse* dari matriks *training samples*.

Fungsi keluaran dari ELM didapatkan dengan mengalikan vektor keluaran dari *hidden layer* dengan vektor bobot keluaran antara *hidden layer* dan *output layer*. Fungsi keluaran ELM

merupakan vektor yang berisi nilai *saliency* dari setiap *superpixel*. *Trained saliency map* dihasilkan dengan memetakan nilai *saliency* dari setiap *superpixel* ke piksel yang dikandungnya. *Smoothing trained saliency map* dilakukan dengan metode *Graph Cut* yang bertujuan untuk mereduksi *noise* dan lebih menonjolkan nilai *saliency* dari *salient object*.



Gambar 3.8 Diagram Alir Proses *Saliency Detection* dengan ELM *Classifier*

3.2.1.5 Segmentasi Citra

Segmentasi citra dilakukan untuk membentuk *object map*. Pertama, dilakukan integrasi empat skala *smoothed trained saliency map* yang merupakan keluaran dari tahap *saliency detection* dengan ELM *classifier*. Integrasi dilakukan dengan

mengambil rata-rata nilai *saliency* setiap piksel dari empat skala yang berbeda. *Object map* dibentuk dari *trained saliency map* hasil integrasi yang disegmentasi berdasarkan nilai *threshold*. Nilai *threshold* yang dihasilkan oleh metode Otsu ditingkatkan nilainya sebanyak 1,5 kali untuk hasil segmentasi yang lebih baik. Hasil akhir dari sistem adalah *object map* atau citra biner yang menandakan area *salient object* dan area non-*salient object* pada citra masukan.

3.2.2 Perancangan Antarmuka Aplikasi

Antarmuka aplikasi segmentasi citra area *salient object* terdiri dari satu jendela yang digunakan, di mana jendela tersebut adalah jendela utama aplikasi. Rancangan antarmuka aplikasi ditunjukkan pada **Gambar 3.9**. Dalam jendela utama aplikasi terdapat empat area proses yang harus dilakukan secara bertahap untuk mendapatkan hasil akhir aplikasi.

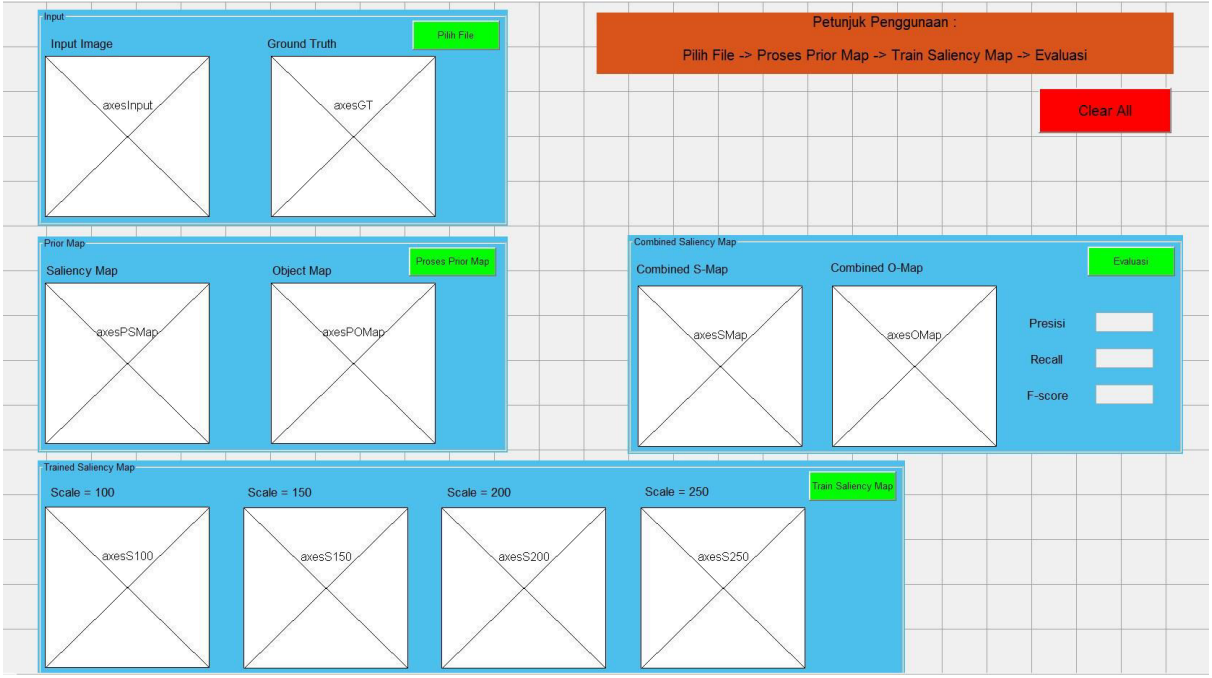
Proses pertama adalah memilih citra masukan yang akan disegmentasi, hal ini dapat dilakukan dengan memilih tombol "Pilih File", jika proses berhasil dilakukan maka area "Input" akan menampilkan citra masukan beserta *ground truth*.

Proses kedua adalah melakukan *preprocessing*, *saliency detection* dengan model *Spectral Residual* dan ekstraksi fitur, hal ini dilakukan dengan memilih tombol "Proses Prior Map", jika proses berhasil dilakukan maka area "Prior Map" akan menampilkan *prior saliency map* dan *object map* yang dibuat berdasarkan *prior saliency map*.

Proses ketiga adalah melakukan *saliency detection* dengan ELM *classifier*, hal ini dilakukan dengan memilih tombol "Train Saliency Map", jika proses berhasil dilakukan maka area "Trained Saliency Map" akan menampilkan *object map* dari masing-masing skala *trained saliency map*.

Proses terakhir adalah melakukan segmentasi citra dan melakukan evaluasi terhadap hasil segmentasi, hal ini dilakukan dengan memilih tombol "Evaluasi", jika proses berhasil dilakukan maka area "Combined Saliency Map" akan menampilkan *trained saliency map* hasil integrasi empat skala

beserta *object map* yang merupakan hasil segmentasi citra masukan. Selain itu, ditampilkan pula nilai presisi, *recall* dan *F1 score* sebagai hasil evaluasi metode *Saliency Extreme Learning Machine* dalam melakukan segmentasi citra area *salient object*.



Gambar 3.9 Rancangan Antarmuka Aplikasi

BAB IV
IMPLEMENTASI

Pada bab ini diuraikan mengenai implementasi perangkat lunak dari rancangan metode yang telah dibahas pada Bab III meliputi kode program dalam perangkat lunak. Selain itu, implementasi dari tiap proses, parameter masukan, keluaran, dan beberapa keterangan yang berhubungan dengan program juga dijelaskan.

4.1 Lingkungan Implementasi

Dalam implementasi segmentasi citra area *salient object* menggunakan metode *Saliency Extreme Learning Machine*, digunakan perangkat-perangkat yang dijelaskan pada sub-bab berikut ini.

4.1.1 Perangkat Keras

Lingkungan implementasi pada tugas akhir ini adalah sebuah *personal computer* (PC). Perangkat PC yang digunakan adalah DELL Inspiron 7537 dengan sistem operasi Windows 10 Home Single Language 64-bit. Spesifikasi dari PC yang digunakan pada tugas akhir ini memiliki prosesor Intel Core i7-4500U dengan kecepatan 1,8 GHz dan *Random Access Memory* (RAM) untuk proses menjalankan program sebesar 8,00 GB.

4.1.2 Perangkat Lunak

Spesifikasi PC dari sisi perangkat lunak menggunakan bahasa pemrograman MATLAB dengan kakas bantu IDE MATLAB 9.1 (R2016b) pada platform *desktop*. Penggunaan MATLAB didukung dengan empat *toolbox* utama yaitu *image processing toolbox, neural network toolbox, computer vision system toolbox* dan *statistics and machine learning toolbox*. Selain itu, digunakan alat bernama GUIDE untuk membuat GUI (*Graphical User Interface*) aplikasi pada MATLAB. Kakas bantu pendukung lain di antaranya *draw.io* untuk dokumentasi dan *Microsoft Excel* sebagai pengolah angka.

4.2 Implementasi Tahap *Preprocessing*

Tahap *preprocessing* pada tugas akhir ini terdiri dari dua tahap yang terpisah, yaitu pengubahan ukuran citra masukan ke skala yang kecil dan pembagian citra masukan ke dalam empat skala *superpixels*. Penjelasan dari masing-masing tahap tersebut terdapat pada sub-bab berikutnya.

4.2.1 Implementasi *Resize* Citra

Tahap ini mengubah ukuran citra masukan ke skala 64x64 sebagai data masukan untuk membuat *prior saliency map*. Implementasi dari *resize* citra ditunjukkan pada **Kode Sumber 4.1**. Pada baris 1 dilakukan pembacaan citra dari direktori file. Pada baris 3 citra di-*resize* ke skala 64x64 menggunakan mode *bilinear* (nilai piksel keluaran menggunakan nilai rata-rata dari piksel tetangga berukuran 2x2).

```
1 I=imread('path/ke/direktori/citra/dan/nama_file.jpg');
2 [rows, cols, depth] = size(I);
3 Irez = imresize(I,[64,64],'bilinear');
```

Kode Sumber 4.1 Implementasi *Resize* Citra

4.2.2 Implementasi *Multi-Scale Superpixels*

Pembagian citra masukan ke dalam empat skala *superpixels* bertujuan untuk mendeteksi *salient object* pada berbagai skala. Citra masukan dibagi ke dalam empat skala *superpixels* yang masing-masing berukuran 100, 150, 200 dan 250 piksel. Meskipun termasuk dalam tahap preprocessing, proses ini dilakukan beberapa kali dalam aplikasi untuk memproses informasi dalam setiap *superpixels*. Implementasi dari pembentukan *multi-scale superpixels* ditunjukkan pada **Kode Sumber 4.2**.

Pada baris 1 dilakukan inisialisasi *array* yang berisi empat skala *superpixels* yang digunakan. Pada baris 2 dan seterusnya dilakukan perulangan sebanyak empat kali untuk membentuk dan memproses *superpixels* dalam empat skala. Pembentukan

superpixels dilakukan pada baris 3, di mana *L* adalah matriks label *superpixel* untuk setiap piksel dan *N* adalah jumlah *superpixels* yang dihasilkan. Matriks label dikonversi ke *cell* pada baris 4, setiap *cell* merupakan label *superpixel* yang mengandung matriks berisi piksel anggotanya. Pada baris 5 dan seterusnya dilakukan perulangan sebanyak jumlah *superpixels* yang dihasilkan untuk memproses informasi seperti fitur atau nilai *saliency* yang dikandung oleh seluruh piksel dalam *superpixel* yang bersangkutan.

```
1  scales = [100,150,200,250];
2  for i = 1:4
3    [L,N] = superpixels(I,scales(i));
4    idx = label2idx(L);
5    for labelVal = 1:N
6      firstIdx = idx{labelVal};
7      ...
```

Kode Sumber 4.2 Implementasi *Multi-Scale Superpixels*

4.3 Implementasi *Saliency Detection* dengan Model *Spectral Residual*

Tahap *saliency detection* dengan model *Spectral Residual* pada tugas akhir ini terdiri dari dua tahap utama. Tahap pertama adalah membentuk *prior saliency map* dengan pendekatan *Spectral Residual*, kemudian membentuk *training samples* positif dan negatif untuk setiap skala *superpixels* dengan memanfaatkan *prior saliency map* yang dibentuk. Penjelasan dari masing-masing tahap tersebut terdapat pada sub-bab berikutnya.

4.3.1 Implementasi Pembentukan *Prior Saliency Map*

Pembentukan *prior saliency map* dilakukan menggunakan model *saliency detection* secara *bottom-up* yaitu model *Spectral Residual*. Citra yang digunakan sebagai data masukan adalah citra masukan yang telah di-*resize* pada tahap *preprocessing*. Implementasi dari pembentukan *prior saliency map* ditunjukkan pada **Kode Sumber 4.3**.

Pada baris 1, citra masukan yang telah di-*resize* diubah ke bentuk *grayscale*, kemudian dilakukan *Fourier transform* pada baris 2 untuk mengubah representasi citra ke dalam domain frekuensi. Pada baris 3 dan 4 dilakukan ekstraksi *magnitude spectrum* dan *phase spectrum* dari citra *Fourier*. Pada baris 5, *log spectrum* didapatkan dengan menerapkan transformasi logaritmik pada *magnitude spectrum*. Baris 6-8 merupakan *log spectrum* citra yang dikonvolusi menggunakan *local average filter* atau *mean filter* untuk mendapatkan aproksimasi. Pada baris 9, *spectral residual* didapatkan dengan mengurangi *log spectrum* citra tersebut dengan *log spectrum* yang dikonvolusi. *Spectral residual* yang direpresentasikan dalam domain frekuensi diubah ke dalam domain spasial menggunakan *Inverse Fourier transform* pada baris 10 untuk mendapatkan *prior saliency map*. *Prior saliency map* di-*smoothing* menggunakan *guided filter* pada baris 11 untuk efek visual yang lebih baik. *Prior saliency map* di-*resize* ke skala semula pada baris 13 agar dapat digunakan untuk tahap pelabelan *training samples*.

1	`IG = im2double(rgb2gray(I));`
2	`FT = fft2(IG);`
3	`Magnitude = abs(FT);`
4	`Phase = angle(FT);`
5	`logMagnitude = log(Magnitude);`
6	`f = ones(3);`
7	`favg = (1/3^2)*f;`
8	`smoothedLogMagnitude = imfilter(logMagnitude,favg,'replicate');`
9	`spectralResidual = logMagnitude - smoothedLogMagnitude;`
10	`saliencyMap = abs(ifft2(exp(spectralResidual + 1i*Phase))).^2;`
11	`saliencyMapGuidedFilter = imguidedfilter(saliencyMap);`
12	`priorSMap = mat2gray(saliencyMapGuidedFilter);`

```
    priorSMap = imresize(priorSMap,[rows,
13  cols],'bilinear');
```

Kode Sumber 4.3 Implementasi Pembentukan *Prior Saliency Map*

4.3.2 Implementasi Pelabelan *Training Samples*

Prior saliency map yang dihasilkan oleh *saliency detection* dengan model *Spectral Residual* dimanfaatkan sebagai solusi untuk menghindari proses *labeling training samples* secara manual. Implementasi dari pelabelan *training samples* ditunjukkan pada **Kode Sumber 4.4**.

Baris 1-7 mendefinisikan piksel yang termasuk dalam batas-batas citra. Pada baris 8 dan 9 ditentukan nilai *threshold* positif dan negatif untuk menentukan label *training samples*. Baris 10 dan seterusnya menggunakan tahap iterasi *multi-scale superpixels* untuk mengolah informasi nilai *saliency* dalam *prior saliency map*. Pada baris 11, dilakukan inisialisasi *array* label *training samples* berukuran sesuai dengan jumlah *superpixels* yang dihasilkan. Baris 13 adalah inisialisasi indeks piksel yang dikandung dalam *superpixel*. Pada baris 14, nilai *saliency superpixel* diambil dari rata-rata nilai *saliency* semua piksel yang dikandung *superpixel* tersebut. Pada baris 15-17, *superpixel* yang mengandung piksel yang terletak pada batas citra diberi label -1 sebagai *training samples* negatif. Baris 19-22 menentukan label *training samples*, *superpixel* dengan nilai *saliency* yang lebih besar dari *threshold* positif diberi label +1 sebagai *training samples* positif, sebaliknya *superpixel* dengan nilai *saliency* yang lebih kecil dari *threshold* negatif diberi label -1 sebagai *training samples* negatif.

```
1  jmlPiksel = rows * cols;
2  lastColumn = jmlPiksel - rows + 1;
3  leftBoundary = linspace(1,rows,rows);
4  rightBoundary =
   linspace(lastColumn,jmlPiksel,rows);
5  topBoundary = 1:rows:lastColumn;
6  botBoundary = rows:rows:jmlPiksel;
```

```
7   boundary = [leftBoundary,rightBoundary,topBoundary,botBoundary];
8   thresPositif = mean(priorSMap(:))*1.5;
9   thresNegatif = 0.05;
10  ...
11    trainingLabel = zeros(N,1);
12    for labelVal = 1:N
13      firstIdx = idx{labelVal};
14      meanSPixel = mean(priorSMap(firstIdx));
15      isBoundary = ismember(firstIdx,boundary);
16      if any(isBoundary > 0)
17        trainingLabel(labelVal) = -1;
18      else
19        if(meanSPixel > thresPositif)
20          trainingLabel(labelVal) = 1;
21        if(meanSPixel < thresNegatif)
22          trainingLabel(labelVal) = -1;
```

Kode Sumber 4.4 Implementasi Pelabelan *Training Samples*

4.4 Implementasi Ekstraksi Fitur

Tahap ekstraksi fitur pada tugas akhir ini terdiri dari dua tahap yang terpisah, yaitu ekstraksi fitur warna RGB dan CIELAB, serta ekstraksi fitur teksktur *Uniform Local Binary Pattern* (*Uniform* LBP). Pada tahap ekstraksi fitur dihasilkan vektor fitur berukuran 65 dimensi yang terbentuk dari gabungan RGB (3 fitur), CIELAB (3 fitur) dan *Uniform* LBP (59 fitur). Penjelasan dari masing-masing tahap tersebut terdapat pada sub-bab berikutnya.

4.4.1 Implementasi Ekstraksi Fitur Warna RGB dan CIELAB

Ekstraksi fitur warna RGB mengambil informasi *channel red, green* dan *blue* dari setiap *superpixel*. Sedangkan ekstraksi fitur warna CIELAB mengambil informasi L*a*b* dari setiap

superpixel. Implementasi dari ekstraksi fitur warna RGB ditunjukkan pada **Kode Sumber 4.5**.

Pada baris 1, dilakukan konversi nilai RGB ke L*a*b* pada setiap piksel citra masukan. Baris 2 dan seterusnya menggunakan tahap iterasi *multi-scale superpixels* untuk mengambil informasi fitur warna dari citra masukan. Pada baris 3, dilakukan inisialisasi matriks fitur berukuran *N*x65, *N* adalah jumlah *superpixels* yang dihasilkan, 65 adalah jumlah fitur yang akan diekstraksi. Baris 7-9 merupakan inisialisasi indeks piksel yang dikandung dalam *superpixel*, dibutuhkan tiga indeks piksel yang berbeda untuk citra yang memiliki tiga dimensi. Pada baris 10-15, masing-masing nilai fitur warna *red*, *green, blue* dan L*a*b* milik *superpixel* diambil dari rata-rata nilai fitur warna semua piksel yang dikandung *superpixel* tersebut.

```
1  ILab = rgb2lab(I);
2  ...
3    featureVector = zeros(N,65);
4    numRows = size(I,1);
5    numCols = size(I,2);
6    for labelVal = 1:N
7      firstIdx = idx{labelVal};
8      secondIdx =
       idx{labelVal}+numRows*numCols;
9      thirdIdx =
       idx{labelVal}+2*numRows*numCols;
10     featureVector(labelVal,60) =
       mean(I(firstIdx));
11     featureVector(labelVal,61) =
       mean(I(secondIdx));
12     featureVector(labelVal,62) =
       mean(I(thirdIdx));
13     featureVector(labelVal,63) =
       mean(ILab(firstIdx));
14     featureVector(labelVal,64) =
       mean(ILab(secondIdx));
15     featureVector(labelVal,65) =
```

```
    mean(ILab(thirdIdx));
```

Kode Sumber 4.5 Implementasi Ekstraksi Fitur Warna RGB dan CIELAB

4.4.2 Implementasi Ekstraksi Fitur Tekstur *Uniform Local Binary Pattern*

Ekstraksi fitur tekstur *Uniform Local Binary Pattern* (*Uniform* LBP) mengambil informasi tekstur lokal dari setiap *superpixel*. *Uniform* LBP menggunakan tetangga berukuran 3x3, yang kemudian diberi nilai atau label dengan *range* 0 sampai 58, nilai 58 ditetapkan untuk semua pola *non-uniform*. Implementasi ekstraksi fitur tekstur *Uniform* LBP ditunjukkan pada **Kode Sumber 4.6**.

Pada baris 1, dilakukan inisialisasi *array* yang berfungsi sebagai tabel *mapping* yang memetakan semua kemungkinan kombinasi pola ke salah satu dari 58 pola *uniform*. Parameter jumlah tetangga yang digunakan adalah 8, maka kemungkinan nilai LBP yang dihasilkan memiliki 256 pola yang akan direduksi menjadi 58 pola *uniform* dan 1 pola *non-uniform*. Baris 2 merupakan inisialisasi citra masukan yang diubah ke bentuk *grayscale* dan bertipe *double*. Baris 3 merupakan inisialisasi matriks berukuran sama dengan citra masukan yang menampung nilai *Uniform* LBP setiap piksel. Pada baris 4-25, dilakukan iterasi sebanyak jumlah piksel pada citra masukan, dimulai dari indeks kedua karena piksel pada batas citra tidak digunakan. Pada baris 10-17, dilakukan iterasi untuk mendapatkan pola biner piksel terpilih, jika nilai *grayscale* piksel tetangga lebih besar dari atau sama dengan nilai *grayscale* piksel terpilih, maka nilai biner untuk piksel tetangga tersebut bernilai 1, untuk kondisi sebaliknya, nilai biner untuk piksel tetangga bernilai 0. Pada baris 18-20, jika pola biner piksel terpilih mengandung lebih dari dua transisi *bitwise* dari 0 ke 1 atau sebaliknya, maka piksel terpilih diberi label bernilai 58 sebagai pola *non-uniform*. Pada baris 22-24, dilakukan iterasi sebanyak jumlah tetangga untuk mendapatkan nilai LBP piksel terpilih. Pemetaan nilai LBP piksel

terpilih ke salah satu dari 58 pola *uniform* (label bernilai 0-57) dilakukan pada baris 25 menggunakan bantuan tabel *mapping* yang telah diinisialisasi sebelumnya. Baris 27 dan seterusnya menggunakan tahap iterasi *multi-scale superpixels* untuk mengambil informasi fitur teksktur dari matriks yang menampung nilai *Uniform* LBP. Pada baris 28 dan 29, nilai fitur tekstur *superpixel* diambil dari jumlah setiap pola *uniform* dan pola *non-uniform* yang dimiliki seluruh piksel dalam *superpixel*.

```
1  map = getmapping(8,'u2');
2  IG_d = double(IG);
3  IG_out = double(zeros(rows,cols));
4  for i=2:1:rows-1
5    for j=2:1:cols-1
6      LBP_value = 0;
7      polabiner = zeros(1,8);
8      selected_pixel = IG_d(i,j);
9      index = 1;
10     for k=1:3
11       for l=1:3
12         if (k == 2) && (l == 2)
13           continue;
14         else
15           if (IG_d(i+k-2,j+l-2) >=
           selected_pixel)
16             polabiner(index) = 1;
17           index = index + 1;
18     if (nnz(diff(polabiner([1:end, 1]))) >
       2)
19       IG_out(i,j) = 58;
20       continue;
21     pangkat = 0;
22     for x=1:8
23       LBP_value = LBP_value +
         (2^pangkat)*polabiner(x);
24       pangkat = pangkat + 1;
25     IG_out(i,j) = map(LBP_value + 1);
```

```
26  uniLBP = uint8(IG_out);
27  ...
28      [pixelCounts, grayLevels] =
        imhist(uniLBP(firstIdx));
29      featureVector(labelVal,1:59) =
        pixelCounts(1:59);
```

Kode Sumber 4.6 Implementasi Ekstraksi Fitur Tekstur *Uniform Local Binary Pattern*

4.5 Implementasi *Saliency Detection* dengan ELM *Classifier*

Tahap *saliency detection* dengan ELM *classifier* terdiri dari tiga tahap utama. Tahap pertama adalah menghitung bobot keluaran antara *hidden layer* dan *output layer*, kemudian tahap kedua adalah melakukan *feature mapping* untuk memetakan *testing samples* ke *hidden layer*, dan tahap terakhir adalah membentuk *trained saliency map*. Penjelasan dari masing-masing tahap tersebut terdapat pada sub-bab berikutnya.

4.5.1 Implementasi Penghitungan Bobot Keluaran

Data yang digunakan dalam tahap ini adalah *training samples* yang dihasilkan dari tahap *saliency detection* dengan model *Spectral Residual* dan tahap ekstraksi fitur. Vektor bobot keluaran terkecil antara *hidden layer* dan *output layer* dihitung dengan memanfaatkan *kernel* ELM dan *pseudoinverse* dari matriks *training samples*. Implementasi dari penghitungan bobot keluaran ditunjukkan pada **Kode Sumber 4.7**.

Pada baris 1 dan 2, dilakukan inisialisasi *cell* yang menampung matriks fitur yang telah dinormalisasi (*testing samples*) dan *training samples* untuk setiap skala *superpixels*. Baris 3 hingga 7 menggunakan tahap iterasi *multi-scale superpixels* untuk mengambil *testing samples* dan *training samples*. Pada baris 4, matriks fitur yang dihasilkan pada tahap ekstraksi fitur dinormalisasi dalam *range* [-1, 1] dan disimpan sebagai *testing samples*. Pada baris 5, *training samples* dibentuk dari gabungan vektor fitur yang telah dinormalisasi dan *array*

label *training samples* (dihasilkan pada tahap *saliency detection* dengan model *Spectral Residual*). Pada baris 6 dan 7, *training samples* yang memiliki label *training samples* bernilai 0 tidak digunakan sebagai *training samples*. *Training samples* dari keempat skala *superpixels* digabung menjadi satu pada baris 8. *Kernel* ELM dihitung pada baris 13 dengan mengalikan matriks *training samples* dengan *transpose* dari matriks *training samples*. Parameter *C* yang menyediakan *tradeoff* antara jarak dari *separating margin* dan *training error* diinisialisasi dengan nilai 1 pada baris 14. Bobot keluaran antara *hidden layer* dan *output layer* dihitung pada baris 16. Bobot keluaran tersebut digunakan untuk tahap pembentukan *trained saliency map*.

```
1  normFV = cell(4,1);
2  trainingSample = cell(4,1);
3  ...
4     normFV{i} = -1 + 2.*(featureVector -
      min(featureVector))./(max(featureVector
      ) - min(featureVector));
5     trainingSample{i} = [normFV{i},
      trainingLabel];
6     unused = trainingSample{i}(:,66) == 0;
7     trainingSample{i}(unused,:) = [];
8  tSample = cat(1,trainingSample{:});
9  fVector = tSample(:,(1:65));
10 tLabel = tSample(:,66);
11 jmlSample = size(tSample,1);
12 trainingSTransposed = tSample.';
13 kernel = tSample * trainingSTransposed;
14 C = 1;
15 identity = eye(jmlSample);
16 outputWeight = (kernel + identity/C) \
   tLabel;
```

Kode Sumber 4.7 Implementasi Penghitungan Bobot Keluaran

4.5.2 Implementasi *Feature Mapping*

Data *testing samples* digunakan ketika mencari vektor keluaran dari *hidden layer*. *Feature mapping* menggunakan *linear kernel* berfungsi untuk memetakan *testing samples* yang berdimensi *d* ke *hidden layer feature space* yang berdimensi *L* secara linier. Implementasi dari *feature mapping* ditunjukkan pada **Kode Sumber 4.8**.

Setiap skala *superpixels* memiliki *testing samples*-nya masing-masing, oleh sebab itu dilakukan iterasi sesuai dengan jumlah skala *superpixels* yang digunakan. *Feature mapping* menggunakan *linear kernel* dilakukan pada baris 3 dengan mengalikan matriks *testing samples* dengan *transpose* dari matriks *testing samples*. Vektor keluaran dari *hidden layer* setiap skala digunakan untuk tahap pembentukan *trained saliency map*.

```
1  for i = 1:4
2    testingSampleTransposed = normFV{i}.';
3    hiddenLayer = fVector *
     testingSampleTransposed;
```

Kode Sumber 4.8 Implementasi *Feature Mapping*

4.5.3 Implementasi Pembentukan *Trained Saliency Map*

Trained saliency map merupakan fungsi keluaran dari ELM yang diproses dengan memetakan nilai *saliency* dari setiap *superpixel* ke piksel yang dikandungnya. *Smoothing trained saliency map* dilakukan dengan metode *Graph Cut* yang bertujuan untuk mereduksi *noise* dan lebih menonjolkan nilai *saliency* dari *salient object*. Implementasi dari pembentukan *trained saliency map* ditunjukkan pada **Kode Sumber 4.9**.

Pada baris 1, dilakukan inisialisasi *cell* yang menampung *trained saliency map* untuk setiap skala *superpixels*. Baris 2 dan seterusnya menggunakan tahap iterasi *multi-scale superpixels* untuk membentuk keempat skala *trained saliency map*. Fungsi keluaran dari ELM pada baris 3 didapatkan dengan mengalikan

vektor keluaran dari *hidden layer* skala *superpixels* yang sedang diproses dengan vektor bobot keluaran antara *hidden layer* dan *output layer*. Fungsi keluaran ELM merupakan vektor yang berisi nilai *saliency* dari setiap *superpixel*. Pada baris 4-9, dilakukan pembentukan *trained saliency map* dengan memetakan nilai *saliency* dari setiap *superpixel* ke piksel yang dikandungnya. Pada baris 10 dan 11, *trained saliency map* dinormalisasi dalam *range* [0, 1].

Baris 12-20 merupakan proses *smoothing trained saliency map* menggunakan metode *Graph Cut*. Pada baris 12-14, dilakukan inisialisasi *data cost* setiap piksel yang diambil dari nilai *saliency* pada *trained saliency map* yang telah dinormalisasi. Setiap piksel memiliki dua *data cost* masing-masing untuk label *foreground* (*salient object*) dan *background* (*non-salienct object*). Pada baris 15 dan 16, dilakukan inisialisi *smoothness cost* yang merupakan biaya untuk menentukan label dari piksel tetangga. Piksel tetangga yang dimaksud adalah piksel yang berada di bawah dan di kanan piksel terpilih. *Sc* bersifat *spatialy invariant* dan tidak dapat mendeteksi detil dari citra, oleh karena itu digunakan bersamaan dengan *Hc* dan *Vc* yang bersifat *spatialy variant* untuk mendeteksi detil seperti tepian citra. *Hc* dan *Vc* merupakan matriks yang menampung hasil konvolusi antara citra masukan dengan filter khusus yang merupakan gabungan dari *Gaussian filter* dan *sobel filter*. Pada baris 17-19, dibentuk *graph* dengan parameter yang sudah diinisialisasi sebelumnya dan dilakukan *Graph Cut* pada *graph* tersebut, menghasilkan *labels* atau *binary map* yang mengandung label *foreground* atau *background* untuk setiap piksel. *Binary map* tersebut digabungkan dengan *trained saliency map* pada baris 20 untuk menghasilkan *smoothed trained saliency map*.

```
1  smoothedMap = cell(4,1);
2  for i = 1:4
3     saliencyMapSPixel = hiddenLayer.' *
      outputWeight;
4     saliencyMapPixel = zeros(rows,cols);
```

```
5    [L,N] = superpixels(I,scales(i));
6    idx = label2idx(L);
7    for labelVal = 1:N
8      firstIdx = idx{labelVal};
9      saliencyMapPixel(firstIdx) =
       saliencyMapSPixel(labelVal);
10   normSMap = saliencyMapPixel -
     min(saliencyMapPixel(:));
11   normSMap = normSMap ./
     max(normSMap(:));
12   Dc = zeros([rows,cols,2],'single');
13   Dc(:,:,1) = normSMap;
14   Dc(:,:,2) = 1 - normSMap;
15   Sc = ones(2) - eye(2);
16   [Hc,Vc] = SpatialCues(im2double(I));
17   gch = GraphCut('open', Dc, 10*Sc, exp(-
     Vc*5), exp(-Hc*5));
18   [gch,labels] = GraphCut('expand',gch);
19   gch = GraphCut('close', gch);
20   smoothedMap{i} = (saliencyMapPixel +
     double(labels)) / 2;
```

Kode Sumber 4.9 Implementasi Pembentukan *Trained Saliency Map*

4.6 Implementasi Segmentasi Citra

Tahap segmentasi citra hanya terdiri dari satu tahap, yang bertujuan untuk membentuk *object map*. *Object map* dibentuk dari *trained saliency map* hasil integrasi yang disegmentasi menggunakan metode Otsu. Hasil akhir dari sistem adalah *object map* atau citra biner yang menandakan area *salient object* dan area non-*salient object* pada citra masukan. Implementasi dari segmentasi citra ditunjukkan pada **Kode Sumber 4.10**.

Pada baris 1 dan 2, dilakukan integrasi empat skala *smoothed trained saliency map* yang merupakan keluaran dari tahap *saliency detection* dengan ELM *classifier.* Integrasi dilakukan dengan mengambil rata-rata nilai *saliency* setiap piksel dari empat skala yang berbeda. Nilai *threshold* untuk segmentasi citra dihitung pada baris 3 dengan menggunakan metode Otsu

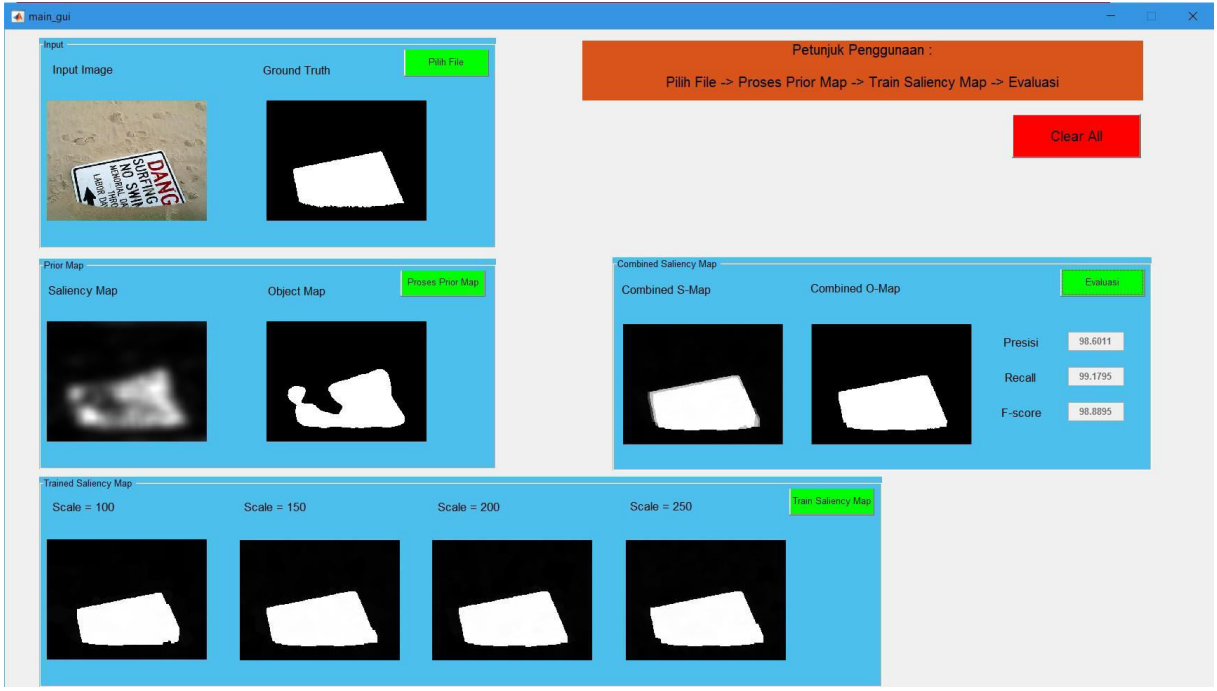
yang ditingkatkan nilainya sebanyak 1,5 kali untuk hasil segmentasi yang lebih baik. Pada baris 4-8, dilakukan pembentukan *object map* dari *trained saliency map* hasil integrasi yang disegmentasi berdasarkan nilai *threshold*.

```
1  totalSaliency = cat(4,smoothedMap{:});
2  meanSMap = mean(totalSaliency,4);
3  thresholdS = graythresh(meanSMap)*1.5;
4  objectMap = zeros(rows,cols);
5  for i=1:rows
6     for j=1:cols
7        if (meanSMap(i,j) > thresholdS)
8           objectMap(i,j) = 1;
```

Kode Sumber 4.10 Implementasi Segmentasi Citra

4.7 Implementasi Antarmuka Aplikasi

Implementasi antarmuka aplikasi segmentasi citra area *salient object* dilakukan dengan membuat jendela utama aplikasi. Dalam jendela utama aplikasi tersebut terdapat empat area proses yang harus dilakukan secara bertahap untuk mendapatkan hasil akhir aplikasi. Proses pertama adalah memilih citra masukan yang akan disegmentasi. Proses kedua adalah melakukan *preprocessing*, *saliency detection* dengan model *Spectral Residual* dan ekstraksi fitur. Proses ketiga adalah melakukan *saliency detection* dengan ELM *classifier*. Proses terakhir adalah melakukan segmentasi citra dan melakukan evaluasi terhadap hasil segmentasi. Hasil implementasi antarmuka aplikasi ditunjukkan pada **Gambar 4.1**.



Gambar 4.1 Hasil Implementasi Antarmuka Aplikasi

BAB V
UJI COBA DAN EVALUASI

Dalam bab ini dibahas mengenai hasil uji coba sistem yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan.

5.1 Lingkungan Uji Coba

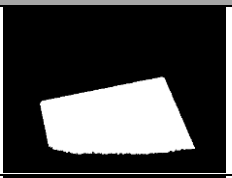
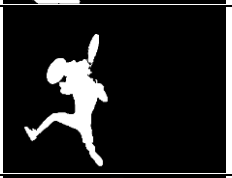
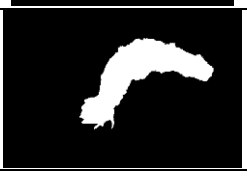
Lingkungan uji coba pada tugas akhir ini adalah sebuah *personal computer* (PC). Perangkat PC yang digunakan adalah DELL Inspiron 7537 dengan sistem operasi Windows 10 Home Single Language 64-bit. Spesifikasi PC dari sisi perangkat keras adalah memiliki prosesor Intel Core i7-4500U dengan kecepatan 1,8 GHz dan *Random Access Memory* (RAM) untuk proses menjalankan program sebesar 8,00 GB.

Pada sisi perangkat lunak, uji coba pada tugas akhir ini dilakukan dengan menggunakan *software* MATLAB 9.1 (R2016b). Penggunaan MATLAB didukung dengan empat *toolbox* utama yaitu *image processing toolbox, neural network toolbox, computer vision system toolbox* dan *statistics and machine learning toolbox*. Dokumentasi hasil uji coba dilakukan dengan menggunakan *Microsoft Excel.*

5.2 Data Uji Coba

Data uji coba yang digunakan sebagai masukan adalah *natural images* dalam berbagai ukuran yang diperoleh dari *MSRA Salient Object Database*. Pada setiap skenario uji coba, digunakan 50 buah *natural images* yang diambil secara acak. Untuk menguji kebenaran dari hasil segmentasi, digunakan data *ground truth* berupa citra hitam putih. Data *ground truth* diperoleh dari sumber yang sama dengan citra masukan. Beberapa *natural images* beserta *ground truth* yang digunakan dalam uji coba ditunjukkan pada **Tabel 5.1**.

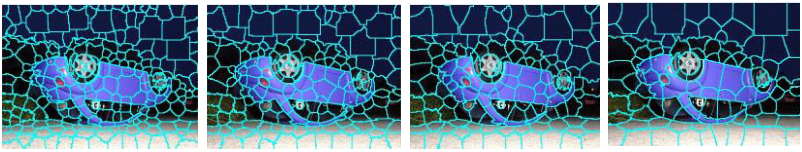
Tabel 5.1 Beberapa Data Uji Coba yang Digunakan

Citra Asli	*Ground Truth*
	
	
	
	
	

5.3 Hasil Uji Coba

Hasil uji coba setiap proses dalam aplikasi akan dijelaskan sebagai berikut. Proses pertama yaitu *preprocessing,* yang terdiri dari dua sub-proses yaitu proses pembentukan *multi-scale superpixels* dan proses *resize* citra. Citra masukan dibagi ke dalam empat skala *superpixels* yang masing-masing berukuran 100, 150, 200 dan 250 piksel. Hasil proses pembentukan *multi-*

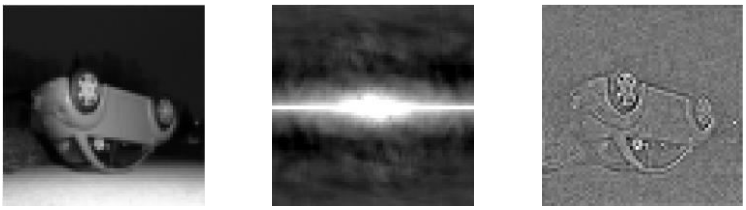
scale superpixels ditunjukkan pada **Gambar 5.1**. Citra masukan di-*resize* ke skala 64x64 sebagai data masukan untuk proses kedua.



(a) (b) (c) (d)

Gambar 5.1 Hasil Pembentukan *Multi-Scale Superpixels*, (a) Skala 100, (b) Skala 150, (c) Skala 200, (d) Skala 250

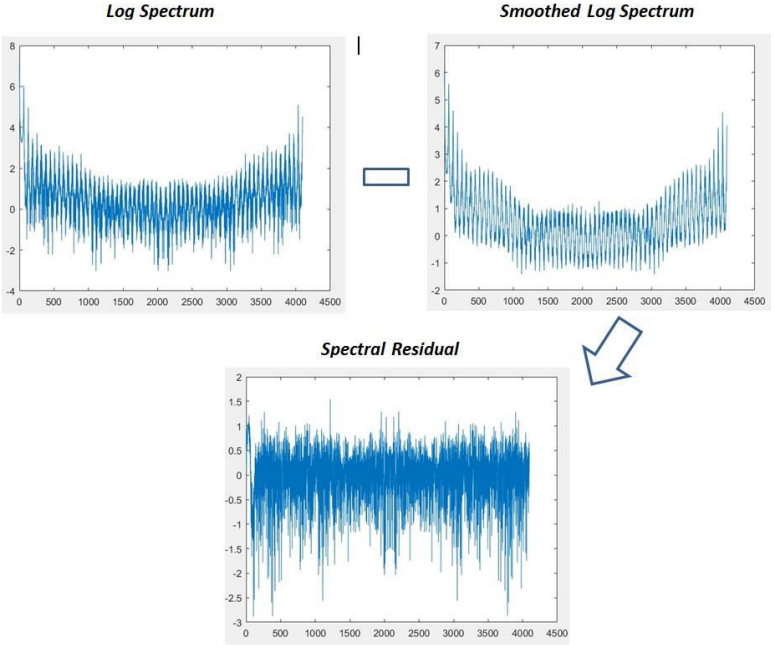
Proses kedua adalah *saliency detection* dengan model *Spectral Residual* yang terdiri dari beberapa sub-proses. Pertama, citra yang telah di-*resize* saat *preprocessing* diubah ke bentuk *grayscale*, kemudian dilakukan *Fourier transform* untuk mengubah representasi citra ke dalam domain frekuensi. *Fourier transform* menghasilkan citra keluaran bernilai bilangan kompleks yang dapat ditampilkan dengan dua komponen, yaitu *magnitude spectrum* dan *phase spectrum*. Hasil citra *grayscale*, citra yang dibentuk hanya dari *magnitude spectrum* dan citra yang dibentuk hanya dari *phase spectrum* ditunjukkan pada **Gambar 5.2**.



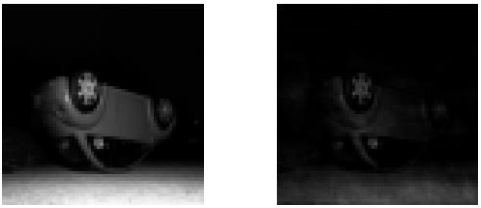
(a) (b) (c)

Gambar 5.2 (a) Citra *Grayscale*, Rekonstruksi Citra Hanya dari (b) *Magnitude Spectrum* dan (c) *Phase Spectrum*

Log spectrum didapatkan dengan menerapkan transformasi logaritmik pada *magnitude spectrum*. *Spectral residual* dari sebuah citra didapatkan dengan mengurangi *log spectrum* citra tersebut dengan rata-rata atau *smoothed log spectrum*. Proses pembentukan *spectral residual* dalam bentuk plot dengan *x-axis* berupa frekuensi dan *y-axis* berupa nilai intensitasnya ditunjukkan pada **Gambar 5.3**. Umumnya untuk mengambil rata-rata *log spectrum* dibutuhkan banyak citra, namun pada model ini hanya digunakan satu *log spectrum* citra yang dikonvolusi menggunakan *local average filter* atau *mean filter* untuk mendapatkan aproksimasi. Perbedaan antara citra yang dibentuk dari *log spectrum* dan *smoothed log spectrum* (menggunakan *Inverse Fourier Transform*) ditunjukkan pada **Gambar 5.4**.



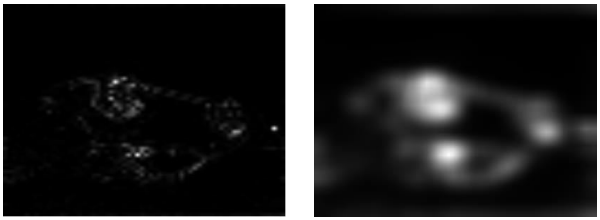
Gambar 5.3 Proses Pembentukan *Spectral Residual* dalam Bentuk Plot



(a) (b)

Gambar 5.4 Rekonstruksi Citra dari (b) *Log Spectrum* dan (c) *Smoothed Log Spectrum*

Spectral residual yang direpresentasikan dalam domain frekuensi diubah ke dalam domain spasial menggunakan *Inverse Fourier transform* untuk mendapatkan *prior saliency map*. *Prior saliency map* di-*smoothing* menggunakan *guided filter* untuk efek visual yang lebih baik. *Prior saliency map* di-*resize* ke skala semula agar dapat digunakan untuk proses labeling *training samples*. Perbedaan antara *prior saliency map* yang tidak di-*smoothing* dan *prior saliency map* yang telah di-*smoothing* menggunakan *guided filter* ditunjukkan pada **Gambar 5.5**.



(a) (b)

Gambar 5.5 (a) *Prior Saliency Map* Tanpa Filter (Citra *Spectral Residual*) dan (b) *Prior Saliency Map* yang Diterapkan *Guided Filter*

Setiap piksel dari *prior saliency map* mempunyai nilai *saliency,* sehingga nilai *saliency* dari setiap *superpixel* didapatkan dari rata-rata nilai *saliency* semua piksel yang dikandung *superpixel* tersebut. *Training samples* untuk mempelajari

classifier mencakup *training samples* positif dari *salient object* dan *training samples* negatif dari *background*. Untuk menentukan *training samples* positif dan negatif digunakan dua *threshold*, yaitu *threshold* positif dan *threshold* negatif. *Superpixels* dengan nilai *saliency* yang lebih besar dari *threshold* positif akan diberi label +1 sebagai *training samples* positif, sebaliknya *superpixels* dengan nilai *saliency* yang lebih kecil dari *threshold* negatif akan diberi label -1 sebagai *training samples* negatif. *Superpixels* yang memiliki nilai *saliency* di antara *threshold* positif dan negatif tidak digunakan sebagai *training samples*. Dengan asumsi bahwa *salient object* biasanya tidak tampak pada batas-batas citra, *superpixels* di sepanjang batas citra diberi label -1 sebagai *training samples* negatif. Hasil dari proses *saliency detection* dengan model *Spectral Residual* adalah matriks label *training samples* berukuran *N*x1 untuk setiap skala *superpixels*, di mana *N* adalah jumlah *superpixels* yang terbentuk dari skala 100, 150, 200 atau 250.

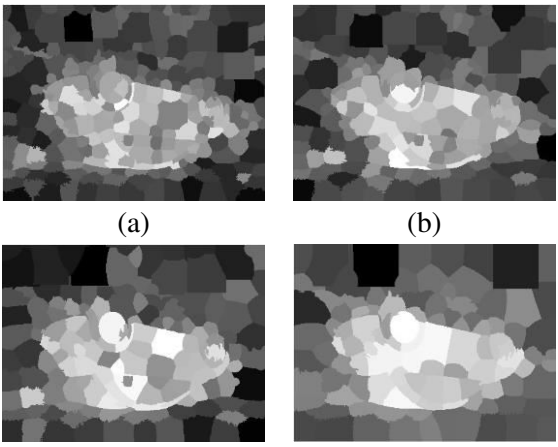
Proses ketiga adalah melakukan ekstraksi fitur untuk setiap skala *superpixels*. Untuk melakukan ekstraksi fitur warna digunakan RGB dan CIELAB, sedangkan untuk fitur tekstur digunakan *Uniform Local Binary Pattern* (*Uniform* LBP). Nilai RGB dan CIELAB untuk satu *superpixel* didapatkan dari rata-rata nilai RGB dan CIELAB semua piksel yang dikandung *superpixel* tersebut. *Uniform* LBP menggunakan tetangga berukuran 3x3, yang kemudian setiap pikselnya diberi nilai atau label dengan *range* 0 sampai 58, nilai 58 ditetapkan untuk semua pola non-*uniform*. *Uniform* LBP untuk satu *superpixel* didapatkan dari histogram nilai *Uniform* LBP semua piksel yang dikandung *superpixel* tersebut. Sehingga pada tahap ekstraksi fitur dihasilkan matriks fitur berukuran *N*x65 yang terbentuk dari gabungan RGB (3 fitur), CIELAB (3 fitur) dan *Uniform Local Binary Pattern* (59 fitur), di mana *N* adalah jumlah *superpixels* yang terbentuk dari skala 100, 150, 200 atau 250.

Proses keempat adalah *saliency detection* dengan ELM *classifier* yang terdiri dari beberapa sub-proses. Data masukan

dalam proses ini adalah *training* dan *testing samples* yang dihasilkan dari proses *saliency detection* dengan model *Spectral Residual* dan proses ekstraksi fitur. Matriks label *training samples* berukuran *N*x1 dan matriks fitur berukuran *N*x65 untuk setiap skala *superpixels* digabungkan membentuk matriks *training samples* berukuran *K*x66, di mana *N* adalah jumlah *superpixels* yang terbentuk dari skala 100, 150, 200 atau 250 dan *K* adalah jumlah *superpixels* yang terbentuk dari gabungan seluruh skala. Matriks *testing samples* terdiri dari matriks fitur untuk skala *superpixels* yang akan diujikan (terdapat empat matriks *testing samples* sesuai dengan jumlah skala *superpixels* yang digunakan).

Testing samples digunakan ketika mencari vektor keluaran dari *hidden layer*, sedangkan *training samples* digunakan ketika mencari vektor bobot keluaran antara *hidden layer* dan *output layer*. *Feature mapping* menggunakan *linear kernel* berfungsi untuk memetakan *testing samples* ke *hidden layer* yang menghasilkan vektor keluaran dari *hidden layer*. Vektor bobot keluaran terkecil antara *hidden layer* dan *output layer* dihitung dengan memanfaatkan *kernel* ELM dan *pseudoinverse* dari matriks *training samples*.

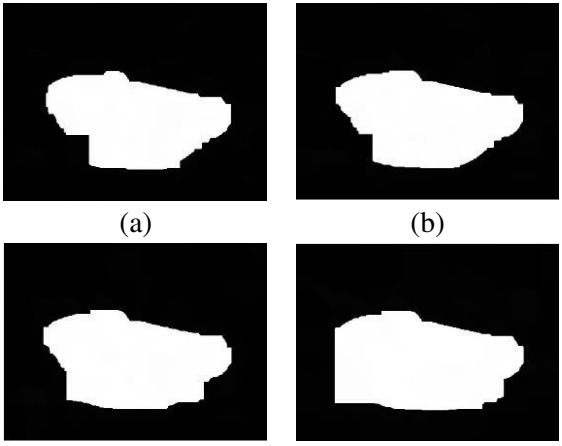
Fungsi keluaran dari ELM didapatkan dengan mengalikan vektor keluaran dari *hidden layer* dengan vektor bobot keluaran antara *hidden layer* dan *output layer*. Fungsi keluaran ELM merupakan vektor yang berisi nilai *saliency* dari setiap *superpixel*. *Trained saliency map* dihasilkan dengan memetakan nilai *saliency* dari setiap *superpixel* ke piksel yang dikandungnya. Empat skala *trained saliency map* yang terbentuk ditunjukkan pada **Gambar 5.6**. *Smoothing trained saliency map* dilakukan dengan metode *Graph Cut* yang bertujuan untuk mereduksi *noise* dan lebih menonjolkan nilai *saliency* dari *salient object*. Hasil dari proses *saliency detection* dengan ELM *classifier* adalah empat skala *trained saliency map* yang telah di-*smoothing* menggunakan metode *Graph Cut*, seperti yang ditunjukkan pada **Gambar 5.7**.



(a) (b)

(c) (d)

Gambar 5.6 Empat Skala *Trained Saliency Map*, (a) Skala 100, (b) Skala 150, (c) Skala 200, (d) Skala 250

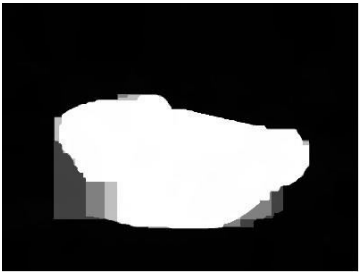


(a) (b)

(c) (d)

Gambar 5.7 Empat Skala *Smoothed Trained Saliency Map*, (a) Skala 100, (b) Skala 150, (c) Skala 200, (d) Skala 250

Proses kelima dan yang terakhir adalah melakukan segmentasi citra, yang terdiri dari dua sub-proses yaitu proses integrasi empat skala *trained saliency map* dan proses pembentukan *object map*. Integrasi dilakukan dengan mengambil rata-rata nilai *saliency* setiap piksel dari empat skala yang berbeda. Hasil integrasi dari empat skala *smoothed trained saliency map* ditunjukkan pada **Gambar 5.8**. *Object map* dibentuk dari *trained saliency map* hasil integrasi yang disegmentasi berdasarkan nilai *threshold*. Nilai *threshold* yang dihasilkan oleh metode Otsu ditingkatkan nilainya sebanyak 1,5 kali untuk hasil segmentasi yang lebih baik. Hasil akhir dari uji coba adalah *object map* atau citra biner yang menandakan area *salient object* dan area non-*salient object* pada citra masukan. Citra masukan, *ground truth* dan *object map* dari hasil uji coba ditunjukkan pada **Gambar 5.9**.



Gambar 5.8 *Trained Saliency Map* Hasil Integrasi dari Empat Skala *Smoothed Trained Saliency Map*

(a) (b) (c)

Gambar 5.9 (a) Citra Masukan, (b) *Ground Truth*, (c) *Object Map*

5.4 Skenario Uji Coba

Uji coba dilakukan untuk mengetahui nilai-nilai parameter yang tepat untuk digunakan pada masing-masing proses. Nilai parameter yang tepat penting untuk diketahui karena penggunaan parameter yang tepat akan memberikan hasil yang terbaik pada keluaran tiap proses.

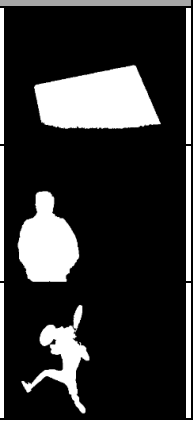
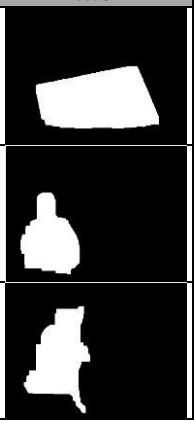
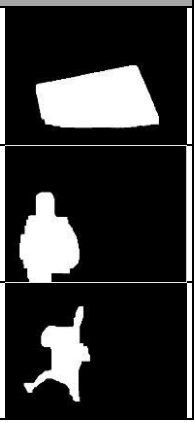
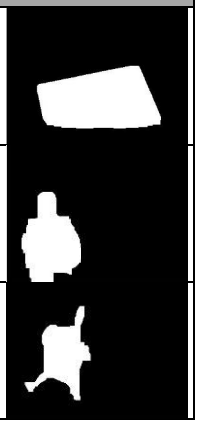
Skenario pengujian terdiri dari lima macam yaitu:

1. Uji coba pengaruh jenis filter dalam proses *smoothing prior saliency map*.
2. Uji coba pengaruh jenis fungsi *feature mapping* atau *kernel* dalam proses pemetaan *testing samples* ke *hidden layer*.
3. Uji coba penentuan nilai parameter *C* dalam proses penghitungan bobot keluaran antara *hidden layer* dan *output layer*.
4. Uji coba penentuan nilai *threshold* dalam proses pembentukan *object map*.
5. Uji coba menggabungkan *trained saliency map* dengan *prior saliency map*.

5.5 Uji Coba dan Evaluasi Skenario 1

Skenario 1 adalah uji coba yang membandingkan hasil segmentasi dengan *ground truth* berdasarkan perbedaan jenis filter dalam proses *smoothing prior saliency map*. Pengaruh dari pemilihan filter saat melakukan *smoothing prior saliency map* terhadap hasil segmentasi akan dievaluasi pada uji coba ini. Beberapa jenis filter yang akan diujicobakan adalah *disk filter*, *Gaussian filter* dan *guided filter*.

Tabel 5.2 Hasil Segmentasi Uji Coba Skenario 1

Ground Truth	*Gaussian Filter*	*Disk Filter*	*Guided Filter*
			

Parameter yang digunakan untuk membangun setiap jenis filter adalah parameter yang memberikan hasil terbaik (berdasarkan uji coba), yaitu *Gaussian filter* dengan σ bernilai 6, *disk filter* dengan radius bernilai 3 dan *guided filter* dengan tetangga berukuran 5x5. Hasil uji coba skenario 1 ditunjukkan pada **Tabel 5.2** dan **Tabel 5.3**.

Tabel 5.3 Hasil Rata-Rata *F1 Score* Uji Coba Skenario 1

Jenis Filter	***F1 Score* (%)**
Gaussian filter	80,16
Disk filter	88,87
Guided filter	**90,39**

Dari hasil segmentasi uji coba skenario 1, dapat dilihat bahwa dari segi visual, penggunaan *disk filter dan guided filter* tidak menghasilkan perbedaan *object map* yang cukup signifikan, namun *object map* yang dihasilkan dari penggunaan *Gaussian filter* memiliki kekurangan dalam bentuk detil dari *salient object*.

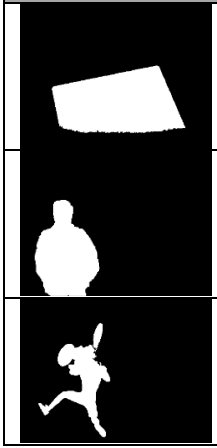
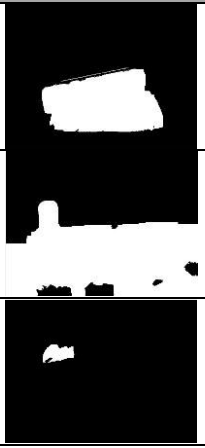
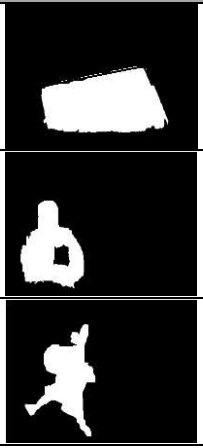
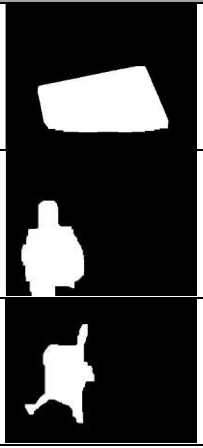
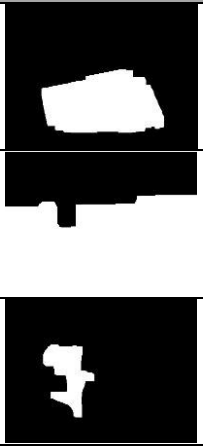
Apabila dilihat dari hasil rata-rata *F1 score* seluruh hasil segmentasi data uji coba, hasil terbaik didapatkan dari penggunaan *guided filter* dengan rata-rata *F1 score* bernilai 90,39%.

Kesimpulan yang dapat diambil dari hasil uji coba skenario 1 adalah pemilihan filter dalam melakukan *smoothing* menjadi salah satu faktor penting terhadap kualitas *prior saliency map* yang secara langsung mempengaruhi kualitas dari *training samples* yang dibentuk. *Training samples* yang lebih representatif akan memberikan hasil segmentasi yang lebih baik.

5.6 Uji Coba dan Evaluasi Skenario 2

Skenario 2 adalah uji coba yang membandingkan hasil segmentasi dengan *ground truth* berdasarkan perbedaan fungsi *feature mapping* atau *kernel* dalam proses pemetaan *testing samples* ke *hidden layer*. Proses pemetaan tersebut merupakan salah satu proses dalam tahap *saliency detection* dengan ELM *classifier*. Pengaruh dari pemilihan fungsi *feature mapping* atau *kernel* saat melakukan *feature mapping* terhadap hasil segmentasi akan dievaluasi pada uji coba ini. Beberapa fungsi *feature mapping* atau *kernel* yang akan diujicobakan adalah *linear kernel*, *polynomial kernel*, *Radial Basis Function* (RBF) *kernel* dan fungsi *sigmoid*.

Tabel 5.4 Hasil Segmentasi Uji Coba Skenario 2

Ground Truth	*Polynomial Kernel*	RBF *Kernel*	*Linear Kernel*	Fungsi *Sigmoid*
				

Parameter yang digunakan dalam setiap fungsi *feature mapping* atau *kernel* adalah parameter yang memberikan hasil terbaik (berdasarkan uji coba), yaitu *polynomial kernel* dengan bias bernilai 0 dan derajat bernilai 2, RBF *kernel* dengan *gamma* bernilai 0,1 dan fungsi *sigmoid* dengan konstanta bernilai acak untuk setiap *testing samples*. Hasil uji coba skenario 2 ditunjukkan pada **Tabel 5.4** dan **Tabel 5.5**.

Tabel 5.5 Hasil Rata-Rata *F1 Score* Uji Coba Skenario 2

Jenis Fungsi *Feature Mapping* atau Kernel	***F1 Score* (%)**
Polynomial kernel	72,63
RBF *kernel*	87,73
Linear kernel	**90,39**
Fungsi *sigmoid*	76,12

Dari hasil segmentasi uji coba skenario 2, dapat dilihat bahwa dari segi visual terdapat perbedaan yang cukup signifikan antar fungsi *feature mapping* atau *kernel* yang berbeda. Penggunaan RBF *kernel* dan *linear kernel* mampu menghasilkan *object map* yang cukup representatif, namun *object map* yang dihasilkan dari penggunaan *polynomial kernel* dan fungsi *sigmoid* kurang representatif. Apabila dilihat dari hasil rata-rata *F1 score* seluruh hasil segmentasi data uji coba, hasil terbaik didapatkan dari penggunaan *linear kernel* dengan rata-rata *F1 score* bernilai 90,39%.

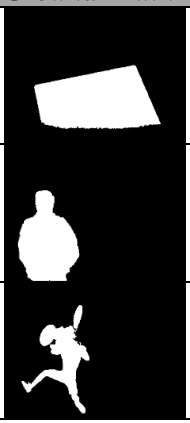
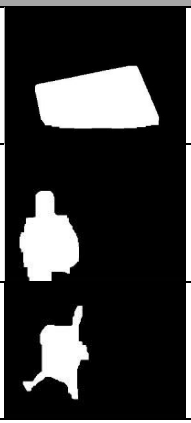
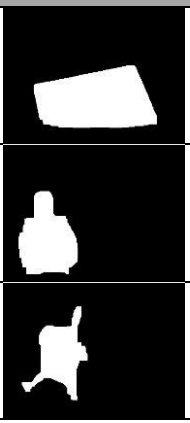
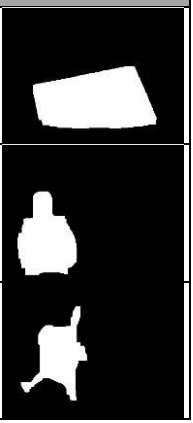
Kesimpulan yang dapat diambil dari hasil uji coba skenario 2 adalah meskipun ELM *classifier* memiliki kemampuan untuk melakukan aproksimasi fungsi aktivasi secara universal, namun pemilihan fungsi *feature mapping atau kernel* yang tepat dalam proses pemetaan *testing samples* ke *hidden layer* dapat meningkatkan akurasi dari *trained saliency map*. Peningkatan akurasi dari *trained saliency map* berbanding lurus dengan akurasi dari *object map*. Selain itu, *testing samples* yang diujikan

sudah bersifat *linearly separable*, karena penggunaan *kernel* linier menunjukkan performa yang lebih baik dibandingkan dengan penggunaan fungsi atau *kernel* non-linier.

5.7 Uji Coba dan Evaluasi Skenario 3

Skenario 3 adalah uji coba yang membandingkan hasil segmentasi dengan *ground truth* berdasarkan perbedaan nilai parameter *C* dalam proses penghitungan bobot keluaran antara *hidden layer* dan *output layer*. Proses penghitungan bobot keluaran tersebut merupakan salah satu proses dalam tahap *saliency detection* dengan ELM *classifier*. Pengaruh dari penentuan nilai parameter *C* saat penghitungan bobot keluaran terhadap hasil segmentasi akan dievaluasi pada uji coba ini. Beberapa nilai parameter *C* yang akan diujicobakan adalah 1, 32 dan 128.

Tabel 5.6 Hasil Segmentasi Uji Coba Skenario 3

Ground Truth	*C* = 1	*C* = 32	*C* = 128
			

Saat menentukan nilai bobot keluaran antara *hidden layer* dan *output layer*, diperlukan parameter *C* yang ditentukan oleh pengguna, yang menyediakan *tradeoff* antara jarak dari *separating margin* dan *training error*. Nilai *C* yang relatif kecil

menyebabkan *classifier* memilih jarak dari *separating margin* dengan kedua kelas relatif besar, meskipun terjadi misklasifikasi pada beberapa data, rentan terjadi *underfitting*. Sedangkan nilai *C* yang relatif besar, menyebabkan *classifier* memilih *separating margin* yang menghasilkan *training error* terkecil, namun jarak dari *separating margin* dengan kedua kelas relatif dekat, rentan terjadi *overfitting*. Hasil uji coba skenario 3 ditunjukkan pada **Tabel 5.6** dan **Tabel 5.7**.

Tabel 5.7 Hasil Rata-Rata *F1 Score* Uji Coba Skenario 3

Nilai Parameter *C*	***F1 Score* (%)**
1	**90,39**
32	89,22
128	88,99

Dari hasil segmentasi uji coba skenario 3, dapat dilihat bahwa dari segi visual tidak terdapat perbedaan yang cukup signifikan antar nilai *C* yang berbeda, setiap nilai *C* dapat menghasilkan *object map* yang cukup baik. Namun apabila dilihat dari hasil rata-rata *F1 score* seluruh hasil segmentasi data uji coba, hasil terbaik didapatkan dari penggunaan parameter *C* bernilai 1 dengan rata-rata *F1 score* bernilai 90,39%.

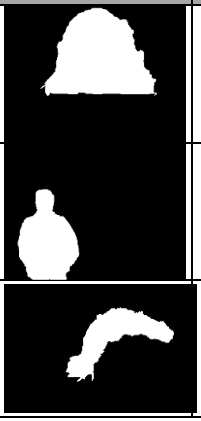
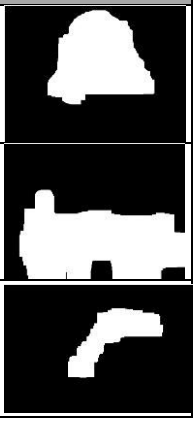
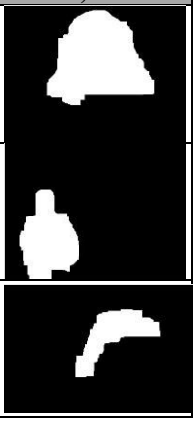
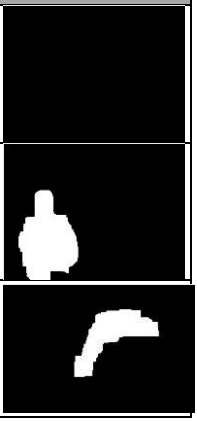
Kesimpulan yang dapat diambil dari hasil uji coba skenario 3 adalah penentuan nilai parameter *C* dalam menghitung bobot keluaran antara *hidden layer* dan *output layer* tidak terlalu berpengaruh terhadap akurasi dari *trained saliency map*. Hasil segmentasi terbaik didapatkan dari penggunaan nilai parameter *C* yang kecil, hal ini berarti *training samples* sudah bersifat *linearly separable*.

5.8 Uji Coba dan Evaluasi Skenario 4

Skenario 4 adalah uji coba yang membandingkan hasil segmentasi dengan *ground truth* berdasarkan perbedaan nilai *threshold* dalam proses pembentukan *object map*. Pengaruh dari

penentuan nilai *threshold* saat pembentukan *object map* akan dievaluasi pada uji coba ini. Nilai *threshold* yang akan diujicobakan adalah beberapa kelipatan dari nilai *threshold* awal yang dihasilkan dari metode Otsu. Hasil uji coba skenario 4 ditunjukkan pada **Tabel 5.8** dan **Tabel 5.9**.

Tabel 5.8 Hasil Segmentasi Uji Coba Skenario 4

Ground Truth	***Threshold* = 1x**	***Threshold* = 1,5x**	***Threshold* = 2x**
			

Tabel 5.9 Hasil Rata-Rata *F1 Score* Uji Coba Skenario 4

Kelipatan Nilai *Threshold*	***F1 Score* (%)**
1x	88,64
1,5x	**90,39**
2x	84,74

Dari hasil segmentasi uji coba skenario 4, dapat dilihat bahwa dari segi visual, penggunaan nilai *threshold* sebesar 2x lipat dari nilai *threshold* awal pada citra pertama menghasilkan *object map* yang tidak dapat menemukan area *salient object*. Pada citra kedua, penggunaan nilai *threshold* sebesar 1,5x lipat dan 2x lipat dari nilai *threshold* awal menghasilkan *object map* yang

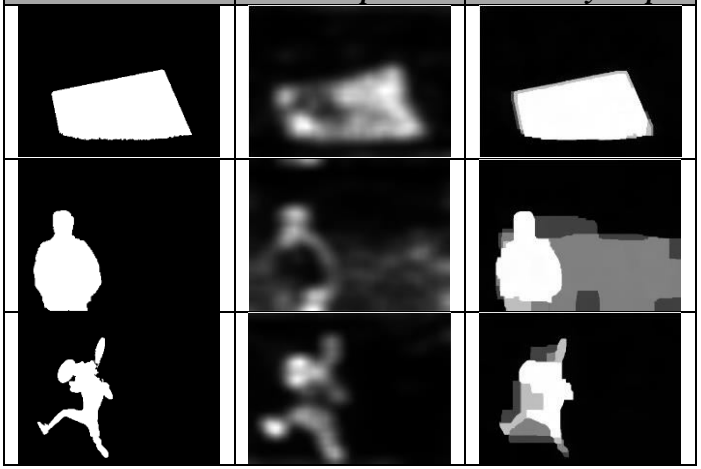
lebih akurat dibandingkan dengan penggunaan nilai *threshold* awal, namun sebaliknya penggunaan nilai *threshold* awal pada citra ketiga mampu menghasilkan *object map* yang lebih akurat. Apabila dilihat dari hasil rata-rata *F1 score* seluruh hasil segmentasi data uji coba, hasil terbaik didapatkan dari penggunaan kelipatan nilai *threshold* sebesar 1,5x dengan rata-rata *F1 score* bernilai 90,39%.

Kesimpulan yang dapat diambil dari hasil uji coba skenario 4 adalah penentuan nilai *threshold* dalam pembentukan *object map* bergantung dari tingkat kompleksitas fitur yang dimiliki citra masukan. Citra dengan fitur yang cukup kompleks biasanya memiliki banyak *proto object*, hal tersebut menyebabkan nilai *threshold* awal yang dihasilkan dari metode Otsu menjadi kurang akurat dalam melakukan segmentasi. Untuk mengatasi masalah tersebut dilakukan peningkatan nilai *threshold* awal, namun nilai *threshold* yang terlalu tinggi juga dapat menghasilkan *object map* yang kurang akurat.

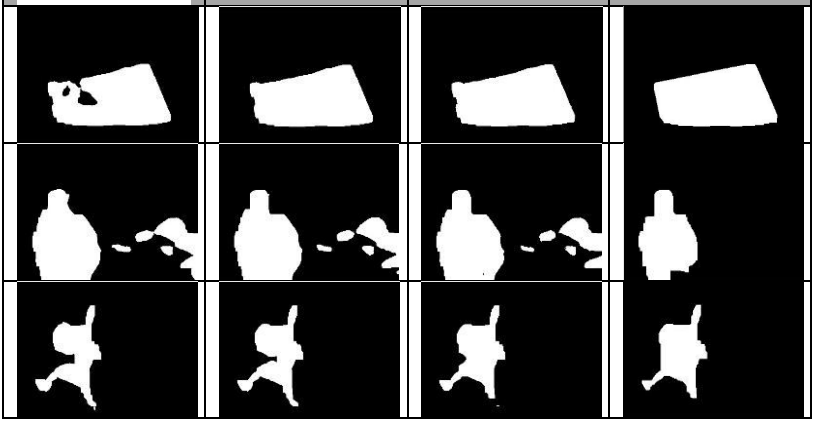
5.9 Uji Coba dan Evaluasi Skenario 5

Skenario 5 adalah uji coba yang membandingkan akurasi dari hasil segmentasi yang dibentuk dari *trained saliency map* dengan hasil segmentasi yang dibentuk dari gabungan *trained saliency map* dengan *prior saliency map*. Penggabungan tersebut didasari dari kelebihan yang dimiliki masing-masing *saliency map*, *prior saliency map* memiliki kelebihan dalam mendeteksi bentuk detil seperti tepi dari *salient object*, sedangkan *trained saliency map* memiliki kelebihan dalam mendeteksi bentuk *salient object* secara global dengan berbagai ukuran dan kategori. Penggabungan dilakukan dengan mengambil nilai *saliency* dari *trained saliency map* $SMap_{trained}$ sebanyak X% dan dijumlahkan dengan nilai *saliency* dari *prior saliency map* sebanyak (100-X)%. Persentase dari $SMap_{trained}$ yang akan diujicobakan adalah 60%, 70%, 80% dan 100% (nilai 100% berarti hanya menggunakan *trained saliency map*). Hasil uji coba skenario 5 ditunjukkan pada **Tabel 5.10**, **Tabel 5.11** dan **Tabel 5.12**.

Tabel 5.10 *Prior Saliency Map* dan *Trained Saliency Map* dari Setiap Citra

Ground Truth	*Prior Saliency Map*	*Trained Saliency Map*
		

Tabel 5.11 Hasil Segmentasi Uji Coba Skenario 5

$SMap_{trained} = 60\%$	$SMap_{trained} = 70\%$	$SMap_{trained} = 80\%$	$SMap_{trained} = 100\%$
			

Tabel 5.12 Hasil Rata-Rata *F1 Score* Uji Coba Skenario 5

$SMap_{trained}$	***F1 Score*** **(%)**
60%	86,90
70%	89,24
80%	89,96
100%	**90,39**

Dari hasil segmentasi uji coba skenario 5, dapat dilihat bahwa dari segi visual, *object map* dari citra pertama dan kedua yang dibentuk dari gabungan *trained saliency map* dengan *prior saliency map* memiliki akurasi yang kurang baik, namun pada citra ketiga penggabungan tersebut dapat menghasilkan *object map* yang lebih baik dibandingkan dengan *object map* yang dibentuk dari *trained saliency map* saja. Apabila dilihat dari hasil rata-rata *F1 score* seluruh hasil segmentasi data uji coba, *object map* hasil penggabungan dengan beberapa nilai $SMap_{trained}$ (%) memiliki *F1 score* yang inferior dibandingkan *object map* yang dibentuk dari *trained saliency map* ($SMap_{trained}$ = 100%). Hasil terbaik didapatkan dari hasil segmentasi yang dibentuk dari *trained saliency map* saja dengan rata-rata *F1 score* bernilai 90,39%.

Kesimpulan yang dapat diambil dari hasil uji coba skenario 5 adalah hasil segmentasi yang dibentuk dari *trained saliency map* lebih stabil dan akurat. Untuk citra tertentu, hasil segmentasi yang dibentuk dari gabungan *trained saliency map* dengan *prior saliency map* dapat memiliki hasil yang lebih baik. Namun hal tersebut hanya berlaku untuk citra yang memiliki *salient object* yang sangat kontras dengan *background* dan pada bagian *background* tidak memiliki fitur yang kompleks (untuk menghindari *noise*). Pada umumnya citra memiliki fitur yang cukup kompleks, sehingga penggunaan *trained saliency map* untuk membentuk hasil segmentasi lebih dipilih.

5.10 Evaluasi Umum Skenario Uji Coba

Berdasarkan skenario uji coba yang telah dilakukan, didapatkan bahwa metode *Saliency Extreme Learning Machine* mampu melakukan segmentasi citra area *salient object* dari 50 citra uji coba dengan rata-rata *F1 score* paling besar 90,39%. Dari seluruh data uji coba, didapatkan *F1 score* terendah dengan nilai 76,27% dan *F1 score* tertinggi dengan nilai 98,81%. Hasil segmentasi dari citra yang memiliki *F1 score* terendah dan tertinggi ditunjukkan pada **Gambar 5.10** dan **Gambar 5.11**.

(a) (b) (c)

Gambar 5.10 (a) Citra Asli, (b) *Ground Truth* dan (c) Hasil Segmentasi Citra dengan *F1 Score* Terendah

(a) (b) (c)

Gambar 5.11 (a) Citra Asli, (b) *Ground Truth* dan (c) Hasil Segmentasi Citra dengan *F1 Score* Tertinggi

Pada skenario uji coba 1 didapatkan bahwa penggunaan *guided filter* dalam proses *smoothing prior saliency map* memberikan hasil terbaik. Kesimpulan yang dapat diambil dari hasil uji coba skenario 1 adalah pemilihan filter dalam melakukan *smoothing* menjadi salah satu faktor penting terhadap kualitas *prior saliency map* yang secara langsung mempengaruhi kualitas

dari *training samples* yang dibentuk. *Training samples* yang lebih representatif akan memberikan hasil segmentasi yang lebih baik.

Pada skenario uji coba 2 didapatkan bahwa penggunaan *linear kernel* dalam proses pemetaan *testing samples* ke *hidden layer* memberikan hasil terbaik. Kesimpulan yang dapat diambil dari hasil uji coba skenario 2 adalah meskipun ELM *classifier* memiliki kemampuan untuk melakukan aproksimasi fungsi aktivasi secara universal, namun pemilihan fungsi *feature mapping atau kernel* yang tepat dapat meningkatkan akurasi dari *trained saliency map*. Selain itu, *testing samples* yang diujikan sudah bersifat *linearly separable*, karena penggunaan *kernel* linier menunjukkan performa yang lebih baik dibandingkan dengan penggunaan fungsi atau *kernel* non-linier.

Pada skenario uji coba 3 didapatkan bahwa penggunaan parameter *C* bernilai 1 dalam proses penghitungan bobot keluaran antara *hidden layer* dan *output layer* memberikan hasil terbaik. Kesimpulan yang dapat diambil dari hasil uji coba skenario 3 adalah penentuan nilai parameter *C* dalam menghitung bobot keluaran antara *hidden layer* dan *output layer* tidak terlalu berpengaruh terhadap akurasi dari *trained saliency map*. Hasil segmentasi terbaik didapatkan dari penggunaan nilai parameter *C* yang kecil, hal ini berarti *training samples* sudah bersifat *linearly separable*.

Pada skenario uji coba 4 didapatkan bahwa penggunaan nilai *threshold* sebesar 1,5 kali lipat dari nilai *threshold* awal yang dihasilkan dari metode Otsu memberikan hasil terbaik. Kesimpulan yang dapat diambil dari hasil uji coba skenario 4 adalah penentuan nilai *threshold* dalam pembentukan *object map* bergantung dari tingkat kompleksitas fitur yang dimiliki citra masukan. Citra dengan fitur yang cukup kompleks biasanya memiliki banyak *proto object* dengan beragam nilai *saliency*, hal tersebut menyebabkan nilai *threshold* awal kurang akurat dalam melakukan segmentasi.

Pada skenario uji coba 5 didapatkan bahwa penggunaan *trained saliency map* dalam pembentukan *object map*

memberikan hasil terbaik. Kesimpulan yang dapat diambil dari hasil uji coba skenario 5 adalah hasil segmentasi yang dibentuk dari *trained saliency map* lebih stabil dan akurat. Untuk citra tertentu, hasil segmentasi yang dibentuk dari gabungan *trained saliency map* dengan *prior saliency map* dapat memiliki hasil yang lebih baik. Namun hal tersebut hanya berlaku untuk citra yang memiliki *salient object* yang sangat kontras dengan *background* dan pada bagian *background* tidak memiliki fitur yang kompleks (untuk menghindari *noise*).

[*Halaman ini sengaja dikosongkan*]

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah. Selain itu juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Kesimpulan yang diperoleh dari uji coba dan evaluasi adalah sebagai berikut:

1. Metode *Saliency Extreme Learning Machine* berhasil digunakan untuk segmentasi citra area *salient object* dengan rata-rata presisi, *recall* dan *F1 score* masing-masing sebesar 90,26%, 91,52%, dan 90,39%.
2. Pelabelan *training samples* secara otomatis dapat dilakukan dengan menggunakan metode *thresholding* untuk menentukan *training samples* positif dan *training samples* negatif dari *prior saliency map* yang dihasilkan oleh tahap *saliency detection* dengan model *Spectral Residual* (*prior saliency map* di-*smoothing* menggunakan *guided filter* dengan tetangga berukuran 5x5 untuk hasil terbaik).
3. Hasil segmentasi citra dengan metode *Saliency Extreme Learning Machine* dipengaruhi oleh fungsi *feature mapping* atau *kernel* dan parameter *C* pada tahap *saliency detection* dengan ELM *classifier*. Performa terbaik dihasilkan oleh penggunaan *linear kernel* dan parameter *C* dengan nilai 1.
4. Hasil segmentasi citra dengan metode *Saliency Extreme Learning Machine* dipengaruhi oleh nilai *threshold* pada proses pembentukan *object map*. Nilai *threshold* sebesar 1,5 kali lipat dari nilai *threshold* yang dihasilkan menggunakan metode Otsu dapat memberikan performa yang terbaik.
5. Hasil segmentasi citra yang dibentuk dari *trained saliency map* lebih stabil dan akurat dibandingkan hasil segmentasi

citra yang dibentuk dari gabungan *prior saliency map* dengan *trained saliency map*.

6.2 Saran

Saran yang hendak disampaikan terkait dengan pengembangan tugas akhir ini adalah sebagai berikut:

1. Aplikasi segmentasi citra area *salient object* menggunakan metode *Saliency Extreme Learning Machine* tidak dapat menghasilkan performa yang maksimal pada citra yang memiliki banyak *proto object* dan citra yang memiliki *salient object* dengan fitur yang mirip dengan *background*, sehingga dibutuhkan metode yang dapat menangani hal tersebut.
2. Aplikasi segmentasi citra area *salient object* menggunakan metode *Saliency Extreme Learning Machine* tidak dapat menghasilkan *object map* dengan bentuk *salient object* yang sangat detil, terutama pada *salient object* yang memiliki bentuk yang kompleks. Penggabungan *prior saliency map* dengan *trained saliency map* merupakan salah satu cara yang dicoba dalam tugas akhir ini untuk menangani hal tersebut, namun ternyata masalah ini masih belum dapat teratasi.
3. Menggunakan varian dari ELM yaitu *Multi-layer Extreme Learning Machine* saat tahap *saliency detection* dengan ELM *classifier* untuk mendapatkan hasil segmentasi yang lebih baik, dikarenakan kemampuannya dalam melakukan integrasi *prior saliency map* yang dihasilkan oleh beberapa metode *bottom-up* dengan kelebihannya masing-masing.

DAFTAR PUSTAKA

[1] "Image Segmentation," MathWorks, [Online]. Available: https://www.mathworks.com/discovery/image-segmentation.html. [Diakses 28 April 2017].

[2] G.-B. Huang, X. Ding, H. Zhou, "Optimization method based extreme learning machine for classification," *Neurocomputing 74*, 2010.

[3] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing 70*, 2006.

[4] L. Zhang, J. Li, H. Lu, "Saliency detection via extreme learning machine," *Neurocomputing 218*, 2016.

[5] X. Hou a. L. Zhang, "Saliency Detection: A Spectral Residual Approach," *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[6] N. Tong, H. Lu, X. Ruan, M.-H. Yang, "Salient object detection via boostrap learning," *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[7] M.-M. Cheng, "MSRA10K Salient Object Database," Media Computing Laboratory Nankai University, 28 Juli 2014. [Online]. Available: http://mmcheng.net/msra10k/. [Diakses 25 Februari 2017].

[8] E. Niebur, "Saliency map," Scholarpedia, 28 Agustus 2007. [Online]. Available: http://www.scholarpedia.org/article/Saliency_map. [Diakses 29 April 2017].

[9] F. Weinhaus, A. Thyssen, "Fourier Transforms," ImageMagick, 27 Oktober 2011. [Online]. Available: http://www.imagemagick.org/Usage/fourier/. [Diakses 1 Mei 2017].

[10] R. Fisher, S. Perkins, A. Walker dan E. Wolfart, “Fourier Tranform,” University of Edinburgh School of Informatics, [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm. [Diakses 1 Mei 2017].

[11] R. Fisher, S. Perkins, A. Walker dan E. Wolfart, “Logarithm Operator,” University of Edinburgh School of Informatics, [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/pixlog.htm. [Diakses 1 Mei 2017].

[12] R. Fisher, S. Perkins, A. Walker dan E. Wolfart, “Digital Filters,” University of Edinburgh School of Informatics, [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/filtops.htm. [Diakses 3 Mei 2017].

[13] R. Fisher, S. Perkins, A. Walker and E. Wolfart, "Spatial Filters - Mean Filter," University of Edinburgh School of Informatics, [Online]. Available: homepages.inf.ed.ac.uk/rbf/HIPR2/mean.htm. [Diakses 3 Mei 2017].

[14] “fspecial - Create predefined 2-D filter,” MathWorks, [Online]. Available: https://www.mathworks.com/help/images/ref/fspecial.html. [Diakses 4 Mei 2017].

[15] “imgaussfilt - 2-D Gaussian filtering of images,” MathWorks, [Online]. Available: https://www.mathworks.com/help/images/ref/imgaussfilt.html. [Diakses 4 Mei 2017].

[16] K. He, J. Sun, X. Tang, “Guided image filtering,” *ECCV*, 2010.

[17] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Susstrunk, “SLIC superpixels compared to state-of-the-art

superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[18] "Introduction to Color Spaces," Tutorialspoint, [Online]. Available: https://www.tutorialspoint.com/dip/introduction_to_color_spaces.htm. [Diakses 5 Mei 2017].

[19] "CIELAB," Adobe, [Online]. Available: http://dba.med.sc.edu/price/irf/Adobe_tg/models/cielab.html. [Diakses 5 Mei 2017].

[20] M. Pietikainen, "Local Binary Patterns," Scholarpedia, 3 Maret 2010. [Online]. Available: http://www.scholarpedia.org/article/Local_Binary_Patterns. [Diakses 5 Mei 2017].

[21] T. Ojala, M. Pietikainen, T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell. 24*, 2002.

[22] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst. Man Cybern*, 2012.

[23] Z. Simayijiang, S. Grimm, "Segmentation with Graph Cuts," Matematikcentrum Lunds Universitet. [Online]. Available: http://www.maths.lth.se/matematiklth/personal/petter/rapporter/graph.pdf. [Diakses 8 Mei 2017].

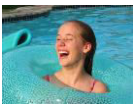
[24] Y. Boykov, O.Veksler, R.Zabih, "Fast Approximate Energy Minimization via Graph Cuts," *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999.

[25] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, 1979.

[26] “Confusion Matrix, ” Department of Computer Sciences University of Regina, [Online]. Available: http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion_matrix/confusion_matrix.html. [Diakses 10 Mei 2017].

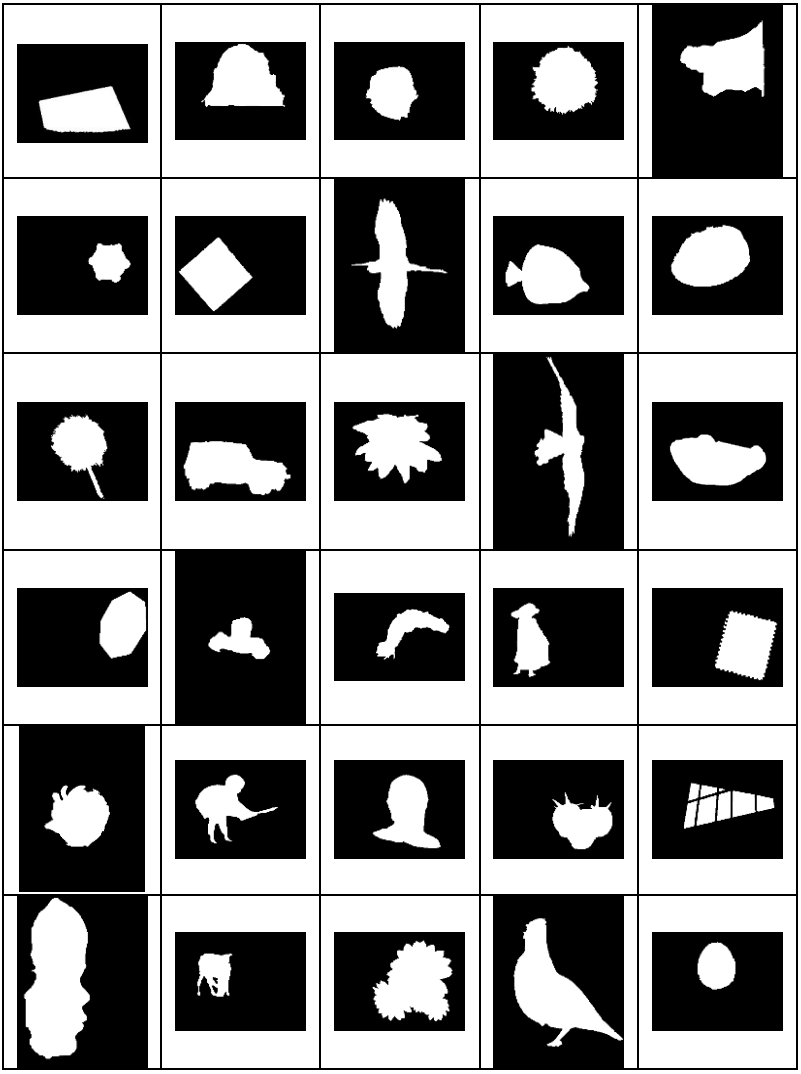
LAMPIRAN

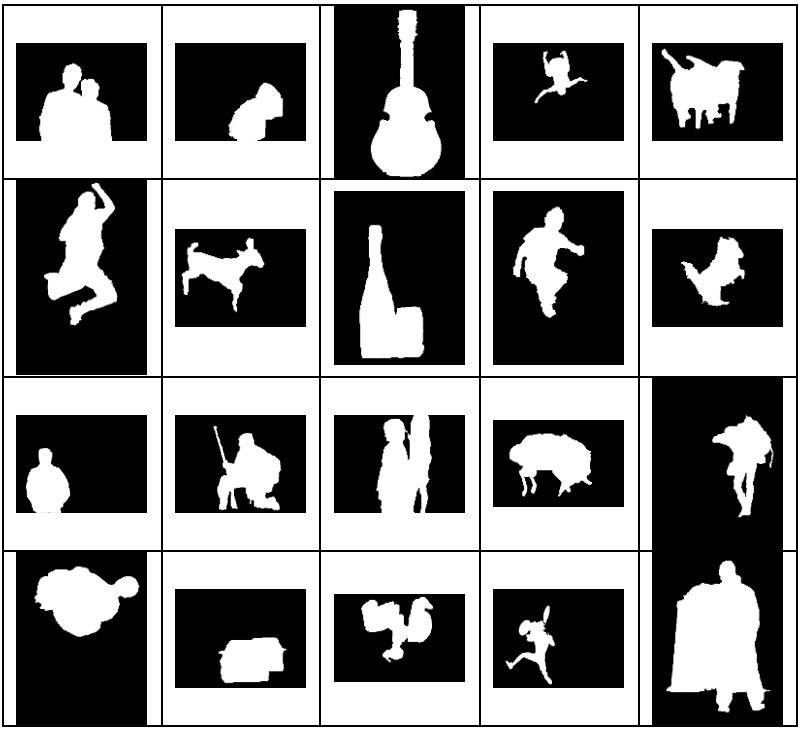
A. Citra Masukan

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

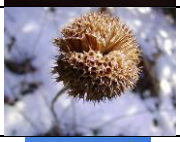
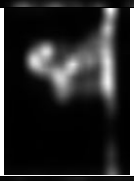
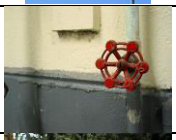
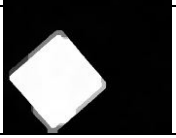
 26.jpg	 27.jpg	 28.jpg	 29.jpg	 30.jpg
 31.jpg	 32.jpg	 33.jpg	 34.jpg	 35.jpg
 36.jpg	 37.jpg	 38.jpg	 39.jpg	 40.jpg
 41.jpg	 42.jpg	 43.jpg	 44.jpg	 45.jpg
 46.jpg	 47.jpg	 48.jpg	 49.jpg	 50.jpg

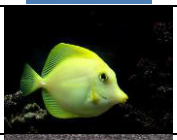
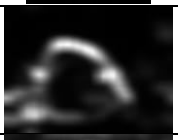
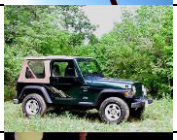
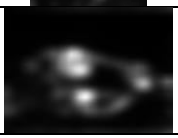
B. Ground Truth

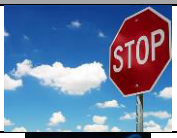
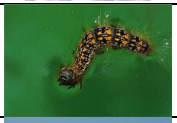
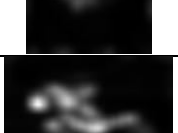
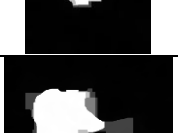
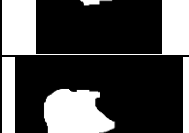
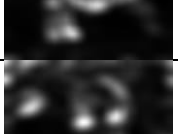


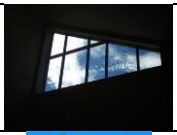
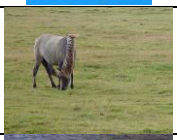
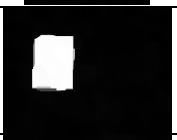
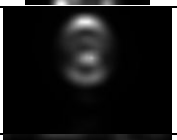
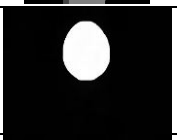
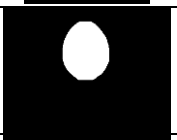
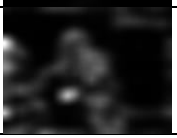


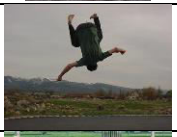
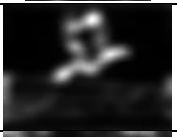
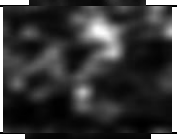
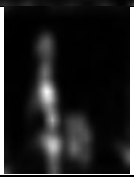
C. Hasil Citra dari Tahapan-Tahapan

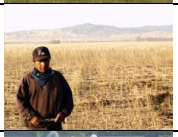
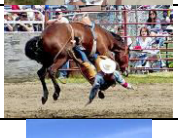
Citra Asli	*Prior Saliency Map*	*Trained Saliency Map*	*Object Map*
			
			
			
			
			
			
			

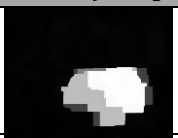
Citra Asli	*Prior Saliency Map*	*Trained Saliency Map*	*Object Map*
			
			
			
			
			
			
			
			

Citra Asli	*Prior Saliency Map*	*Trained Saliency Map*	*Object Map*
			
			
			
			
			
			
			
			

Citra Asli	*Prior Saliency Map*	*Trained Saliency Map*	*Object Map*
			
			
			
			
			
			
			
			

Citra Asli	*Prior Saliency Map*	*Trained Saliency Map*	*Object Map*
			
			
			
			
			
			
			

Citra Asli	*Prior Saliency Map*	*Trained Saliency Map*	*Object Map*
			
			
			
			
			
			
			
			

Citra Asli	***Prior Saliency Map***	***Trained Saliency Map***	***Object Map***
			
			
			
			

D. Performa Uji Coba Skenario 1

Nama Citra	Gaussian Filter			Disk Filter			Guided Filter		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
1	98,37	99,25	98,81	98,66	99,18	98,92	99,57	97,13	98,34
2	76,72	96,56	85,50	98,69	96,27	97,47	95,83	96,21	96,02
3	88,96	94,36	91,58	99,62	89,56	94,32	99,60	89,39	94,22
4	97,44	85,94	91,33	95,27	95,99	95,63	95,05	97,79	96,40
5	95,62	99,32	97,43	97,14	98,36	97,75	97,00	98,23	97,61
6	36,26	99,73	53,19	96,89	98,91	97,89	96,99	98,83	97,90
7	98,85	98,78	98,82	89,28	99,02	93,90	89,84	99,06	94,22
8	75,45	96,99	84,87	96,70	71,44	82,17	98,13	92,43	95,20
9	82,03	98,87	89,67	76,58	64,57	70,06	83,67	98,62	90,53
10	99,02	90,64	94,65	96,36	95,99	96,17	92,89	96,82	94,81
11	96,81	88,28	92,35	98,35	87,76	92,76	97,50	91,35	94,33
12	92,37	92,11	92,24	92,60	93,51	93,05	92,68	93,20	92,94
13	99,59	95,26	97,37	99,31	94,11	96,64	99,31	94,19	96,68
14	54,56	95,88	69,55	90,79	87,98	89,36	91,09	87,36	89,18
15	88,00	97,76	92,63	99,21	93,11	96,06	99,25	92,47	95,74
16	96,61	99,79	98,17	96,97	99,65	98,29	94,23	99,64	96,86

Nama Citra	Gaussian Filter			Disk Filter			Guided Filter		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
17	1,12	0,87	0,98	98,39	89,17	93,56	98,77	85,63	91,73
18	57,17	98,92	72,46	97,05	82,62	89,26	95,75	68,55	79,90
19	50,84	99,83	67,37	96,51	95,06	95,78	96,12	95,30	95,71
20	97,44	99,17	98,30	97,44	99,15	98,29	88,09	99,26	93,34
21	96,53	85,92	90,92	97,29	75,55	85,06	97,48	82,21	89,20
22	46,91	99,31	63,72	86,34	81,29	83,74	80,16	83,52	81,80
23	98,97	85,26	91,60	98,89	91,11	94,84	98,89	90,86	94,71
24	91,28	97,86	94,46	97,62	85,38	91,09	98,10	83,46	90,19
25	59,59	99,90	74,65	85,18	99,08	91,61	85,64	97,11	91,01
26	98,52	99,27	98,89	98,47	99,09	98,78	98,53	99,08	98,81
27	70,40	97,39	81,73	81,62	97,60	88,90	82,23	96,73	88,89
28	92,11	98,82	95,34	91,82	98,96	95,26	92,45	98,87	95,55
29	80,61	91,11	85,54	98,12	93,29	95,64	97,56	94,41	95,96
30	70,72	95,78	81,36	99,59	94,65	97,06	99,67	94,03	96,77
31	64,81	85,05	73,56	89,10	68,86	77,68	86,10	76,70	81,13
32	75,90	99,35	86,06	97,61	82,29	89,30	97,17	82,98	89,51
33	92,88	76,02	83,61	93,89	71,26	81,02	95,34	74,94	83,92

Nama Citra	Gaussian Filter			Disk Filter			Guided Filter		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
34	58,63	95,83	72,75	85,73	81,54	83,58	77,22	89,65	82,97
35	89,78	94,74	92,20	83,30	98,09	90,09	83,73	96,23	89,54
36	75,23	95,07	83,99	88,96	90,47	89,71	92,27	86,83	89,47
37	59,42	88,01	70,95	54,30	97,56	69,77	66,87	97,05	79,18
38	92,46	98,81	95,53	99,28	87,78	93,18	99,33	87,14	92,84
39	43,37	24,11	30,99	87,59	87,91	87,75	82,19	89,79	85,82
40	43,96	99,04	60,89	72,95	97,10	83,31	71,70	97,61	82,67
41	95,00	86,66	90,64	96,24	94,50	95,36	91,75	95,36	93,52
42	73,19	94,75	82,59	63,11	96,51	76,31	62,53	97,73	76,27
43	86,95	94,19	90,43	86,92	89,84	88,36	87,15	91,97	89,49
44	99,34	18,03	30,52	66,25	76,71	71,10	89,95	72,51	80,30
45	29,32	97,11	45,05	76,54	96,04	85,19	75,76	96,10	84,73
46	74,33	99,88	85,23	76,45	99,47	86,45	79,73	99,43	88,50
47	96,96	45,08	61,55	94,90	35,18	51,34	96,87	80,32	87,82
48	78,82	96,66	86,83	86,06	95,81	90,67	84,98	95,83	90,08
49	54,62	99,37	70,49	82,78	95,50	88,69	76,38	95,34	84,81
50	89,79	87,54	88,65	88,15	65,80	75,35	83,82	80,70	82,23

Nama Citra	Gaussian Filter			Disk Filter			Guided Filter		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
AVG	77,27	89,09	80,16	90,34	89,11	88,87	90,26	91,52	90,39
MIN	1,12	0,87	0,98	54,30	35,18	51,34	62,53	68,55	76,27
MAX	99,59	99,90	98,89	99,62	99,65	98,92	99,67	99,64	98,81

E. Performa Uji Coba Skenario 2

Nama Citra	Polynomial Kernel			RBF Kernel			Linear Kernel			Fungsi Sigmoid		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
1	99,38	88,87	93,83	99,15	92,88	95,91	99,57	97,13	98,34	99,87	76,46	86,61
2	99,54	83,68	90,92	99,45	95,56	97,47	95,83	96,21	96,02	99,46	77,22	86,94
3	99,97	52,81	69,11	97,42	92,42	94,85	99,60	89,39	94,22	99,16	86,45	92,37
4	94,50	92,39	93,43	94,39	97,89	96,11	95,05	97,79	96,40	93,15	87,46	90,22
5	99,17	71,09	82,82	98,50	96,92	97,70	97,00	98,23	97,61	98,44	91,21	94,69
6	98,02	92,58	95,22	97,29	98,75	98,01	96,99	98,83	97,90	97,57	90,86	94,09
7	57,71	70,36	63,41	85,65	68,82	76,32	89,84	99,06	94,22	99,70	73,16	84,40
8	98,66	52,45	68,49	97,93	93,71	95,77	98,13	92,43	95,20	97,74	65,75	78,61
9	90,80	44,48	59,71	78,50	53,85	63,88	83,67	98,62	90,53	66,43	90,77	76,72
10	76,22	38,46	51,12	96,97	71,82	82,52	92,89	96,82	94,81	43,88	98,62	60,73
11	99,19	84,07	91,00	98,59	89,51	93,84	97,50	91,35	94,33	98,58	83,06	90,15
12	92,52	95,42	93,95	93,05	91,77	92,41	92,68	93,20	92,94	92,13	36,23	52,01
13	99,82	96,15	97,95	95,71	97,97	96,82	99,31	94,19	96,68	99,63	66,12	79,49
14	90,59	84,76	87,58	90,17	90,88	90,52	91,09	87,36	89,18	88,60	73,17	80,15
15	99,24	92,95	95,99	99,58	91,18	95,20	99,25	92,47	95,74	99,75	83,99	91,19

Nama Citra	Polynomial Kernel			RBF Kernel			Linear Kernel			Fungsi Sigmoid		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
16	81,58	99,69	89,73	94,09	99,68	96,80	94,23	99,64	96,86	87,71	93,30	90,42
17	22,86	98,48	37,11	98,34	74,27	84,63	98,77	85,63	91,73	54,49	87,94	67,28
18	63,24	96,96	76,55	86,92	94,19	90,41	95,75	68,55	79,90	68,59	85,13	75,97
19	33,00	99,90	49,61	71,71	99,21	83,24	96,12	95,30	95,71	79,91	75,99	77,90
20	99,12	98,18	98,65	99,37	98,20	98,78	88,09	99,26	93,34	99,33	50,16	66,66
21	100,0	36,28	53,25	97,87	77,75	86,66	97,48	82,21	89,20	97,86	81,45	88,90
22	88,69	71,13	78,95	90,12	72,56	80,39	80,16	83,52	81,80	68,32	86,32	76,27
23	31,38	98,74	47,63	98,23	97,15	97,69	98,89	90,86	94,71	97,80	72,36	83,18
24	99,56	49,86	66,45	91,44	88,49	89,94	98,10	83,46	90,19	96,13	75,60	84,64
25	88,05	96,31	91,99	85,98	98,93	92,00	85,64	97,11	91,01	85,64	86,02	85,83
26	98,38	25,21	40,14	94,84	59,09	72,81	98,53	99,08	98,81	97,28	40,97	57,65
27	93,29	94,39	93,84	87,66	95,81	91,55	82,23	96,73	88,89	74,44	61,51	67,36
28	99,51	84,23	91,24	96,69	92,25	94,42	92,45	98,87	95,55	95,69	98,53	97,09
29	90,96	93,90	92,41	84,53	98,23	90,87	97,56	94,41	95,96	99,49	53,02	69,18
30	99,97	75,43	85,98	99,60	93,22	96,31	99,67	94,03	96,77	99,73	94,37	96,97
31	47,28	94,38	63,00	96,14	60,88	74,55	86,10	76,70	81,13	67,52	66,08	66,79

Nama Citra	Polynomial Kernel			RBF Kernel			Linear Kernel			Fungsi Sigmoid		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
32	97,57	66,21	78,89	97,09	74,77	84,48	97,17	82,98	89,51	98,21	98,51	98,36
33	92,11	77,64	84,26	91,95	92,67	92,31	95,34	74,94	83,92	92,12	36,80	52,59
34	57,32	91,25	70,41	64,38	96,19	77,13	77,22	89,65	82,97	41,17	90,73	56,64
35	79,95	98,04	88,08	90,96	96,94	93,85	83,73	96,23	89,54	82,63	70,62	76,15
36	25,37	99,23	40,41	85,10	93,77	89,23	92,27	86,83	89,47	90,32	78,85	84,20
37	53,79	86,54	66,34	53,68	97,31	69,20	66,87	97,05	79,18	69,58	62,39	65,79
38	94,77	86,17	90,27	99,50	81,71	89,73	99,33	87,14	92,84	61,67	96,40	75,22
39	46,69	83,10	59,78	77,29	88,32	82,44	82,19	89,79	85,82	85,73	50,74	63,75
40	75,80	96,36	84,85	76,67	98,23	86,12	71,70	97,61	82,67	78,37	95,22	85,98
41	29,12	91,15	44,14	97,39	78,08	86,67	91,75	95,36	93,52	98,92	68,05	80,63
42	72,75	82,83	77,47	83,24	94,26	88,41	62,53	97,73	76,27	74,35	64,80	69,25
43	44,13	92,36	59,72	86,88	88,13	87,50	87,15	91,97	89,49	100,0	6,03	11,37
44	46,17	89,85	61,00	61,33	81,14	69,86	89,95	72,51	80,30	61,18	70,42	65,48
45	28,46	97,80	44,08	76,36	96,20	85,14	75,76	96,10	84,73	90,32	56,57	69,57
46	98,93	91,29	94,95	71,33	97,53	82,40	79,73	99,43	88,50	46,16	93,85	61,88
47	25,49	95,60	40,25	94,05	80,91	86,98	96,87	80,32	87,82	93,64	75,07	83,33

Nama Citra	Polynomial Kernel			RBF Kernel			Linear Kernel			Fungsi Sigmoid		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
48	87,23	89,26	88,23	85,10	95,90	90,18	84,98	95,83	90,08	99,75	40,26	57,37
49	99,36	16,89	28,87	75,21	99,49	85,66	76,38	95,34	84,81	68,92	90,89	78,39
50	99,77	23,64	38,22	96,00	56,07	70,80	83,82	80,70	82,23	92,84	69,57	79,54
AVG	77,74	80,18	72,63	89,19	88,23	87,73	90,26	91,52	90,39	85,40	74,10	76,12
MIN	22,86	16,89	28,87	53,68	53,85	63,88	62,53	68,55	76,27	41,17	6,03	11,37
MAX	100,0	99,90	98,65	99,60	99,68	98,78	99,67	99,64	98,81	100,0	98,62	98,36

F. Performa Uji Coba Skenario 3

Nama Citra	C = 1			C = 32			C = 128		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
1	99,57	97,13	98,34	98,67	99,22	98,94	98,66	99,22	98,94
2	95,83	96,21	96,02	95,87	96,20	96,04	95,87	96,23	96,05
3	99,60	89,39	94,22	99,49	90,03	94,52	99,49	90,02	94,52
4	95,05	97,79	96,40	93,81	97,12	95,44	93,02	98,61	95,73
5	97,00	98,23	97,61	96,02	98,93	97,46	96,03	98,93	97,46
6	96,99	98,83	97,90	96,08	98,99	97,51	96,08	98,99	97,51
7	89,84	99,06	94,22	87,60	99,45	93,15	87,16	99,54	92,94
8	98,13	92,43	95,20	98,28	88,69	93,24	98,19	90,07	93,95
9	83,67	98,62	90,53	79,18	79,25	79,22	79,17	79,17	79,17
10	92,89	96,82	94,81	87,81	96,37	91,89	87,36	96,35	91,63
11	97,50	91,35	94,33	96,49	92,28	94,34	96,15	92,31	94,19
12	92,68	93,20	92,94	92,10	93,70	92,89	92,10	93,70	92,89
13	99,31	94,19	96,68	99,24	94,54	96,83	99,23	95,86	97,51
14	91,09	87,36	89,18	89,15	88,21	88,67	89,14	88,21	88,67
15	99,25	92,47	95,74	99,01	92,72	95,76	98,99	92,80	95,80
16	94,23	99,64	96,86	93,61	99,67	96,54	93,61	99,67	96,54

Nama Citra	C = 1			C = 32			C = 128		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
17	98,77	85,63	91,73	98,56	85,82	91,75	98,61	68,24	80,66
18	95,75	68,55	79,90	96,64	56,74	71,50	96,64	56,74	71,50
19	96,12	95,30	95,71	87,64	97,09	92,12	81,88	98,90	89,59
20	88,09	99,26	93,34	83,95	99,56	91,09	86,46	99,51	92,53
21	97,48	82,21	89,20	97,38	75,24	84,89	97,38	75,23	84,88
22	80,16	83,52	81,80	79,02	83,47	81,18	79,02	83,47	81,18
23	98,89	90,86	94,71	98,57	90,90	94,58	98,56	90,90	94,57
24	98,10	83,46	90,19	98,92	80,77	88,93	98,92	80,16	88,56
25	85,64	97,11	91,01	85,60	98,49	91,60	85,50	98,55	91,56
26	98,53	99,08	98,81	98,41	99,09	98,75	98,01	99,11	98,56
27	82,23	96,73	88,89	80,67	98,29	88,61	80,67	98,30	88,62
28	92,45	98,87	95,55	93,22	98,93	95,99	94,11	98,82	96,41
29	97,56	94,41	95,96	87,24	97,00	91,86	87,00	97,12	91,78
30	99,67	94,03	96,77	99,57	94,76	97,11	99,57	94,74	97,09
31	86,10	76,70	81,13	86,01	76,86	81,18	85,97	76,59	81,01
32	97,17	82,98	89,51	97,15	82,74	89,37	97,17	82,70	89,36
33	95,34	74,94	83,92	94,86	75,50	84,08	94,86	75,50	84,08

Nama Citra	C = 1			C = 32			C = 128		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
34	77,22	89,65	82,97	77,61	91,74	84,09	85,78	86,56	86,17
35	83,73	96,23	89,54	85,12	95,79	90,14	85,12	95,79	90,14
36	92,27	86,83	89,47	87,67	91,68	89,63	87,69	91,97	89,78
37	66,87	97,05	79,18	55,41	97,58	70,68	53,80	97,61	69,36
38	99,33	87,14	92,84	99,26	90,73	94,80	99,26	90,75	94,81
39	82,19	89,79	85,82	82,03	85,64	83,80	83,07	85,70	84,36
40	71,70	97,61	82,67	68,81	98,99	81,18	69,15	98,94	81,41
41	91,75	95,36	93,52	96,97	90,06	93,39	96,97	90,05	93,38
42	62,53	97,73	76,27	64,91	96,82	77,72	64,22	96,82	77,22
43	87,15	91,97	89,49	89,48	87,29	88,37	89,45	87,05	88,24
44	89,95	72,51	80,30	89,58	72,62	80,21	89,72	72,63	80,28
45	75,76	96,10	84,73	76,41	96,12	85,14	76,41	96,12	85,14
46	79,73	99,43	88,50	98,42	99,27	98,85	98,40	99,26	98,83
47	96,87	80,32	87,82	95,30	52,81	67,96	95,26	52,32	67,54
48	84,98	95,83	90,08	84,49	96,35	90,03	84,49	96,35	90,03
49	76,38	95,34	84,81	76,38	95,33	84,81	76,37	95,33	84,80
50	83,82	80,70	82,23	85,05	81,51	83,24	83,65	81,71	82,67

Nama Citra	C = 1			C = 32			C = 128		
	Presisi	**Recall**	**F1 Score**	**Presisi**	**Recall**	**F1 Score**	**Presisi**	**Recall**	**F1 Score**
AVG	90,26	91,52	90,39	89,57	90,34	89,22	89,59	89,98	88,99
MIN	62,53	68,55	76,27	55,41	52,81	67,96	53,80	52,32	67,54
MAX	99,67	99,64	98,81	99,57	99,67	98,94	99,57	99,67	98,94

G. Performa Uji Coba Skenario 4

Nama Citra	Threshold = 1x			Threshold = 1,5x			Threshold = 2x		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
1	97,89	99,48	98,68	99,57	97,13	98,34	99,61	91,97	95,64
2	95,78	96,46	96,12	95,83	96,21	96,02	0,00	0,00	0,00
3	97,66	92,62	95,07	99,60	89,39	94,22	99,70	88,72	93,89
4	93,56	98,77	96,09	95,05	97,79	96,40	95,52	95,63	95,57
5	95,51	99,26	97,35	97,00	98,23	97,61	97,23	98,00	97,61
6	96,22	99,14	97,66	96,99	98,83	97,90	98,57	72,48	83,53
7	84,01	99,71	91,19	89,84	99,06	94,22	89,68	99,08	94,14
8	97,21	95,17	96,18	98,13	92,43	95,20	97,69	92,26	94,90
9	82,78	98,95	90,14	83,67	98,62	90,53	0,00	0,00	0,00
10	86,95	97,22	91,80	92,89	96,82	94,81	96,32	96,08	96,20
11	95,88	93,19	94,51	97,50	91,35	94,33	96,94	91,89	94,34
12	91,75	94,57	93,14	92,68	93,20	92,94	93,26	90,93	92,08
13	99,22	94,69	96,90	99,31	94,19	96,68	99,51	93,92	96,63
14	89,59	88,73	89,16	91,09	87,36	89,18	91,15	86,81	88,92
15	96,11	95,37	95,74	99,25	92,47	95,74	99,05	93,27	96,08
16	92,38	99,70	95,90	94,23	99,64	96,86	97,67	99,60	98,63

Nama Citra	Threshold = 1x			Threshold = 1,5x			Threshold = 2x		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
17	98,37	88,68	93,27	98,77	85,63	91,73	98,84	85,94	91,94
18	90,06	88,45	89,25	95,75	68,55	79,90	95,61	79,02	86,53
19	85,97	97,08	91,19	96,12	95,30	95,71	96,82	95,27	96,04
20	76,99	99,66	86,87	88,09	99,26	93,34	89,57	99,47	94,26
21	96,86	88,65	92,57	97,48	82,21	89,20	97,25	83,45	89,83
22	70,76	84,48	77,01	80,16	83,52	81,80	76,89	83,87	80,23
23	98,39	92,07	95,12	98,89	90,86	94,71	99,08	90,18	94,42
24	96,23	88,88	92,41	98,10	83,46	90,19	98,75	82,17	89,70
25	84,70	99,32	91,43	85,64	97,11	91,01	85,35	98,73	91,55
26	97,65	99,38	98,51	98,53	99,08	98,81	98,33	96,85	97,58
27	80,15	98,43	88,35	82,23	96,73	88,89	82,65	95,25	88,50
28	89,80	99,30	94,31	92,45	98,87	95,55	93,84	98,69	96,21
29	74,29	97,27	84,24	97,56	94,41	95,96	97,54	94,43	95,96
30	99,32	95,29	97,26	99,67	94,03	96,77	99,69	89,88	94,53
31	67,14	99,73	80,25	86,10	76,70	81,13	77,55	77,99	77,77
32	91,59	99,42	95,35	97,17	82,98	89,51	97,15	83,07	89,56
33	94,89	92,44	93,65	95,34	74,94	83,92	94,85	75,59	84,14

Nama Citra	Threshold = 1x			Threshold = 1,5x			Threshold = 2x		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
34	56,58	97,57	71,62	77,22	89,65	82,97	78,29	89,77	83,63
35	83,19	96,98	89,55	83,73	96,23	89,54	92,58	93,28	92,93
36	86,91	91,73	89,26	92,27	86,83	89,47	92,09	87,57	89,77
37	55,57	97,61	70,82	66,87	97,05	79,18	66,71	97,16	79,10
38	99,12	89,91	94,29	99,33	87,14	92,84	99,32	85,42	91,85
39	76,97	92,83	84,16	82,19	89,79	85,82	83,23	89,89	86,43
40	65,19	97,63	78,17	71,70	97,61	82,67	68,78	97,35	80,61
41	34,76	98,61	51,40	91,75	95,36	93,52	96,72	93,90	95,29
42	57,68	98,50	72,75	62,53	97,73	76,27	65,01	96,97	77,84
43	87,38	99,15	92,89	87,15	91,97	89,49	87,19	93,76	90,36
44	72,41	76,77	74,53	89,95	72,51	80,30	78,92	75,21	77,02
45	74,46	96,41	84,02	75,76	96,10	84,73	0,00	0,00	0,00
46	74,18	99,50	85,00	79,73	99,43	88,50	80,29	99,45	88,85
47	93,40	81,54	87,07	96,87	80,32	87,82	96,47	81,17	88,16
48	83,99	96,52	89,82	84,98	95,83	90,08	87,30	94,91	90,95
49	64,20	98,09	77,61	76,38	95,34	84,81	76,37	95,37	84,82
50	83,22	81,74	82,47	83,82	80,70	82,23	83,70	81,60	82,63

Nama Citra	Threshold = 1x			Threshold = 1,5x			Threshold = 2x		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
AVG	84,70	94,85	88,64	90,26	91,52	90,39	85,29	85,07	84,74
MIN	34,76	76,77	51,40	62,53	68,55	76,27	0,00	0,00	0,00
MAX	99,32	99,73	98,68	99,67	99,64	98,81	99,70	99,60	98,63

H. Performa Uji Coba Skenario 5

Nama Citra	$SMap_{trained}$ = 60%			$SMap_{trained}$ = 70%			$SMap_{trained}$ = 80%			$SMap_{trained}$ = 100%		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
1	99,07	91,04	94,89	99,11	96,31	97,69	99,10	96,44	97,75	99,57	97,13	98,34
2	94,08	95,77	94,92	95,46	96,43	95,94	95,75	96,35	96,05	95,83	96,21	96,02
3	98,66	91,28	94,83	99,06	90,55	94,62	99,52	89,95	94,49	99,60	89,39	94,22
4	95,05	95,71	95,38	95,26	95,84	95,55	95,16	95,97	95,56	95,05	97,79	96,40
5	96,52	96,45	96,48	96,80	98,59	97,69	96,89	98,39	97,64	97,00	98,23	97,61
6	97,11	96,90	97,00	97,11	98,75	97,92	97,16	98,75	97,95	96,99	98,83	97,90
7	89,55	99,22	94,14	90,66	99,10	94,69	89,58	99,08	94,09	89,84	99,06	94,22
8	96,00	94,57	95,28	97,44	94,24	95,82	97,59	94,02	95,77	98,13	92,43	95,20
9	80,99	98,82	89,02	82,83	98,79	90,11	83,61	98,69	90,53	83,67	98,62	90,53
10	90,55	67,63	77,43	94,90	96,55	95,72	96,45	96,19	96,32	92,89	96,82	94,81
11	97,36	90,93	94,04	97,75	91,39	94,47	97,81	91,39	94,50	97,50	91,35	94,33
12	93,62	79,52	86,00	92,87	92,61	92,74	92,88	92,84	92,86	92,68	93,20	92,94
13	97,80	94,33	96,03	99,29	94,30	96,73	99,29	94,30	96,73	99,31	94,19	96,68
14	90,67	87,00	88,80	90,73	87,65	89,16	90,78	87,64	89,18	91,09	87,36	89,18
15	99,19	84,20	91,08	99,25	90,94	94,92	99,25	91,01	94,95	99,25	92,47	95,74

Nama Citra	$SMap_{trained} = 60\%$			$SMap_{trained} = 70\%$			$SMap_{trained} = 80\%$			$SMap_{trained} = 100\%$		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
16	92,43	95,42	93,90	93,91	99,69	96,71	94,80	99,65	97,17	94,23	99,64	96,86
17	91,92	85,41	88,55	98,57	86,22	91,98	98,56	86,70	92,25	98,77	85,63	91,73
18	96,76	75,39	84,75	96,57	76,66	85,47	95,96	78,62	86,43	95,75	68,55	79,90
19	91,89	84,66	88,12	94,16	95,19	94,67	95,43	95,20	95,31	96,12	95,30	95,71
20	93,48	93,08	93,28	94,37	99,46	96,85	93,50	99,45	96,38	88,09	99,26	93,34
21	96,42	76,55	85,34	96,96	76,02	85,22	97,03	75,52	84,93	97,48	82,21	89,20
22	82,24	81,02	81,63	81,75	83,30	82,52	81,84	83,19	82,51	80,16	83,52	81,80
23	98,17	90,02	93,92	98,65	90,88	94,61	98,65	90,88	94,61	98,89	90,86	94,71
24	92,70	84,37	88,34	95,42	83,85	89,26	96,43	83,55	89,53	98,10	83,46	90,19
25	85,09	95,86	90,15	85,27	96,07	90,35	85,34	96,68	90,66	85,64	97,11	91,01
26	96,79	74,55	84,22	97,75	80,36	88,21	98,30	99,20	98,75	98,53	99,08	98,81
27	76,76	70,01	73,23	81,49	92,33	86,57	81,93	95,56	88,23	82,23	96,73	88,89
28	90,15	85,88	87,96	91,53	99,05	95,14	91,41	99,08	95,09	92,45	98,87	95,55
29	88,01	97,24	92,39	91,38	96,95	94,08	95,70	96,06	95,88	97,56	94,41	95,96
30	99,09	95,15	97,08	99,60	94,69	97,08	99,60	94,48	96,97	99,67	94,03	96,77
31	77,03	72,14	74,51	75,97	78,74	77,33	76,95	78,36	77,65	86,10	76,70	81,13

Nama Citra	$SMap_{trained} = 60\%$			$SMap_{trained} = 70\%$			$SMap_{trained} = 80\%$			$SMap_{trained} = 100\%$		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
32	91,59	87,20	89,34	94,61	85,67	89,92	96,68	83,74	89,75	97,17	82,98	89,51
33	91,34	70,10	79,32	94,55	70,14	80,53	94,55	73,61	82,78	95,34	74,94	83,92
34	67,67	91,87	77,94	70,65	94,73	80,94	74,47	93,18	82,78	77,22	89,65	82,97
35	83,53	84,58	84,05	84,49	94,86	89,37	84,67	96,29	90,11	83,73	96,23	89,54
36	91,73	86,19	88,87	92,12	85,95	88,93	92,08	85,48	88,65	92,27	86,83	89,47
37	65,72	81,37	72,71	66,77	97,35	79,21	66,96	97,07	79,25	66,87	97,05	79,18
38	99,27	82,85	90,32	99,29	85,85	92,08	99,29	85,77	92,04	99,33	87,14	92,84
39	82,41	86,13	84,23	83,79	86,32	85,03	83,65	86,69	85,14	82,19	89,79	85,82
40	71,68	92,29	80,69	72,57	97,30	83,14	72,23	97,30	82,91	71,70	97,61	82,67
41	60,90	97,80	75,06	62,94	97,72	76,56	64,43	97,70	77,65	91,75	95,36	93,52
42	77,88	94,62	85,44	76,71	94,83	84,81	75,22	95,25	84,06	62,53	97,73	76,27
43	87,14	80,96	83,93	87,78	89,01	88,39	87,64	89,47	88,54	87,15	91,97	89,49
44	78,14	70,75	74,26	78,16	72,17	75,05	78,65	74,31	76,42	89,95	72,51	80,30
45	76,84	93,73	84,45	74,56	96,51	84,13	75,83	96,26	84,83	75,76	96,10	84,73
46	87,77	88,95	88,35	85,47	90,64	87,98	80,26	99,30	88,77	79,73	99,43	88,50
47	93,53	56,66	70,57	95,18	59,05	72,89	96,29	77,92	86,13	96,87	80,32	87,82

Nama Citra	$SMap_{trained} = 60\%$			$SMap_{trained} = 70\%$			$SMap_{trained} = 80\%$			$SMap_{trained} = 100\%$		
	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score	Presisi	Recall	F1 Score
48	85,30	95,80	90,25	85,34	95,76	90,25	85,37	95,75	90,26	84,98	95,83	90,08
49	79,56	98,32	87,95	79,95	97,86	88,00	77,68	95,60	85,71	76,38	95,34	84,81
50	81,37	68,71	74,50	81,80	69,43	75,11	81,81	70,11	75,51	83,82	80,70	82,23
AVG	88,37	86,58	86,90	89,37	90,25	89,24	89,60	91,28	89,96	90,26	91,52	90,39
MIN	60,90	56,66	70,57	62,94	59,05	72,89	64,43	70,11	75,51	62,53	68,55	76,27
MAX	99,27	99,22	97,08	99,60	99,69	97,92	99,60	99,65	98,75	99,67	99,64	98,81

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Gian Sebastian Anjasmara, lahir di Bandung, pada tanggal 18 Mei 1995. Penulis menempuh pendidikan mulai dari SD Permata Harapan (2001-2007), SMP Santa Angela (2007-2010), SMA Santa Angela (2010-2013) hingga terakhir di Institut Teknologi Sepuluh Nopember Surabaya (2013-2017) di jurusan Teknik Informatika, Fakultas Teknologi Informasi angkatan tahun 2013.

Selama belajar di kampus Teknik Informatika, penulis berkesempatan menjadi asisten dosen Komputasi Numerik (2015), asisten dosen Pemrograman Web (2016), dan asisten dosen Kecerdasan Buatan (2017). Selain mengikuti kegiatan akademik, penulis mengikuti kegiatan organisasi dengan berpartisipasi sebagai staf Biro Keamanan dan Transportasi Schematics pada tahun 2014 dan menjadi anggota Sie Dana dan Usaha Schematics pada tahun 2015.

Penulis memiliki bidang minat Komputasi Cerdas Visi (KCV) dengan fokus studi pada bidang *image processing* dan *data mining*. Komunikasi dengan penulis dapat melalui email: **giantcrabi@gmail.com**.