

DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

**SEMI-SUPERVISED LEARNING FOR IMAGE
SEGMENTATION**

by
Gökhan TİĞİLSEL

February, 2016
İZMİR

SEMI-SUPERVISED LEARNING FOR IMAGE SEGMENTATION

**A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science of
Electrical and Electronics Engineering**

**by
Gökhan TIĞİLSEL**

**February, 2016
İZMİR**

M.Sc. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled “SEMI-SUPERVISED LEARNING FOR IMAGE SEGMENTATION” completed by GÖKHAN TIĞILSEL under supervision of ASST. PROF. DR. GÜLESER KALAYCI DEMİR and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



Asst. Prof. Dr. Güleser Kalaycı Demir

Supervisor



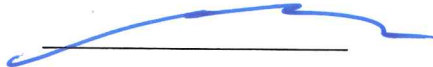
Doç. Dr. M. Alper SELVER

Jury Member



Asst. Prof. Dr. Derya BİRANT

Jury Member



Prof. Dr. Ayşe OKUR

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

I wish to express my special thanks to my supervisor Asst. Prof. Dr. Güleser Kalaycı Demir, whose support, knowledge and insight throughout all stages of my work were of enormous benefit and value.

I would like to thank my colleagues Aykut Yanık, Kemal Büyükabalı, Mehmet Erişir, Hakan Yılmaz, Mahir Ulutaş and Altay Erakman, for their encouragement and generous support over the years.

This thesis is dedicated to my wife, my daughter, my sister and to my dear parents.

Gökhan TİĞİLSEL

SEMI-SUPERVISED LEARNING FOR IMAGE SEGMENTATION

ABSTRACT

In this thesis, a new approach to semi-supervised image segmentation is proposed, by implementing Unconstraint Least Square Importance Fitting (ULSIF) algorithm. ULSIF estimates importance value without determining probability distribution functions of the labeled and test class separately. Different methods based on color or texture features are applied on semi-supervised image segmentation problems and similarities and differences between these methods have been explored. The performance of ULSIF based segmentation is compared with the state of art method one-class SVM in terms of segmentation accuracy and run time. According to the results, both color and texture based semi-supervised one-class ULSIF gave better results than one-class SVM does.

Additionally, ULSIF algorithm is implemented as a two-class image segmentation method. In two class image segmentation methods, features of the foreground and background areas are used together during the learning stages. Results of supervised two-class ULSIF is compared with SVM and neural networks. According to the results, two-class SVM gave better results than two-class ULSIF and neural networks do.

Keywords : Semi-supervised learning, image segmentation, ULSIF, one class SVM.

GÖRÜNTÜ BÖLÜTLEME UYGULAMALARI İÇİN YARI-EĞİTİCİLİ ÖĞRENME

ÖZ

Bu tez çalışmasında, Unconstraint Least Square Importance Fitting (ULSIF) algoritmasını kullanan, yeni bir yarı-eğitici öğrenme metodu önerilmiştir. ULSIF algoritması önem değerini doğrudan hesaplabilmektedir, bunun için test sınıflarına ait olasılık dağılım fonksiyonlarını ayrı ayrı hesaplamaya ihtiyaç duymamaktadır. Görüntü bölütleme problemi üzerinde renk ve örüntü özelliklerini kullanan farklı yarı-eğitici öğrenme metodları kullanılmış, bu metodların benzerlikleri ve farkları incelenmiştir. ULSIF algoritmasının bölütleme performansı, yaygın olarak bilinmekte olan tek-sınıflı SVM metoduyla, tutarlılık ve çalışma süresi bakımından karşılaştırılmıştır. Sonuç olarak, tek-sınıflı ULSIF metodu hem renk hem de örüntü tabanlı görüntü bölütleme probleminde tek-sınıflı SVM'den daha iyi performans göstermiştir.

Ayrıca, ULSIF algoritması iki-sınıflı görüntü bölütleme metodu olarak da uyarlanmıştır. İki-sınıflı görüntü bölütleme metodlarında, öğrenme işlemi sırasında hem önplan hem de arkaplan özellikleri birlikte kullanılmıştır. İki-sınıflı ULSIF görüntü bölütleme metodunun çıktıları, iki-sınıflı SVM ve yapay zeka algoritmaları ile kıyaslanmıştır. Sonuç olarak, iki-sınıflı SVM algoritması iki-sınıflı ULSIF ve yapay zeka algoritmalarına göre daha iyi performans göstermiştir.

Anahtar Kelimeler : Yarı-eğitici öğrenme, görüntü bölütleme, ULSIF, tek sınıflı SVM.

CONTENTS

	Page
THESIS EXAMINATION RESULT FORM.....	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZ.....	v
LIST OF FIGURES.....	viii
LIST OF TABLES	x
CHAPTER ONE - INTRODUCTION	1
CHAPTER TWO – FEATURE EXTRACTION	4
2.1 Features.....	4
2.1.1 Color Feature Set	4
2.1.2 Texture Feature Set	5
CHAPTER THREE – IMAGE SEGMENTATION GRAPHICAL USER INTERFACE IMPLEMENTATION	8
3.1 Image Segmentation GUI Implementation.....	8
3.1.1 Selecting Input Image and Displaying in GUI.....	9
3.1.2 Extracting Texture Features of Input Image	9
3.1.3 Selecting Labeled Foreground and Background Pixels	10
3.1.4 Extracting Features of User Labeled Pixels.....	11
3.2 Proposed Segmentation Methods According to Feature Set	11
CHAPTER FOUR – IMAGE SEGMENTATION ALGORITHMS	14
4.1 Segmentation Methods	14
4.1.1 ULSIF	14
4.1.2 Support Vector Machines for Image Segmentation.....	16
4.1.2.1 Two-Class SVM.....	16

4.1.2.2 One-Class SVM (OSVM)	17
4.1.3 Neural Networks for Image Segmentation	19
4.2 Implementation of Image Segmentation Algorithms	22
4.2.1 Semi-Supervised Methods	22
4.2.1.1 Semi-Supervised C_ULSIF.....	22
4.2.1.2 Semi-Supervised T_ULSIF.....	24
4.2.1.3 Semi-Supervised C_OSVM	25
4.2.1.4 Semi-Supervised T_OSVM	25
4.2.2 Supervised Methods.....	26
4.2.2.1 Supervised C2_ULSIF	26
4.2.2.2 Supervised T2_ULSIF	31
4.2.2.3 Supervised C2_SVM.....	31
4.2.2.4 Supervised T2_SVM.....	32
4.2.2.5 Supervised C2_NN.....	32
4.2.2.6 Supervised T2_NN.....	33
CHAPTER FIVE – TEST AND RESULTS.....	38
5.1 Performance of Semi-Supervised One-Class Segmentation Methods Using Color Feature Set	38
5.2 Performance of Supervised Two-Class Segmentation Methods Using Color Feature Set	40
5.3 Performance of Semi-Supervised One-Class Segmentation Methods Using Texture Feature Set	43
5.4 Performance of Supervised Two-Class Segmentation Methods Using Texture Feature Set.....	48
CHAPTER SIX – CONCLUSION.....	51
REFERENCES	52

LIST OF FIGURES

	Page
Figure 2.1	Image examples which contain texture information 5
Figure 2.2	GLCM directions and neighborhood region..... 6
Figure 3.1	Image segmentation tool GUI menu..... 8
Figure 3.2	Input image selection and displaying in GUI..... 9
Figure 3.3	Extracting full image texture features by block processing 9
Figure 3.4	Extracting texture feature with mxn pixel neighborhood region..... 10
Figure 3.5	Selection foreground pixels on the image by mouse cursor..... 10
Figure 3.6	Flowchart of selecting foreground or background pixels..... 10
Figure 3.7	Extracting labeled foreground and background features..... 12
Figure 3.8	Categorization of proposed algorithms 12
Figure 4.1	SVM examples 16
Figure 4.2	SVM optimal hyperplane 16
Figure 4.3	Representation of human neuron system..... 20
Figure 4.4	Single layer feedforward neural network diagrams..... 21
Figure 4.5	(a) Multilayer feedforward neural network example with its weights (b) Proposed pattern recognition network in this thesis 21
Figure 4.6	Flowchart of semi-supervised C_ULSIF algorithm 23
Figure 4.7	Flowchart of semi-supervised T_ULSIF algorithm 24
Figure 4.8	Flowchart of semi-supervised C_OSVM algorithm..... 27
Figure 4.9	Flowchart of semi-supervised T_OSVM algorithm..... 28
Figure 4.10	Flowchart of supervised C2_ULSIF algorithm 29
Figure 4.11	Flowchart of supervised T2_ULSIF algorithm 30
Figure 4.12	Flowchart of supervised C2_SVM algorithm..... 34
Figure 4.13	Flowchart of supervised T2_SVM algorithm..... 35
Figure 4.14	Flowchart of supervised C2_NN algorithm 36
Figure 4.15	Flowchart of supervised T2_NN algorithm..... 37
Figure 5.1	Semi-supervised color segmentation results for sea-star..... 38
Figure 5.2	Semi-supervised color segmentation results for flower 39
Figure 5.3	Semi-supervised color segmentation results for church..... 39

Figure 5.4	Supervised color segmentation results for sea-star	41
Figure 5.5	Supervised color segmentation results for flower	41
Figure 5.6	Supervised color segmentation results for church	42
Figure 5.7	Semi-supervised texture segmentation results while choosing parameter b for ULSIF and outlier fraction parameter for OSVM..	43
Figure 5.8	Semi-supervised texture segmentation results while choosing size of texture blocks.....	45
Figure 5.9	Semi-supervised texture segmentation results while testing 20x20 sized overlapping texture blocks	46
Figure 5.10	Semi-supervised texture segmentation results while testing 20x20 sized non-overlapping texture blocks	46
Figure 5.11	Supervised texture segmentation results while testing 20x20 sized overlapping texture blocks.....	48
Figure 5.12	Supervised texture segmentation results while testing 20x20 sized overlapping texture blocks, for different labeled regions.....	50

LIST OF TABLES

	Page
Table 2.1 Hue angle and its representative color.....	4
Table 2.2 Haralick feature definition and formulas	7
Table 5.1 Performance of semi-supervised methods using color feature set	40
Table 5.2 Performance of supervised methods using color feature set	42
Table 5.3 Performance of semi-supervised methods while choosing parameter b for ULSIF and outlier fraction parameter for OSVM.....	44
Table 5.4 Performance of semi-supervised methods with 20x20 sized overlapping texture blocks	47
Table 5.5 Performance of semi-supervised methods with 20x20 sized non- overlapping texture blocks	47
Table 5.6 Performance of supervised methods in Figure 5.11	49
Table 5.7 Performance of supervised methods in Figure 5.12	50

CHAPTER ONE

INTRODUCTION

Image segmentation has an important role in computer vision, machine learning, image enhancement, medical image processing, video understanding, video tracking, and etc. Kapur, Grimson, Wells, & Kikinis (1996) worked on segmentation of brain tissue from magnetic resonance images. Mikic, Cosman, Kogut, & Triverdi (2000) presented an algorithm for segmentation of traffic scenes that distinguishes moving objects from their moving cast shadows. Needham & Boyle (2001) presented a framework that tracks the movements of sports (specifically football) players, from a single fixed camera, on an indoor court. Stringa, & Regazzoni (2002) proposed a system which aims to detect the presence of abandoned objects in a guarded environment.

Many segmentation algorithms have been applied on different problems, and these algorithms can be divided into three main categories according to the learning methods they use. These three learning methods are i) unsupervised learning, ii) supervised learning, iii) semi-supervised learning.

i. The aim of unsupervised learning: is to understand hidden structure in the unlabelled data and this type of learning method is closely related to statistical density estimation. The most common and known unsupervised learning techniques are k-means and self-organizing map (SOM).

ii. The goal of supervised learning is here to find a transfer function by using a training set. This set consists of pairs (x_i, y_i) , each x_i is a sample (contains features), and y_i is called as class label. In the training set, we need to have samples from different classes. For example in an image segmentation problem, we need to have some labeled samples from foreground and background. Artificial neural networks, decision tree learning, nearest neighbor algorithm, support vector machines (SVM), and naïve bayes classifier are some common supervised learning techniques.

iii. Semi-supervised learning is a method between supervised and unsupervised learning. A large set of unlabelled data with some (very few) labeled data is provided for this kind of learning algorithms. To produce a considerable improvement in

learning accuracy, this type of algorithms does not use only labeled data, they also use the unlabelled data for training. Obtaining supervised data has a cost of money, time and effort. Because of that, semi-supervised learning algorithms offer some advantages (Chapelle, Schölkopf, & Zien, 2006). For example, Supervised hyperspectral image classification is a difficult problem because of the unbalance between the high dimensionality of the data, also there is a limited availability of labeled training samples.

Guillaumin, Verbeek, & Schmid (2010), introduced Multimodal semi-supervised learning method for image categorization problem on a large image database. They used image features and keywords of labeled images together. Multiple Kernel Learning (MKL), least-squares regression (LSR), and SVM are used in their problem.

Tuia, & Camps-Valls (2011) had bird's-eye view images of urban places with very high resolution. They proposed a semi-supervised method SVM using bagged kernels.

Dopido, Jun Li, Marpu, & Plaza (2012) developed a new semi-supervised method to analyse surface of the Earth by using images obtained from aerospace laboratories. They used Multinomial Logistic Regression (MLR) and Probabilistic SVM as semi-supervised segmentation methods.

By using both of the labeled foreground and labeled background pixels during training step, Ensemble of Laplacian SVMs are used as a semi-supervised learning method. This method performs less error-rate (with the percentage of mislabeled pixels) compared to other algorithms such as Grabcut, Random Walker, and etc (Jiazhen, Xinmeng, & Xuejuan, 2008).

Novelty selection method is adapted into a semi-supervised segmentation algorithm which classifies unlabelled data by using the label of their nearest neighbor

in the representative set. This method performs faster segmentation with respect to algorithms without novelty selection (Paiva, & Tasdizen, 2010).

Gaussian mixture model introduced for image segmentation as a semi-supervised method. This method uses semi-supervised expectation maximization technique and gives better classification rate (Martínez-Usó, Pla, & Sotoca, 2010).

In another previous work (Peng, Zhang, Zhang, & Yang, 2011), Iterated Region Merging with Localized Graph Cuts (IRM-LGC) algorithm is introduced as a semi-supervised method. In each iteration only local neighboring regions are labeled. IRM-LGC gives better results with respect to other Grabcut algorithms.

All these algorithms above use both of the labeled foreground and labeled background pixels in the training set. This thesis is aimed to introduce a new semi-supervised image segmentation algorithm which is using only labeled one class information.

Kanamori, Hido, & Sugiyama (2009), introduced Unconstrained Least-squares Importance Fitting (ULSIF) algorithm which can be used as inlier based outlier detector. We are going to integrate this method into semi-supervised image segmentation problems directly. Our semi-supervised ULSIF based image segmentation algorithm is going to use color or texture features of the labeled pixels.

Furthermore, semi-supervised ULSIF based image segmentation method can be applied to supervised ULSIF by using features belong to the second class. Its algorithm detail is explained in subsequent chapters.

Performance of color or texture based ULSIF methods are compared with each other and also with the well known methods in the literature such as one-class SVM, and Neural Networks.

CHAPTER TWO

FEATURE EXTRACTION

2.1 Features

Hue-saturation-value (HSV) color set and Gray level co-occurrence matrix (GLCM) Haralick feature set are decided to use in this thesis. Detailed explanation about these features are given in the subsequent sections.

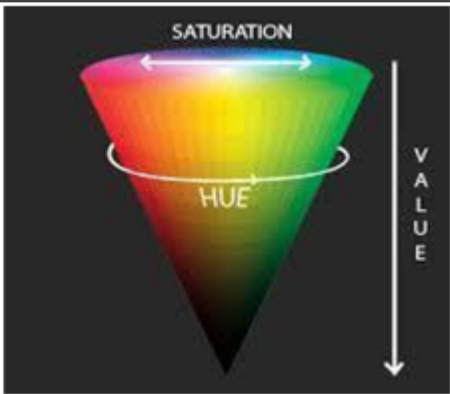
2.1.1 Color Feature Set

For image segmentation problems, it is preferred to use HSV color information as feature. The algorithms using HSV colorspace as a feature set performs better segmentation results with respect to other algorithms using RGB or LAB colorspace (Sural, Gang Qian, & Pramanik, 2002). (Bora, Gupta, & Khan, 2015).

Hue-Saturation-Value color model is introduced by Alvy Ray Smith in 1978. In this model:

Hue is defined between 0 to 360° for separating different colors. Saturation has a range 0% to 100% and represents the amount of gray. Value takes the ratio between 0% to 100% and represents brightness or intensity of the color and Table 2.2 shows the relation between Hue Saturation and Value.

Table 2.1 Hue angle and its representative color (D'Silva, 2008).

Angle	Color	
0-60	Red	
60-120	Yellow	
120-180	Green	
180-240	Cyan	
240-300	Blue	
300-360	Magenta	

2.1.2 Texture Feature Set

Rao (2009), explains texture in his lecture notes with the words “An image obeying some statistical properties. Similar structures repeated over and over again. Often has some degree of randomness ... A texture is a set of texture elements or texels occurring in some regular or repeated pattern.” Texture can give us information about all pixels of the image or a selected region of it. Below Figure 2.4 gives texture image examples.

Several studies based on the texture features have been proposed to improve segmentation results in the literature. Eleyan, & Demirel (2011), used GLCM method to extract Haralick features while working on face recognition algorithm. Then, Mohanaiah, Sathyanarayana, & Lokku 2013, used again GLCM method while extracting texture features of cartoon images.

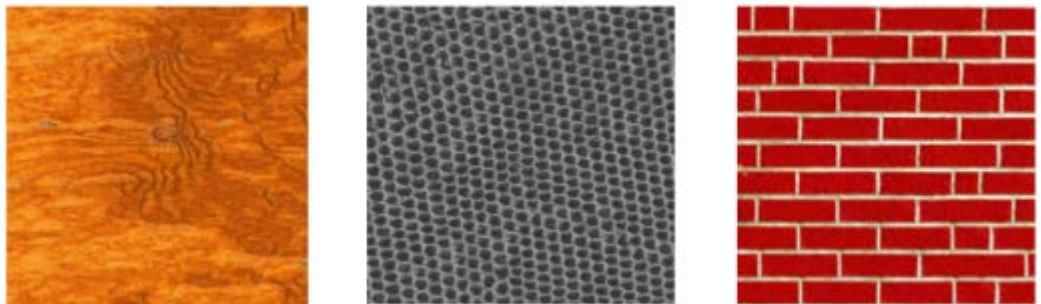


Figure 2.1 Image examples which contain texture information (Rao, & Chen, 2009).

GLCM is a statistical method which calculates spatial relationship of pixels, it shows how often specific pixel values, pairs as (i, j) , are neighboring each other (Eleyan & Demirel, 2011). By using GLCM, several statistical texture features have been proposed in the literature (ref: Haralick, etc.).

After changing our image colorspace, from RGB to Gray-level, Ng is the number of gray levels in the converted image.

Let G is our square form GLCM and $i = \{1, 2, \dots, N_g\}$, $j = \{1, 2, \dots, N_g\}$. Then G can be written as:

$$G = \begin{bmatrix} p(1,1) & \dots & p(1, N_g) \\ \vdots & \ddots & \vdots \\ p(N_g, 1) & \dots & p(N_g, N_g) \end{bmatrix} \quad (2.1)$$

where $p(i, j)$ for $i = \{1, 2, \dots, N_g\}$, $j = \{1, 2, \dots, N_g\}$ is the number of cooccurrence value in the given direction and in the neighborhood region of GLCM. This direction and neighborhood region is shown in Figure 2.2.

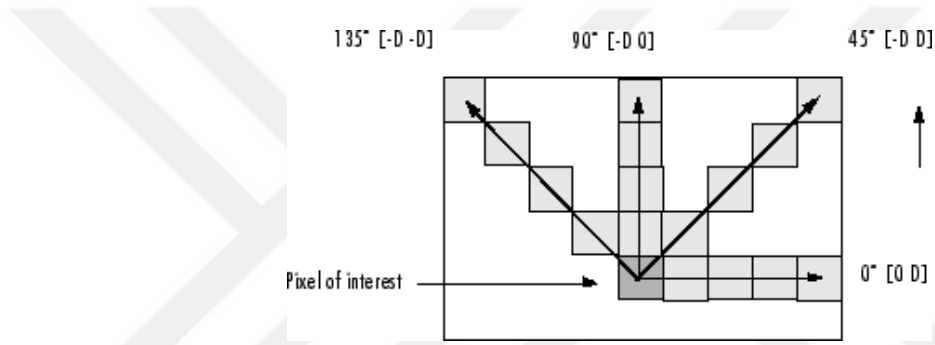


Figure 2.2 GLCM directions and neighborhood region (Specify Offset Used in GLCM Calculation, n.d.)

By using GLCM some statistical features can be extracted, and Haralick introduced 14 different statistics, but generally not all of these texture features are used at the same time, because of efficiency, complexity and run time issues. Therefore, Contrast, Energy, and Homogeneity are chosen as texture features in this thesis study.

Contrast measures the local variations in the gray-level co-occurrence matrix. Returns a measure of the intensity contrast between a pixel and its neighbor over the whole image. Contrast is 0 for a constant image.

Energy, provides the sum of squared elements in the GLCM. Also known as uniformity or the angular second moment. Returns the sum of squared elements in the GLCM. Energy is 1 for a constant image.

Homogeneity, measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal. Returns a value that measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal. Homogeneity is 1 for a diagonal GLCM.

Mathematical formulas of the Haralick's Contrast Energy and Homogeneity features are given below in Table2.2.

Table 2.2 Haralick feature definitions and formulas

Statistic	Formula
Contrast	$\sum_{i,j} i - j ^2 p(i, j)$
Energy	$\sum_{i,j} p(i, j)^2$
Homogeneity	$\sum_{i,j} \frac{p(i, j)}{1 + i - j }$

CHAPTER THREE

IMAGE SEGMENTATION GRAPHICAL USER INTERFACE IMPLEMENTATION

3.1 Image Segmentation GUI Implementation

In this study, our aim is to apply semi-supervised learning techniques on image segmentation problem. A new GUI has implemented which helps user to extract input image features and to apply different semi-supervised or supervised image segmentation algorithms on the same platform.

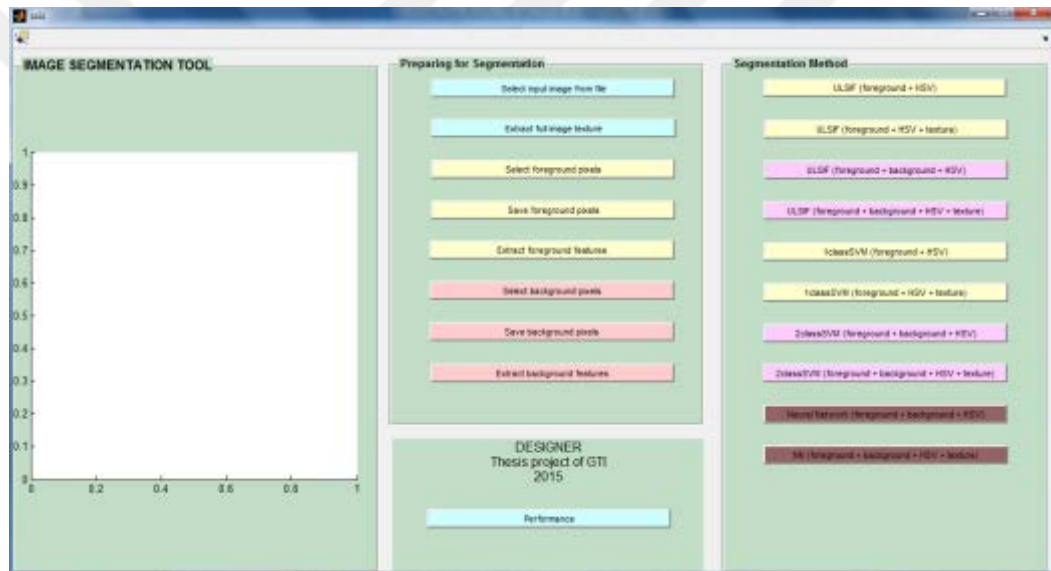


Figure 3.1 Image segmentation tool GUI menu

This GUI consists of vertically arranged three main panel; left section is an input and output image display area, middle section is the feature extraction and performance calculation area, and right side includes different segmentation methods. An example snapshot of the developed GUI is given in Figure3.1.

3.1.1 Selecting Input Image and Displaying in GUI

If “Select input image from file” button is pressed, a Windows pop-up page is appeared asking for an opening image. After selection, input image is shown on the left panel of the GUI as Figure 3.2 shows.

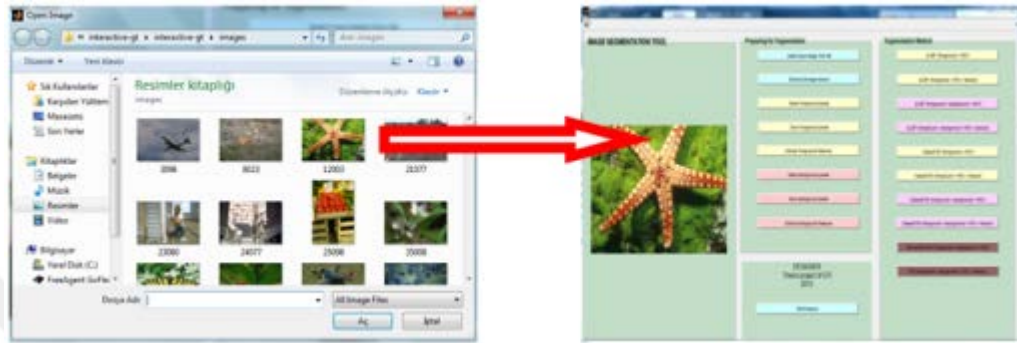


Figure 3.2 Input image selection and displaying in GUI

3.1.2 Extracting Texture Features of Input Image

When “Extract full image texture” button is pressed, our algorithm starts processing on 4x4 pixel blocks as shown in Figure3.3. This subprogram uses block processing function of Matlab.



Figure 3.3 Extracting full image texture features by block processing

After selecting input image by using GUI function, we need to extract Text Features of the original image. If we press “Extract full image texture” button, the algorithm runs like in Figure 3.4 below.

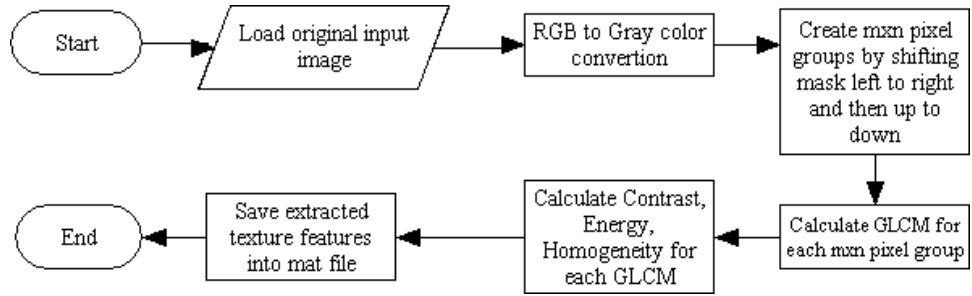


Figure 3.4 Extracting texture features with mxn pixel neighborhood region

3.1.3 Selecting Labeled Foreground and Background Pixels

When “Select foreground pixels” or “Select background pixels” buttons are pressed, on the original input image “+” icon will appear, which helps us to copy pixel locations. Each continuous mouse line shown in Figure 3.5 is called as spline.



Figure 3.5 Selecting foreground pixels on the image by mouse cursor (The Berkeley Segmentation Dataset, 2001).

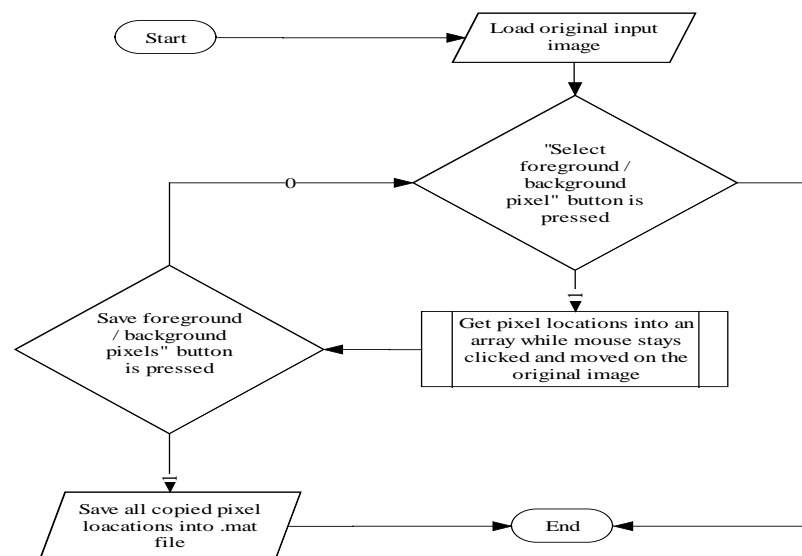


Figure 3.6 Flowchart of selecting foreground or background pixels

In this step, we have the locations of selected foreground and selected background pixels in our hand. If we have moved the mouse cursor on the input image for n times while labeling foreground pixels, this means we have n set of different sized labeled pixel locations. Then these locations are stored in a file, and we need to extract features of these entire labeled pixel sets.

3.1.4 Extracting Features of User Labeled Pixels'

At first labeled pixel locations are loaded from the stored files from previous steps. Then input image color space is converted from RGB to HSV. The HSV values of the labeled foreground or background locations are stored. Same operation is repeated for each labeled pixel set.

Then original input image's colorspace is converted from RGB to Gray, and GLCM is calculated. By using statistical information of GLCM; Contrast, Energy and Homogeneity features will be calculated for all labeled pixel set.

Finally; all of the Hue, Saturation, Value, Contrast, Energy and Homogeneity features are stored.

All these feature extraction steps are shown in Figure 3.7.

3.2 Proposed Segmentation Methods According to Feature Set

Some images are rich with its color information, but some images are containing more edge and texture information. According to the segmentation problem and the input image properties, it is necessary to choose one of these feature sets, color or texture. Both of these feature sets can be used in semi-supervised and supervised image segmentation methods.

The proposed algorithms in this thesis can be categorized like in Figure 3.8 below. Semi-supervised algorithms use only labeled foreground data and unlabelled data.

On the other hand, supervised algorithms use both of the labeled foreground and labeled background data during learning stages.

All these methods given in Figure 3.8 are going to be tested on the same input image, their optimal settings are going to be done, and the output performances will be compared in Chapter 5 and in Chapter 6.

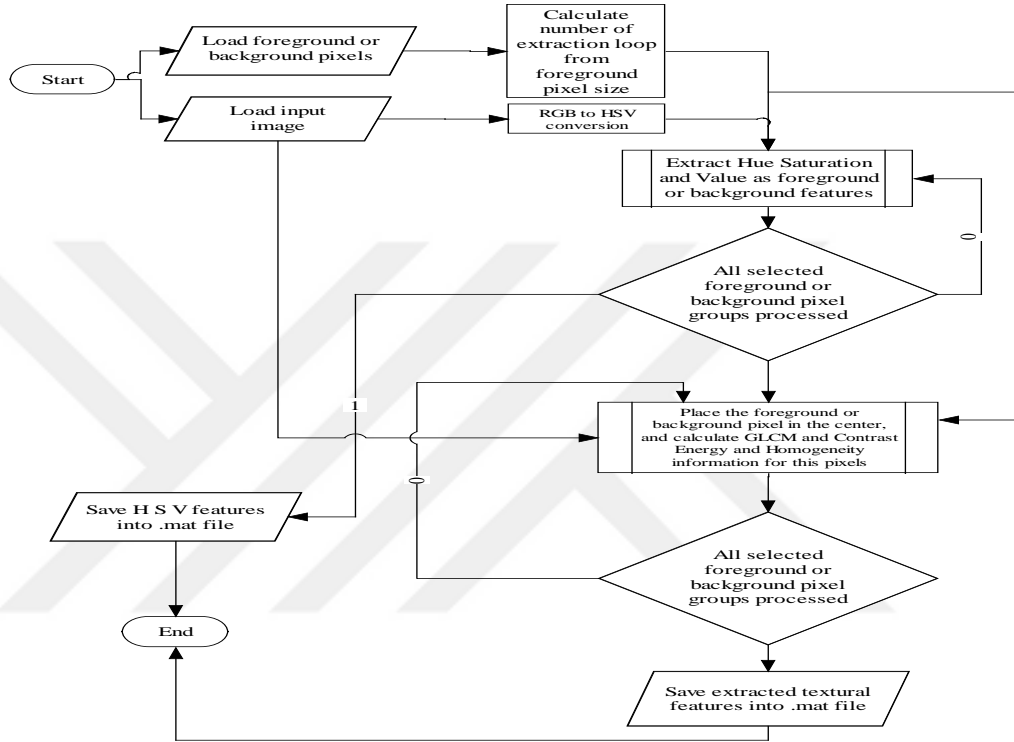


Figure 3.7 Extracting labeled foreground and background features

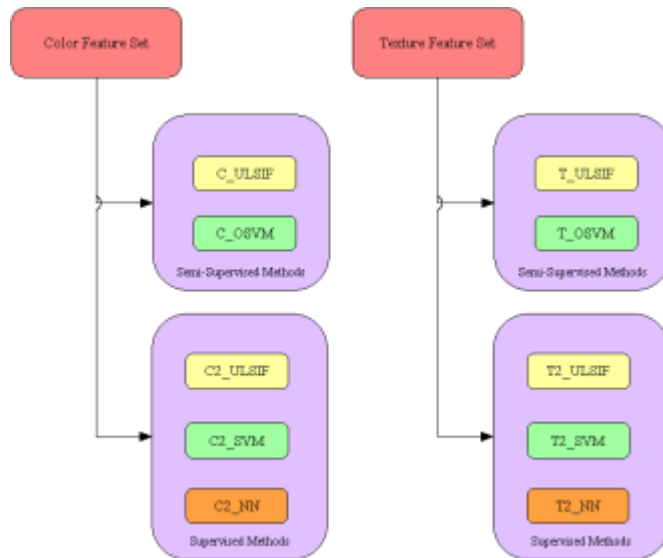


Figure 3.8 Categorization of proposed algorithms

We apply 10 different methods for image segmentation, and they use the features below:

Semi-Supervised methods with color feature set:

- ❖ C_ULSIF is semi-supervised version of ULSIF algorithm which uses only HSV color feature set of labeled foreground pixels
- ❖ C_OSVM is One Class SVM (semi-supervised version of SVM) algorithm which uses only HSV color feature set of labeled foreground pixels

Semi-Supervised methods with texture feature set:

- ❖ T_ULSIF is semi-supervised version of ULSIF algorithm which uses GLCM Haralick's texture feature set of labeled foreground pixels
- ❖ T_OSVM is One Class SVM (semi-supervised version of SVM) algorithm which uses GLCM Haralick's texture feature set of labeled foreground pixels

Supervised methods with color feature set:

- ❖ C2_ULSIF is supervised version of ULSIF algorithm which uses HSV color feature set of the labeled foreground and the labeled background pixels
- ❖ C2_SVM is SVM algorithm which uses HSV color feature set of the labeled foreground and the labeled background pixels
- ❖ C2_NN is Artificial Neural Network which uses HSV color feature set of the labeled foreground and the labeled background pixels

Supervised methods with texture feature set:

- ❖ T2_ULSIF is supervised version of ULSIF algorithm which uses GLCM Haralick's texture feature set of the labeled foreground and the labeled background pixels
- ❖ T2_SVM is SVM algorithm which uses GLCM Haralick's texture feature set of the labeled foreground and the labeled background pixels
- ❖ T2_NN is Artificial Neural Network which uses GLCM Haralick's texture feature set of the labeled foreground and the labeled background pixels

CHAPTER FOUR

IMAGE SEGMENTATION ALGORITHMS

4.1 Segmentation Methods

4.1.1 ULSIF

In this thesis, we aim to introduce Unconstrained Least-squares Importance Fitting (ULSIF) algorithm as a semi-supervised method to an image segmentation problem. We assume that labeled data and our original image is two different probability density functions (pdf). And the ratio between these two pdf can give us the similarity information between labeled pixels and unlabeled pixels.

Kanamori, Hido, & Sugiyama (2009), introduced ULSIF, which estimates the ratio of two pdf without calculating explicitly any pdf. They explain ULSIF in their paper:

Importance can be explained as the ratio of two probability density functions. ULSIF algorithm approach to the estimation of importance problem as least-squares function fitting problem.

Let $\{x_i^{tr}\}_{i=1}^{n_{tr}}$ be our independent and identically distributed training samples with distribution density $p_{tr}(x)$, and $\{x_j^{te}\}_{j=1}^{n_{te}}$ be our independent and identically distributed training samples with distribution density $p_{te}(x)$.

$$\{x_i^{tr}\}_{i=1}^{n_{tr}} \overset{i.i.d.}{\sim} p_{tr}(x) \quad (4.1)$$

$$\{x_j^{te}\}_{j=1}^{n_{te}} \overset{i.i.d.}{\sim} p_{te}(x) \quad (4.2)$$

Normally importance, the ratio between two probability distribution functions can be formulated as:

$$w(x) = \frac{p_{te}(x)}{p_{tr}(x)} \quad (4.3)$$

In ULSIF algorithm, importance is represented as the weighted sum of basis functions as given in (4.4).

$$\hat{w}(x) = \sum_{l=1}^b \alpha_l \varphi_l(x) \quad (4.4)$$

In this equation $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_b)^T$ are the parameters to be learned from labeled features, and $\{\varphi_l(x)\}_{l=1}^b$ are basis functions such:

$$\varphi_l(x) \geq 0 \text{ for all } x \in D \text{ and for } l = 1, 2, \dots, b. \quad (4.5)$$

Although there are various possibilities, gaussian kernels are considered as the basis functions in this thesis study.

Gaussian kernel model centered at the test samples $\{x_j^{tr}\}_{j=1}^{n_{te}}$ is given below:

$$\varphi_l(x) = K_\sigma(x, x_l^{te}), \quad (4.6)$$

$$K_\sigma(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right) \quad (4.7)$$

where σ is the kernel width. When n_{te} is large, it is not offered to use all test points $\{x_j^{tr}\}_{j=1}^{n_{te}}$ as Gaussian centers, because this is computationally ineffective. Because of that, it is suggested to use a subset of $\{x_j^{tr}\}_{j=1}^{n_{te}}$ as Gaussian centers and the equation takes form:

$$\hat{w}(x) = \sum_{l=1}^b \alpha_l K_\sigma(x, c_l), \quad (4.8)$$

where c_l is the randomly selected points from $\{x_j^{tr}\}_{j=1}^{n_{te}}$, and $b (\leq n_{te})$ is the number of basis functions.

ULSIF algorithm optimizes kernel width σ and regularization parameter λ by using cross validation and grid search methods. For regularization steps, it is referred to read Section 2.6, 4.3, Appendix D in Kanamori, T., Hido, S., & Sugiyama, M. (2009).

Our work is a kind of outlier detection problem and we implement ULSIF algorithm into our software to detect foreground pixels from entire image. If the importance value of any pixel is close to one it means there is a high possibility it

belongs to foreground object. If importance value of the pixel is significantly deviated from one, it means this pixel tends to be background.

In our GUI we have 4 different ULSIF methods, each of them call same ULSIF function. The difference between the methods is the feature size used in training and test data, and the post processes applied after obtaining ULSIF output to get a single binary output image.

4.1.2 Support Vector Machines for Image Segmentation

4.1.2.1 Two-Class SVM

Support vector machines concentrated on decision planes. These decision planes separate objects belonging to different classes. Many classification problems are not so easy to handle. For example in Figure 4.1 (a), separation of green and red samples needs to have a smooth curve, and SVM have capability to handle this kind of tasks.

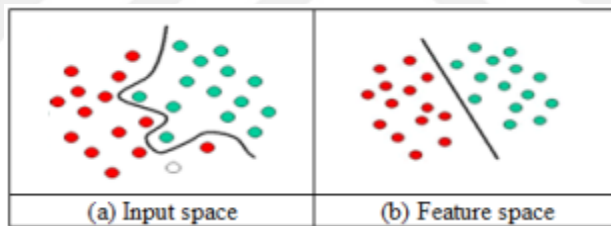


Figure 4.1 SVM examples (Support Vector Machines Introductory Overview, 2016).

SVM rearranges the input samples by using some mathematical functions (called as kernel) into a linearly separable feature space. This process is known as mapping.

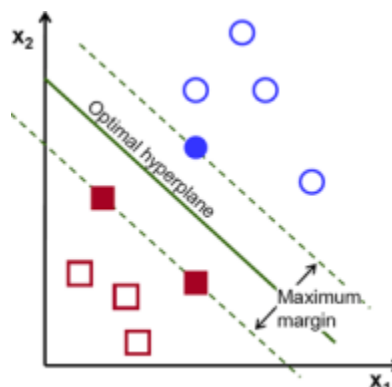


Figure 4.2 SVM optimal hyperplane (What is a SVM?, n.d.)

Let's have a look how we find the optimal hyperplane:

Normally hyperplane definition can be given with the formula below:

$$f(x) = w_0 + w_x^T \quad (4.12)$$

where w is weight and w_0 is the bias.

Between all possible hyperplanes, we select one of them that satisfies:

$$|w_0 + w_x^T| = 1 \quad (4.13)$$

where x represents the training samples closest to the hyperplane. Training samples, x , are called support vectors.

Now, we will calculate the distance between x and the hyperplane (w, w_0)

$$dist = \frac{|w_0 + w_x^T|}{\|w\|} \quad (4.14)$$

For canonical hyperplane distance takes the value:

$$dist_{support_vectors} = \frac{1}{\|w\|} \quad (4.15)$$

And the margin shown in Figure 4.2, is twice of that value:

$$M = \frac{2}{\|w\|} \quad (4.16)$$

Finally, we want to maximize M , and this is equal to minimizing function $L(w)$ by using Lagrange multipliers to find weight and bias values:

$$\min_{w, w_0} L(w) = \frac{1}{2} \|w\|^2 \quad \text{subject to} \quad y_i(w_{x_i}^T + w_0) \geq 1 \quad \forall i \quad (4.17)$$

where y_i is the label of each training samples.

4.1.2.2 One-Class SVM (OSVM)

If we look at again the SVM method first, its objective function is:

$$\min_{w, b, \xi_i} \frac{\|w\|^2}{2} + C \sum_{i=1}^n \xi_i \quad (4.18)$$

subject to

$$y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i \quad \text{for all } i = 1, \dots, n \quad (4.19)$$

$$\xi_i \geq 0 \text{ for all } i = 1, \dots, n \quad (4.20)$$

where w is vector of coefficients,

b is constant,

ξ_i is parameter of non-separable input,

C is capacity constant, decides the smoothness,

$\phi(x_i)$ is input mapping function to the feature space,

$y_i \in \pm 1$ is the class labels,

x_i Independent inputs.

Then, classification (decision function) rule for x is:

$$f(x) = \text{sgn}(\sum_{i=1}^n \alpha_i y_i K(x, x_i) + b) \quad (4.21)$$

Here α_i are the Lagrange multipliers, and $K(x, x_i) = \phi(x)^T \phi(x_i)$ is the *kernel function*. In the literature, the most popular choice for kernel function is the Gaussian Radial Base Function, that is:

$$K(x, x') = \exp(-\frac{\|x - x'\|^2}{2\sigma^2}) \quad (4.22)$$

where $\sigma \in R$ is a kernel parameter and $\|x - x'\|$ is the dissimilarity measure.

One-Class SVM according to Schölkopf et al. (1999) has a quadratic programming minimization function which is slightly different from the original stated above:

$$\min_{w, \xi_i, \rho} \frac{1}{2} \|w\|^2 + \frac{1}{vn} \sum_{i=1}^n \xi_i - \rho \quad (4.23)$$

subject to:

$$(w \cdot \phi(x_i)) \geq \rho - \xi_i \text{ for all } i = 1, \dots, n \quad (4.24)$$

$$\xi_i \geq 0 \text{ for all } i = 1, \dots, n \quad (4.25)$$

In this formula ν parameter characterizes the solution. First, it sets an upper bound on the fraction of outliers (training examples regarded out-of-class) . Second, it is a lower bound on the number of training examples used as Support Vector.

By using Lagrange multipliers and using a kernel function for the dot-product calculations, the decision function or classification becomes:

$$f(x) = \text{sgn}((w \cdot \phi(x)) - \rho) = \text{sgn}(\sum_{i=1}^n \alpha_i K(x, x_i) - \rho) \quad (4.26)$$

This method thus creates a hyperplane characterized by w and ρ which has maximal distance from the origin in feature space and separates all the data points from the origin. Function returns +1 in a “small” region (capturing the training data points) and -1 elsewhere.

4.1.3 *Neural Networks for Image Segmentation*

Neural networks stayed is a very popular alternative to other pattern recognition algorithms even in more complex problems. Because if we have a complex problem and higher dimensions, all other methods may need a dimension reduction as a pre-process. However, multi layer neural networks can learn a complex transfer function in high dimensional space without requiring any additional process for feature extraction. This allow designer to concentrate more on training process and less engineering on feature selection and extraction (LeCun & Bengio, 1995).

Labbe, Herault, & Chatelain (2009) propose a framework based on deep neural networks in order to deal with high dimensional input problems such as image classification on toy datasets and USPS character reconstruction. And their results are promising.

Neural networks are inspired from human brain through there are still lots of unknowns how brain is training itself. Typically, A neuron collects signals from

dendrites and sends electro - chemical signals (pulses) through a single wire called axon. This axon divided into many branches; at the end of each branch it has a synapse.

During the learning process of neural networks, effectiveness of the synapses is changed, by this way the influence of one neuron on another is changed (Stergiou & Siganos, 2015). Similarity between human and artificial neuron system is given in Figure 4.3.

Pattern recognition networks are feedforward networks and it consists of larger number simple neurons (processing-units) and organized in layers. Every neuron in a layer is connected to the other layers' neurons and each connection has a different weight. During the normal operation, data enters to the inputs, and passes through all layers and neurons, and there is no feedback between these layers, this is why we call it feedforward neural network (Boersma & Weenink, 2004).

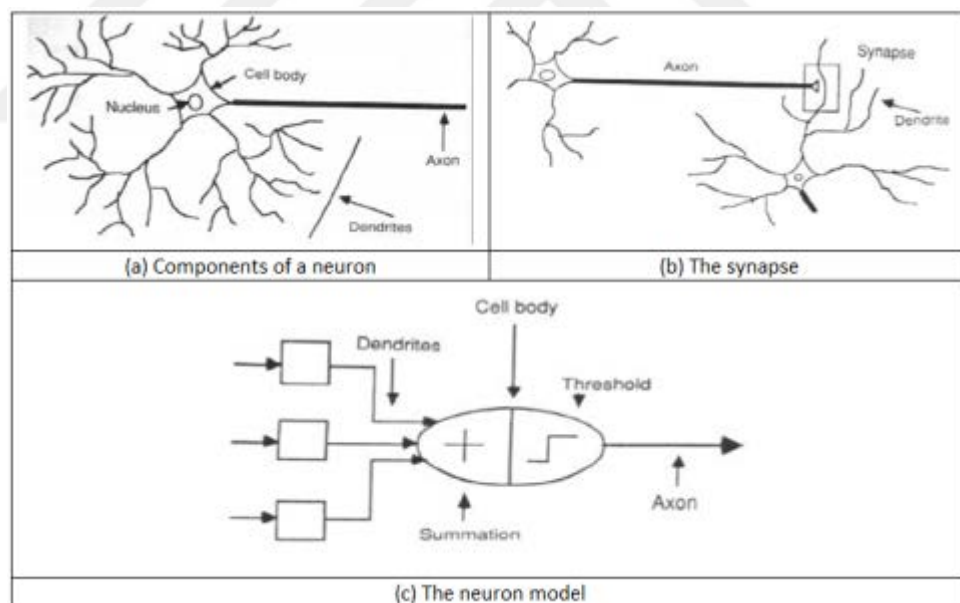


Figure 4.3 Representation of human neuron system (Stergiou, & Siganos, 1997).

A single layer feedforward neural network example is given in Figure 4.4. And a multi layer feedforward neural network example is given in Figure 4.5.

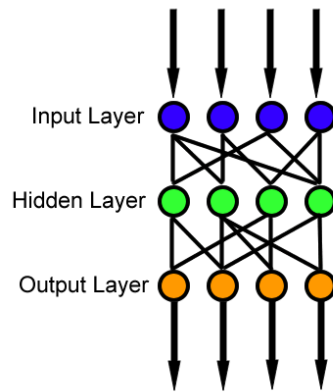
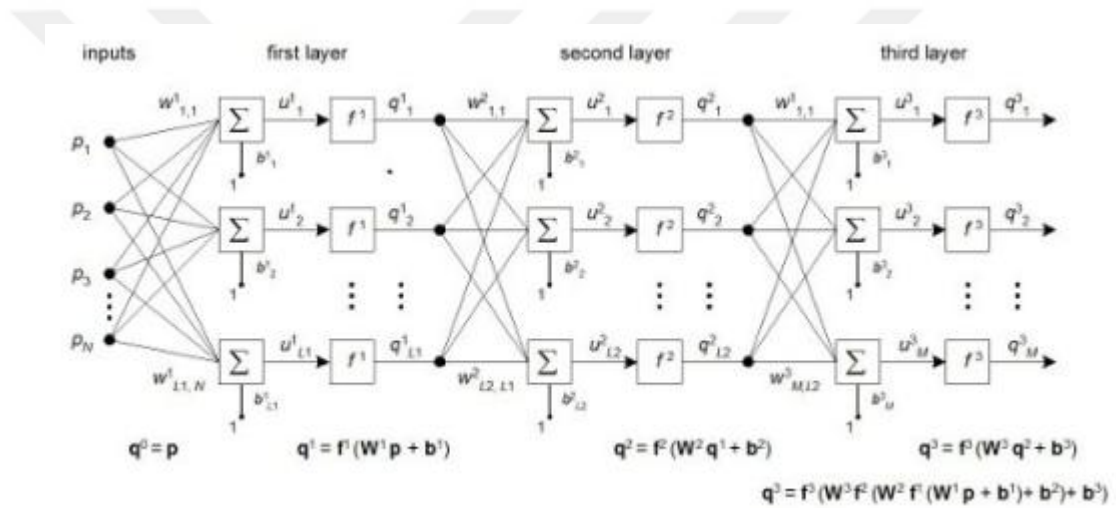
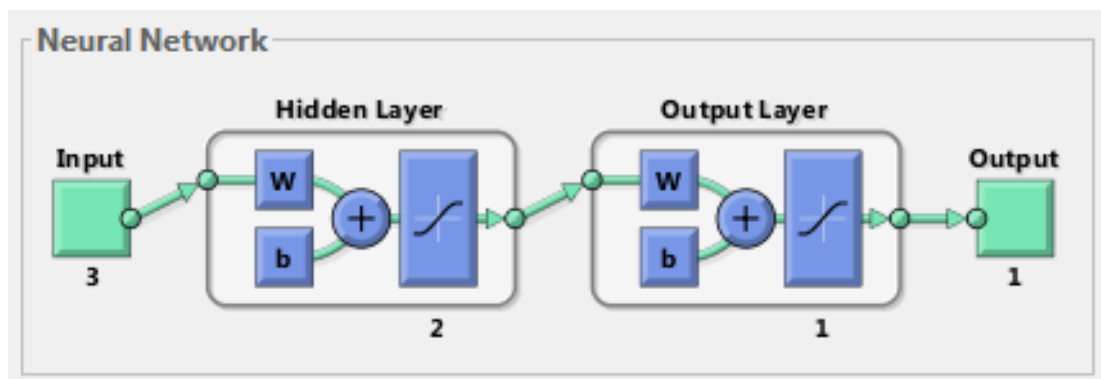


Figure 4.4 Single layer feedforward neural network diagrams (Feedforward neural network, n.d.).



(a)



(b)

Figure 4.5 (a) Multi layer feedforward neural network example with its weights (Application of Neural Networks to Modeling and Control of Parallel Manipulators, Three-layer feedforward network , n.d.), (b) Proposed pattern recognition network in this thesis

In this thesis pattern recognition network is used in Matlab as supervised image segmentation tool. Proposed pattern recognition network is shown in Figure 4.5 (b). It works with network training function that updates weight and bias values according to the scaled conjugate gradient method (`trainsgc`) and uses the cross-entropy performance method. C2_NN has 3 input layers (for H-S-V features), T2_NN has 9 input layers (for energy, contrast and homogeneity). The number of hidden layers is 10. And the output layer number 1 to get a binary pixel values.

4.2 Implementation of Image Segmentation Algorithms

4.2.1 Semi Supervised Methods

4.2.1.1 *Semi-Supervised C_ULSIF*

This method is using only one labeled class information. User label only foreground pixels and program runs ULSIF algorithm as an outlier detection function.

It calculates the importance value of each pixel. Higher importance value means that pixel belongs to foreground and lower importance value means that pixel belongs to background.

As we can see from the flow chart in Figure 4.6:

- Step1: Original image is loaded into the workspace, color space is changed RGB to HSV, and all Hue Saturation and Value is sorted into `x_nu` matrix
- Step2: Extracted features are loaded in to the workspace, if user selected `n` times different spline for labeling foreground, the first labeled pixel group's HSV information transferred into `x_de` matrix
- Step3: ULSIF function is called by the program and importance values of the original image pixels are calculated and stored in `wh_x_nu` in a row vector format. This row vector is reshaped with the size of input image. Then, this reshaped importance image is converted to a binary image by Otsu (thresholding) method and

is stored into $bwout\{i\}$, where $i = 1, 2, \dots, n$ represents the used spline number for training step.

- Step4: Algorithm returns to Step2, gets the features of next user selected spline, performs the Step3 and stores the new binary output into the $bwout\{k\}$, and this cycle is repeated till the latest foreground spline.

- Step5: All calculated binary images enter an OR operator, and finally we get a single binary image. Indeed it is our final segmentation output.

$$bwout = \mathbf{OR}_{i=1}^n(bwout, bwout\{i\}) \quad (4.27)$$

- Step6: We can use this binary segmentation output as a mask, we can multiply Hue Saturation and Value with this mask separately, and the HSV output image will be converted into an RGB image again.

- Finally, we get a color output image and all background pixels have black color, only foreground pixels have their own color.

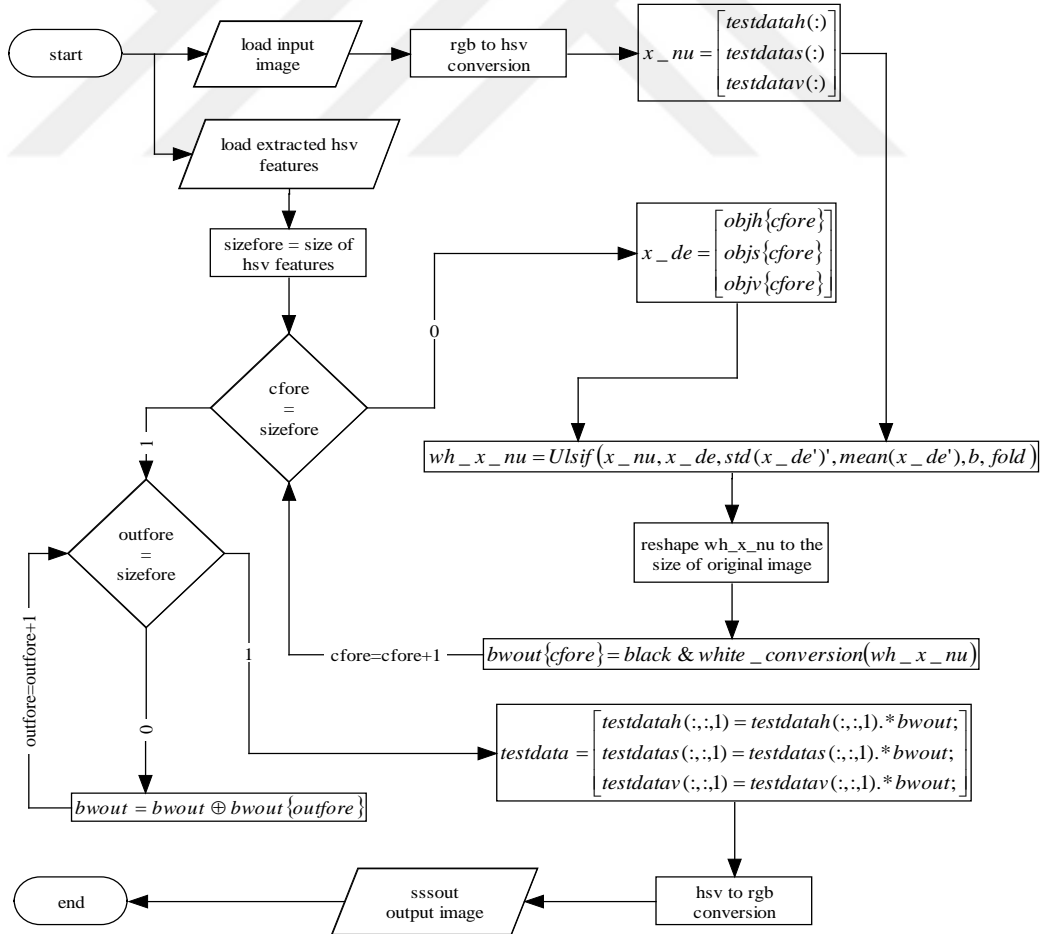


Figure 4.6 Flowchart of semi-supervised C_ULSIF algorithm

4.2.1.2 Semi-Supervised T_ULSIF

This method is very similar to C_ULSIF. It is using only one class labeled features, instead of HSV color feature set texture features are taken into calculation.

x_nu includes contrast, energy and homogeneity as texture features of the original input image, also x_de includes contrast, energy and homogeneity as texture features of the labeled foreground pixels. Its flowchart is given in Figure 4.7.

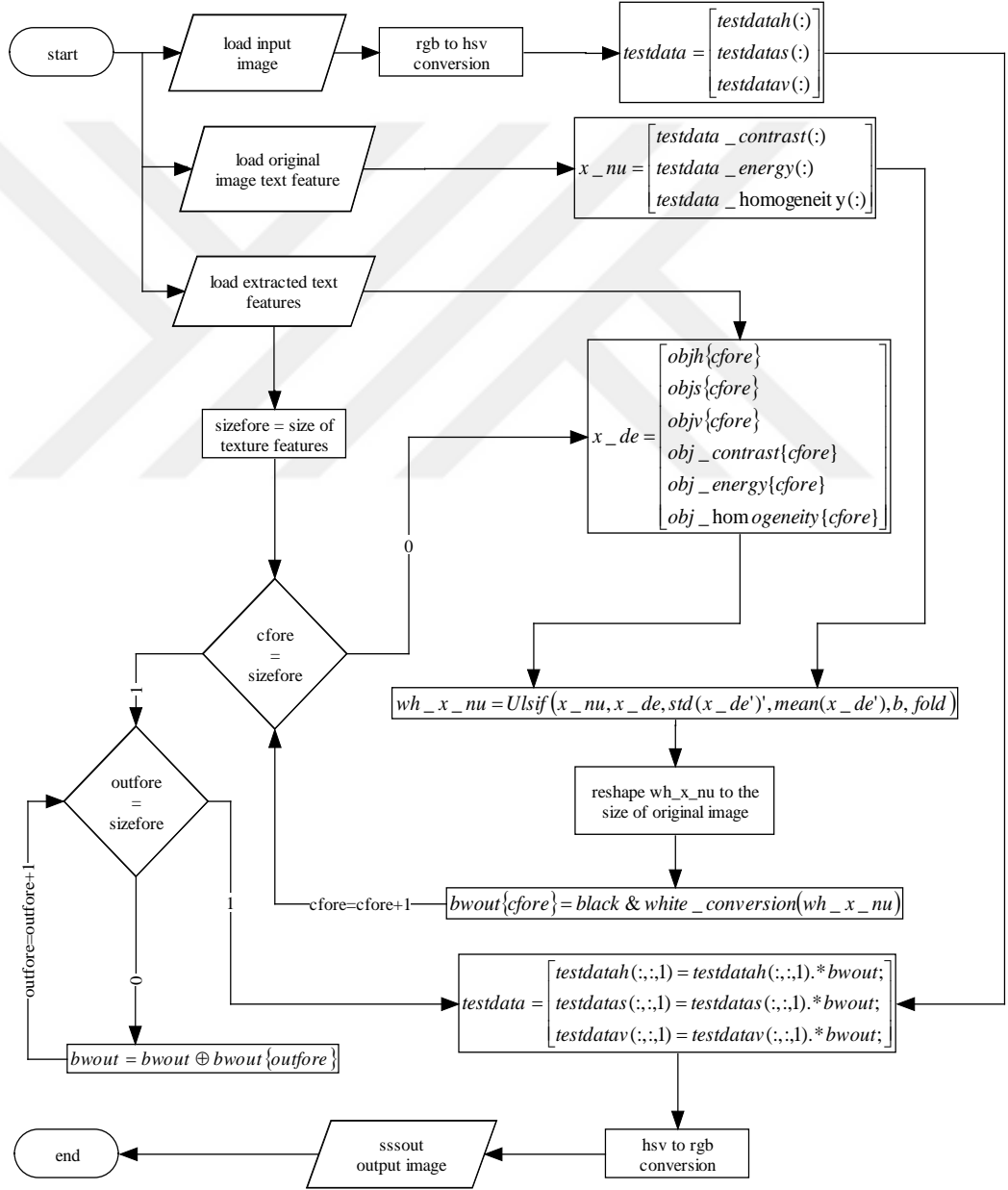


Figure 4.7 Flowchart of semi-supervised T_ULSIF algorithm

4.2.1.3 *Semi-Supervised C_OSVM*

This algorithm is a semi-supervised image segmentation method which uses only labeled a few foreground pixels.

- The original input image is loaded into workspace, then its color space is converted into HSV, and all pixel's HSV features stored into x_{nu} matrix
- All of the user selected foreground splines' features sorted in x_{del} and they are labeled with value 1, these labels stored in a vector F
- "Fitsvm" function is called to calculate optimal weight and bias of OSVM, during the training step we assumed that 1% of the training samples are misclassified (depends another class).
- "Predict" function is called with original image feature (x_{nu}). Score vector is reshaped into the original image size (resizing), and a threshold is applied to get a binary output image
- Finally this binary output is multiplied with the HSV values of the image pixelwisely. And the resultant HSV image is converted in to RGB color space.
- Figure 4.8 shows the flowchart of the C_OSVM segmentation algorithm.

4.2.1.4 *Semi-Supervised T_OSVM*

This method is too similar with C_OSVM, the only difference is that, it uses GLCM Haralick texture set as feature.

- The original input image is loaded into workspace, then its color space is converted into HSV
- Original image's all extracted texture features loaded from file and texture features stored into x_{nu} matrix
- Texture feature of user selected foreground splines' sorted in x_{del} and they are labeled with value 1, these labels stored in a vector F

- “Fitsvm” function is called to calculate optimal weight and bias of OSVM, during the training step we assumed that 1% of the training samples are misclassified (depends another class).
- “Predict” function is called with original image feature (x_nu). Score vector is reshaped into the original image size (resizing), and a threshold is applied to get a binary output image
- Finally binary output is multiplied with the HSV values of the original image pixelwisely. The resultant HSV image is converted in to RGB color space.
- Figure 4.9 shows the flowchart of the T_OSVM segmentation algorithm.

4.2.2 Supervised Segmentation Methods

4.2.2.1 Supervised C2_ULSIF

This method uses labeled features of two classes. For example user select n times different spline for background pixels and also n times different spline for foreground pixels. C2_ULSIF uses only HSV color features.

- Program runs C_ULSIF algorithm (from Step1 to the end of Step4 explained in 4.2.1.1.) for n different foreground spline, here the binary outputs generated for each spline is stored into the matrix $bwf\{1\}to\ bwf\{n\}$

- Again, program runs C_ULSIF algorithm (from Step1 to the end of Step4 explained in 4.2.1.1.) for n different background spline, here the binary outputs generated for each spline is stored into the matrix $bwb\{1\}to\ bwb\{n\}$

- Then each binary background matrix is subtracted pixel-wise

$$\begin{aligned} bw\{1\} &= bwf\{1\} - bwb\{1\} \\ &\vdots \\ bw\{n\} &= bwf\{n\} - bwb\{n\} \end{aligned} \tag{4.28}$$

- Now program makes the same logical OR operation including all binary outputs from

- Finally OR operation is performed between all binary outputs to get final binary image

$$bwout = \mathbf{OR}_{i=1}^n(bwout, bw\{i\}) \tag{4.29}$$

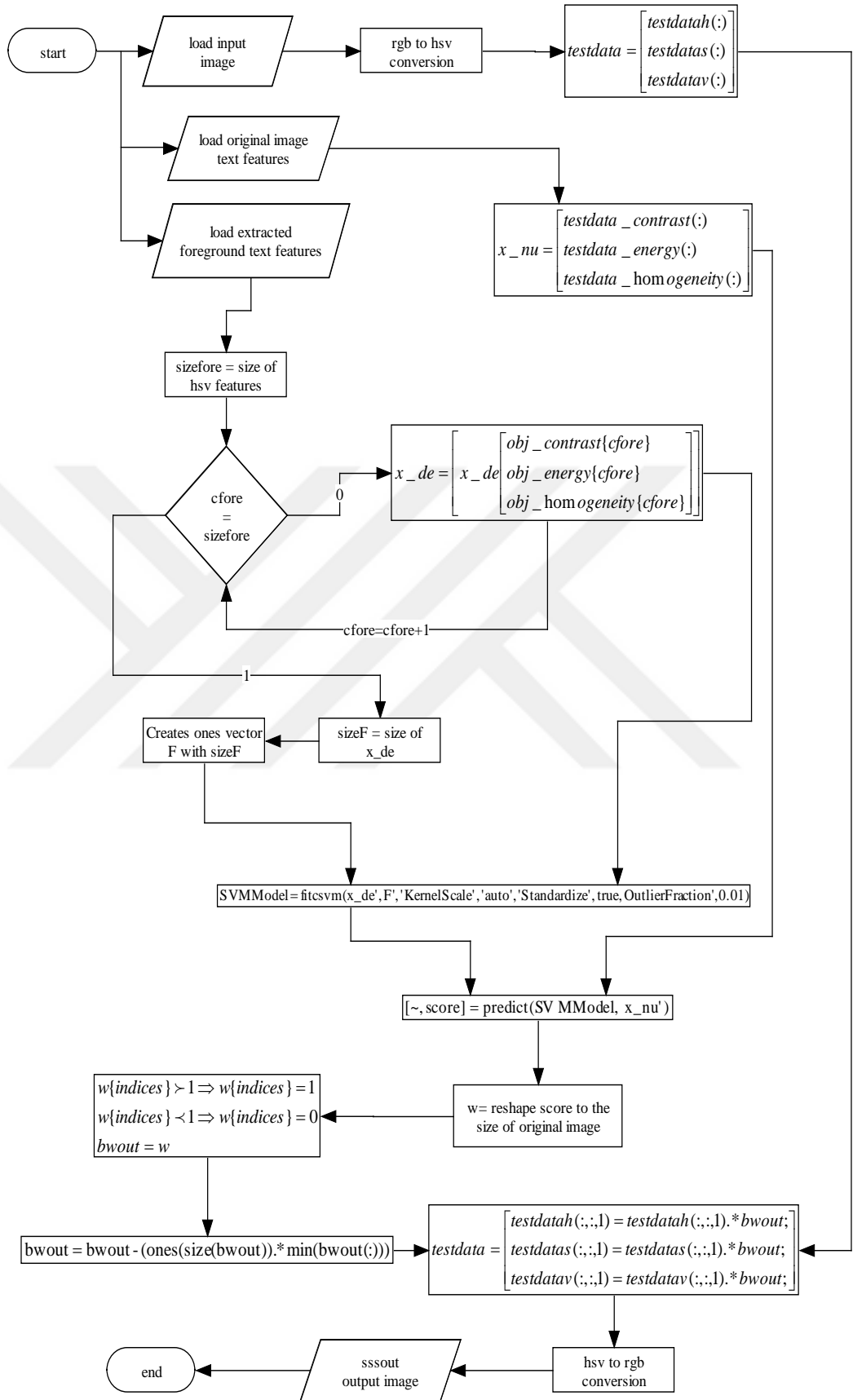


Figure 4.9 Flowchart of semi-supervised T_OSVM algorithm

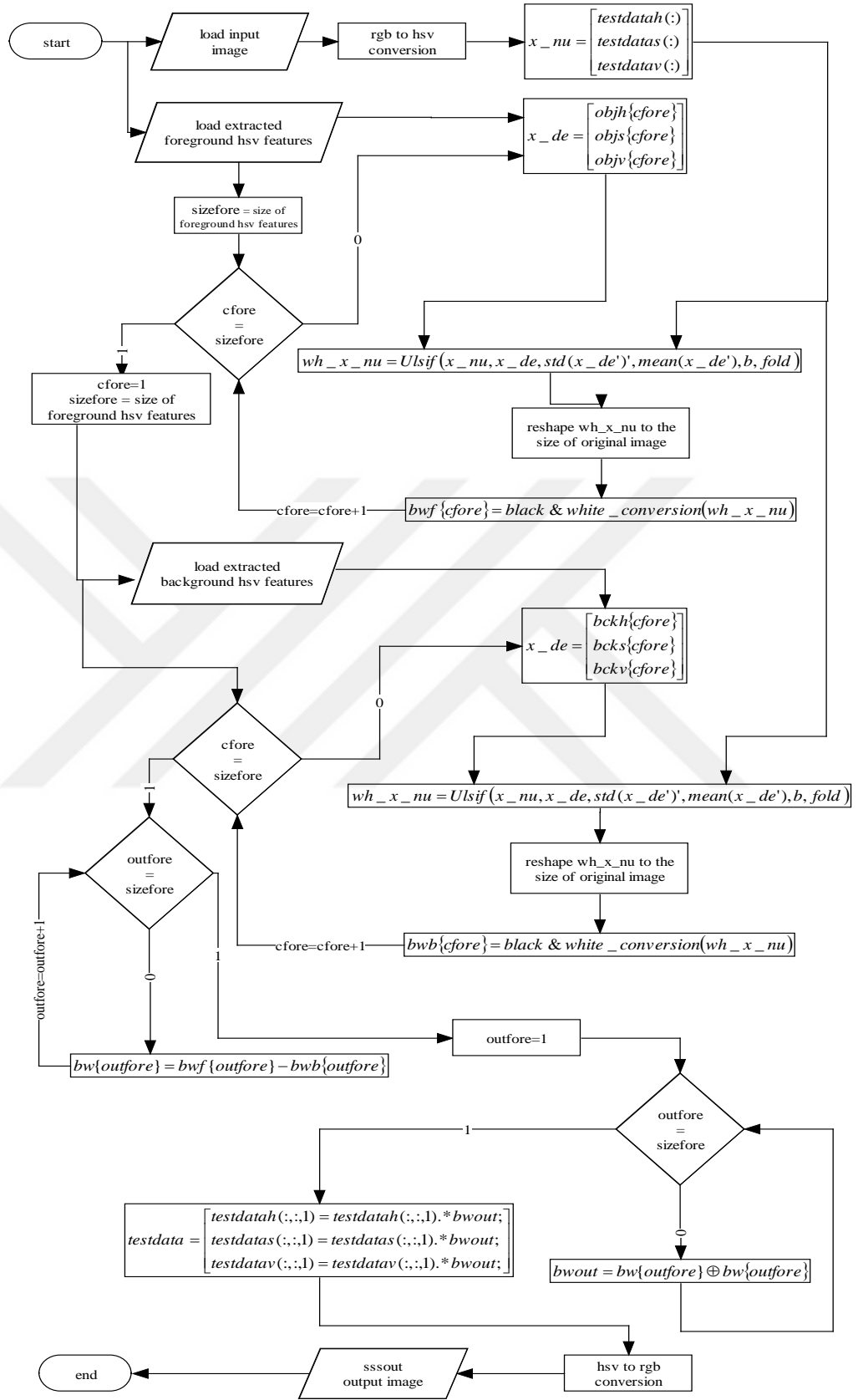


Figure 4.10 Flowchart of supervised C2_ULSIF algorithm

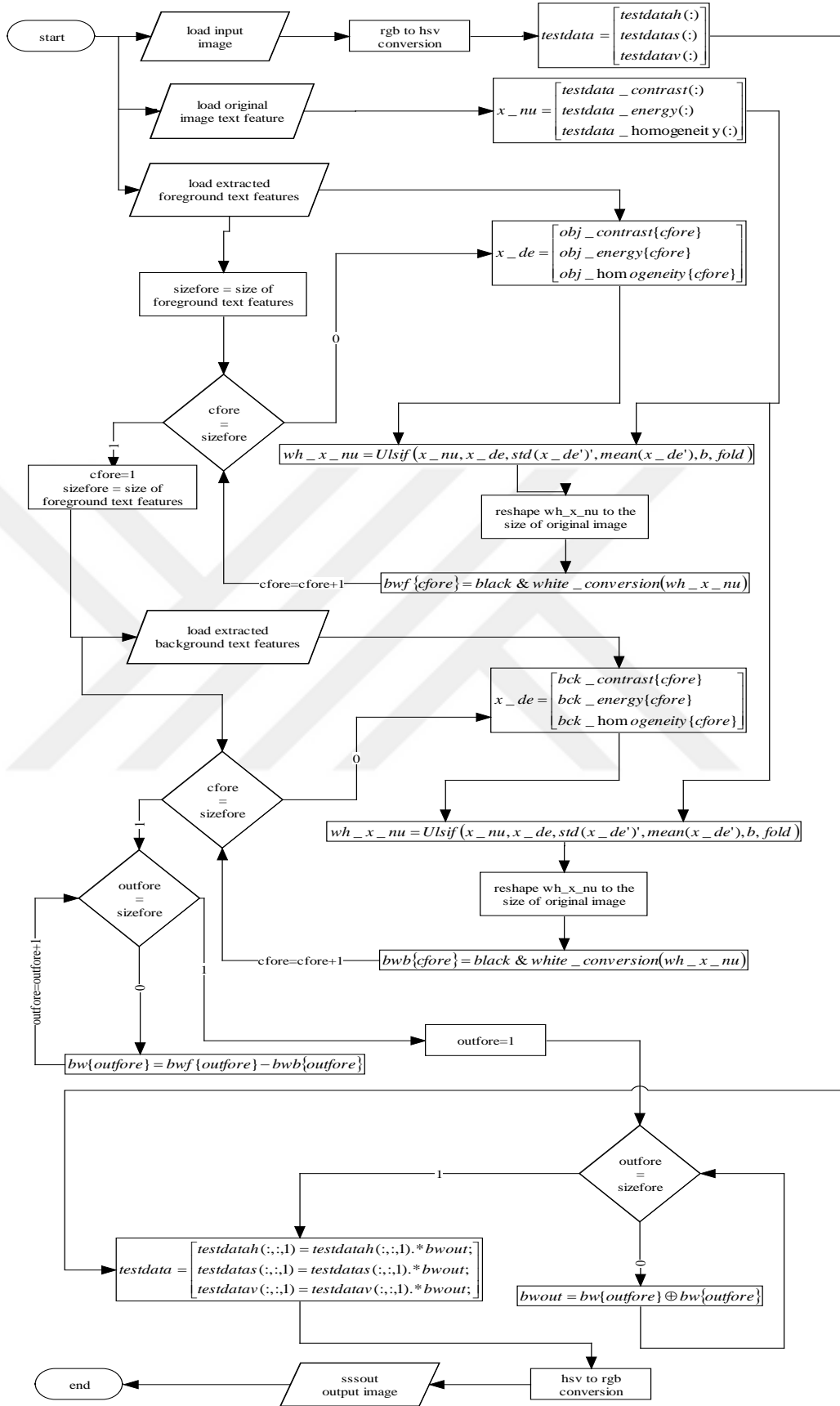


Figure 4.11 Flowchart of supervised T2_ULSIF algorithm

4.2.2.2 *Supervised T2_ULSIF*

This method is very similar to C2_ULSIF. The only difference is that, T2_ULSIF algorithm uses texture features of the labeled data and the original input image.

x_{nu} includes contrast, energy and homogeneity as texture features of the original input image, also x_{de} includes contrast, energy and homogeneity as texture features of the labeled foreground pixels.

All cycles explained in 4.2.2.1 for C2_ULSIF is completely same for T2_ULSIF.

Figure 4.11 shows the flow chart of T2_ULSIF algorithm for better understanding.

4.2.2.3 *Supervised C2_SVM*

It is a supervised image segmentation algorithm which uses only HSV features of the image. This algorithm needs labeled samples from both of the foreground and background.

- The original input image is loaded into workspace, then its color space is converted into HSV, and all pixel's HSV features stored into x_{nu} matrix
- All of the user selected foreground splines' features sorted in x_{de1} and they are labeled with value 1, these labels stored in a vector F
- All of the user selected background splines' features sorted in x_{de2} and they are labeled with value 0, these labels stored in a vector B
- SVM training function is called to calculate optimal weight and bias, then original image features x_{nu} and SVM classification function are called to label entire image
- Score vector is reshaped into the original image size (resizing), and a threshold is applied to get a binary output image

- Finally this binary output is multiplied with the HSV values of the image pixelwisely. And the resultant HSV image is converted in to RGB color space.
- Figure 4.12 shows the flowchart of the segmentation algorithm.

4.2.2.4 *Supervised T2_SVM*

In Figure 4.13 the flow chart of the T2_SVM algorithm is shown. This method is too similar to C2_SVM, the only difference is that, it uses only GLCM Haralick texture features.

4.2.2.5 *Supervised C2_NN*

This algorithm is a kind of supervised image segmentation method, which uses neural network to find pixel classes. It uses only HSV color information during the training and classification processes.

- Step1: Original input image is loaded and its colorspace is converted RGB to HSV, then all pixels HSV information is stored in matrix x_{nu} .
- Step2: All extracted HSV information of the labeled foreground splines are loaded and they are sorted end-to-end in the matrix x_{de1} .
- Step3: Is the same process like Step2. This time user selected background splines' HSV information stored in x_{de2} .
- Step4: Create a pattern recognition neural network, and train it by using x_{de1} , x_{de2} and their labels.
- Step5: Simulate the weighted pattern recognition network on the entire image (x_{nu}) to find all pixel classes.
- Step6: Reshape the output vector into the size of original input image. This reshaped output image will have double pixel values.
- Step7: Thresholding is applied to the pixel values
 - If any pixel value is greater or equal to 0.75, it will be labeled as foreground with binary value 1
 - Else this pixel will be labeled as background with binary value 0.

- Step8 : The binary image is multiplied pixelwisely with the original image's HSV colors
- Step9: The labeled HSV color image is converted in to RGB colorspace and stored in a Matlab file.
- Its flowchart is shown in Figure 4.14.

4.2.2.6 *Supervised T2_NN*

This algorithm is very similar to C2_NN. The only difference is it uses texture features instead of HSV color features.

- Step1: Original input image is loaded and its colorspace is converted RGB to HSV, also all image's texture features are loaded into the worksapece and then stored in matrix x_nu.
- Step2: All extracted texture feature of the labeled foreground splines are loaded and they are sorted end-to-end in the matrix x_de1.
- Step3: Is the same process like Step2. This time user selected background splines' texture features stored in x_de2.
- All steps from Step4 to Step 9 are same with C2_NN.
- Its flowchart is shown in Figure 4.15.

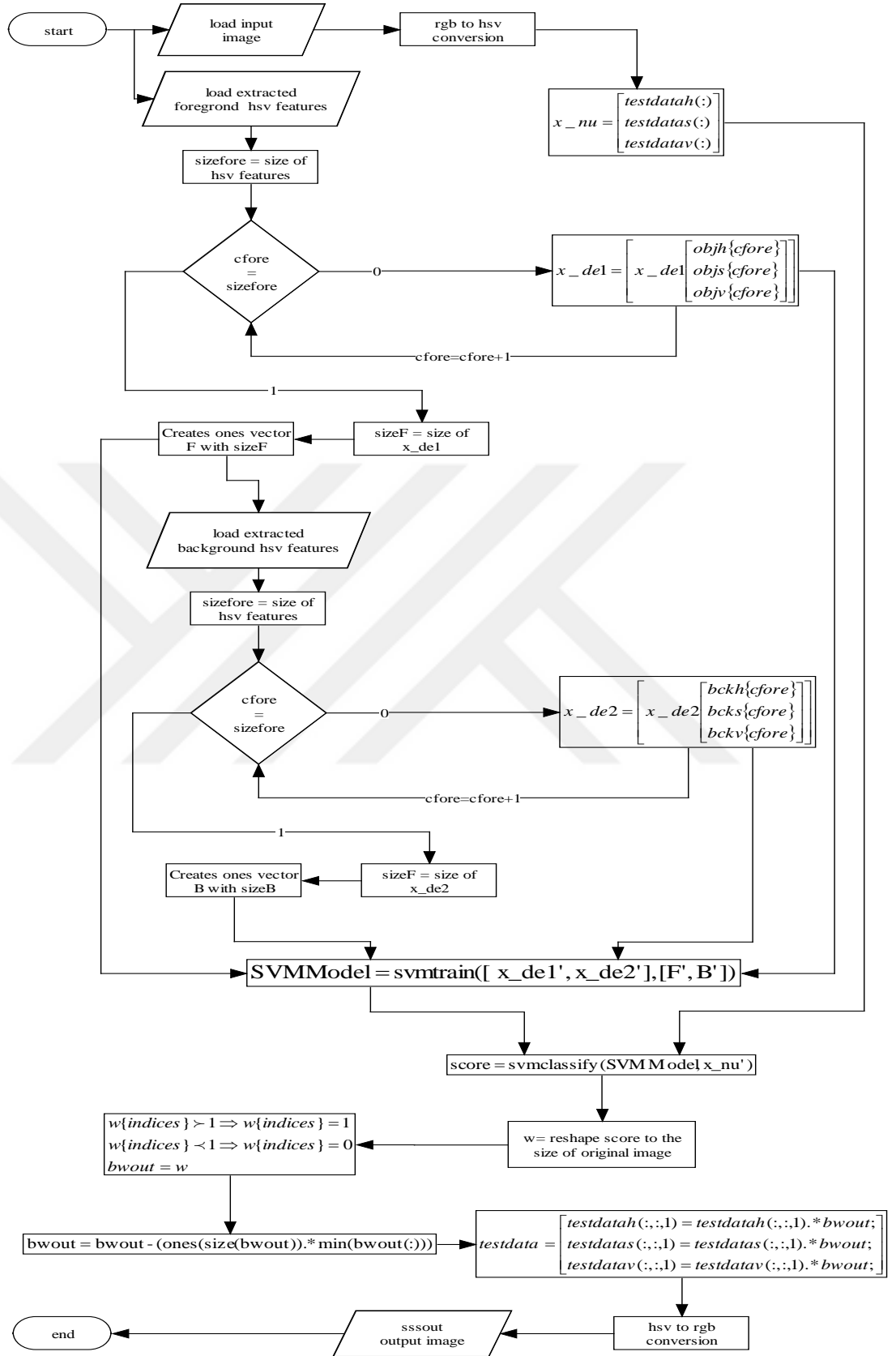


Figure 4.12 Flowchart of supervised C2_SVM algorithm

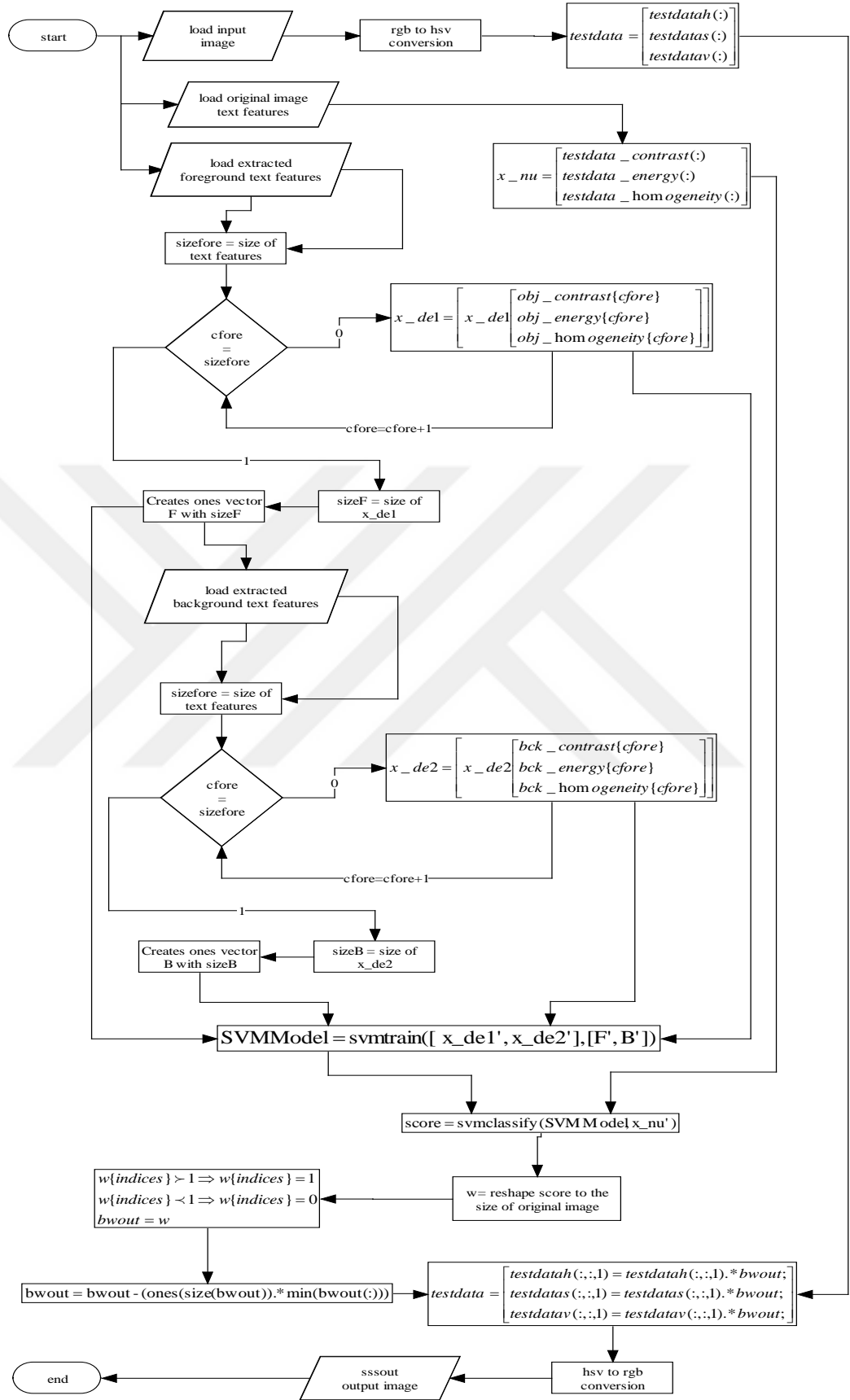


Figure 4.13 Flowchart of supervised T2_SVM algorithm

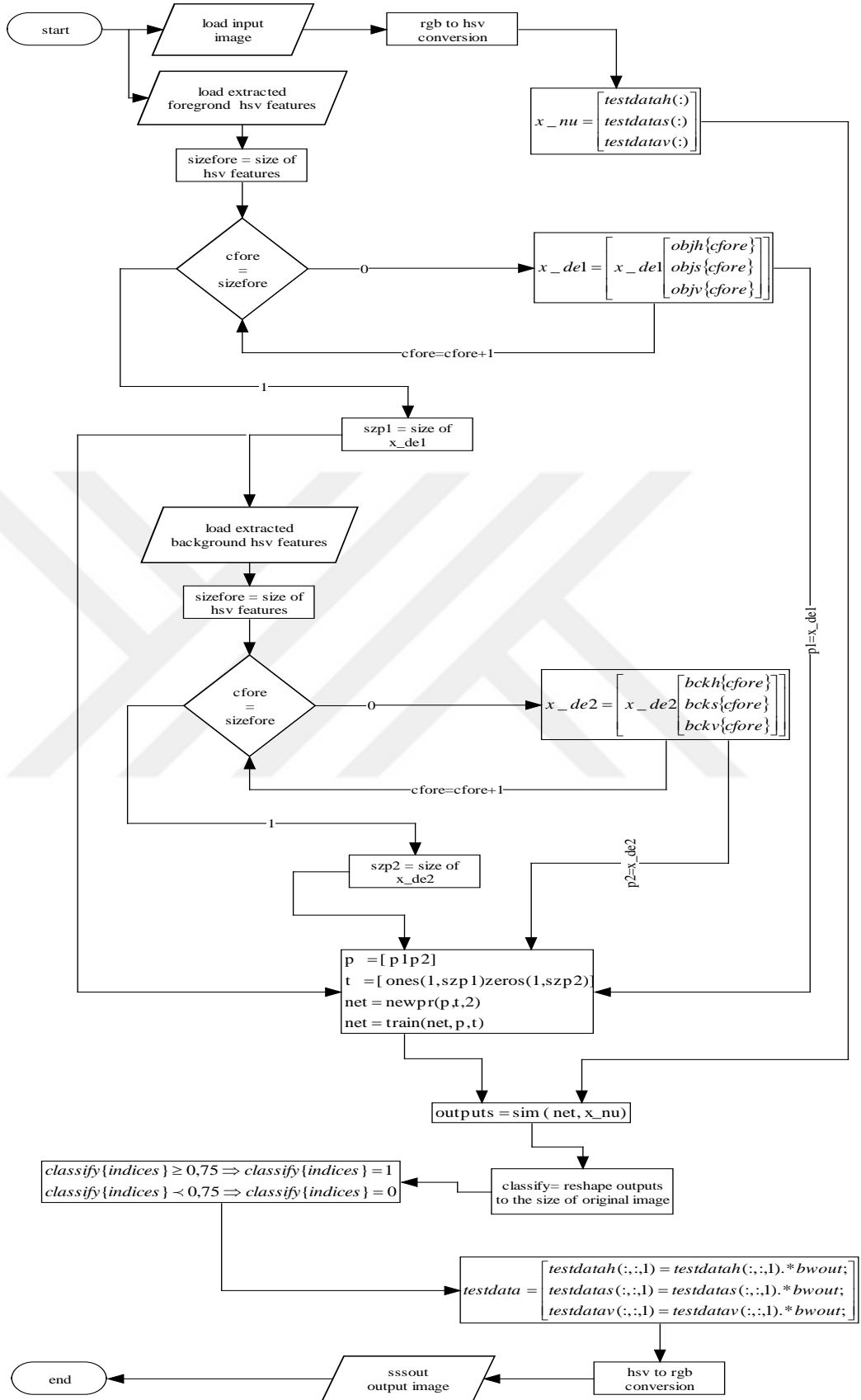


Figure 4.14 Flowchart of supervised C2_NN algorithm

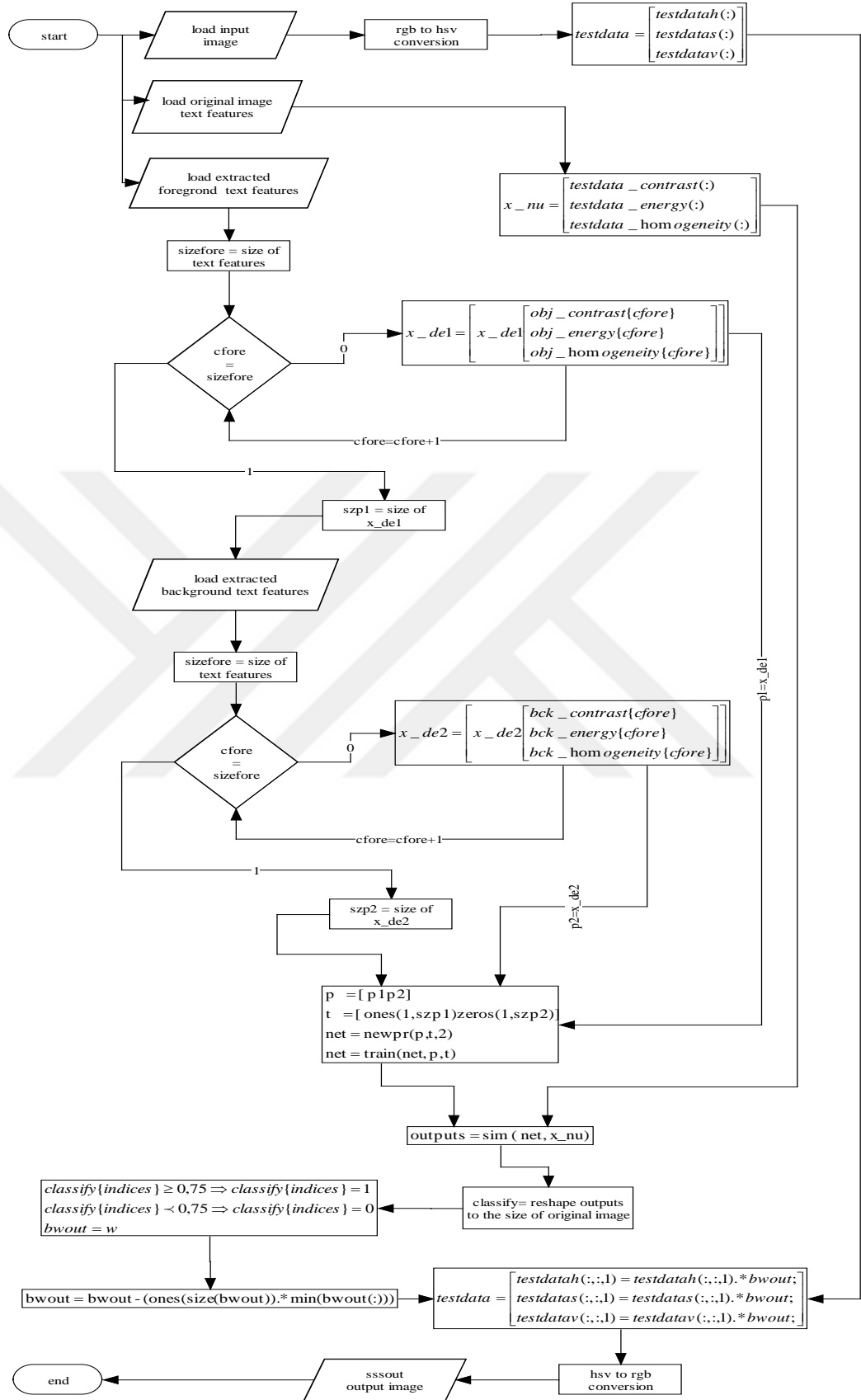


Figure 4.15 Flowchart of supervised T2_NN algorithm

CHAPTER FIVE

TESTS AND RESULTS

In this chapter, we applied different methods explained in chapter 4 on the same input image, and we compare their outputs performances of them. Color image samples are taken from The Berkeley Segmentation Dataset, and texture samples are taken from the Brodatz Texture Book.

There is an expected binary output image in our hand, and we count correctly classified or misclassified pixel quantities. Six different performance parameters are used in this thesis. True Positive Flag (TPF) is the proportion of foreground pixels that were correctly identified. True Negative Flag (TNF) is proportion of background pixels that were classified correctly. False Positive Flag (FPF) is proportion of background pixels that were incorrectly classified as foreground. False Negative Flag (FNF) is misclassified proportion of foreground pixels that were incorrectly classified as background. Pper is proportion of the total number of pixels that were correctly classified. Nper is just opposite of Pper. TIME is segmentation algorithm runtime in terms of second.

5.1 Performance of Semi-Supervised One-Class Segmentation Methods Using Color Feature Set

Semi-supervised C_ULSIF and C_OSVM methods are applied on seastar, flower and church images.

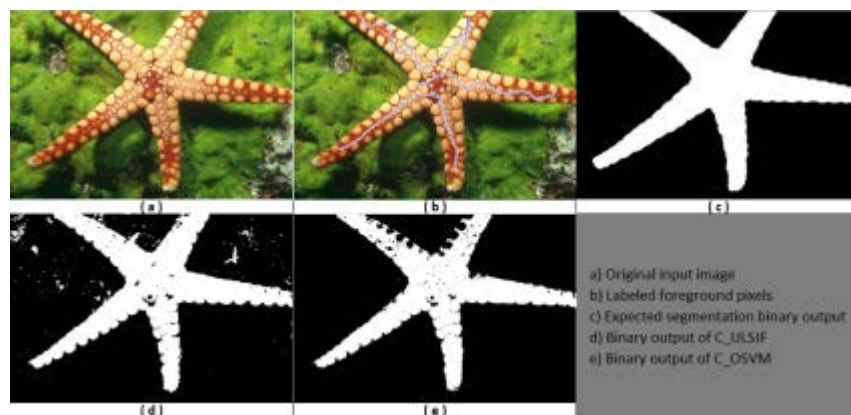


Figure 5.1 Semi-supervised color segmentation results for sea star

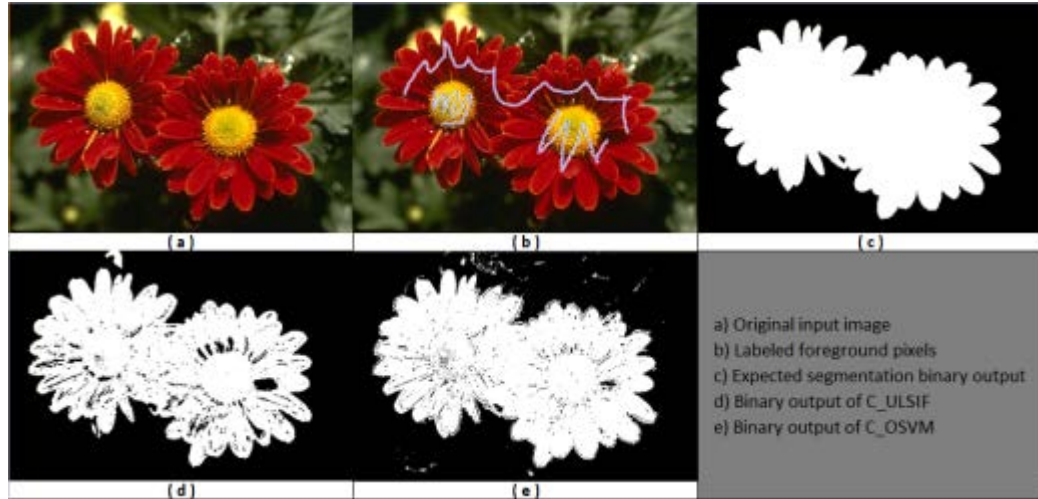


Figure 5.2 Semi-supervised color segmentation results for flower

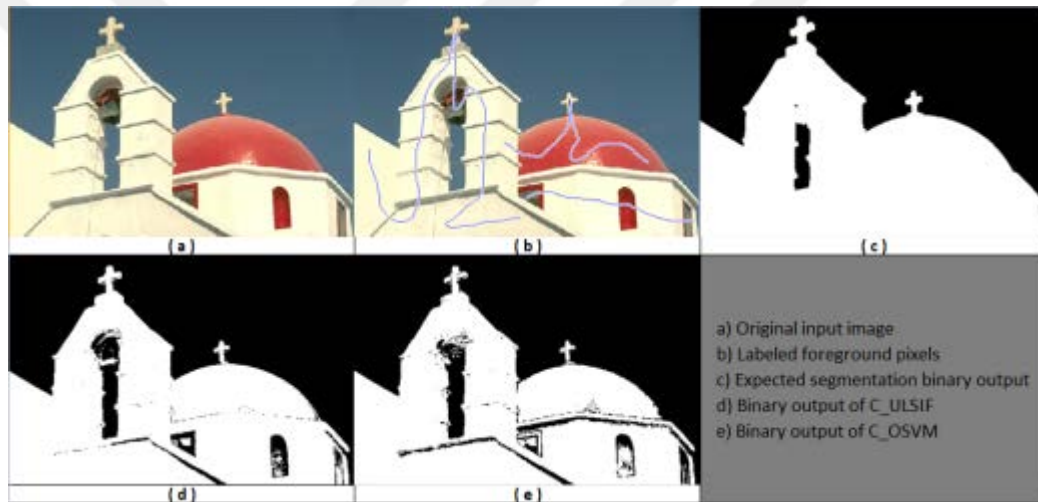


Figure 5.3 Semi-supervised color segmentation results for church

According to the performance results given in Table 5.1, average runtime of C_ULSIF is just 200msec longer. Therefore, C_ULSIF has better average segmentation performance because it has:

- Higher TPF, better segmentation performance on foreground pixels
- Higher TNF, better segmentation performance on background pixels
- Less FPF, means less mistake on background pixels
- Less FNF, means less mistake on foreground pixels
- Higher Pper, true classified pixel ratio on entire image is more
- Less Nper, wrong classified pixel ratio on entire image is less

Table 5.1 Performance of semi-supervised methods using color feature set

Applied Image Segmentation Methods			
Input Image	Performance Measurements	C_ULSIF	C_OSVM
Performance Measurements for Figure 5.1 Sea Star	TIME(sec.)	3.3571	3.5059
	FNF	0.0614	0.1666
	FPF	0.0214	0.0580
	Nper	0.0397	0.0445
	Pper	0.9646	0.9591
	TNF	0.9786	0.9420
	TPF	0.9386	0.8334
Performance Measurements for Figure 5.2 Flower	TIME(sec.)	5.8752	3.5800
	FNF	0.0915	0.0333
	FPF	0.0724	0.0264
	Nper	0.0480	0.0408
	Pper	0.9591	0.9674
	TNF	0.9276	0.9736
	TPF	0.9085	0.9667
Performance Measurements for Figure 5.3 Church	TIME(sec.)	2.5262	4.0300
	FNF	0.0669	0.0988
	FPF	0.0744	0.1099
	Nper	0.0363	0.0537
	Pper	0.9666	0.9502
	TNF	0.9256	0.8901
	TPF	0.9331	0.9012
Applied Image Segmentation Methods			
Input Image	Performance	C_ULSIF	C_OSVM
AVERAGE PERFORMANCES	TIME(sec.)	3.9195	3.7053
	FNF	0.0732	0.0996
	FPF	0.0561	0.0648
	Nper	0.0413	0.0464
	Pper	0.9634	0.9589
	TNF	0.9439	0.9352
	TPF	0.9268	0.9004

5.2 Performance of Supervised Two-Class Segmentation Methods Using Color Feature Set

In this section, two-class supervised C2_ULSIF, C2_SVM and C2_NN methods are applied on the same images in Section 5.1.

These methods are using background color features additionally, so they have an extra labeling labor for supervisor, and also additional time for background feature extraction.

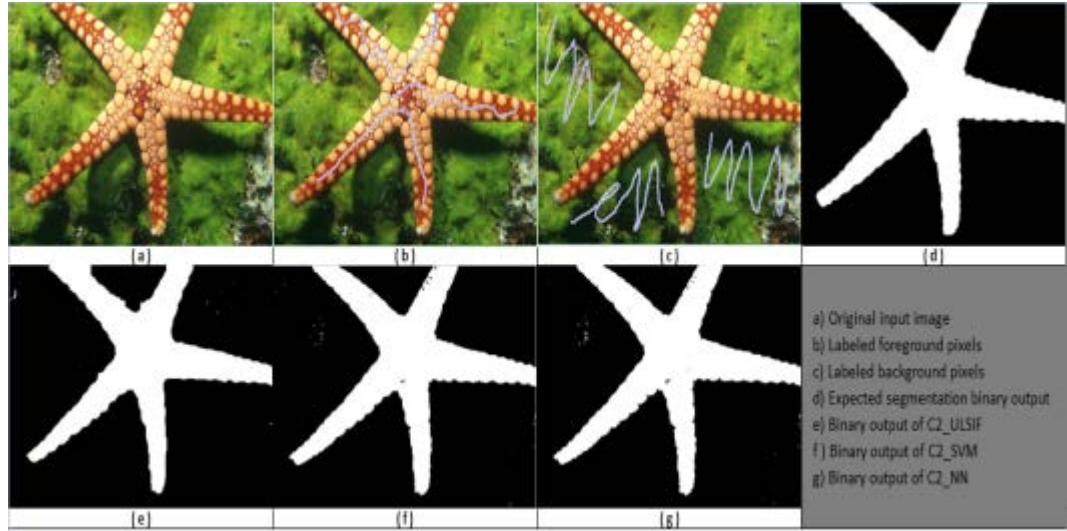


Figure 5.4 Supervised color segmentation results for sea-star



Figure 5.5 Supervised color segmentation results for flower

When segmentation performances are compared between Table 5.1 and Table 5.2 we can observe that TIME values are higher for two-class supervised methods, which means longer runtime. On the other hand, TPF TNF and Pper values are higher for two-class supervised methods which means better segmentation performance on overall image. Although C2_ULSIF and C2_NN algorithms are slower, C2_SVM algorithm is the fastest one between all supervised methods using color feature set.

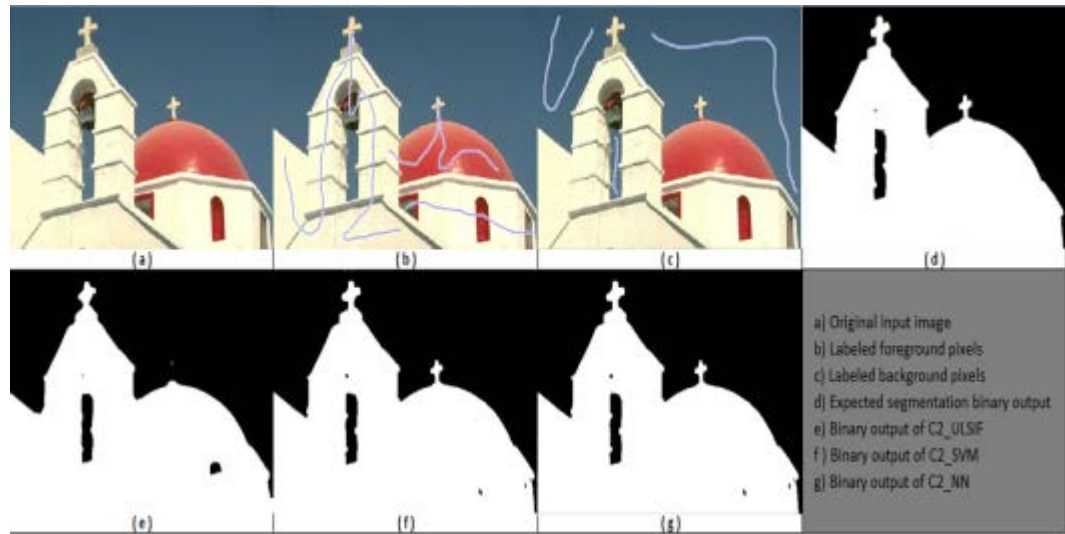


Figure 5.6 Supervised color segmentation results for church

Table 5.2 Performance of supervised methods using color feature set

Applied Image Segmentation Methods				
Input Image	Performance Measurements	C2_ulsif	C2_svm	C2_nn
Performance Measurements for Figure 5.4	TIME(sec.)	6.6486	0.6039	4.5601
	FNF	0.0752	0.0544	0.0641
	FPF	0.0262	0.0190	0.0223
	Nper	0.0276	0.0155	0.0183
	Pper	0.9744	0.9869	0.9842
	TNF	0.9738	0.9810	0.9777
	TPF	0.9248	0.9456	0.9359
Performance Measurements for Figure 5.5	TIME(sec.)	8.8835	0.8844	0.9434
	FNF	0.0958	0.0128	0.0722
	FPF	0.0759	0.0101	0.0572
	Nper	0.0582	0.0324	0.0496
	Pper	0.9444	0.9731	0.9578
	TNF	0.9241	0.9899	0.9428
	TPF	0.9042	0.9872	0.9278
Performance Measurements for Figure 5.6	TIME(sec.)	7.2739	0.6204	1.0203
	FNF	0.0119	0.0074	0.0084
	FPF	0.0132	0.0082	0.0093
	Nper	0.0112	0.0056	0.0052
	Pper	0.9902	0.9957	0.9961
	TNF	0.9868	0.9918	0.9907
	TPF	0.9881	0.9926	0.9916
Applied Image Segmentation Methods				
Input Image	Performance	C2_ulsif	C2_svm	C2_nn
AVERAGE PERFORMANCES	TIME(sec.)	7.6020	0.7029	2.1746
	FNF	0.0610	0.0249	0.0482
	FPF	0.0384	0.0124	0.0296
	Nper	0.0323	0.0178	0.0244
	Pper	0.9696	0.9852	0.9794
	TNF	0.9616	0.9876	0.9704
	TPF	0.9390	0.9751	0.9518

Semi-supervised and supervised color segmentation methods have some critical settings which effect output performance directly. For example b is kernel number of ULSIF, outlier fraction ratio of OSVM, and hidden layer numbers in NN. Finding optimal values of these parameters are explained in sections 5.3 and 5.4.

5.3 Performance of Semi-Supervised One-Class Segmentation Methods Using Texture Feature Set

Input image which consists of 4 different 128x128 pixel sub-images, with different texture information, is tested on different algorithms.

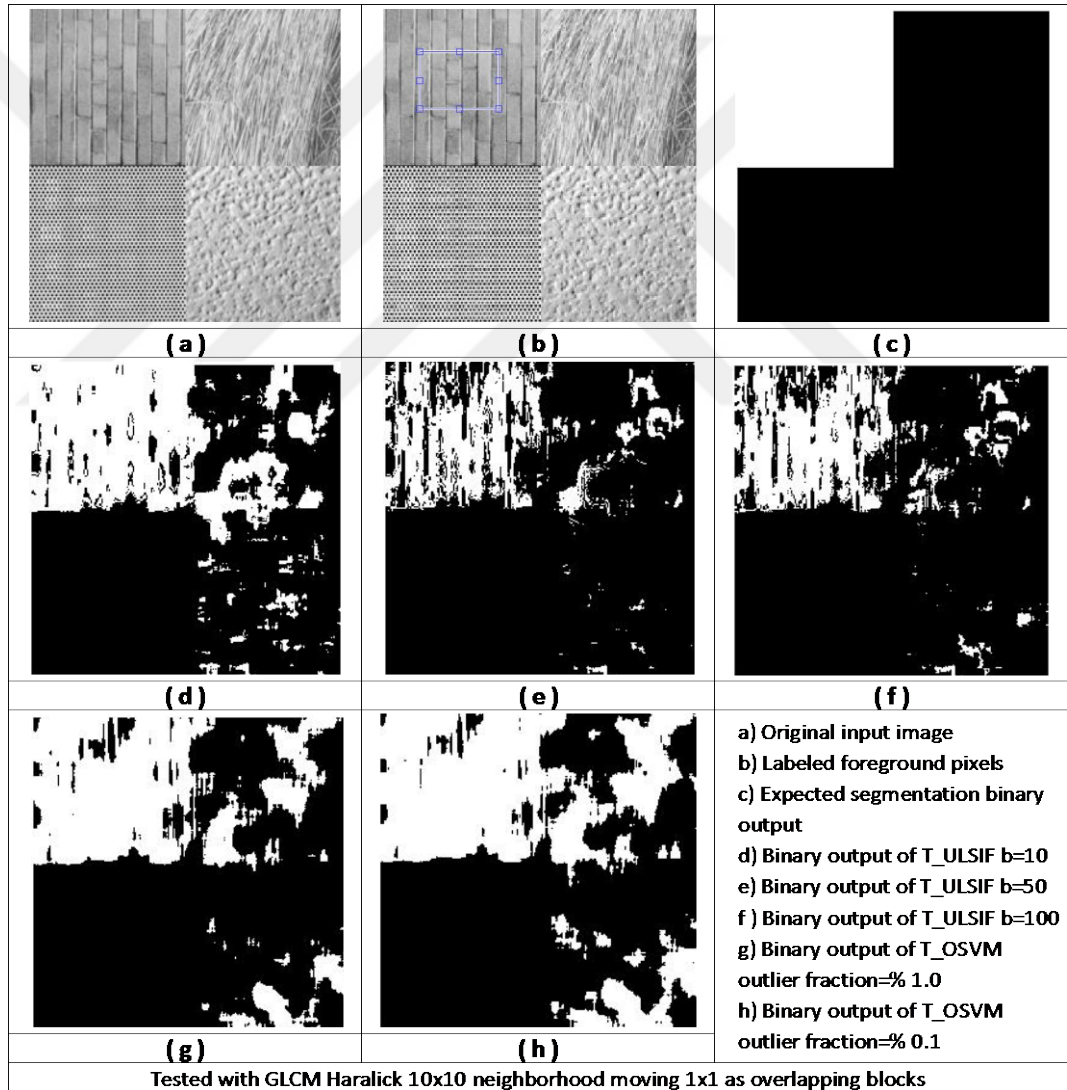


Figure 5.7 Semi-Supervised texture segmentation results while choosing parameter b for ULSIF and outlier fraction parameter for OSVM

Table 5.3 Performance of semi-supervised methods while choosing parameter b for ULSIF and outlier fraction parameter for OSVM

Applied Image Segmentation Methods			Applied Image Segmentation Methods		
Input Image	Performance Measurements	T_ULSIF	Input Image	Performance Measurements	T_OSVM
Performance Measurements for Figure 5.7 d	TIME(sec.)	150.0100	Performance Measurements for Figure 5.7 g	TIME(sec.)	108.4740
	TPF	0.8442		TPF	0.7867
	FPF	0.0519		FPF	0.0711
	TNF	0.9481		TNF	0.9289
	FNF	0.1558		FNF	0.2133
	pper	0.8466		pper	0.8562
	nper	0.1584		nper	0.1479
Performance Measurements for Figure 5.7 e	TIME(sec.)	406.8200	Performance Measurements for Figure 5.7 h	TIME(sec.)	110.3820
	TPF	0.6575		TPF	0.5237
	FPF	0.1142		FPF	0.1588
	TNF	0.8858		TNF	0.8412
	FNF	0.3425		FNF	0.4763
	pper	0.8632		pper	0.8086
	nper	0.1457		nper	0.1954
Performance Measurements for Figure 5.7 f	TIME(sec.)	754.4240			
	TPF	0.7844			
	FPF	0.0719			
	TNF	0.9281			
	FNF	0.2156			
	pper	0.8706			
	nper	0.1371			

According to the output images in Figure 5.7 and performance results in Table 5.3, increasing number of kernels in ULSIF has positive effect but when b is greater than 50 this becomes a disadvantage. Because of that we selected b=50 as an optimal setting for all semi-supervised or supervised methods using ULSIF.

When outlier fraction ratio is %1 OSVM gives better performance. Giving lower fraction ratio is reducing the segmentation performance since it reduces Pper and increase runtime.

For one-class semi-supervised texture segmentation, T_ULSIF has better performance than T_OSVM, with higher TPF-TNF- Pper and lower FPF-FNF-Nper.

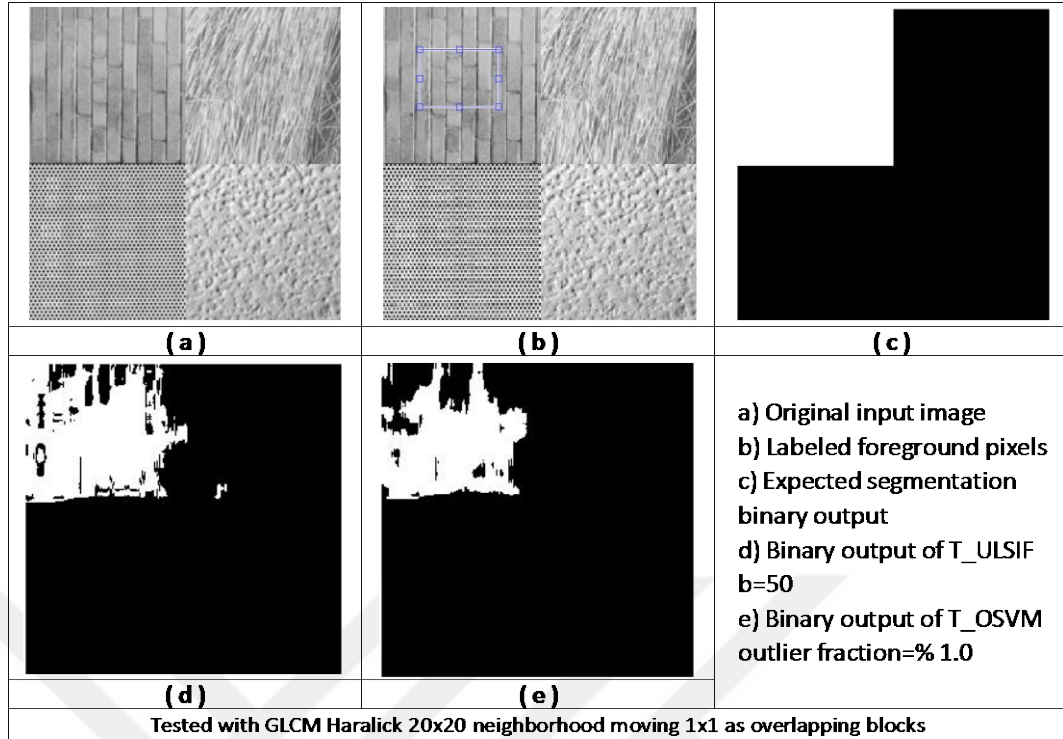


Figure 5.8 Semi-supervised texture segmentation results while choosing size of texture blocks

During the texture extraction stage blocproc function is used. At the beginning 10x10 block size is chosen and the center point of the block is moved 1 pixel right or down to produce overlapping blocks on original input image while calculating GLCM. In addition, 16 different neighborhoods are chosen for $D = \{1,2,3,4\}$ to all directions shown in Figure 2.2.

Then we decided to increase block size to 20x20. According to the output images in Figure 5.7, Figure 5.8, Figure 5.9 and in Table 5.4 ; after increasing block size, T_ulsif and T_osvm have performed better segmentation results. Because of that we decided to use 20x20 block size as standard for all texture extraction stages.

Also, effect of using non-overlapping 20x20 sized blocks (by shifting center point of the block 20 pixel right or down in each stage) is tested. According to the segmentation outputs in Figure 5.10 and Table 5.5, this has reduced the performance of T_ulsif and T_osvm. There is no advantage of using non-overlapping blocks, so using 20x20 overlapping blocks is preferred.

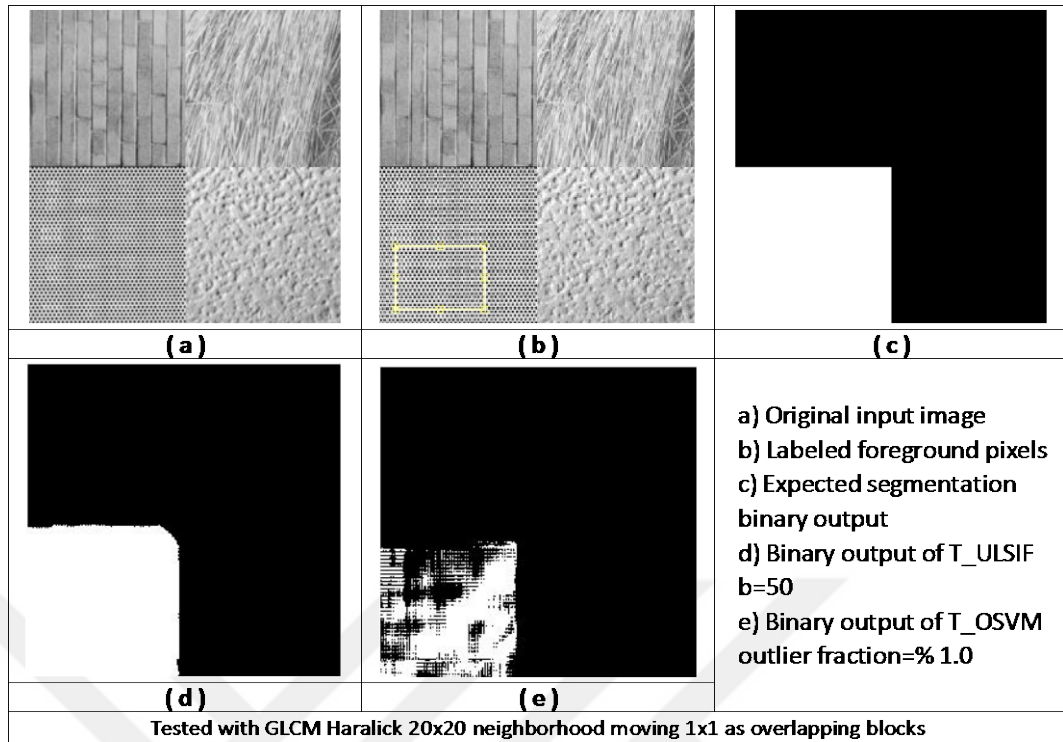


Figure 5.9 Semi-supervised texture segmentation results while testing 20x20 sized overlapping texture blocks

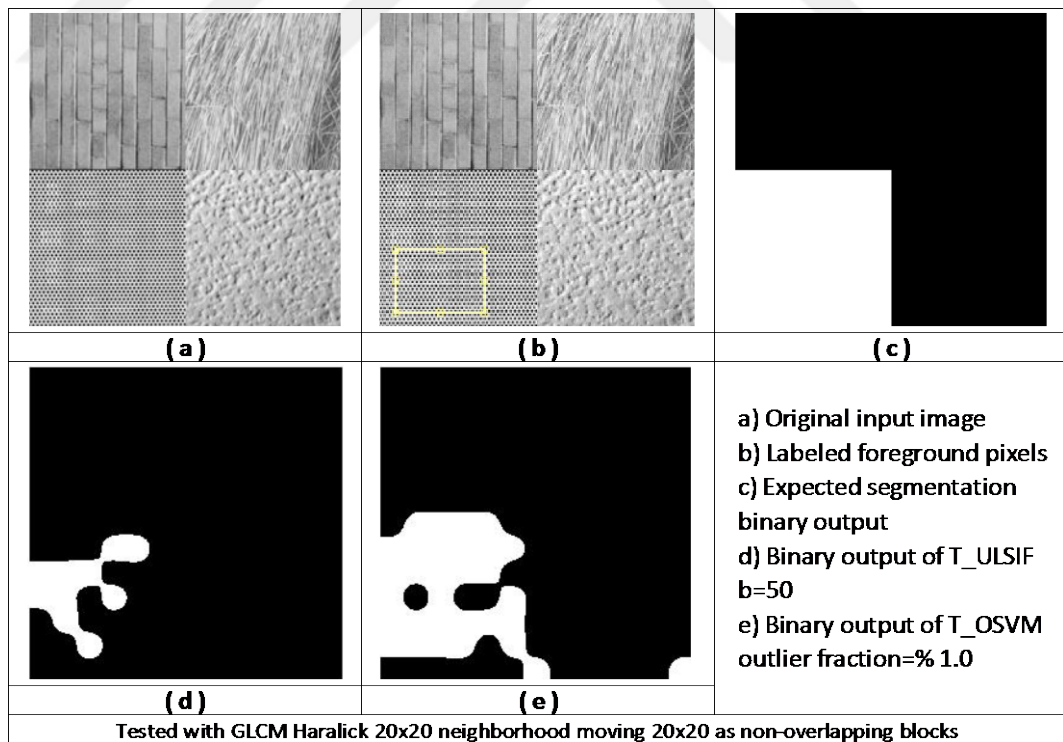


Figure 5.10 Semi-supervised texture segmentation results while testing 20x20 sized non-overlapping texture blocks

Table 5.4 Performance of semi-supervised methods with 20x20 sized overlapping texture blocks

Applied Image Segmentation Methods			
Input Image	Performance Measurements	T_ULSIF	T_OSVM
Performance Measurements for Figure 5.8	TIME(sec.)	328.8060	166.9590
	TPF	0.6346	0.5064
	FPF	0.1218	0.1645
	TNF	0.8782	0.8355
	FNF	0.3654	0.4936
	pper	0.9065	0.8772
	nper	0.0951	0.1239
Performance Measurements for Figure 5.9	TIME(sec.)	514.7270	229.4570
	TPF	0.9370	0.4508
	FPF	0.0210	0.1831
	TNF	0.9790	0.8169
	FNF	0.0630	0.5492
	pper	0.9844	0.8653
	nper	0.0159	0.1399

Table 5.5 Performance of semi-supervised methods with 20x20 sized non-overlapping texture blocks

Applied Image Segmentation Methods			
Input Image	Performance Measurements	T_ULSIF	T_OSVM
Performance Measurements for Figure 5.10	TIME(sec.)	0.2110	0.7845
	TPF	0.2403	0.7371
	FPF	0.2532	0.0876
	TNF	0.7468	0.9124
	FNF	0.7597	0.2629
	pper	0.8103	0.8958
	nper	0.1901	0.1049

5.4 Performance of Supervised Two-Class Segmentation Methods Using Texture Feature Set

In this section two-class supervised texture segmentation methods are used. One texture region in Figure 5.11(b) is labeled as foreground, and three different regions in Figure 5.11 (c) (d) (e) are labeled as background.

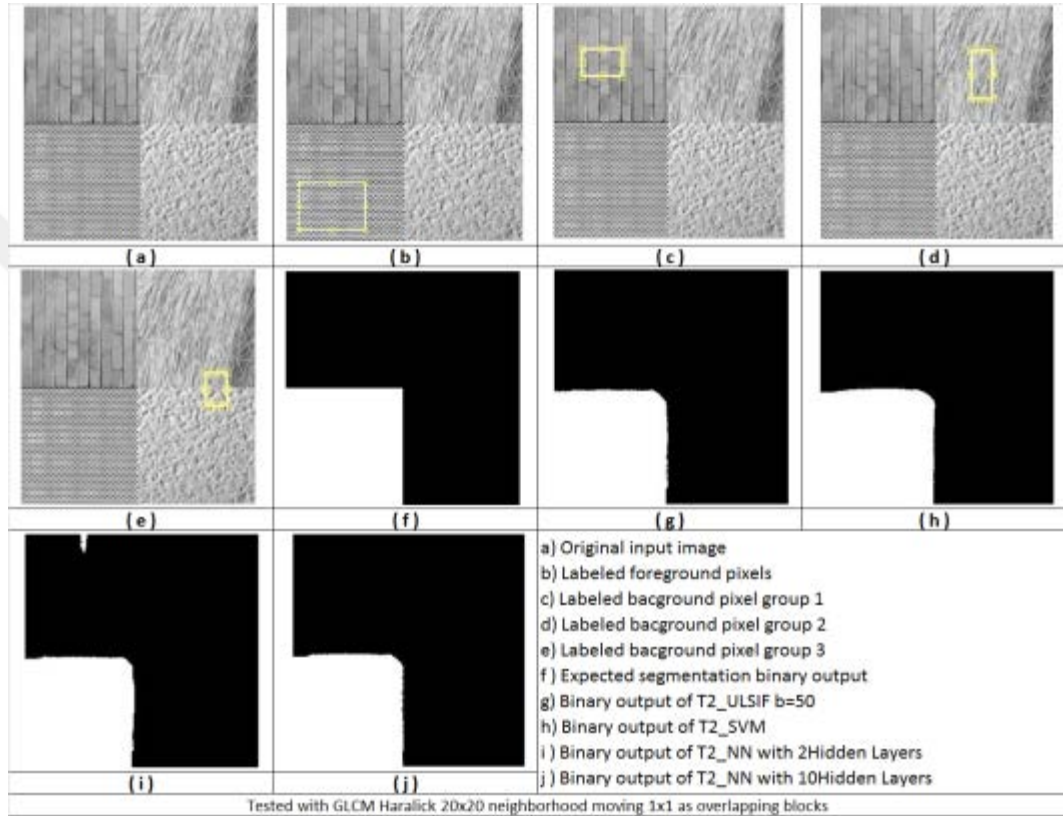


Figure 5.11 Supervised texture segmentation results while testing 20x20 overlapping texture blocks

According to output images in Figure 5.11 and Table 5.6, C2_ulsif and C2_SVM have very close and better performance than C2_NN. The only disadvantage of C2_ulsif is that it is the slowest supervised texture segmentation algorithm between these three methods. Also, we can observe that the more hidden layer number in Neural Network, the better segmentation performance.

Table 5.6 Performance of supervised methods in Figure 5.11

Input Image	Performance Measurements		Method
Performance Measurements for Figure 5.11 g	elapsedTime	126.0909	T2_ULSIF
	TPF	0.9293	
	FPF	0.0236	
	TNF	0.9764	
	FNF	0.0707	
	pper	0.9826	
	nper	0.0179	
Performance Measurements for Figure 5.11 h	elapsedTime	0.484	T2_SVM
	TPF	0.9506	
	FPF	0.0165	
	TNF	0.9835	
	FNF	0.0494	
	pper	0.9878	
	nper	0.0125	
Performance Measurements for Figure 5.11 i	elapsedTime	33,522	T2_NN 2 Hidden Layers
	TPF	0.8799	
	FPF	0.04	
	TNF	0.96	
	FNF	0.1201	
	pper	0.967	
	nper	0.0334	
Performance Measurements for Figure 5.11 j	elapsedTime	35,933	T2_NN 10 Hidden Layers
	TPF	0.9102	
	FPF	0.0299	
	TNF	0.9701	
	FNF	0.0898	
	pper	0.9777	
	nper	0.0226	

In Figure 5.12 we selected another texture region for foreground labeling. This time, according to Table 5.7, C2_ULSIF has lower performance than other two alternative methods and C2_SVM is the best performer again.

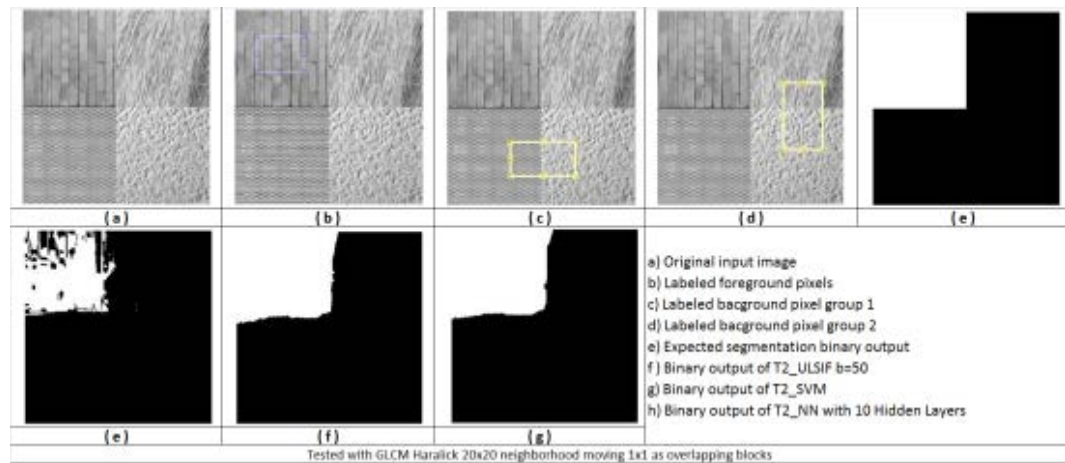


Figure 5.12 Supervised texture segmentation results while testing 20x20 sized overlapping texture blocks, for different labeled region

Table 5.7 Performance of supervised methods in Figure 5.12

Input Image	Performance Measurements		Method
Performance Measurements for Figure 5.12 e	elapsedTime	241.2920	T2_ULSIF
	TPF	0.5961	
	FPF	0.1346	
	TNF	0.8654	
	FNF	0.4039	
	pper	0.8999	
	nper	0.1018	
Performance Measurements for Figure 5.12 f	elapsedTime	16,164	T2_SVM
	TPF	0.9313	
	FPF	0.0229	
	TNF	0.9771	
	FNF	0.0687	
	pper	0.9665	
	nper	0.0338	
Performance Measurements for Figure 5.12 g	elapsedTime	56,042	T2_NN 10 Hidden Layers
	TPF	0.9031	
	FPF	0.0323	
	TNF	0.9677	
	FNF	0.0969	
	pper	0.9648	
	nper	0.0354	

CHAPTER SIX

CONCLUSION

In this thesis, a new approach to semi-supervised segmentation is proposed, by implementing Unconstraint Least Square Importance Fitting (ULSIF) algorithm. ULSIF is an algorithm based on importance estimation which is used for anomaly and outlier detection problems.

For semi-supervised methods, only color or texture features of one-class is used. Ratio of the labeled class' probability distribution function and the test data probability distribution function is estimated, and this ratio is called importance. If the importance value is greater, test sample is belonging to the labeled class. The biggest advantage of using ULSIF is, there is no need to estimate probability distribution functions of the labeled class and the test data separately. It estimates directly the ratio of them. Because of that feature, it is not needed to tune pdf parameters which is important to set for multi-dimensional space. As we know, there is no previous use of ULSIF for image segmentation in the literature. Because of that different features are used to observe the performance of ULSIF, and hue, saturation and value are used as color features, also energy, contrast and homogeneity are used as texture features. The performance of ULSIF is compared with the state of art method one-class SVM. According to the results, both color and texture based semi-supervised one-class ULSIF gave better results than OSVM does.

Additionally, labeled features of two classes are used for supervised image segmentation methods. For supervised two-class ULSIF, foreground importance and background importances are estimated separately. If foreground importance value of a test data is greater than the background importance of that data, it means this test data is belonging to the foreground, or vice versa. Results of supervised two-class ULSIF is compared with SVM and Neural networks. Supervised two-class SVM is the best method in supervised methods.

REFERENCES

- Application of Neural Networks to Modeling and Control of Parallel Manipulators Backpropagation. Three-layer feedforward network.* (2008). Retrieved August 24, 2015, from http://www.intechopen.com/books/parallel_manipulators_new_developments/application_of_neural_networks_to_modeling_and_control_of_parallel_manipulators
- Boersma, P., & Weenink, D. (2004). *Feedforward neural networks*. Retrieved September 19, 2015, from http://www.fon.hum.uva.nl/praat/manual/Feedforward_neural_networks_1__What_is_a_feedforward_ne.html
- Bora, D.J., Gupta, A.K., & Khan, F.A. (2015). Comparing the Performance of L*A*B* and HSV Color Spaces with Respect to Color Image Segmentation. *Cornell University Library, Computer Science, Computer Vision and Pattern Recognition*. Retrieved August 13, 2015, from <http://arxiv.org/abs/1506.01472>
- Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. The MIT Press: Cambridge, Massachusetts London, England
- Dopido, I., Jun Li, Marpu, P.R., & Plaza, A. (2013). Semisupervised Self-Learning for Hyperspectral Image Classification. *Geoscience and Remote Sensing, IEEE Transactions*, 51, 4032-4044.
- D'Silva, P. (2008). *A Little About Color: HSV vs. RGB*. Retrived January 12, 2016, from http://www.kirupa.com/design/little_about_color_hsv_rgb.htm
- Eleyan, A., & Demirel, H. (2011). Co-occurrence matrix and its statistical features as a new approach for face recognition. *Turkish Journal of Electrical Engineering & Comp Science*, 19, No.1

Feedforward neural network, (2015). Retrieved August 24, 2015, from https://en.wikipedia.org/wiki/Feedforward_neural_network

Gray-Level Co-Occurrence Matrix (GLCM), (2015). Retrieved August 23, 2015, from <http://www.mathworks.com/help/images/gray-level-co-occurrence-matrix-gldm.html>

Guillaumin, M., Verbeek, J., & Schmid, C. (2010). Multimodal semi-supervised learning for image classification. *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference*, 902-909.

HSV (Hue, Saturation and Value), (2014). Retrieved August 23, 2015, from <http://www.tech-faq.com/hsv.html>

Image texture, (2015). Retrieved August 23, 2015, from https://en.wikipedia.org/wiki/Image_texture

Jiazhen, X., Xinmeng, C., & Xuejuan, H. (2008). Interactive image segmentation by semi-supervised learning ensemble. *International Symposium on Knowledge Acquisition and Modeling*, 645-648.

Kanamori, T., Hido, S., & Sugiyama, M. (2009). A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 10, 1391-1445

Kapur, T., Grimson, E., Wells, W., & Kikinis, R. (1996). Segmentation of brain tissue from magnetic resonance images. *Medical Image Analysis*, 1, 109-127.

Labbe, B., Herault, R., & Chatelain, C. (2009). Learning deep neural networks for high dimensional output problems. *Machine Learning and Applications (ICMLA) 2009 International Conference*, 63-68.

- LeCun, Y. & Bengio, Y. (1995). *Pattern recognition and neural networks, The Handbook of Brain Theory and Neural Networks*. MIT Press. Retrieved September 13, 2015 from <http://yann.lecun.com/exdb/publis/pdf/lecun-bengio-95b.pdf>
- Martínez-Usó, A., Pla, F., & Sotoca, J.M. (2010). A semi-supervised gaussian mixture model for image segmentation, *Pattern Recognition (ICPR). 20th International Conference*, 2941 – 2944.
- Mikic, I., Cosman, P.C., Kogut, G.T., & Trivedi, M.M. (2000). Moving shadow and object detection in traffic scenes. *Pattern Recognition Proceedings 2000 15th International Conference*, 1, 321-324.
- Mohanaiah, P., Sathyanarayana, P., & GuruKumar, L. (2013). Image texture feature extraction using GLCM approach. *International Journal of Scientific and Research Publications 2013*, 3.
- Multilayer Feedforward Neural Networks*, (2015). Retrieved August 25, 2015, from <http://docs.roguewave.com/imsl/c/6.0/stat/default.htm?url=multilayerfeedforwardneuralnetworks.htm>
- Needham, C.J., & Boyle, R.D. (2001). Tracking multiple sports players through occlusion, congestion and scale. *University of Leeds School of Computing Research Report Series 2001*, 07.
- Paiva, A.R.C., & Tasdizen, T. (2010). Fast semi-supervised image segmentation by novelty selection. *Acoustics Speech and Signal Processing (ICASSP) IEEE International Conference*, 1054 – 1057.
- Patternnet*, (2015). Retrieved August 24, 2015, from <http://www.mathworks.com/help/nnet/ref/patternnet.html>

- Peng, B., Zhang, L., Zhang, D., & Yang, J. (2011). Image segmentation by iterated region merging with localized graph cuts. *Semi-Supervised Learning for Visual Content Analysis and Understanding*, 44, 10–11, 2527–2538.
- Rao, R. P. N., & Chen, J.-H. (2009). *Computer vision lecture 12 texture*. Retrieved August 23, 2015, from <https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect12.pdf>
- Specify Offset Used in GLCM Calculation*, (2015). Retrieved January 21, 2015, from <http://www.mathworks.com/help/images/gray-level-co-occurrence-matrix-glcm.html?requestedDomain=www.mathworks.com>
- Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., & Williamson, R.C. (1999). Estimating the support of a high-dimensional distribution. *Technical report, Microsoft Research*, MSR-TR-99-87.
- Semi-supervised learning*, (2015). Retrieved August 20, 2015, from https://en.wikipedia.org/wiki/Semi-supervised_learning
- Stergiou, C., & Siganos, S. (1997). *Neural networks*. Retrieved September 19, 2015, from http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html
- Stringa, E., & Regazzoni, C.S. (2000). Real-time video-shot detection for scene surveillance applications. *Image Processing, IEEE Transactions*, 9, 69-79.
- Support vector machine*, (2015). Retrieved August 24, 2015, from https://en.wikipedia.org/wiki/Support_vector_machine
- Support Vector Machines (SVM) Introductory Overview*. (2016). Retrieved August 23, 2015, from <http://www.statsoft.com/Textbook/Support-Vector-Machines>

Sural, S., Gang Qian, & Pramanik, S. (2002). Segmentation and histogram generation using the HSV color space for image retrieval. *Image Processing Proceedings International Conference 2002*, 2, 589-592 .

The Berkeley Segmentation Dataset, (2001). Retrieved August 23, 2015, from <https://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/BSDS300-images.tgz>

Tuia, D., & Camps-Valls, G. (2011). Urban image classification with semisupervised multiscale cluster kernels. *Selected Topics in Applied Earth Observations and Remote Sensing*, 4, 65-74.

Vlasveld, R. (2013). *Introduction to one-class support vector machines*. Retrieved July 12, 2015, from <http://rvlasveld.github.io/blog/2013/07/12/introduction-to-one-class-support-vector-machines/>

What is a SVM? (2015). Retrieved August 23, 2015, from http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

Zhu, X. (2007). Semi-supervised learning tutorial. *The International Machine Learning Society*. Retrieved September 25, 2015, from <http://pages.cs.wisc.edu/~jerryzhu/pub/sslicml07.pdf>