Automated Resolution Selection for Image Segmentation

by

Fares Al-Qunaieer

A thesis
submitted to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2014

Abstract

It is well known in image processing in general, and hence in image segmentation in particular, that computational cost increases rapidly with the number and dimensions of the images to be processed. Several fields, such as astronomy, remote sensing, and medical imaging, use very large images, which might also be 3D and/or captured at several frequency bands, all adding to the computational expense.

Multiresolution analysis is one method of increasing the efficiency of the segmentation process. One multiresolution approach is the coarse-to-fine segmentation strategy, whereby the segmentation starts at a coarse resolution and is then fine-tuned during subsequent steps. Until now, the starting resolution for segmentation has been selected arbitrarily with no clear selection criteria.

The research conducted for this thesis showed that starting from different resolutions for image segmentation results in different accuracies and speeds, even for images from the same dataset. An automated method for resolution selection for an input image would thus be beneficial. This thesis introduces a framework for the selection of the best resolution for image segmentation. First proposed is a measure for defining the best resolution based on user/system criteria, which offers a trade-off between accuracy and time. A learning approach is then described for the selection of the resolution, whereby extracted image features are mapped to the previously determined best resolution.

In the learning process, class (i.e., resolution) distribution is imbalanced, making effective learning from the data difficult. A variant of AdaBoost, called RAMOBoost, is therefore used in this research for the learning-based selection of the best resolution for image segmentation. RAMOBoost is designed specifically for learning from imbalanced data. Two sets of features are used: Local Binary Patterns (LBP) and statistical features.

Experiments conducted with four datasets using three different segmentation algorithms show that the resolutions selected through learning enable much faster segmentation than the original ones, while retaining at least the original accuracy. For three of the four datasets used, the segmentation results obtained with the proposed framework were significantly better than with the original resolution with respect to both accuracy and time.

Acknowledgements

I have faced many obstacles and difficulties while pursuing my PhD degree and writing this thesis. Several individuals made this journey possible and relieved some of the stress.

First, I would like to thank my supervisor, Dr. Hamid Tizhoosh, and co-supervisor, Dr. Shahryar Rahnamayan, for their time and support. I have learned valuable lessons from them. I appreciated their fast response and availability when needed. I also want to thank all the committee members for their time and valuable suggestions on how to improve this thesis.

Special thanks go to my wife and children for their support and patience during the entire time that I worked on the thesis, and to my parents for their endless and constant motivation and encouragement.

I would like to thank King Abdulaziz City for Science and Technology for their scholarship and support. I would also like to thank Dr. Haibo He for his fruitful discussion on the topic of learning from imbalanced class distribution. Many thanks go to everyone who discussed and reviewed this thesis with me.

The Saudi Student Association of Waterloo has been helping me since I arrived at Waterloo. They made me feel as though I were still at my home. I would like to thank everyone who made this possible. Your support has been invaluable to me.

Dedication

to my wife and children for their endless support

and

to my parents, who always encouraged me to learn and grow

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Image segmentation is an essential component of many image processing and computer vision applications. Extracted regions are used for a variety of purposes, such as object detection and recognition [1], size/volume measurement [2], and content-based image retrieval [3]. Because of its importance, tremendous effort has been invested in designing accurate and efficient segmentation algorithms. Concepts from numerous fields, such as optimization [4], statistics [5], information theory [6], and graph theory [7], have been utilized in this area.

An inherent challenge in image processing in general, and in image segmentation in particular, is the rapid increase in computational cost that accompanies an expanding number of images and their dimensions. This issue is crucial because, in many fields, such as astronomy [8], remote sensing [9], and medical imaging [10], the number of images and their dimensions is extensive. Some of these images also have three dimensions and/or have been captured at several frequency bands (i.e., multi-spectral).

Processing such images requires either powerful hardware (e.g., GPUs), or enhanced algorithms (e.g., parallel design), or both. As presented in the literature, an established method of satisfying these requirements is multiresolution analysis [11, 12, 13]. In addition to speeding up the segmentation, multiresolution analysis also increases the accuracy of many techniques because images at different resolutions can be viewed in different perspectives. Large objects and major structures and edges can be analyzed at coarse resolutions, while fine details and small objects are examined at finer resolutions. At lower resolutions, noise is also reduced, and regions become more homogeneous.

Multiresolution analysis has been used extensively for developing a variety of image segmentation algorithms. The majority of these methods utilize information from some or all resolutions, and some of these methods are reviewed in Section 3.2. This work was targeted at a multiresolution technique for increasing image segmentation efficiency, namely, the coarse-to-fine segmentation strategy, in which the segmentation process starts with a coarse resolution and is then fine-tuned at finer ones. Because it simplifies the input image rather than altering the segmentation algorithm, numerous segmentation algorithms can be used with this method.

1.1 Motivation and Problem Formulation

Processing large images is computationally expensive. The computation dramatically increases with images captured in 3D (multiple slices) or/and in multiple frequency bands. Recently, new technology enables the capturing of images in much greater detail, which results in more data to process. Medical imaging is an example in which an image could require Gigabytes or Terabytes of memory [10]. These images can be very large or/and consist of thousands of slices [10]. Another example is hyperspectral images capturing hundreds of spectral bands [14]. Furthermore, these images could be processed in patches. One of the challenges for such images is the slow processing time [10]. Even a slight increase in speed for processing an image or slice of it (for 3D or multi-spectral images) can amount to much greater overall speed.

In some cases, it is not affordable to use powerful hardware for image processing. Therefore, several research projects have been conducted on developing software solutions, such as parallel algorithms [15] and multiresolution schemes [11, 12, 13]. The multiresolution coarse-to-fine image segmentation proved to be an efficient method for increasing both the speed and accuracy of the used segmentation algorithm as discussed in Section 3.2.

Some researchers have used image segmentation algorithms in a coarse-to-fine fashion with arbitrarily chosen initial resolutions for the processing of all images [16, 17, 18, 19]. However, segmenting an image at a variety of resolutions results in different accuracies and running times, as discussed in Sections 4.2 and 6.2.1 and in [20, 21]. Different images yield varying segmentation accuracies and times per resolution, even with images from the same dataset as explained in Section 6.2.1. To the best of the author's knowledge, no work has been reported with respect to the automated selection of resolutions for image segmentation, for which a framework is proposed in this thesis. Studies have been conducted on scale selection for the scale-space representation, which are reviewed in Section 4.1, but they are not directly related to this research, as will be explained.

Defining the best resolution for image segmentation is empirical and is directly related to the nature of the application. Two factors can be used for defining the best resolution for image segmentation: accuracy and time. This research is concerned with the simultaneous consideration of both time and accuracy; hence investigating an intelligent approach toward establishing a trade-off between time and accuracy becomes crucial. For critical systems, such as medical applications, accuracy is the primary consideration, while in other applications, such as robot navigation or image retrieval, speed can be more important than perfect accuracy. There must therefore be a trade-off between accuracy and speed, and without an appropriate criterion, selecting a resolution that yields the desired accuracy and speed is difficult.

1.2 Objectives

The goal of this research was to develop a method for the automated selection of an initial resolution for image segmentation in a coarse-to-fine scheme. The overall goal was broken down into the following objectives:

- To investigate trade-off measures with respect to time and accuracy as a means of defining the best resolution for image segmentation according to application requirements. Best resolution generally has no clear definition and is application oriented.

The trade-off between accuracy and speed requirements varies enormously according to the applications.

- To develop a general framework for the resolution selection for image segmentation. Images are usually complex and can contain several objects of different sizes and with different characteristics, various noise types and levels, crisp or fuzzy edges, and varying degrees of uniformity in the regions. Therefore, with the exception of very specific cases, simple rules seems to be insufficient for estimating the best resolution for an image. In this research, machine learning methods are investigated to learn and estimate the best resolutions from training images.

- To identify both an efficient learning method and features suitable for the learning purposes of this research.

While this work intends to find the best resolution for image segmentation, it is not concerned with developing or modifying image segmentation algorithms, nor with multiresolution analysis methods.

1.3 Outline of the Thesis

This thesis consists of seven chapters organized as follows:

Chapter 1: The thesis is introduced, the motivation behind the work is explained, and the specific objectives are discussed.

Chapter 2: An overview of image segmentation techniques is provided. Image segmentation is first defined, and then associated approaches are briefly described.

Chapter 3: Multiresolution image segmentation is discussed: a number of multiresolution techniques are described, and several strategies of published multiresolution image segmentation in the literature are reviewed, with an emphasis on the coarse-to-fine strategy, the approach used in this research.

Chapter 4: Resolution selection for image segmentation is explained, and previous work on scale selection for the scale-space representation is distinguished from the research in this thesis. The performance of image segmentation at different resolutions is then analyzed based on simple experimental results.

Chapter 5: The proposed framework is described in detail. A trade-off measure for defining the best resolution is proposed. Then, image segmentation and the obtaining of the best resolution using the proposed trade-off measure are discussed, followed by a description of the proposed features to be used for learning. The learning method selected is presented, and the measures of classification performance are then discussed.

Chapter 6: This chapter explains the process of experimentally verifying the proposed framework and assessing the performance of the trade-off measure, along with an analysis of the behaviour of the trade-off measure with different segmentation algorithms for the same dataset. The classification performances of the learning method used with the features chosen are compared with those of other learning methods. The results of the examination of the impact of misclassification on both accuracy and time are presented. The results are discussed in the last section of this chapter.

Chapter 7: The main findings of this research are summarized, the contributions are highlighted, and the direction of future work is suggested.

Chapter 2

Image Segmentation

Image segmentation is an essential part of many computer vision applications. Numerous tasks, such as object recognition, first require the desired object to be extracted (segmented). Let I denote the input image; the aim of image segmentation is to partition I into n regions I_j, such that [22]

1. $\bigcup_{j=1}^{n} I_j = I$ (an image consists of multiple segments).

2. I_j is a connected set, where $j = 1, 2, \ldots, n$.

3. $I_j \cap I_k = \emptyset$ for all j and k, where $j \neq k$.

4. $Q(I_j) =$ TRUE for $j = 1, 2, \ldots, n$, where $Q(I_x)$ is a logical predicate defined over the pixels in the region I_x. An example of such a predicate is the similarity of intensity level.

5. $Q(I_j \cup I_k) =$ FALSE, for any adjacent regions I_j and I_k.

Because of the importance of image segmentation, numerous algorithms have already been proposed for performing this task accurately and efficiently. Existing segmentation algorithms can be categorized as follows: thresholding-based segmentation [23, 6], edge-based segmentation [24, 25], region-based segmentation [22, 26], active contour segmentation [4, 27], and other segmentation algorithms.

This chapter provides an overview of existing methods for image segmentation and the details of the methods used in this research.

2.1 Thresholding-Based Segmentation

Image thresholding algorithms represent the simplest segmentation methods. The goal of image thresholding is to discriminate objects from the background. Bi-level thresholding usually involves only one object and a widely uniform background, but thresholding techniques can also be applied to multiple objects (i.e., multi-level thresholding). Thresholding can be used for segmentation on its own or can be an element of a more complex processing chain as a pre-stage. Intensive effort has been devoted to the design of accurate thresholding algorithms, resulting in the reporting of numerous algorithms in the literature. Sezgin and Sankur [28] conducted an extensive review of more than 40 existing thresholding techniques. They categorized them into the following groups: histogram-shape-based methods, clustering-based methods, entropy-based methods, object-attribute-based methods, spatial methods, and local methods.

The Otsu method [23] is the most popular thresholding algorithm. From the histogram of an image, Otsu proposed an evaluation criterion based on the zeroth and first-order statistics. The aim of the algorithm is to minimize the measure of separability among the classes. Kapur et al. [6] proposed an algorithm based on the entropy concept. They defined the entropies that correspond to the distributions of the different classes. Then, in order to achieve the maximum information between the classes (hence, the optimal thresholding), the sum of the defined entropies is maximized. Kittler and Illingworth [29] modeled the classes (i.e., object and background) as a mixture of Gaussians obtained from the image histogram. The objective of their suggested algorithm is to minimize the classification error rate in order to obtain the minimum thresholding error.

A variety of techniques have been utilized for designing image thresholding algorithms. Tao et al. [30] proposed the use of normalized graph cuts. The weights of the graph were computed from the grey levels of the image. Yang et al. [31] proposed a Spatially Weighted Fuzzy C-Means (SWFCM) algorithm and incorporated the spatial information into the Fuzzy C-Mean (FCM) clustering algorithm. Opposition-Based Differential Evolution (ODE) was applied in order to find the best threshold value [32], and the thresholding task was modeled as a minimization problem. To solve the problem more quickly, the ODE with a very small population size ($N_p = 5$), called micro-ODE, was used. Rahnamayan et al. [33] proposed a method for the fusion of the thresholding results from different algorithms. They applied a weighted voting scheme as a means of finding the best threshold value. An algorithm for image thresholding based on opposite fuzzy sets was proposed by Al-Qunaieer et al. [34]. The proposed algorithm searches for the two fuzzy sets with minimum similarity in order to determine the threshold.

Computational intelligence techniques have also been used for image thresholding. To find the best thresholding value, Kang and Zhang [35] utilized the Cellular Neural Network (CNN) in conjunction with histogram analysis. A genetic algorithm was used by Ren [36] to estimate the optimal threshold value, and Particle Swarm Optimization (PSO) was utilized for image thresholding by Lin et al. [37], who considered each pixel as a particle and the optimal threshold as the food source.

Because the thresholding algorithms work on the intensity histogram, they work globally. This introduces problems in situations such as regions with multiple grey-scales (i.e., non-homogenous) and noisy images. Variants were suggested to overcome this problem, such as adaptive thresholding [38].

2.2 Edge-Based Segmentation

Edges can be detected through a search for sharp and local changes (discontinuities) in the intensity levels of an image. Since the goal is to find intensity changes, first and second order derivatives are commonly used for detecting edges. The simplest algorithms are those that use discrete masks for calculating the gradient of the image by convolving the image with the mask. Roberts [39], Prewitt [40], and Sobel [24] edge detectors are examples of the use of such masks (Figure 2.1). Roberts masks are used to detect diagonal edges by taking the difference between adjacent diagonals. Prewitt masks calculate the horizontal gradient (x-direction edges) H_g by subtracting the first and last rows and, in the same way, calculate the vertical gradient (y-direction edges) V_g by subtracting the first and last columns. Sobel masks perform the same operations as Prewitt masks but add additional weight to the centre pixel to smooth the result. Both the strength (magnitude M) and direction θ of the edges can be calculated from H_g and V_g, as follows:

$$M = \sqrt{H_g^2 + V_g^2}, \tag{2.1}$$

$$\theta = tan^{-1}\left[\frac{V_g}{H_g}\right]. \tag{2.2}$$

Prewitt and Sobel masks can be modified for enhanced detection of diagonal edges, as shown in Figure 2.2.

-1	0
0	1

0	-1
1	0

(a) Roberts

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

(b) Prewitt

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

(c) Sobel

Figure 2.1: Roberts, Prewitt, and Sobel masks for edge detection.

0	1	1
-1	0	1
-1	-1	0

-1	-1	0
-1	0	1
0	1	1

(a) Prewitt

0	1	2
-1	0	1
-2	-1	0

-2	-1	0
-1	0	1
0	1	2

(b) Sobel

Figure 2.2: Prewitt and Sobel masks for diagonal edge detection.

Marr and Hildreth [41] proposed a more sophisticated edge detection algorithm, which requires more processing steps than pure convolution methods. They used Laplacian of a Gaussian (LoG) as the edge detection operator, which is defined as

$$\nabla^2 G(x,y) = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\dfrac{x^2+y^2}{2\sigma^2}}, \tag{2.3}$$

where G is a 2D Gaussian function and ∇^2 is a Laplacian operator. The former is used for image smoothing and the latter to obtain the second derivative of the image. After the image is convolved with the operator in Eq. 2.3, the edges are found by searching for zero-crossings of the resulting image.

Canny [25] proposed one of the most popular edge detection algorithms. First, the input image is smoothed through convolving with a Gaussian function, and the gradient magnitude and angles are then calculated from the smoothed image. To thin the edges in the gradient magnitude image, non-maxima suppression is applied. The resulting image is thresholded, and connectivity analysis is conducted to find and connect the edges.

Several researchers have applied fuzzy logic for the detection of edges. Law et al. [42] considered edge detection to be a fuzzy reasoning problem and smoothed the input image based on that approach. After edge membership values for each pixel are evaluated, candidate edges are selected, and pixels with high edge membership are then connected using fuzzy reasoning. Fuzzy morphology was utilized by Gonzalez-Hidalgo et al. [43] to detect edges and denoise the image. They used a residual operator based on fuzzy opening and closing operations.

Neural network is a well-known soft computing tool that has been utilized for edge detection by several researchers [44, 45, 46], who applied backpropagation neural networks with different training patterns and network settings. Bhandarkar et al. [47] employed genetic algorithms for edge detection, with consideration of minimum cost edge configuration and the definition of edge configurations as 2D chromosomes.

Because edge-based image segmentation methods are local, they are sensitive to noise and are not working well with smooth edges and low contrast images. In addition, post-procession is required for edge linking.

2.3 Region-Based Segmentation

While edge detection algorithms depend primarily on discontinuities in the images, region-based algorithms look for similarities within regions. The characteristics of the regions, such as intensity, colours, and texture, are used to determine the homogeneity (uniformity) inside each region. The goal of such algorithms is to maximize the similarity within similar regions and the dissimilarity among different regions. Region-based algorithms are generally less sensitive to noise than edge-based algorithms, but they are also usually more complex.

Region growing [22] is considered one of the simplest categories of region-based algorithms. The central concept behind this technique is for regions to grow from predetermined points termed "seed points." Starting from each seed, neighbouring pixels that meet specified similarity criteria are added to the growing region. Many similarity criteria are related to intensity, colour, and other image properties. The choice of the initial seed points has a considerable effect on the performance of region-growing algorithms [48]. These points can be specified manually by allowing user interaction, or they can be obtained through an automated preprocessing step.

In contrast to region growing is the concept of split and merge [22]: this process starts with the whole image as one region. Recursively, regions are then split into subregions until pixels within all regions exhibit the same properties (homogeneity). When no further possibilities for splitting exist, adjacent regions with the same properties are merged into larger homogeneous regions. One method of implementing a split-and-merge process is to split the whole image into four regions, following which, as described previously, each subregion is then recursively divided into four regions [49]. With this method, the image and its subregions can be represented as a tree data structure called "quad-trees," in which each node has four children.

Clustering algorithms have also been used for region-based image segmentation. In such algorithms, similar pixels are clustered based on their properties, such as intensity or colour. Each cluster contains pixels with similar properties that represent a region. Tan et al. [26] proposed a region-based method based on fuzzy similarity. The image is first divided into blocks of pixels, and then based on the fuzzy similarity of neighbouring blocks similar blocks are merged.

Numerous researchers have employed computational intelligence techniques for image segmentation. Malki [50] utilized neural networks for image segmentation. A genetic algorithm was introduced by Visa [51] as a post-processing stage for improving the results of image segmentation. Bhandarkar and Zhang [52] proposed a method that utilizes genetic algorithms, with an algorithmic objective of minimizing a cost function defined based on both edge information and regional grey-scale uniformity.

Region-based segmentation methods are generally more complex than thresholding and edge-based methods and may not be suitable for segmenting objects having excess variation of intensities or textures. As stated previously, the choice of the initial seed can greatly affect the performance of region-growing methods. Statistical region merging algorithm, which is used in this research, is described next.

2.3.1 Statistical Region Merging

Statistical Region Merging (SRM) is a region merging image segmentation algorithm proposed by Nock and Nielsen [5] based on the formulation of image segmentation as an inference problem. An image I is considered to be sampled from a perfect unknown scene I^*, where pixels are represented by a family of distributions.

For regions R and R', the following merging logical predicate was suggested for grey-level regions:

$$\Gamma(R, R') = \begin{cases} true, & \text{if } |\overline{R} - \overline{R'}| \leq \sqrt{b^2(R) + b^2(R')}, \\ false, & \text{otherwise,} \end{cases} \tag{2.4}$$

where $\overline{R}$ is the average intensity of region R and $b(\cdot)$ is a merging threshold. The merging predicate for color regions is defined by

$$\Gamma(R, R') = \begin{cases} true, & \text{if } \forall a \in R, G, B,\ |\overline{R_a} - \overline{R'_a}| \leq \sqrt{b^2(R_a) + b^2(R'_a)}, \\ false, & \text{otherwise.} \end{cases} \tag{2.5}$$

The SRM segmentation method is described in Algorithm 1. Nock and Nielsen [5] tested two choices of function $f(p, p')$. The first is defined by

$$f(p, p') = |p - p'|, \tag{2.6}$$

and the Sobel convolution filter was the second choice.

Algorithm 1 Statistical Region Merging (SRM) [5]

```
Inputs
$S_I$: set of adjacent pixels in a 4-connectivity
$f(p, p')$: real valued function of $p$ and $p'$ pixels

Sort($S_I$) in increasing order of $f(p, p')$
for all $(p, p')$ pixels $\in S_I$ do
  if $\Gamma(R(p), R(p')) =$ true then
    Merge($R(p), R(p')$)
  end if
end for
```

The statistical complexity can be controlled by a parameter Q, which is used in the definition of the merging threshold, $b(\cdot)$, in Eq. 2.4 as detailed in [5]. Changing this parameter controls the scale of segmentation. A small value of Q enables the segmentation of only large regions, and as the value of Q increases, smaller regions can be segmented.

2.4 Active Contours

This section discusses algorithms for curve evolution. The objective of these methods is the evolution of curves based on forces derived from the characteristics of the curve itself and the content of the image. Cost functions are defined based on these forces, and optimal segmentation results are obtained through the minimization of the defined cost function.

The active contours presented in this section fall into two main categories: snakes and level sets. Snakes active contours are parametric in nature, and the curve evolution is performed explicitly. In the case of level set methods, the curve evolves implicitly through the deformation of a defined level set function. Snakes active contours and level set methods are discussed in greater details in the following subsections.

2.4.1 Snakes

Kass et al. [4] introduced the so-named snakes model, because of the behaviour of the curve evolution: the contour is deformed in order to minimize the contour's energy. In the snakes approach, active contours are parameterized curves, and a contour parameterized by arc length s is defined as

$$C(s) = \left\{(x(s), y(s)) : 0 \leq s \leq L\right\}, \tag{2.7}$$

where L is the length of the contour C. The energy function $E(C)$ that deforms the contour can be defined as

$$E(C) = E_{int} + E_{ext}, \tag{2.8}$$

where E_{int} denotes the internal energy, and E_{ext} denotes the external energy. The internal energy determines the regularity of the contour and can be defined as

$$E_{int} = \int_0^L \alpha|C'(s)|^2 + \beta|C''(s)|^2 \, ds, \tag{2.9}$$

where α controls the tension of the contour, and β controls the rigidity of the contour. The external energy determines the criteria of contour evolution depending on the image $I(x, y)$, which can be defined as

$$E_{ext} = \int_0^L E_{img}(C(s))\, ds, \tag{2.10}$$

where $E_{img}(x, y)$ is a function defined on the image plane. A popular choice of edge attraction function is

$$E_{img}(x, y) = \frac{1}{\lambda |\nabla G_\sigma * I(x, y)|}, \tag{2.11}$$

where G_σ is a Gaussian smoothing filter with a standard deviation σ, λ is a suitability constant, and $*$ represents the convolution.

Early snakes formulations entailed two major problems. First, they should be initialized close to the desired edges because they rely only on local information. This feature makes contour initialization a critical step, which can become a challenging problem, especially in the absence of prior knowledge. Second, the topology of the contours cannot change because they are parametric, which means that boundaries cannot be split or merged.

Cohen [53] proposed a solution for the initialization problem by adding a constant force. The additional force inflates the growth of the contour, hence the method named "balloon snake." In this way, the initial contour can be located farther from the desired boundaries. Rahnamayan et al. [54] proposed automated snakes initialization for ultrasound prostate segmentation, whereby morphological operators are used in order to find a section inside the prostate whose, boundary is used as the initial curve.

A solution for the topology problem was proposed by McInerney and Terzopoulos [55], who utilized Affine Cell Image Decomposition (ACID), which decomposes the image into a set of convex polytopes. The changes in topology are managed through the reparameterization of the curve with each set of iterations.

2.4.2 Level Sets

The level set method was introduced by Osher and Sethian [56] for interface propagation and has been used in numerous applications: computational fluid mechanics [57], computer graphics [58], shape optimization [59], inverse problems [60], and image analysis [27]. The contour is defined implicitly with the use of a Lipschitz continuous function $\phi(x, y)$, called a level set function. The contour is usually defined at the zeroth level of $\phi(x, y)$, such that positive and negative represent different regions. Let C represent the contour; it is defined as

$$C = \left\{(x, y) : \phi(x, y, t) = 0\right\}, \forall (x, y) \in \Omega, \tag{2.12}$$

where Ω denotes the entire image domain, and t is the time through deformation. Figure 2.3 illustrates the representation of curves using a level set function. The evolution of the curve is performed implicitly as the level set function evolves. Since the contour is defined implicitly with a function of higher dimension, changes in the topology are handled automatically. For the evolution of a level set function, the following equation is differentiated with respect to t:

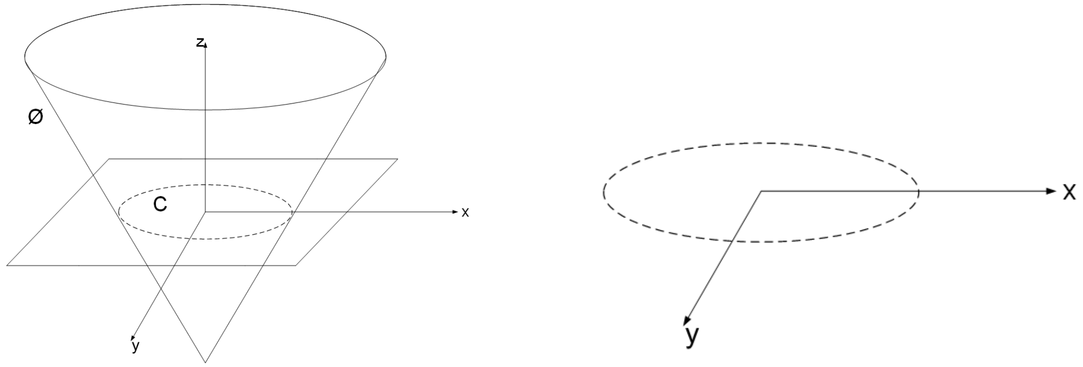
$$\phi(x, y, t) = 0, \tag{2.13}$$

and the result is

$$\phi_t + V \cdot \nabla\phi = 0, \tag{2.14}$$

where V is the velocity field of the contour $C(t)$. The velocity field consists of normal ($\vec{N}$) and tangent ($\vec{T}$) components; Eq. 2.14 thus becomes

$$\phi_t + \left(v_n \cdot \vec{N} + v_t \cdot \vec{T}\right) \cdot \nabla\phi = 0. \tag{2.15}$$



(a) Level set function ϕ with curve C embedded in it

(b) Curve C represented by the level set function ϕ

Figure 2.3: Illustration of a level set function and associated curve representation.

Tangent velocity is known not to affect the deformation of the function; thus $\vec{T} \cdot \nabla\phi = 0$. The unit normal of the curve can also be written as

$$\vec{N} = \frac{\nabla\phi}{|\nabla\phi|}, \tag{2.16}$$

where $|.|$ is the L^2-Norm. Thus, Eq. 2.15 becomes [56]

$$\phi_t + v_n|\nabla\phi| = 0. \tag{2.17}$$

The mean curvature of the level set function, κ, is defined to control the regularity of the curve and can be obtained as the second derivative of $\phi(x, y)$, as follows [56]:

$$\kappa(\phi(x, y)) = \nabla\left(\frac{\nabla\phi}{|\nabla\phi|}\right) = \frac{\phi_{xx}\phi_y^2 - 2\phi_x\phi_y\phi_{xy} + \phi_{yy}\phi_x^2}{(\phi_x^2 + \phi_y^2)^{3/2}}, \tag{2.18}$$

where ϕ_x and ϕ_{xx} are the first and second order partial derivatives of function $\phi(x, y)$ with respect to x, and analogously ϕ_y and ϕ_{yy}, with respect to y. The most common method of defining the initial level set function $\phi_0(x, y)$ is to use the signed distance function of the initial contour.

Two main methodologies for implementing the level set for image segmentation are edge-based [61, 62, 27] and region-based [63, 64, 65]. Edge-based level set methods rely on principles similar to those of edge-based segmentation algorithms. Because the goal is to attract the contours toward the edges of the image, they usually depend on the gradient of the image. On the other hand, for curve evolution, region-based active contour algorithms exploit the characteristics of the regions, such as intensity distribution or texture. In this research, a region-based level set image segmentation is used, which is described next.

Region-Based Level Set

Region-based active contour algorithms exploit regional characteristics, such as intensity distribution or texture, for curve evolution. An early and well-known method is the one proposed by Chan and Vese [63]. The authors employed a level set method to minimize the Mumford-Shah segmentation model [66] for piecewise constant approximation of the image. The cost function is defined by

$$\begin{aligned} E(c_1, c_2, C) = \mu \cdot L(C) + v \cdot A(C_{in}) \\ +\lambda_1 \int_{C_{in}} |I(x,y) - c_1|^2 \, dx \, dy \\ +\lambda_2 \int_{C_{out}} |I(x,y) - c_2|^2 \, dx \, dy, \end{aligned} \tag{2.19}$$

where C denotes the curve; C_{in} is the set of points inside C; C_{out} is the set of points outside C; $L(\cdot)$ and $A(\cdot)$ functions calculate the length and area, respectively; I is the input image; μ, λ_1, and λ_2 are fixed parameters greater than zero; and c_1 and c_2 are averages of the areas inside and outside curve C, respectively. The level set formulation of the function is

$$\begin{aligned} E(c_1, c_2, \phi) = \mu \int_\Omega \delta(\phi(x,y)) |\nabla \phi(x,y)| \, dx \, dy + v \int_\Omega H(\phi(x,y)) \, dx \, dy \\ +\lambda_1 \int_\Omega |I(x,y) - c_1|^2 H(\phi(x,y)) \, dx \, dy \\ +\lambda_2 \int_\Omega |I(x,y) - c_2|^2 (1 - H(\phi(x,y))) \, dx \, dy, \end{aligned} \tag{2.20}$$

where $H(x)$ is the Heaviside function defined by

$$H(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}, \tag{2.21}$$

and $\delta(x)$ is the Dirac measure defined as the first derivative of $H(x)$. Chan and Vese [63] defined a slightly regularized version of $H(x)$ and δ, denoted by $H_\varepsilon(x)$ and $\delta_\varepsilon(x)$. To solve Eq. 2.20, its Euler-Lagrange equation for ϕ is derived to steady state by

$$\frac{\partial \phi}{\partial t} = \delta_\varepsilon(\phi) \left[\mu \, div \left(\frac{\nabla \phi}{|\nabla \phi|} \right) - v + \sum_{j=1}^{2} (-1)^j \cdot \lambda_j (I - c_j)^2 \right] = 0. \tag{2.22}$$

Vese and Chan [67] proposed a piecewise smooth approximation, in which smoothed partial images represent each partition. Similar active contour models and formulations were proposed by Yezzi et al. [64] and Tsai et al. [65].

2.5 Other Segmentation Algorithms

Another common method is “watershed segmentation” [68], which combines region- and edge-based algorithms as well as morphological processing. Because the idea of watershed algorithm came from the field of topography, images are viewed as 3D surfaces. Holes can be imagined as piercing the local minima of the image (usually the gradient of the image is used), with water raised to fill the basins. As water from different basins starts to merge, dams are built. The regions represented by the basins are thus segmented by the dams that have been built. Watershed algorithms usually obtain results characterized by over-segmentation [68]. To prevent this effect, consideration of domain knowledge is included through the use of markers [68]. Appropriate markers can be selected manually or by the inclusion of domain knowledge, such as intensity range, connectivity, size, shape, and location.

Several segmentation algorithms that integrate both edge-based and region-based techniques are proposed in the literature. Xuan et al. [69] suggested the use of edge detection to validate the boundaries of the results of region growing and merging. The algorithm begins with region growing, followed by region merging; Canny edge detection is then used to verify and correct the boundaries of the resulting regions. Yu and Wang [70] first utilized an edge detection method to obtain a Difference-In-Strength (DIS) map, which describes complete edge information in the image. A region growing algorithm is then employed which includes a criterion that determines whether to stop or proceed if a region touches an edge. The next step is for the regions to be merged according to the similarity measure. Moigne and Tilton [71] defined a stopping criterion for region growing based on edge information: the Hausdorff distance is calculated among edges that result from the Canny algorithm and the boundaries of the regions.

In "graph-based segmentation" (e.g., the graph cuts [7]), images are represented as a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. Each pixel of the image is represented as a vertex $v_i \in V$. The edges $(v_i, v_j) \in E$ denote two adjacent vertices v_i and v_j, and each edge has a weight w_{ij}. The weight of the edge is a measure of the dissimilarity of the two vertices connected by that edge. The objective of graph cut algorithms is to partition V into disjoint sets (regions) by removing the edges from non-similar sets. Next, a variant of graph cuts is described, which is used in this research.

2.5.1 Parametric Kernel Graph Cuts

Ben Saleh et al. [72] proposed a multi-region image segmentation technique in which the image data is transformed into a higher dimensional space, and then the graph cuts is applied. The authors used radial basis function kernel to implicitly transform the image data. The mapping of the data into higher dimensions transforms the non-linear problem into a linear one, and the piecewise constant model of the graph cuts can be applied.

For image I, with regions parameters $\{\mu_l\}, l = 1, \ldots, N_{reg}$, where N_{reg} is the number of regions, the proposed functional is defined by

$$F_K(\{\mu_l\}, \lambda) = \sum_{l \in \Psi} \sum_{p \in R_l} J_K(I_p, \mu_l) + \alpha \sum_{\{p,q\} \in \Omega} r(\lambda(p), \lambda(q)), \tag{2.23}$$

where λ is a labeling function, Ψ is a set of region indices, R_l is a region with label l, and Ω is a neighbourhood set containing all pairs of neighbouring pixels. The term $r(\lambda(p), \lambda(q))$ is a smoothness regularization function given by

$$r(\lambda(p), \lambda(q)) = min\left(s^2, |\mu_{\lambda(p)} - \mu_{\lambda(q)}|^2\right), \tag{2.24}$$

where s is a constant. J_K is a non-Euclidean distance measure in the original data space defined by

$$J_K(I_p, \mu) = ||\phi(I_p) - \phi(\mu)||^2 = K(I_p, I_p) + K(\mu, \mu) - 2K(I_p, \mu), \mu \in \{\mu_l\}_{1 \le l \le N_{reg}}, \tag{2.25}$$

where

$$K(x, y) = \phi(x)^T \cdot \phi(y), \tag{2.26}$$

and $\phi(\cdot)$ is a non-linear mapping function to a higher dimensional space.

Ben Saleh et al. [72] comprehensively tested this approach on synthetic and natural images and found that it outperformed many other well-known segmentation algorithms.

This chapter has provided a general overview of image segmentation approaches. The next chapter describes common multiresolution methods and approaches for their use in image segmentation.

Chapter 3

Multiresolution Image Segmentation

Objects in images are extremely diverse with respect to a number of aspects: size, shape, regularity, textures, and edges. It is therefore intuitive to expect that analyzing an object at a variety of scales provides more information than evaluating it at only a single scale. In addition, for any specific scale, some objects are more easily analyzed than others. For these reasons, the understanding and analysis of images at several scales has attracted the attention of numerous researchers, resulting in a substantial number of methods for multiscale and multiresolution analysis. These methods have been designed for different purposes, such as object recognition [73], compression [74, 75], image retrieval [76], and image segmentation [77, 78]. This chapter provides a discussion of the most common multiresolution and multiscale methods and reviews previous work related to multiresolution image segmentation.

Because of the variation in terminology among disciplines (e.g., spatial resolution, spectral resolution, temporal resolution, and radiometric resolution [79]), clearly defining the meaning of resolution and scale in this research is important. In this work, the term "different resolutions" denotes spatial resolution (different subsampled images): an example is the pyramid representation. On the other hand, the term "different scales" refers to images that have different degrees of smoothness but exhibit the same spatial size: scale-space representation illustrates this terminology.

The next sections offer an overview of the most common multiresolution and multiscale methods, followed by a review of multiresolution image segmentation techniques.

3.1 Multiresolution Methods

The literature includes proposals for several multiresolution and multiscale analysis approaches that have emerged from different fields and are based on various concepts. These methods were all proposed for the purpose of analyzing an image at different resolutions/scales. The most common techniques can be grouped in three main categories: pyramid representation, scale-space representation, and wavelets analysis. An overview of these methods are presented in the following subsections.

3.1.1 Pyramid Representation

An image pyramid [74] is a hierarchical representation of an image consisting of several resolutions of different spatial sampling of the image. The basic operations for constructing pyramids are filtering followed by subsampling. Filtering is necessary in order to avoid an aliasing problem after the subsampling. Figure 3.1 illustrates an example of the pyramid representation. Decreased image size in the lower resolutions leads to reduced computational time in processing. In addition, the memory required for coarser resolutions is much less than that needed for the original one. The hierarchical structure of the image pyramids facilitates more efficient hardware implementation. Several types of pyramids have been proposed in the literature, but the best-known ones are Gaussian and Laplacian [74].

Gaussian Pyramids

A given image I can be decomposed into r levels of resolutions using a Gaussian low-pass filter followed by subsampling. Each lower resolution level is obtained by using a 5×5 low-pass filter on the previous level. For levels $0 < l < r - 1$ and an image I of size

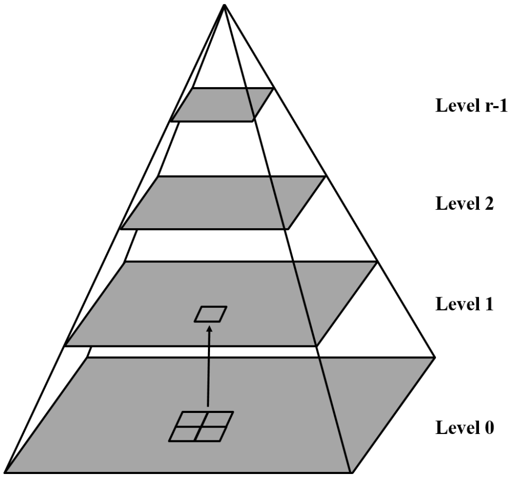


Figure 3.1: Pyramid representation.

$N \times M$, the pyramid is defined as [74]

$$\begin{aligned} &G_0(i,j) = I(i,j), \text{ for level } l = 0, \\ &G_l(i,j) = \textstyle\sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) G_{l-1}(2i+m, 2j+n), \text{ otherwise,} \end{aligned} \tag{3.1}$$

where $i = 1, \ldots N$, $j = 1, \ldots M$, and $w(\cdot)$ is a Gaussian-like kernel. The result of the image decomposition is stored in G_k, where $k = 0, \ldots, r-1$, and $r-1$ is the lowest resolution; thus G_0 is the original image, and G_{r-1} is the coarsest resolution.

An example of the Gaussian pyramid decomposition is provided in Figure 3.2(a).

Laplacian Pyramids

A Laplacian pyramid is a band-pass pyramid created by taking the difference between two adjacent levels of a Gaussian pyramid. A Laplacian pyramid L of r different resolutions can be created from

$$L_l = G_l - Expand(G_{l+1}), \tag{3.2}$$

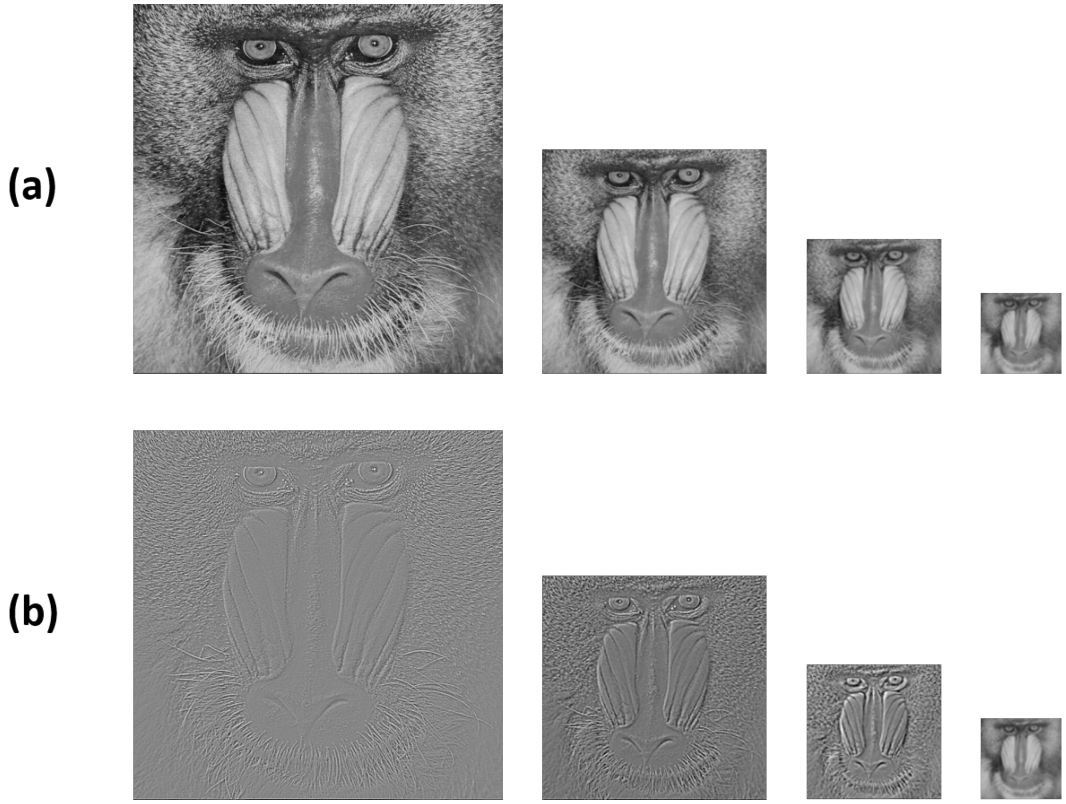


Figure 3.2: (a) Gaussian pyramid; (b) Laplacian pyramid.

where $l = 0, 1, \cdots, r-1$ is the resolution level, and G is a Gaussian pyramid. The expansion of G_{l+1} to G_l is defined as

$$Expand\,(G_{l+1}) = 4 \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) G_{l+1} \left(\frac{i-m}{2}, \frac{j-n}{2} \right). \tag{3.3}$$

An example of the Laplacian pyramid is shown in Figure 3.2(b).

The pyramid described here is an oversampled representation, in which there are more pixels in the representation than in the image. The number of pixels is $\frac{4}{3}$ times that of the original image [22]. Many extensions of the pyramid representation are available, each with specific applications. Examples include irregular [80], stochastic [81], and adaptive [82] pyramids.

3.1.2 Scale-Space Representation

Scale-space, introduced by Witkins [83], is a representation in which an image is decomposed into a stack of different smooth versions. The Gaussian kernel is the best smoothing filter used for linear scale-space, in which new extrema cannot be created with increasing scale [84]. The continuous scale parameter controls the degree of smoothness. The linear scale-space representation of an image I is defined as

$$S(I, \varepsilon) = I * G(\varepsilon), \tag{3.4}$$

where $G(\cdot)$ is the Gaussian kernel defined as

$$G(x, y; \varepsilon) = \frac{1}{2\pi\varepsilon} e^{-(x^2+y^2)/2\varepsilon}, \tag{3.5}$$

and $\varepsilon = \sigma^2$ is the scale parameter. Equivalently, scale-space can be defined as the solution of the diffusion equation

$$S_t = div(c \bigtriangledown I), \tag{3.6}$$

at time t, with the initial condition

$$S_0 = I, \tag{3.7}$$

where the diffusion coefficient $c(x, y, t)$ is a constant. Figure 3.3 illustrates the scale-space representation of an image.

A shortcoming of successive smoothing in the linear scale-space representation is the spatial distortion of important edges. To solve this problem, non-linear scale-space methods have been proposed, whereby the regions are smoothed while the edges are preserved.

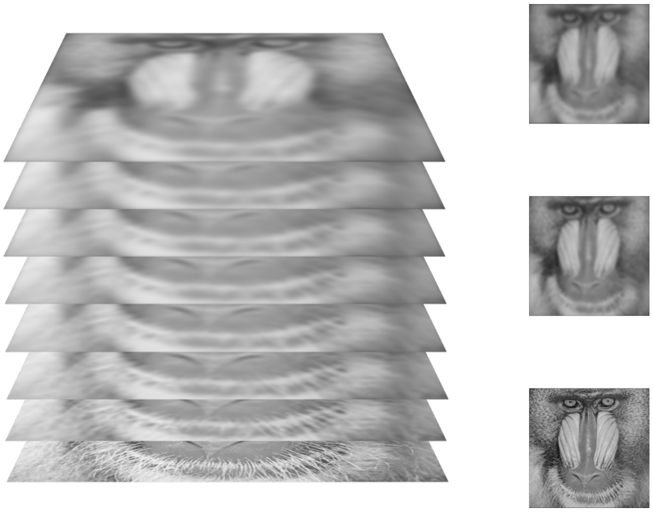


Figure 3.3: Scale-space representation (s=0 at the bottom).

An example of this type of method is the anisotropic diffusion technique suggested by Perona and Malik [85], which is based on the modification of Eq. 3.6 to have an edge function $g(\cdot)$, as follows:

$$S_t = div\left(g(| \bigtriangledown I|) \cdot \bigtriangledown I\right), \tag{3.8}$$

where $\bigtriangledown$ is the gradient operator. The edge function $g(| \bigtriangledown I|)$ can be defined as [85]

$$g(| \bigtriangledown I|) = e^{-(|\bigtriangledown I|/k)^2}, \tag{3.9}$$

or

$$g(| \bigtriangledown I|) = \frac{1}{1 + \left(\frac{|\bigtriangledown I|}{k}\right)^2}, \tag{3.10}$$

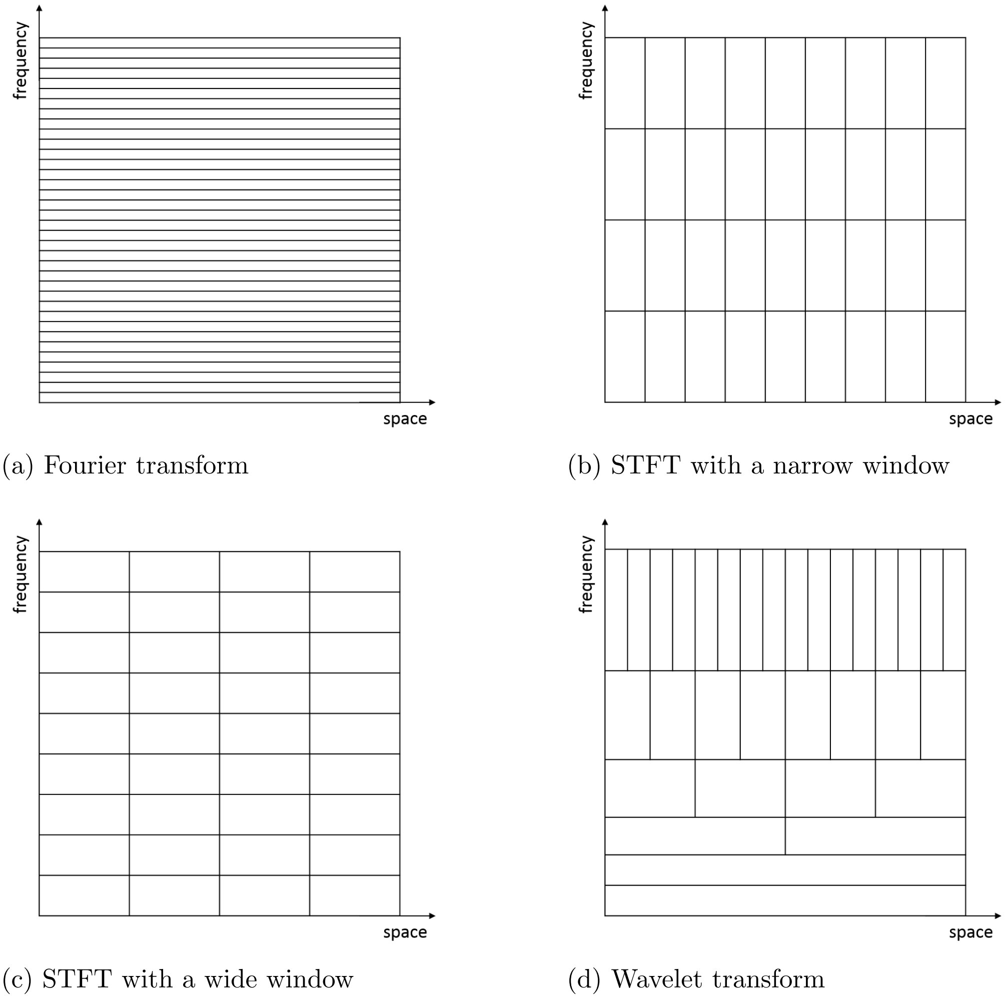
where k is a constant set by the user or by means of a noise estimator.

3.1.3 Wavelets Analysis

Fourier analysis is a well-known tool for function analysis and approximation, in which the approximation is performed as a sum of complex sinusoids. The Fourier transform is used to decompose a signal into its frequency components. A disadvantage of the Fourier transform is its inability to capture space (time) information, as illustrated in Figure 3.4(a). One method of solving this problem is to divide the signal into small windows through multiplication with a sliding window function and then to take the Fourier transform of each window. Such a technique is called Short-Time Fourier Transform (STFT). However, a trade-off between frequency and time resolution is still a factor, as shown in Figures 3.4(b) and 3.4(c). If a narrow window is used, satisfactory time resolution will be achieved but with poor frequency resolution, as shown in Figure 3.4(b). On the other hand, if a wide window is chosen, satisfactory frequency resolution with poor time resolution will result, as shown in Figure 3.4(c). Wavelet transform [86] provides a good solution. A base function, called "mother wavelet," is defined. The signal is decomposed into a set of basis functions, called "wavelets," as different translations and scales of the previously defined mother wavelet. Frequency analysis is performed using wavelets that are an expansion of the mother wavelet (low frequency), while temporal (time) analysis is performed using a contracted version of the wavelets (high frequency). Wavelet analysis is illustrated in Figure 3.4(d). Because of the scaling property of wavelets, it is considered an effective tool for multiresolution analysis. The Wavelet transform is used in many fields: astronomy [87], acoustics [88], optics [89] and image processing. Applications of wavelet transform in image processing include noise removal [90], edge detection [91], and image compression [75]. In contrast to the pyramid representation discussed previously, the wavelet transform is a complete representation (i.e., critically sampled) [86]. The total number of pixels in the wavelet representation is equal to the number of pixels of the original image, in which there is no redundancy. This feature is important for some applications, such as compression [92].

The wavelet function is defined as

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right), \tag{3.11}$$



(a) Fourier transform

(b) STFT with a narrow window

(c) STFT with a wide window

(d) Wavelet transform

Figure 3.4: Frequency and space (time) analysis (a) with a Fourier transform, (b) STFT with a narrow window, (c) STFT with a wide window, and (d) wavelet transform.

where the parameter s indicates scale, and τ represents translation. Wavelets $\psi_{s,\tau}$ are thus generated from the mother wavelet ψ through scaling and translation.

The use of wavelet transform for 2D images requires scaling and wavelet functions to be defined as two variable functions. In 2D, there is one scaling function and three wavelet functions, defined by [86]

$$\phi(x,y) = \phi(x)\phi(y), \tag{3.12}$$

$$\psi^H(x,y) = \psi(x)\phi(y), \tag{3.13}$$

$$\psi^V(x,y) = \phi(x)\psi(y), \tag{3.14}$$

$$\psi^D(x,y) = \psi(x)\psi(y), \tag{3.15}$$

where $\phi(x,y)$ is the 2D scaling function, which is the low frequency component of the previous level. $\phi(x,y)$ is known as the approximation coefficients. The three 2D wavelet functions capture the functional variation information (e.g., edges in the case of images) in three directions: $\psi^H(x,y)$ for horizontal, $\psi^V(x,y)$ for vertical, and $\psi^D(x,y)$ for diagonal. Let $I(x,y)$ be an image of size $M \times N$; the 2D Discrete Wavelet Transform (DWT) is defined as follows:

$$W_\phi(j_0,m,n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I(x,y)\phi_{j_0,m,n}(x,y), \tag{3.16}$$

$$W_\psi^i(j,m,n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I(x,y)\psi_{j,m,n}^i(x,y), i = \{H,V,D\}, \tag{3.17}$$

and the inverse transform is defined by

$$\begin{aligned} I(x,y) &= \frac{1}{\sqrt{MN}} \sum_m \sum_n W_\phi(j_0,m,n)\phi_{j_0,m,n}(x,y) \\ &+ \frac{1}{\sqrt{MN}} \sum_{i=H,V,D} \sum_{j=j_0}^{\infty} \sum_m \sum_n W_\psi^i(j,m,n)\psi_{j,m,n}^i(x,y). \end{aligned} \tag{3.18}$$

The DWT can be used for 2D images if it is applied on the columns followed by the rows, as shown in Figure 3.5. Figure 3.6 illustrates the result of the two-level wavelet decomposition of an image. Clockwise from the upper left of Figure 3.6, the results are the approximation image and the horizontal, vertical, and diagonal details. Approximation image is further decomposed at the second level.

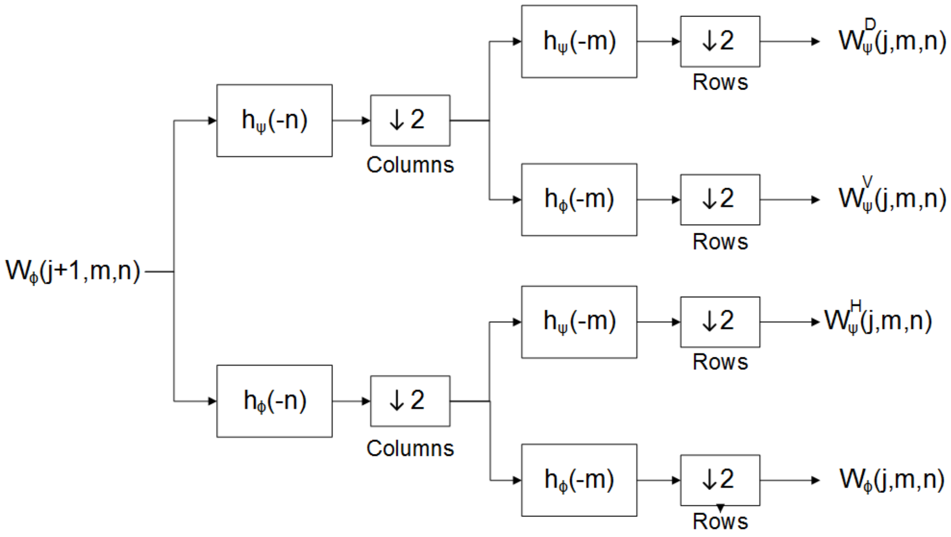


Figure 3.5: The process of 2D wavelet decomposition for one level; $h_\phi(\cdot)$ is scaling function coefficients and $h_\psi(\cdot)$ is wavelet function coefficients.

3.2 Multiresolution Image Segmentation Techniques

Multiresolution analysis has been used extensively for developing numerous image segmentation algorithms, the majority of which utilize information from some or all resolutions. Several methodologies have utilized multiresolution analysis for image segmentation. Categorizing all the methods introduced in the literature is difficult because of the extreme diversity of the concepts. This section provides a review of some of the types of methodologies that have been employed.

Some techniques use a merging strategy, whereby segments from some or all resolutions are merged to construct the final segment. Gaetano et al. [93, 94] proposed a hierarchical

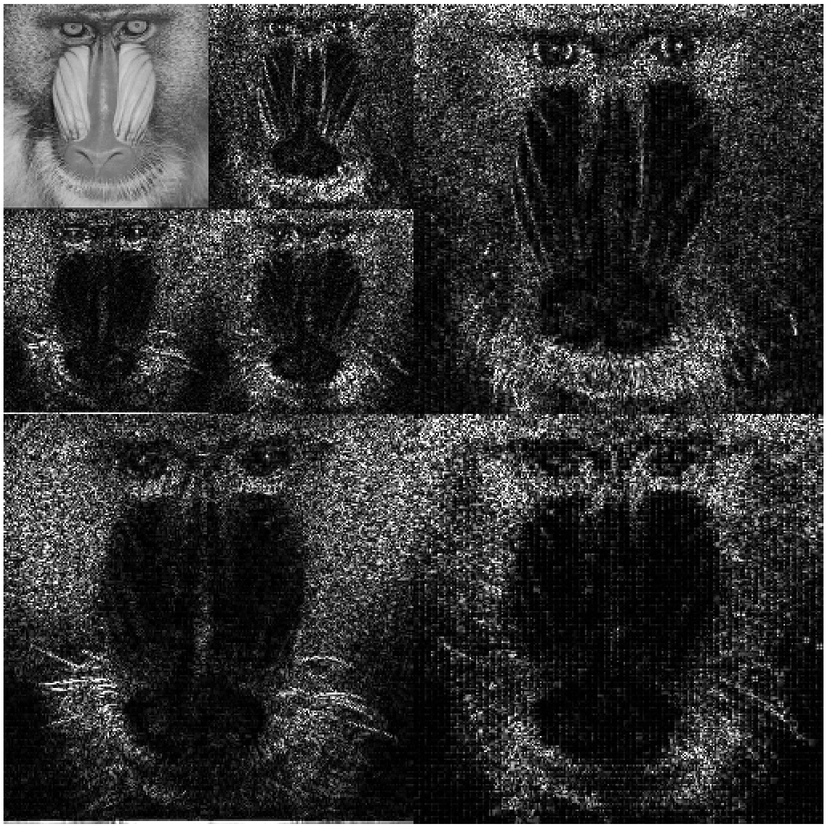


Figure 3.6: The result of the decomposition of a 2D image for two levels.

segmentation scheme for satellite images. The image is first segmented at the original resolution using a tree-structured Markov Random Field (MRF) model. Based on spectral, spatial, and textural features, regions are then clustered at a lower resolution, and these clusters are progressively merged to form the final segments. Rezaee et al. [77] introduced a segmentation technique using pyramids and fuzzy c-means clustering. A root labeling method is used for the initial segmentation of each resolution level of the pyramid. Features from the resulting segments are then used in order to merge regions at the original resolution through fuzzy c-means clustering.

Other methods employ the dependencies among pixels at different resolutions. An example is the method proposed by Saeed et al. [78], in which they employed the pixel dependence between resolutions to extend the Gaussian Mixture Model (GMM) by including the correlation of the pixels at adjacent resolution levels. The image is segmented using the Maximum a Posteriori (MAP) method, in which parameters are learned through the Expectation Maximization (EM) algorithm.

Other methods include a set of algorithms that utilize the features generated at different resolutions for image segmentation [95, 96, 97] as well as numerous additional techniques that cannot be grouped with any of the above categories.

Coarse-to-Fine Strategy

Although the approaches described above can enhance accuracy, they are computationally expensive. The research presented in this thesis is concerned with a multiresolution technique that increases image segmentation efficiency: the coarse-to-fine segmentation strategy. With this technique, the segmentation process starts at a coarse resolution, and then is tuned at finer ones. Many segmentation algorithms can be used with this method because it simplifies the input image rather than modifying the segmentation algorithm. Numerous studies published in the literature report the use of this method to increase the efficiency of several segmentation algorithms. However, as previously mentioned, all of them select the initial resolution for segmentation arbitrarily.

Snakes active contour segmentation algorithm (Section 2.4.1) has been used with multiresolution analysis in several research studies. Leroy et al. [11] enhanced the snakes algorithm by using the pyramid representation. The propagation of the curve starts from the coarsest resolution and iteratively continues toward finer resolutions when the curve evolution has converged at previous ones. This method is faster than the original snakes method because most of the calculations are performed at coarse resolutions that require less computation. Yan et al. [98] proposed an algorithm based on snakes for prostate segmentation in transrectal ultrasound (TRUS) images. They used prior shape models and propagated the curve using the Laplacian pyramid in a manner similar to that employed by Leroy et al. to increase the capturing range of the curve and enhance the efficiency of the initialization. Akgul and Kambhamettu [99] utilized the scale-space representation to construct coarser versions of the original image. The authors used dynamic programming and gradient descent to propagate the parametric contour from coarser representations through finer ones. Dehmeshki et al. [100] improved the snakes algorithm by using wavelets. The contour is initialized at the coarsest resolution of the image, and the initial curve of a finer resolution is the converged one of the previous coarse resolution. This method extends the capturing range and prevents the contour from being trapped into weak edges. The algorithm used four levels of resolutions and was employed for the segmentation of lung and colour CT images. A similar algorithm was proposed by Yoon et al. [101] to improve the Gaussian of Gradient Force (GGF) snakes. The authors reported an increase in speed accompanied by a high degree of accuracy.

The level set is a well-known active contour segmentation algorithm (Section 2.4.2), but it is computationally expensive and thus slow to converge. Numerous researches have been directed at solving this problem through the multiresolution coarse-to-fine strategy. The Gaussian pyramid was utilized by Tsang [16] to enhance the edge-based level set active contours. The curve is initialized at the coarsest resolution, and the propagation proceeds with finer resolutions. A similar methodology has been employed for segmenting ultrasound echocardiographic images [102, 103]. The Curvelet, which is a multiscale and multidirectional geometric wavelet transform, is used to enhance the geodesic active contours [104], and the edge map is obtained using curvelet thresholding. Initialization occurs at the coarsest resolution, and for each subsequent finer resolution, the level set function ϕ is defined as the converged level set function of the previous resolution. Al-Qunaieer et al. proposed a method for accelerating region-based level set image segmentation [12]. The authors used wavelets to decompose the image into three resolutions, with the curve evolution beginning from the coarsest resolution. The results confirmed that using multiresolution reduces the effect of noise for large objects and accelerates the convergence rate of the segmentation algorithm.

Graph-based segmentation algorithms (Section 2.5) have been combined with multiresolution analysis to reduce computation time. Roullier et al. [105] applied the multiresolution approach with graph-based segmentation for mitosis extraction in breast cancer histological whole slide images. The segmentation begins at a coarse resolution, and at each finer resolution, the resulting segmentation is refined through semi-supervised clustering. Lombaert et al. [13] adopted a similar approach, but rather than using clustering for fine-tuning, they applied graph cuts on a narrow banded graph obtained from the resulting minimum cut at the coarser resolution. They showed that their method dramatically increases speed and reduces memory usage without affecting the accuracy of the graph cuts segmentation.

The multiresolution approach has been incorporated to enhance other segmentation methods. Bouman and Liu [106] proposed a multiresolution version of the MRF segmentation, in which at each resolution, the segmentation is implemented as the maximization of the posteriori probability. The segmentation is performed from coarse to fine resolutions, and the authors reported substantial improvement with respect to computation time. Multiresolution Active Shape Model (ASM) was used by Wang et al. [19] for lung segmentation in chest X-ray radiographs. At lower resolutions, the model is fitted to the lung very quickly, after which it is honed at finer resolutions. The use of multiresolution analysis considerably improved the ASM performance. Munoz et al. [107] applied the coarse-to-fine approach for segmentation based on Active Region algorithm. For each resolution level, the initialization is based on the results of the previous (coarser) level. They concluded that multiresolution reduces the noise effect and increases computational efficiency.

These studies all offer either a vague or no explanation of the method of selecting the initial resolution for the commencement of the segmentation. Some adopted a trial-and-error technique to select a specific resolution level for all images of a single or different dataset(s), which is impractical because, as shown in Section 6.2.1, with respect to time and accuracy, performance varies over different resolutions, even for images from the same dataset.

This chapter has reviewed multiresolution techniques and described different approaches for their use with image segmentation. The next chapter discusses previous work related to scale selection methods and highlights the distinction between such methods and the work presented in this thesis. Then, segmentation at different resolutions is analysed.

Chapter 4

Learning Best Resolution for Image Segmentation

Image segmentation at different resolutions yields different accuracies and at varying speeds as presented in Sections 4.2 and 6.2.1 and discussed in [20, 21]. At each resolution, the image is viewed from a different perspective because the global view tends to become local as resolutions become coarser [108]. Such a change in view could contribute to either an increase or a decrease in accuracy. On the other hand, the lower number of grid points (pixels) at low resolutions leads to much faster performance.

The next section discusses previous work conducted with respect to scale selection and highlights the distinguishing features of this work. The following section describes the experimental analysis of the segmentation performance at different resolutions for images of different characteristics.

4.1 Scale Selection for Scale-Space Representation

The problem of selecting the best scale for image processing has attracted the attention of several researchers. A number of proposed methods described in the literature were based on local information obtained from the derivatives of the image. Lindeberg [84] is a pioneer in the scale selection field. In his work, the scale of interest is selected from the scales that have maxima of normalized derivative over scales [109]. This methodology was used for the detection of a variety of features: junctions, edges, ridges, and blobs [110, 109, 111, 112]. Jeong and Kim [113] defined an energy function so that during its minimization it detects the most useful scale and obtains the edge map. It is also defined to take into account the constraints of optimal edge detection. A method for detecting "minimum reliable scale," in which the edges could be reliably detected, was developed by Elder and Zucker [114], who calculated the gradient of each scale and then determined the lowest scale as a function of the amplitude and sensor noise. A drawback of this method is the requirement of prior knowledge about sensor's noise and operator norms.

Concepts borrowed from the information theory have also been used for scale selection. A scale-space measure of information has been proposed by Jagersand [115]. By calculating Kullback's contrast between consecutive scales, information distribution among scales can be obtained. The author demonstrated that this measure could be used for scale selection. Renyi's generalized entropy [116] was employed by Sporring and Weickert [117] for scale selection and size estimation. The authors based their approach on the observation of the monotonic behaviour and smoothness of generalized entropies with respect to the information order and the scale parameter. Hadjidemetriou et al. [118] used Tsallis generalized entropies [119] of the histogram for scale selection. They used histogram entropies because they are non-monotonic with respect to scale, can be used for the selection of multiple scales, and are robust with respect to noise. This approach was utilized to increase the discriminability among images and to improve the performance of an optical flow algorithm. Kadir and Brady [120] proposed an algorithm for detecting salient regions based on scale selection and a local descriptor. In their method, scales with maximum entropy are selected and then weighted using the sum of absolute difference of the grey-level histogram. In an alterative methodology for scale selection, Mirzaalian and Hamarneh [121] considered the correlation of a pixel with its neighbours when selecting its scale. They designed the problem as a Markov Random Field (MRF) multi-label optimization and used this approach to detect the scales for vascular structures in medical images.

Statistical methods have also been utilized for the selection of the most favourable scales. Based on probabilistic models of the sensor and an edge detection operator, Marimont and Rubner [122] suggested a statistical framework. For each pixel in the image at different scales, they calculated the edge and confidence probabilities. A minimum reliable scale is then selected by thresholding the confidence probability and choosing the resulting minimal scale. Pedersen et al. [123] proposed a scale selection scheme based on maximum likelihood estimation. The authors utilized the Brownian image model [124] because of its relation to natural images. After a spatially varying transformation is applied, the selected scale at location x is obtained based on the maximum likelihood of the probability distribution of a Gaussian filter response. The Bayesian estimation theory was used by Gomez et al. [125] for scale selection. At each scale, a decomposition likelihood is associated with both the smoothed image and the residual, and the Minimum Description Length (MDL) principle is used for scale selection. The authors showed that this method can be used for simple edge detection and texture segmentation.

In all of these approaches, the objective of scale selection was related primarily to feature detection, with some works also investigating primitive segmentation tasks (e.g., edge detection [112, 113, 114]). Several studies have incorporated scale selection into the image segmentation process. With some modifications, Bayram et al. [126] applied the "minimum reliable scale" method proposed by [114] in order to find the edges in medical images. Lindeberg's scale selection method [110] was utilized by Piovano and Papadopoulo [127] to guide snakes active contour inside homogeneous regions. Li et al. [128] proposed a scale selection method for supervised image segmentation. For each scale in a training image, features are extracted per pixel and assigned to their corresponding labels. Therefore, for n scales, there are n learned classifiers. For a test image, pixels in each scale are classified based on their corresponding classifier, and the best scale is the one at which the posterior probability calculated from the classification is the highest. Although these methods incorporated scale selection in the segmentation process, several distinctions exists between them and the work presented in this thesis:

- The previous methods were intended for selecting the best Gaussian scale for scale-space approach. Since the scale-space representation does not involve the subsampling of the image, the grid size (number of pixels) is the same for all scales. For the research for this thesis, the pyramid scheme is used, in which each resolution is filtered and subsampled from the previous resolution. The reduced size is the major contributing factor in decreasing the computational complexity and hence accelerating the processing time for image segmentation.

- The scale selection approaches analyze several scales in order to determine the best scale. Moreover, the majority of the previous approaches select a different scale for each pixel. These techniques inherently lead to excessive computation time compared with working only at one resolution (the original one) in order to estimate a single resolution for all pixels, as in the proposed approach, which is much faster.

- With the exception of [128], the scale selection methods are limited in their applicability. For example, the majority of the approaches mentioned search for local changes, an approach that does not work well with images that consist of many homogeneous regions. However, including learning, such as in the approach proposed in this thesis and in [128] allows to select the resolution/scales most suitable for the specific problem at hand.

- The mentioned scale selection approaches proposed specific segmentation algorithms that incorporate scale selection. In contrast, the work introduced in this thesis is a framework for resolution selection for image segmentation approaches, not a segmentation method. The framework is general and can be used with a wide variety of image segmentation algorithms, such as level set, graph cuts, region growing, and watershed. These segmentation algorithms utilize various aspects of images other than edges: region homogeneity, textures, colours, and others.

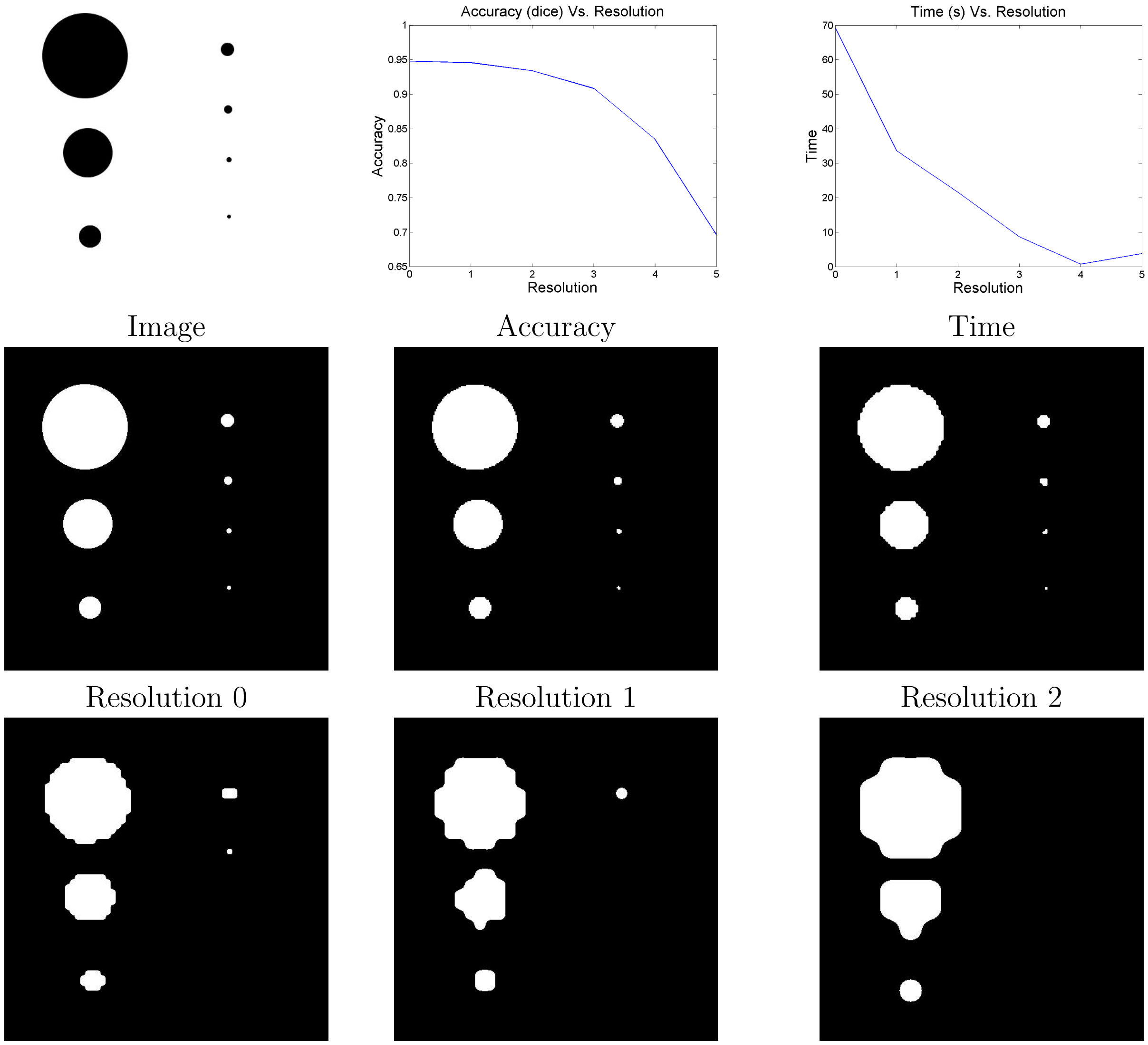
- In the proposed approach, the user is given the weighted option of choosing between accuracy and speed, a feature that broadens the range of applications for which it can be employed.

4.2 Analysis of Segmentation at Different Resolutions

To demonstrate the effect of segmentation at different resolutions on a variety of objects, three simple experiments were conducted. In all of the experiments, the Chan and Vese region-based level set [63] was used to segment the input images, and the output segment of each resolution was simply upsampled to the original image resolution. Each input image was 512×512. Accuracy was measured using dice coefficient as given in Eq. 6.2 (page 72), and time was recorded in seconds. The first image contained several objects of different sizes. Figure 4.1 shows the results of applying the level set for six different resolutions. It is clear from the accuracy figure that the level of accuracy decreases as the resolution is reduced, while, except for a slight decrease from the fourth to fifth resolutions, the speed substantially increases. The sample output of resolutions 3 to 5 reveals the reason for the decreased accuracy. Because small objects disappear at lower resolutions, the segmentation algorithm is unable to retain them during the segmentation process. As the resolution further decreases, additional objects are lost. Another factor contributing to the lower level of accuracy at coarse resolutions is the loss of quality due to the upsampling of the segmentation results.

The same image was tested again but with the introduction of Gaussian noise: the results are illustrated in Figure 4.2. This example shows one advantage of segmenting at a lower resolution for noisy images: as can be clearly seen in the accuracy figure, the level of accuracy at all other resolutions is much better than that at resolution 0 because the noise is suppressed at lower resolutions, as can be observed in the segmented output images. The accuracy begins to fall after resolution 2 because smaller objects start to disappear, as mentioned previously. Except for a slight increase in processing time at resolution 1, the time is greatly decreased in subsequent resolutions. A possible reason for the faster performance of resolution 0 compared to resolution 1 is that the segmentation at resolution 0 may be trapped at local minima, resulting in early convergence. The extremely poor result for resolution 0 supports this possibility.

The third image was intended to test the segmentation of irregular shapes, as shown in Figure 4.3. Accuracy is decreased with respect to resolution because the smaller grid



Image

Accuracy

Time

Resolution 0

Resolution 1

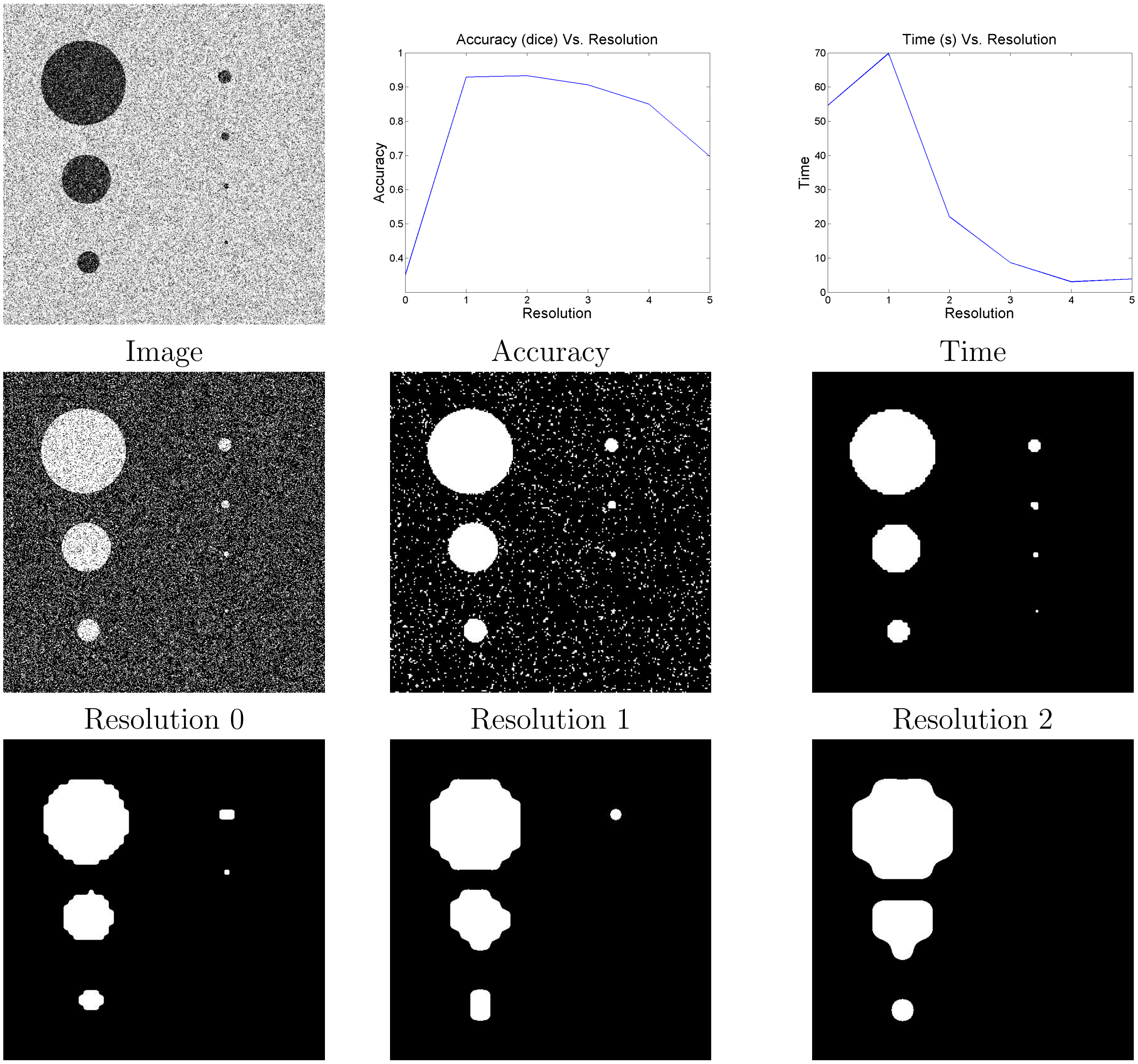
Resolution 2

Resolution 3

Resolution 4

Resolution 5

Figure 4.1: Segmentation of different-sized objects at different resolutions, with the output upsampled to the original resolution.



Image

Accuracy

Time

Resolution 0

Resolution 1

Resolution 2

Resolution 3

Resolution 4

Resolution 5

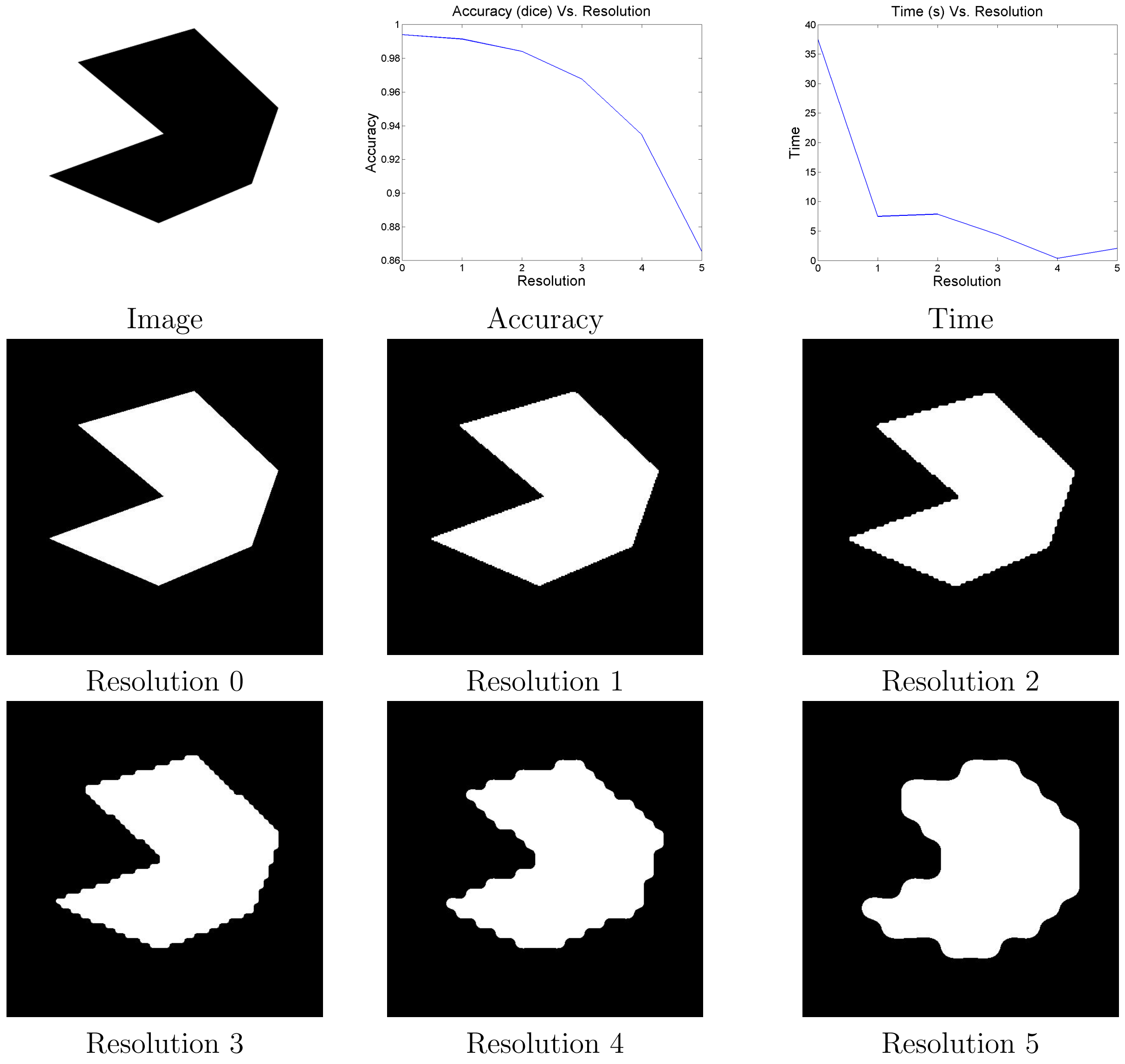
Figure 4.2: Segmentation of different-sized objects with added Gaussian noise at different resolutions, with the output upsampled to the original resolution.

sizes did not permit the correct representation of straight lines, sharp edges, or corners. As with the previous two images, the process is generally much faster at lower resolutions.

It can be noted that for all three images, the segmentation time at resolution 5 is slightly slower than for resolution 4, which could be due to the initial placement of the curve. Based on these results, the following factors were the basis of, and motivation for, the use of the coarse-to-fine multiresolution image segmentation strategy:

- The segmentation results could be better at lower resolutions (Figure 4.2).
- Postprocessing is needed for fine-tuning the segmentation results in order to counter the quality loss due to the upsampling of the output segments. The use of a simple method is described in the next chapter.
- Segmentation at lower resolutions is usually much faster.

This chapter has included a discussion of previous scale selection methods and the results of an analysis of image segmentation at different resolutions. The next chapter presents the proposed framework for the automated resolution selection.



Image

Accuracy

Time

Resolution 0

Resolution 1

Resolution 2

Resolution 3

Resolution 4

Resolution 5

Figure 4.3: Segmentation of an irregular shape at different resolutions, with the output upsampled to the original resolution.

Chapter 5

Proposed Framework

This chapter proposes a machine learning framework for resolution selection for image segmentation. Resolution selecting for image segmentation is a difficult task, in part because of the vague definition of the best resolution, which can also differ according to the segmentation method used, as a result of the utilization of different aspects of the image (e.g., homogeneity, texture, edges). This chapter presents a measure for determining the best resolution for an input image when segmented with a specific segmentation algorithm.

The overall approach is described in the next section, followed by a discussion of the algorithms used in the framework. The trade-off measure used to define the best resolution is discussed in Section 5.3. Then, the process of image segmentation and the determination of the best resolutions are explained in Section 5.4. In Section 5.5, the features extracted from the images for use in the learning are described. The machine learning approach utilized is presented in Section 5.6, along with an explanation of learning from imbalanced data. The final section presents the performance measures for assessing the effectiveness of the learning algorithm.

5.1 Overall Approach

The aim of the learning approach, for segmenting an image with a specific segmentation algorithm, is to map the features extracted from an input image to the best resolution for image segmentation. Figure 5.1 provides an overall view of the framework, illustrating its two main components: training and testing. In the training phase, the system associates features extracted from training images with the best resolution for segmenting them. Each training image is segmented at r different resolutions. The accuracy of, and time required for, segmentation at each resolution are recorded, based on which the trade-off measure, ω, is calculated (Eq. 5.1). The best resolution for each image is the one that obtains the maximum value of ω. These best resolutions obtained from the training images are the labels (i.e., classes), which are then used for training the classifier. The inputs are comprised of features extracted from the training images. Two sets of features are used: Local Binary Patterns (LBP) [129] and a statistical set of features. After training, the classifier is used for estimating the best resolution based on the features extracted from the testing image. Because, as will be shown, the nature of the learning data is imbalanced, a modified version of AdaBoost, namely Rank Minority Oversampling in Boosting (RAMOBoost), is used. RAMOBoost was specifically designed by Chen et al. [130] for learning from imbalanced data. The following sections detail the individual components of the proposed framework.

5.2 Preliminary Settings

For the purpose of this research, a multiresolution representation that includes different subsampling sizes is needed. This requirement excludes the scale-space representation (Section 3.1.2), which has the same spatial resolution for all levels. Although the approximation image of the wavelet representation (Section 3.1.3) can be used, there is information that is not used in this work (e.g., the high-frequency information defining the horizontal, vertical, and diagonal details images). Therefore, the pyramid representation (Section 3.1.1) is chosen for this research. Because of the learning nature of the framework, other multiresolution methods constructing finite levels of resolutions should also work with it.

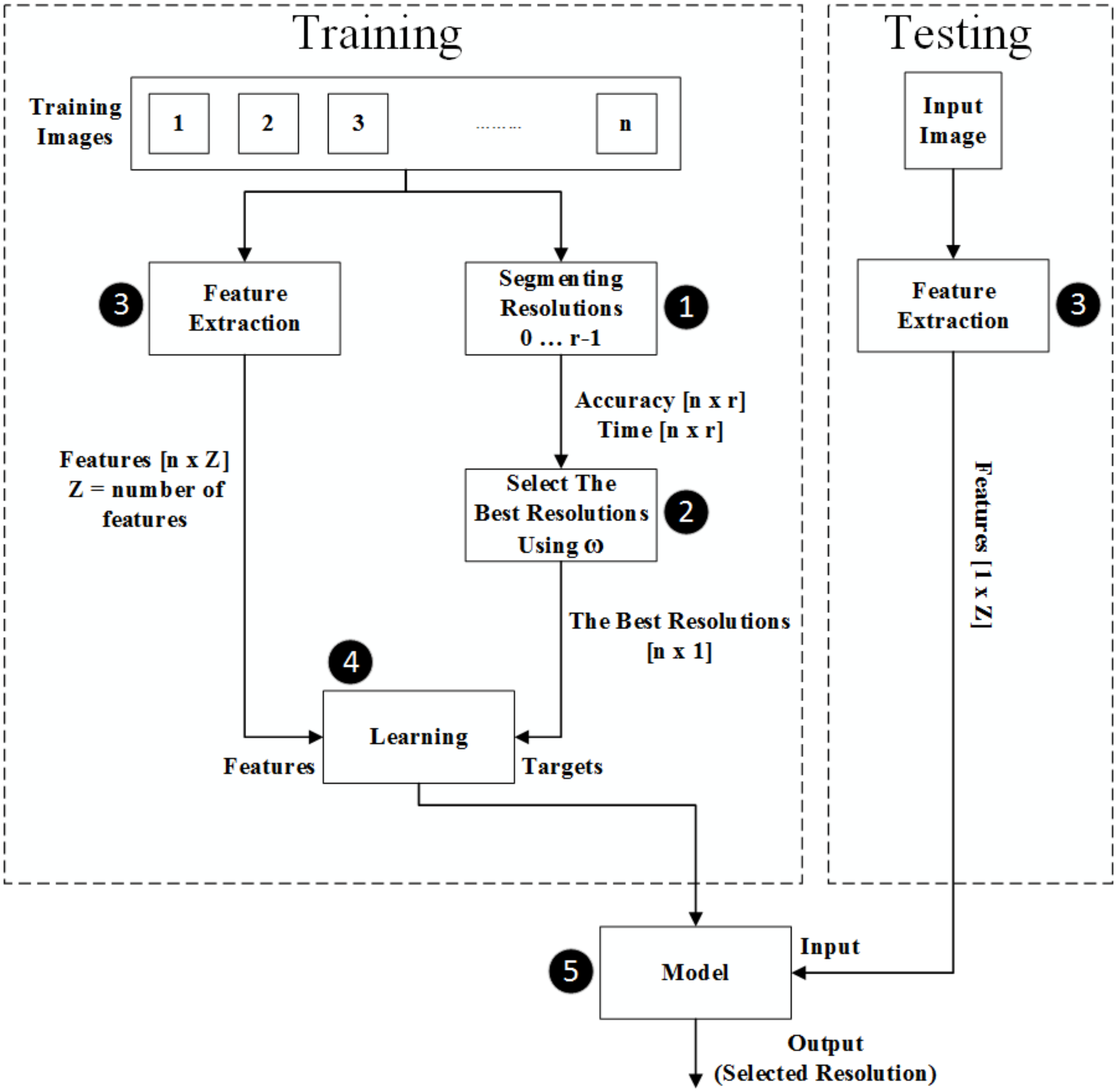


Figure 5.1: Overall approach to the selection of the best resolution for image segmentation through machine learning.

The proposed framework is a generic one, in which numerous feature sets and learning methods can be embedded. Further, the framework can be trained to improve many segmentation algorithms. It cannot be claimed that a set of features or a certain learning technique can always achieve the optimal classification accuracy. The performance of the features varies according to the nature of the images used, and the learning approach can achieve different results based on several factors, such as data distribution and size. However, the proposed framework in this thesis may already cover a wide range of image categories and applications, as demonstrate by using a very diverse set of test images (see Chapter 6).

For this research, many sets of features have been tested, and two of them were selected (Section 5.5): LBP [129] and statistical features. Similarly, numerous learning approaches have been investigated, and RAMOBoost (Section A.4) achieved the best results.

5.3 Defining the Best Resolution

The definition of the best resolution for image segmentation can be different from system to system. For example, in systems such as robot navigation, speed is very important, so it could be favoured more than an increase in accuracy. On the other hand, in critical systems, such as medical applications, accuracy is much more important, because a mistake could be life-threatening. Thus, a method is needed to facilitate decisions about the selection of the best resolution. A measure is thereby proposed that allows the user or system to select a trade-off between speed and accuracy. Weighted geometric mean has been used as an aggregation function of several variables [131, 132] and for trade-off measurement [133]. Given normalized accuracy A_i and time T_i in the range [0 1], the measure is defined as

$$\omega_i = A_i^{\alpha} \times (1 - T_i)^{1-\alpha}, \tag{5.1}$$

where α can be set by an algorithm or chosen by the user in order to determine the desired trade-off and $i = 0, \cdots, r-1$ is the resolution level.. The values of α are in the range [0 1]. Here, A_i and T_i are obtained from resolution i. This trade-off measure is only used during the training phase to define the best (target) resolutions as labels for the classifier's training, where the processing time, T, of all resolutions are available. Therefore, T_i can be normalized by dividing by the maximum time of all resolutions. Higher values of α will favour accuracy over speed, while lower values will favour speed. This measure provides the flexibility of selecting resolutions according to the specifications of the problem at hand, and forms the criterion for the best resolution definition incorporated into the proposed resolution selection framework.

The value of α is set to meet the requirements of the application (e.g., to meet minimum accuracy or time). This is specified by the user based on the domain knowledge. Because at the training phase all accuracies and times are available, and due to the limited number of resolutions, the value of α can be quickly found using a simple searching algorithm or by trial-and-error. The selected α is fixed, and the best resolutions are defined according to it.

The trade-off measure can be defined in several other ways. Appendix B provides a discussion of two other possible choices.

5.4 Computing the Best Resolution for Learning

The proposed framework is based on a supervised learning approach for resolution selection for image segmentation. Supervised methods require labeled instances for proper learning, whereas the labels are the learning targets. For the best resolutions to be learned, they must be provided as labels (i.e., classes) associated with the training dataset (Boxes 1 and 2 in Figure 5.1). As described in Section 5.3, best resolutions can be defined using the trade-off measure, ω. As controlled by the parameter α, the system or user can choose an appropriate trade-off between accuracy and speed based on the specific needs. Higher α values favour resolutions that provide a great degree of accuracy, while lower α values favour speed. The best resolution is the one that produces the maximum ω. To this end, the image should be segmented at each resolution, and the resulting values for accuracy and time should be recorded.

The pyramid representation entails r resolutions, $0, 1, \ldots, r-1$, where 0 denotes the original resolution, and $r-1$ is the coarsest one. The criterion for choosing the lowest level of resolution is the disappearance of image information that is useful for segmentation, which can be found empirically. In the pyramid representation, each level of resolution is a smoothed and subsampled version of the previous level. Coarse resolutions can be used to capture the main features that correspond to large objects and strong edges. Because regions become more homogeneous, capturing similar regions thus becomes easier, and noise and weak edges are eliminated. The segmentation speed at coarse resolutions is much faster than at finer ones, because the grid size (number of pixels) is lower, resulting in a lower computational cost. Fine resolutions capture the details of objects, and are therefore used to fine-tune the regions that are roughly segmented at coarse resolutions.

After an image is segmented at each resolution and the values for accuracy and time at each resolution are obtained, ω can be calculated for each level of resolution. For r values of ω for r resolutions, the best resolution is the one having the maximum value of ω. This process is illustrated in Figure 5.2 and described in detail in Algorithm 2.

This method of obtaining the best resolution for image segmentation can be used with many image segmentation algorithms. Different segmentation methods can yield different results at different resolutions, so that the best resolutions are not the same for different segmentation algorithms. Section 6.2.2 provides a detailed explanation.

After segmentation at lower resolutions and the upsampling of the results to the original resolution, fast fine-tuning is needed in order to compensate the loss in quality (see Figures 4.1, 4.2, and 4.3). The literature reports numerous fine-tuning methods, as discussed in Section 3.2, but the majority are slow, and each method is specific to a particular segmentation algorithm. This work applies a unified approach for fine-tuning, which is appropriate for any segmentation algorithm: region-growing segmentation started from the border of the results. Region-growing was selected for the following reasons:

- It can be used as a boundary-enhancing method following many other segmentation techniques.

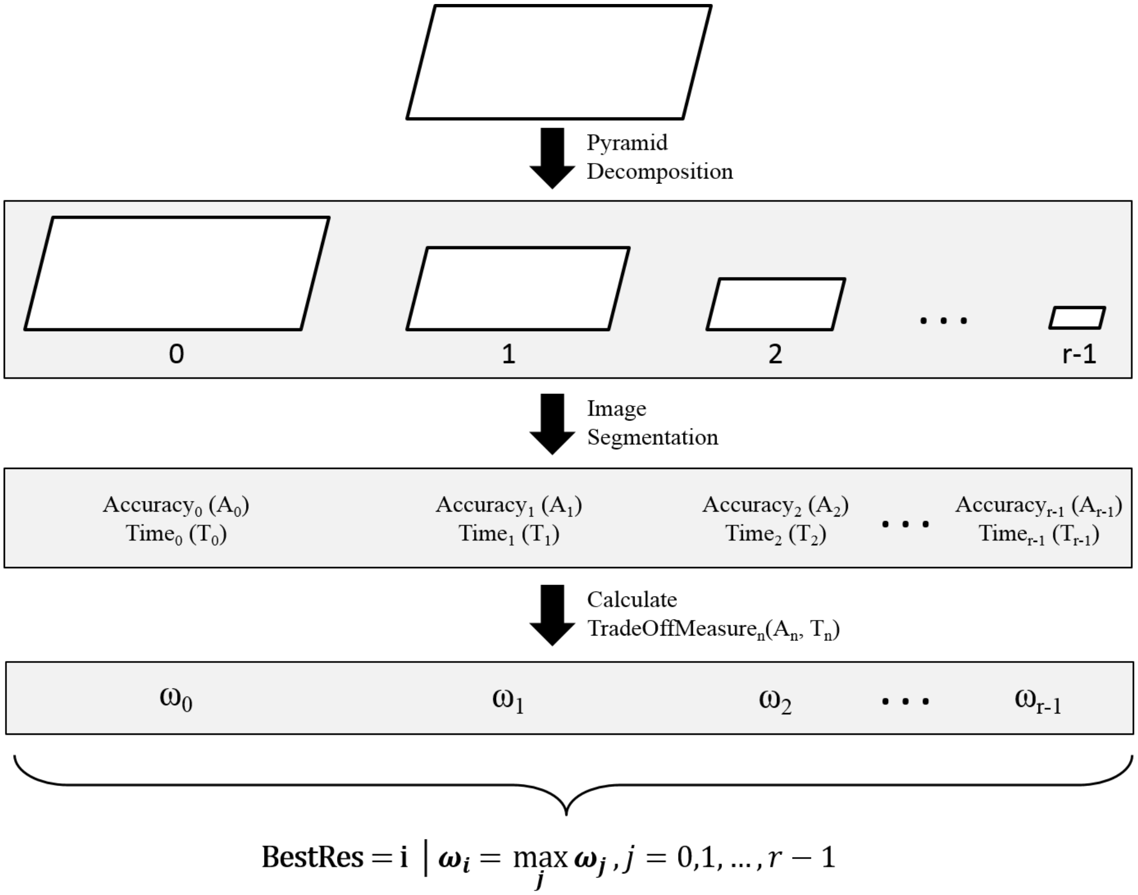

Pyramid
Decomposition
0
1
2
. . .
r-1
Image
Segmentation
$\text{Accuracy}_0\ (A_0)$ $\text{Time}_0\ (T_0)$
$\text{Accuracy}_1\ (A_1)$ $\text{Time}_1\ (T_1)$
$\text{Accuracy}_2\ (A_2)$ $\text{Time}_2\ (T_2)$
. . .
$\text{Accuracy}_{r\text{-}1}\ (A_{r\text{-}1})$ $\text{Time}_{r\text{-}1}\ (T_{r\text{-}1})$
Calculate
$\text{TradeOffMeasure}_n(A_n, T_n)$
ω_0
ω_1
ω_2
. . .
$\omega_{r\text{-}1}$
$\textbf{BestRes} = \textbf{i} \mid \boldsymbol{\omega_i} = \max_{\boldsymbol{j}} \boldsymbol{\omega_j}, j = 0,1,\dots,r-1$

Figure 5.2: Illustration of the process of calculating the best resolution of an image (r = the number of resolutions).

Algorithm 2 Calculating The Best Resolution

```
Inputs
$I$: input image
$G$: gold standard image (prepared manually by an expert)
$r$: number of resolutions
$\alpha$: trade-off between accuracy and speed for calculating the trade-off measure $\omega$

$P = pyramid(I, r)$ [pyramid decomposition of $I$ into $r$ resolutions]
for i=0,1,...,r-1 do
    start time calculation
    seg = segment($P_i$) [segment the image at resolution $i$]
    if $i > 0$ then
        upSeg = upSample(seg) [upsample the segmentation result to the original resolution size]
        finalSeg = fineTune(upSeg) [fine-tune the upsampled segmentation result]
    else
        finalSeg = seg;
    end if
    $T_i$ = stop time calculation
    $A_i$ = calcAccuracy(finalSeg, G) [calculate the segmentation accuracy compared to the gold standard image]
end for
for i=0,1,...,r-1 do
    $\omega_i = A_i^\alpha \times \left(1 - \frac{T_i}{max(T)}\right)^{1-\alpha}$
end for
BestRes = $i \mid \omega_i = \max_j \omega_j,\ j = 0, 1, \cdots, r-1$
```

- Because of its simplicity, it is very fast — an attribute that translates into only minimal increase in time during the fine-tuning step.

- It is consistent in its increasing in time, which enables the learning method to learn the best resolution. In contrast, some methods result in an unpredictable increase in time, which makes the learning impossible, as illustrated in the forthcoming example.

Figure 5.3 shows an image segmented using the ChanVese level set [63]. The accuracy (dice coefficient, Eq. 6.2, page 72) and time in seconds are presented for four cases: no fine-tuning, fine-tuning with ChanVese only at the original resolution, fine-tuning with ChanVese iteratively in each resolution, and fine-tuning with region growing at the original resolution. Although fine-tuning with ChanVese results in better accuracy than region growing, it results in a completely unpredictable increase in time. On the other hand, region growing results in a consistent and slight increase in time.

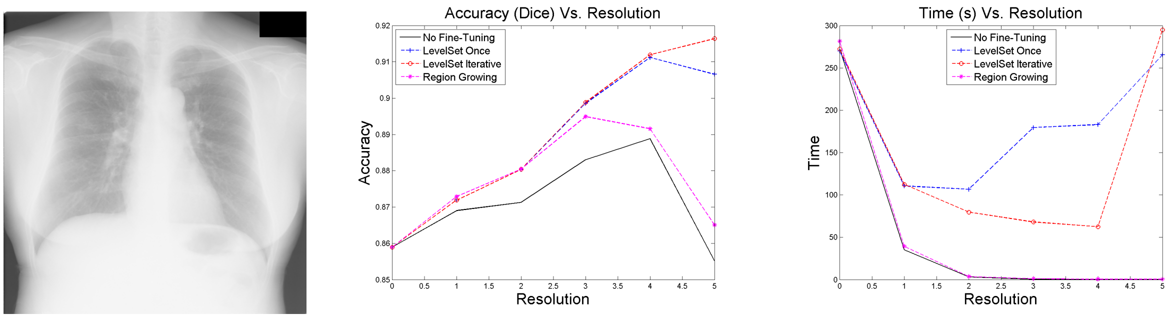


Figure 5.3: An image segmented using ChanVese level set fine-tuned with level set once at the original resolution, iteratively at each resolution, region growing at the original resolution, and without fine-tuning. From left to write: input image, accuracies, and times.

5.5 Feature Extraction

The extraction of relevant features is very important for achieving reasonable classification accuracy (Box 3 in Figure 5.1). For resolution selection, two sets of features were used: Local Binary Patterns (LBP) and statistical features. Two criteria were set for the choice of feature extraction methods in this research: good correlation with the classes (e.g., resolutions) and the time constraint. Determining the best features was performed empirically. The performance of the features in terms of classification accuracy was assessed with the F_1-*measure* described in Section 5.7, and the speed was measured in seconds. Relevant features must be extracted extremely fast so as not to delay the decision about the best resolution. From a set of good features, several of them have therefore been omitted because they would make the process slow. Examples of such features are Gabor [134] and Granulometric features [135].

5.5.1 Local Binary Patterns (LBP)

LBP is a method for extracting texture features proposed by Ojala et al. [129]. It was originally a 3×3 operator, where the intensities inside each block are thresholded by the value of the central pixel. The value of the central pixel is then obtained from the summation of the thresholded values multiplied by powers of two. In this way, $2^8 = 256$ different values can be achieved. This operation is illustrated in Figure 5.4.

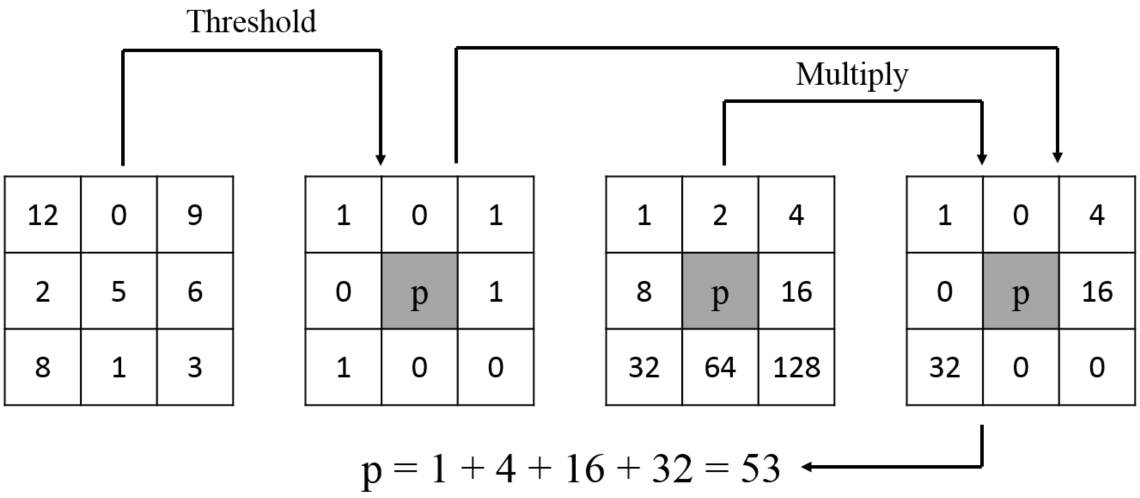


Figure 5.4: The process of calculating LBP; the leftmost block is thresholded by its central pixel value, then multiplied by powers of two. The value of p is the sum of the result.

Constructing features from the histogram of an LBP-labeled image has been proven to be efficient for face recognition [136, 137]. In order to retain spatial information, the image is divided into m regions $R_0, R_1, \cdots, R_{m-1}$, in which the histogram features are calculated as follows [137]:

$$H_{i,j} = \sum_{x,y} \Gamma\{I_{LBP}(x,y) = i\}\Gamma\{(x,y) \in R_j\}, i = 0, \cdots, n-1, j = 0, \cdots, m-1, \tag{5.2}$$

where I_{LBP} is an LBP labeled image, n is the number of labels produced by the LBP operator, and $\Gamma\{A\}$ is defined as

$$\Gamma\{A\} = \begin{cases} 1 & \text{if } A \text{ is True,} \\ 0 & \text{if } A \text{ is False.} \end{cases} \tag{5.3}$$

Examples of LBP output and histogram features are shown in Figure 5.5.

LBP is computationally simple and thus very fast, making it appropriate for the purposes of this research. LBP is also robust to monotonic grey-scale changes. Several LBP variants have been proposed, such as larger operator size [138], rotation invariance [138, 139], and multiscale LBP [140].

5.5.2 Statistical Features

For this research, the second set of features consists of six measures taken from the Region of Interest (ROI), and comprised of four first-order statistical features (mean, standard deviation, skewness, kurtosis), entropy, and a measure of spatial uniformity. Given an image or part of an image of size $M \times N$, the probability of the occurrence of intensity level $\gamma \in \{0, 1, ..., G-1\}$ is calculated by

$$p(\gamma) = \frac{S_\gamma}{M \times N}, \tag{5.4}$$

where S_γ is the number of pixels with intensity γ. The mean m is then calculated by

$$m = \sum_{\gamma=0}^{G-1} \gamma p(\gamma), \tag{5.5}$$

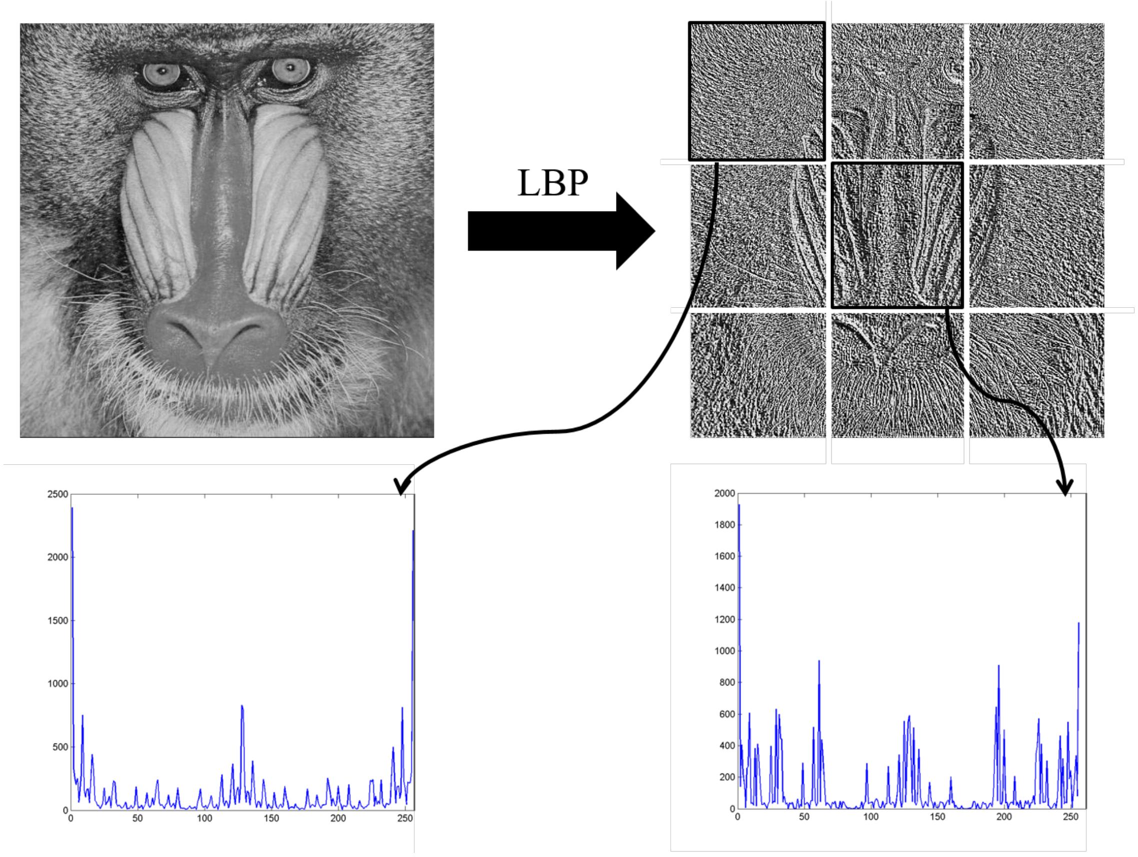


Figure 5.5: Example of an image filtered with an LBP operator (top) and the resulting histogram features (bottom).

the Standard Deviation (STD) is calculated by

$$STD = \sqrt{\sum_{\gamma=0}^{G-1} (\gamma - m)^2 p(\gamma)}, \tag{5.6}$$

the skewness is calculated by

$$SK = \sum_{\gamma=0}^{G-1} (\gamma - m)^3 p(\gamma), \tag{5.7}$$

and the kurtosis is calculated by

$$K = \sum_{\gamma=0}^{G-1} (\gamma - m)^4 p(\gamma). \tag{5.8}$$

These features represent the first four moments of the random variable γ. These simple texture features describe different aspects of the histogram. Standard deviation measures the width of the histogram, which is a measure of intensity contrast; skewness measures the asymmetry of the histogram around the mean; and kurtosis measures the flatness (or sharpness) of the histogram [22, 141].

The fifth measure is the entropy, which measures the randomness (or uniformity) of the intensity distribution [22, 141]. Entropy is calculated by

$$E = -\sum_{\gamma=0}^{G-1} p(\gamma) log_2 p(\gamma). \tag{5.9}$$

These features provide no spatial information because they are taken from the histogram. As described in the previous subsection, spatial information can be retained by dividing the image into several regions and calculating the features from them. The last feature measures spatial uniformity and is basically the standard deviation of the coefficients of the Sobel operator response defined as [142]

$$SU = \sqrt{\frac{1}{N} \sum_i \sum_j (Sobel(I(i,j)))^2 - (\frac{1}{N} \sum_i \sum_j (Sobel(I(i,j))))^2}, \tag{5.10}$$

where I is the input image, and N is the total number of pixels.

These six measures are concatenated and used as features for each ROI.

5.6 Learning the Best Resolution

Objects that have different characteristics can be segmented correctly at different resolutions, as can be observed from the experiments described in Sections 4.2 and 6.2.1 and as well as demonstrated in [20, 21]. Coarse resolutions are sufficient for segmenting large objects with clear boundaries. This is not the case with small objects, fine details, and sharp corners, which must be segmented at finer resolutions. Other factors affect the choice of the best resolution, such as noise, which can be reduced or eliminated as the resolution is decreased, allowing more accurate and faster segmentation. Unfortunately, natural images are complex. The same image can have a variety of noise levels and contain objects with different characteristics. Resolution selection for image segmentation is thus difficult. Simple rules, such as noise level and object size, are insufficient, and object size cannot actually be accurately estimated without segmentation. In this work, a machine learning approach is suggested (Boxes 4 and 5 in Figure 5.1), so that best resolutions can be learned based on features extracted from the training images.

Although the problem is complex, many learning algorithms can be used to learn and construct non-linear models for solving it. Complex problems can be properly learned using learning methods, given the right conditions. The features from the images are already available (Section 5.5), as are the training instance labels (Section 5.4), so a supervised machine learning approach can be used to learn the best resolution. Learning the best resolutions is explained in Algorithm 3, and estimating the best resolution for an input image is presented in Algorithm 4.

The learning objective here is to estimate the best resolution for segmenting an input image from the extracted features. As shown in Section 6.2.1, the learning data in this research have an imbalanced class distribution. This problem is well recognized in the machine learning community, and it can seriously affect learning performance. Learning from imbalanced data is discussed next.

Algorithm 3 Learning the Best Resolution

```
Inputs
Training images set consists of $n$ images ($I_j,\ j = 1, 2, \cdots, n$), and their corresponding gold standard images $G_j$
$r$: number of resolutions
$\alpha$: trade-off between accuracy and speed for calculating the trade-off measure $\omega$

Initialize: inputs = [ ], targets = [ ]
for i=1,2,. . . ,n do
  BestRes = Algorithm2($I_i, G_i, r, \alpha$) [calculate the best resolution]
  $F$ = featuresExtraction($I_i$) [extract features from image $I_i$]
  inputs = append(inputs, $F$) [append extracted features to inputs]
  targets = append(targets, BestRes) [append calculated best resolution to the targets]
end for
model = trainClassifier(inputs, targets) [train the classifier of choice with the inputs and targets]
```

Algorithm 4 Estimating the Best Resolution

```
Inputs
$I$: input image
$LM$: the learned model

$F$ = featuresExtraction($I$) [extract features from image $I$]
estimatedRes = estimate($LM$, $F$) [estimate the best resolution given the model and the inputs]
```

5.6.1 Learning from Imbalanced Data

A dataset is considered imbalanced if it has unequal class distribution, i.e., not all classes are represented equally. Imbalanced data have serious implications with respect to learning, especially if the imbalance is severe, because the learning models become biased toward the majority classes. Many real-life problems exhibit imbalanced data: bioinformatics [143], text classification [144], speech recognition [145], intrusion detection [146], and oil spill detection [147]. An example of extreme class imbalance can be found in a typical cancer detection data set, in which about 2% of the data represent the cancer class [148]. In this case, non-cancer cases will be well modelled, while cancer cases will be undermodelled. It should be noted that the implications of false negative classification have a greater impact than a false positive one. Over the last decade, the machine learning community has witnessed increased interest in this problem, as indicated by the dramatically growing number of published studies [149]. Several solutions have been proposed for increasing the efficiency of learning from imbalanced data. These approaches can be grouped into three general categories: data-level, algorithmic-level, and boosting strategies [150, 149].

Data-Level Strategies

Data-level methods represent an attempt to re-balance the dataset by either decreasing the number of majority class instances (undersampling), increasing the number of minority class instances (oversampling), or both. Sampling methods include several variants: random oversampling [151] randomly replicates instances of minority classes; similarly, random undersampling randomly removes instances from majority classes. Both of these techniques have disadvantages. Random undersampling could remove important instances from a majority class, and random oversampling can lead to possible overfitting in the learning.

Directed, also called focused or informed, oversampling and undersampling address the problems inherent in random sampling [151]. In oversampling, only instances on the boundaries between the classes are replicated, and in undersampling, only majority class instances that are far from the boundaries are removed. Although the directed approach can be superior to random sampling [152], it cannot completely eliminate the shortcomings mentioned.

To reduce the overfitting caused by the replication of the instances in oversampling methods, the Synthetic Minority Over-sampling TEchnique (SMOTE) was proposed by Chawla et al. [148]. In this approach, new instances are generated using randomly selected instances from the k-nearest neighbours multiplied by a random number $\in [0\ 1]$. The new instances are thus similar to the existing ones but not identical. He et al. [153] introduced Adaptive Synthetic sampling (ADASYN), in which the number of synthetic data instances is adaptively determined according to their distribution. ADASYN is described further in Appendix A.2.

Algorithmic-Level Strategies

In algorithmic-level methods, aspects or procedures of the learning algorithm are modified to address the class imbalance problem. Cost-sensitive learning [154] solves the imbalance problem by adjusting the cost of misclassification. This is performed by constructing a cost matrix that represents the loss resulting from the classification of one class as another. The learning is then performed using the defined cost. In this way, a greater penalty can be assigned to the misclassification of a minority class instance and a lesser penalty for a majority class.

Active learning methods represent another technique for learning from imbalanced data [155]. In this approach, the most informative instances are selected from the training set in order to build the model. Other methods based on kernel modification have been proposed for learning that involves class-imbalanced datasets. For example, in several approaches, the SVM boundary separating different classes is adjusted by means of boundary-alignment methods [156, 157, 158].

Boosting Methods

Learning using a combination of classifiers can be an effective approach for enhancing classification performance. Several ensemble methods have been proposed in the literature, such as bagging [159], boosting [160], and stacking [161]. Boosting trains several base classifiers consecutively while adaptively adjusting the weights of the training instances, so that each classifier concentrates on the examples that were difficult for the previous classifier to learn. Several studies reported combining sampling and boosting for learning from imbalanced data [162, 163]. RAMOBoost was proposed by Chen et al. [130] for learning from imbalanced data, which is described in more detail in Appendix A.4.

5.6.2 The Learning Algorithm

In this work, RAMOBoost [130] was used for learning and estimating the best resolution. RAMOBoost is a variant of AdaBoost that is specifically designed for learning from imbalanced data through adaptive generation of synthetic samples of minority class examples in each iteration of the AdaBoost method. RAMOBoost has been chosen because of its good performance, which stems from its two combined components: boosting and sampling as follows:

- Boosting: as described in the previous subsection, boosting trains several base classifiers consecutively while adaptively adjusting the weights of the training instances. In this manner, the decision boundary is adaptively shifted during each boosting iteration, so that the focus is on instances that are difficult to learn. This property provides ability to deal with outliers [164]. The generalization abilities of boosting methods have been proven by Schapire and Freund [165].
- Sampling: In RAMOBoost, the sampling is based on adaptive adjustment to the sampling weights of minority class samples according to their distribution. Greater emphasis is thus placed on rare examples that are inherently difficult to learn.

The RAMOBoost technique is presented in Algorithm 7 in Appendix A. RAMOBoost algorithm is a direct extension of the AdaBoost.M2 algorithm (Algorithm 6 in Appendix A), making it inherently applicable for multiclass problems.

Decision trees are selected as the base classifier for RAMOBoost in this research. It is the most popular choice as the base classifier for AdaBoost [166]. An experimental comparison of the performance of decision trees against Naive Bayes and Bayes Net as base classifiers was presented in [167]. The authors reported that AdaBoost with decision trees as base classifier achieved the highest classification rate with lowest computational time.

The model resulting from training with RAMOBoost consists of t decision trees classifiers, where t is the number of iterations. Classification based on this model is performed by finding the class that maximizes a weighted average of the outputs of all base classifiers (see Algorithm 7 in Appendix A).

5.7 Classifiers' Performance Measures

Measuring classifiers' performance is an important issue in machine learning [168]. A classifier is typically evaluated using a confusion matrix as illustrated in Figure 5.6 by an example for multiclass problems. Elements of the confusion matrix count the number of estimated classes with respect to actual classes. The diagonal elements represent correctly classified instances. The most widely used measure is accuracy, which is calculated from the confusion matrix as follows:

$$Accuracy = \frac{\sum_{i=1}^{n} a_{ii}}{\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}}. \tag{5.11}$$

		Predicted			
		$\mathbf{C_1}$	$\mathbf{C_2}$	$\ldots$	$\mathbf{C_n}$
Actual	$\mathbf{C_1}$	a_{11}	a_{12}	$\ldots$	a_{1n}
	$\mathbf{C_2}$	a_{21}	a_{22}	$\ldots$	a_{2n}
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$\mathbf{C_n}$	a_{n1}	a_{n2}	$\ldots$	a_{nn}

Figure 5.6: Confusion matrix: n = number of classes.

However, the level of accuracy can be misleading when used for imbalanced class problems [149]. For example, given a data with 2% of minority class instances and 98% of majority class instances, an accuracy of 98% can be achieved by blindly classifying all the data instances as the majority class. The literature includes descriptions of several measures for evaluating the performance of the classification of imbalanced class problems [149, 150, 169]. For class C_i, two measures, Precision (P_i) and Recall (R_i), can be calculated from the confusion matrix as follows:

$$P_i = \frac{a_{ii}}{\sum_{j=1}^{n} a_{ji}}, \tag{5.12}$$

and

$$R_i = \frac{a_{ii}}{\sum_{i=1}^{n} a_{ij}}. \tag{5.13}$$

F_1-*measure* is the harmonic mean of precision and recall and is calculated by

$$F_1\text{-measure} = \frac{2P_iR_i}{P_i + R_i}. \tag{5.14}$$

G-*mean* is the geometric mean of precision and recall. It has been extended for multiclass problem evaluation by Sun et al. [169] and has been defined as

$$G\text{-mean} = \left(\prod_{i=1}^{n} R_i\right)^{1/n}, \tag{5.15}$$

where n is the number of classes. However, if any class has 0 recall, the G-*mean* will also be 0, which is misleading as an overall performance indication.

The Area Under Curve (AUC), which is the Receiver Operating Characteristic (ROC) curve, is a well-known method for evaluating a classifier's performance. It is useful for evaluating imbalanced class problems [149, 150]. AUC was originally proposed for binary classification problems, but Hand and Till [170] generalized it to multiclass problems as well. AUC measures the overall performance of a classifier. However, it cannot represent the performance of different parts of the ROC curve [171]. Curves in ROC space might intersect with each other; therefore, classifiers with high AUC value may have worse performance at some regions of ROC space than a classifier with lower value of AUC.

In this research, the F_1-*measure* was used for evaluating the performance of the estimation of the best resolutions for image segmentation. Unlike accuracy (Eq. 5.11), F_1-*measure*, which is a weighted harmonic mean of precision (Eq. 5.12) and recall (Eq. 5.13), is more useful for problems with imbalanced class distribution. By using F_1-*measure*, the performance of classification per class (i.e., resolution) can be assessed. This enables the investigation of the performance of classification of individual classes, in contrast to G-*mean* (Eq. 5.15) and AUC, which measure the overall performance.

This chapter has introduced a new framework for the learning and selection of the best resolution for image segmentation. Experiments were conducted in order to verify the performance of the framework and are presented in the next chapter, along with the results and related discussions.

Chapter 6

Results and Discussion

This chapter describes experiments that were conducted in order to verify the performance of the trade-off measure and the automated resolution selection framework. The settings used in the experiments, including datasets and parameters, are explained in Section 6.1, followed by a discussion of the performance of the trade-off measure in Section 6.2 with an evaluation of it using different segmentation algorithms, and the accuracy and speed of the maximum accuracy resolutions. Section 6.3 provides an evaluation of the RAMOBoost resolution estimation and a comparison of RAMOBoost with other learning approaches. An examination of the impact of misclassification on accuracy and time is presented in Section 6.4, followed by an analysis of overhead time in Section 6.5. The results of the experimentation are discussed in Section 6.6.

6.1 Experimentation Setup

This section explains the datasets and settings used to conduct the experiments described in this chapter, including the specific datasets used, the parameter settings for the segmentation and learning methods, and the implementation environment.

6.1.1 Datasets

Four image datasets were used for the experimental verification of the proposed resolution selection framework. The aim in these experiments is to assess the performance of the proposed framework with different types of images. Therefore, each dataset contains images whose characteristics differ from those of the images in the other datasets. The images in some datasets are of very similar objects, while others contain objects of varied shapes and sizes. All of the datasets include gold standard images that have been segmented manually and that are used for calculating the dice coefficient, as described in Eq. 6.2. The four datasets are described in the following subsections.

Breast Ultrasound Dataset

This dataset consists of 52 breast ultrasound images whose sizes range from 230×390 pixels to 580×760 pixels. The level of variability among the shapes in the images is high. The speckle noise and low local contrast make the images difficult to segment. Segmentation algorithms for this kind of images usually require special preprocessing and/or postprocessing. Because the objective of this research was not to design an image segmentation method for a particular dataset, a semi-automated approach was applied, whereby the user selects one of the several output segments by clicking on it. This has been automated by taking the centroid of the object in the gold standard image as if it were the user's click. Samples of breast ultrasound images can be seen in Figure 6.1.

Statue of Liberty Dataset

The Statue of Liberty dataset (Liberty) is taken from the CMU-Cornell iCoseg image segmentation dataset [172], which contains several categories of images each of specific objects. The Liberty category consists of 41 images with a high degree of shape variability. Some images contain the whole statue, while others contain only part of it. The size of the images is 375×500 pixels. Figure 6.2 shows sample images from this dataset.

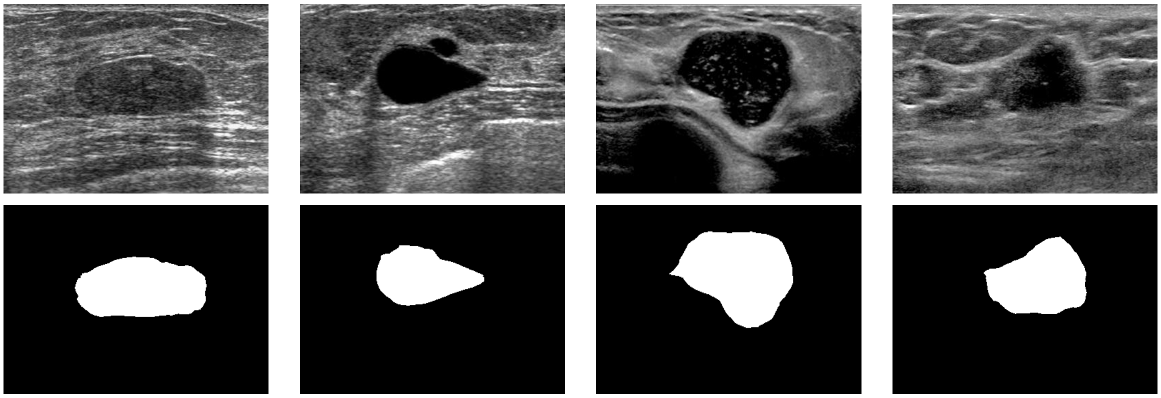


Figure 6.1: Samples of breast ultrasound dataset images (top row) with their corresponding gold standard images (bottom row).

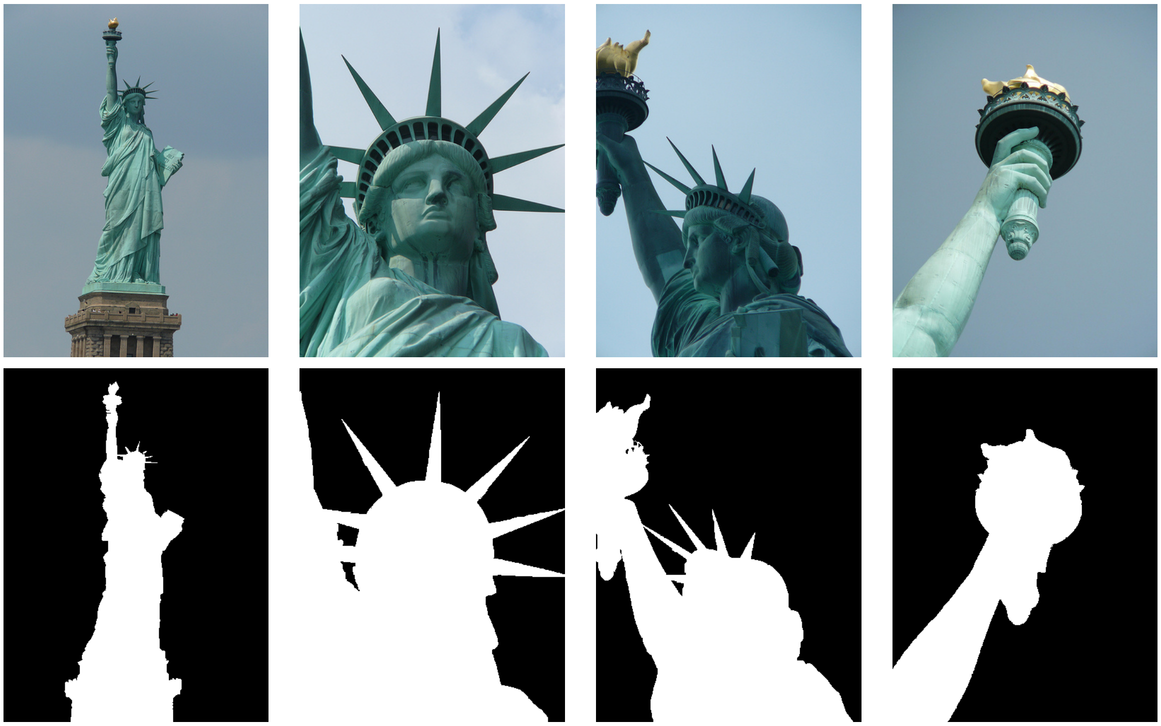


Figure 6.2: Samples of the Liberty dataset images (top row) with their corresponding gold standard images (bottom row).

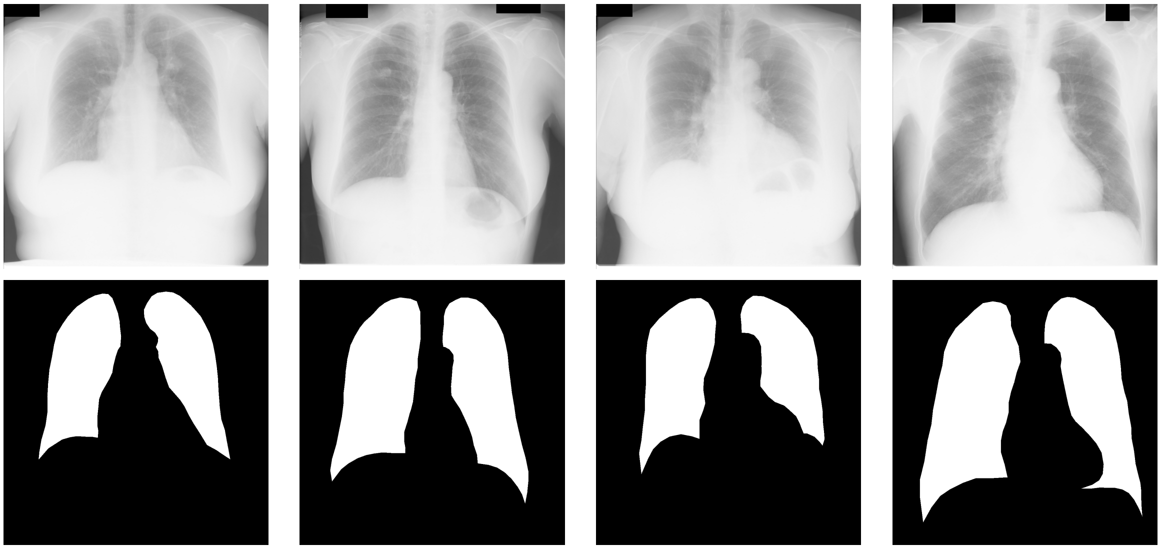


Figure 6.3: Samples of the images in the lung X-ray dataset (top row) with their corresponding gold standard images (bottom row).

Lung X-Ray Dataset

The lung X-ray dataset [173] consists of 98 lung X-ray images that are of 1024×1024 pixels in size. Compared to the images in the previous two datasets, these images exhibit a high degree of similarity. The contrast between the objects (lungs) and the background is low. Selected images from the lung X-ray dataset are presented in Figure 6.3.

Synthetic Images Dataset

The synthetic images dataset consists of 100 randomly created synthetic images. Each image is 1024×1024 pixels and was created by the random addition of three circles with random radius lengths and three rectangles with random lengths and widths. In this way, images will have objects of different shapes. However, the shapes are not as complex as the breast ultrasound and the Liberty datasets' images. Random Gaussian noise was then added to the image. Selected samples of this dataset are shown in Figure 6.4.

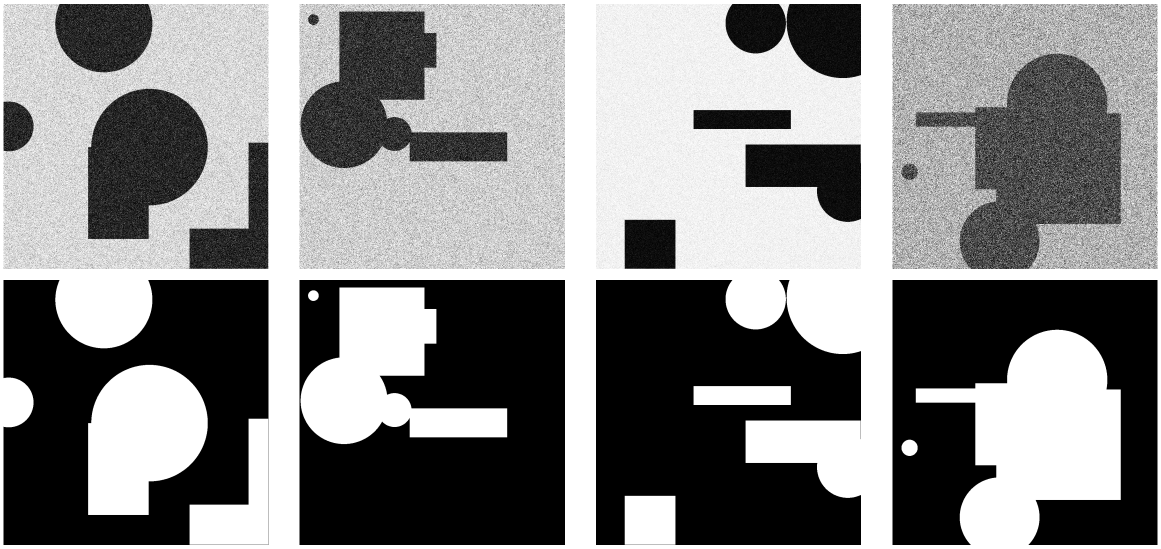


Figure 6.4: Samples of images from the synthetic images dataset (top row) with corresponding gold standard images (bottom row).

6.1.2 Parameter Settings

The parameters for the methods used in the experiments in this chapter are described below.

Image Segmentation

Obtaining the best resolutions as labels (i.e., classes) for training requires the images be segmented at all possible resolutions. For the datasets used, it was found that after the sixth level of resolution, no meaningful information remained in the image. The pyramid representation was thus comprised of six resolutions (i.e., $r = 6$), where 0 is the original image resolution, and 5 is the coarsest one. The pyramid was implemented as described in [74]. A separable 5×5 filter, w, was employed, defined by

$$w = [\frac{1}{4} - \frac{a}{2}, \frac{1}{4}, a, \frac{1}{4}, \frac{1}{4} - \frac{a}{2}]. \tag{6.1}$$

Parameter a was selected as 0.4, so that the filter would be close to a Gaussian shape [74].

The purpose of this research was to select the best resolution for the segmentation of an input image, rather than segmentation enhancement, including optimal parameter selection, preprocessing, or postprocessing. The parameters selected for the employed image segmentation algorithms might therefore not be the optimal choice for the respective datasets. However, some images, such as breast ultrasound and lung X-ray images, are difficult to segment without preprocessing or postprocessing. For the breast ultrasound dataset, a semi-automated segmentation approach was used as described in the previous subsection. Because of the low contrast between objects and background in the lung X-ray dataset images, their contrast was enhanced through contrast-limited adaptive histogram equalization [174].

In order to investigate the resolution selection framework with different image segmentation algorithms, three well-known methods were selected. They were chosen to have different concepts behind them. Parametric Kernel Graph Cuts (PKGraphCuts) is based on graph theory, ChanVese level set is an active contour method, and Statistical Region Merging (SRM) is a region growing technique. PKGraphCuts, implemented by Ben Ayed, was utilized in this research (code available at [175]). The regularization weight parameter α was set to 0.1. The initialization of the ChanVese level set was performed as multiple circles, which have been shown to be effective [63]. The iterations stop if a change less than η $(= 5)$ occurs for five consecutive iterations, or if the number of iterations exceeds a preset threshold value (here 1000). Parameter Q of the SRM algorithm was set to 32. The SRM implemented by Boltz was used in this work (code available at [176]).

Images of each dataset were segmented using different segmentation algorithms. The choice of segmentation algorithm for each dataset was performed empirically based on random samples of images from each dataset, as follows:

- Breast ultrasound and Liberty datasets: PKGraphCuts
- X-ray Lung dataset: ChanVese level set
- Synthetic dataset: SRM

The accuracy of the segmentation is measured using dice coefficient defined as follows [177]:

$$Dice = \frac{2|I_n \cap I_G|}{|I_n| + |I_G|}, \tag{6.2}$$

where I_n is the segmented image, I_G is the gold standard image, and $|\cdot|$ indicates the set cardinality.

Learning Methods

In the experiments conducted for this research, the performance achieved with RAMOBoost was compared with that obtained with AdaBoost (Appendix A.3), Support Vector Machines (SVM) (Appendix A.1), and SVM with re-sampled training data using ADASYN (SVM-ADASYN) (Appendix A.2). LibSVM [178] (code available at [179]) implementation of SVM was used, with a radial basis function for the kernel. The penalty parameter C and the parameter γ for radial basis function were selected through 5-fold cross validation. Joint Mutual Information (JMI) feature selection [180] (Appendix A.5) was used for the selection of the 10 most representative features for training the SVM. For ADASYN, the number of nearest neighbours was 5, and the balance level parameter β was chosen as 0.7. For both AdaBoost and RAMOBoost, the decision tree classifier was used as the base classifier. Both boosting algorithms were run for 10 iterations. LBP features were extracted using 10-bins histogram for each region. For all methods, the training and testing sets were split by 10-fold cross validation. Each experiment was run 10 times, and the average was taken.

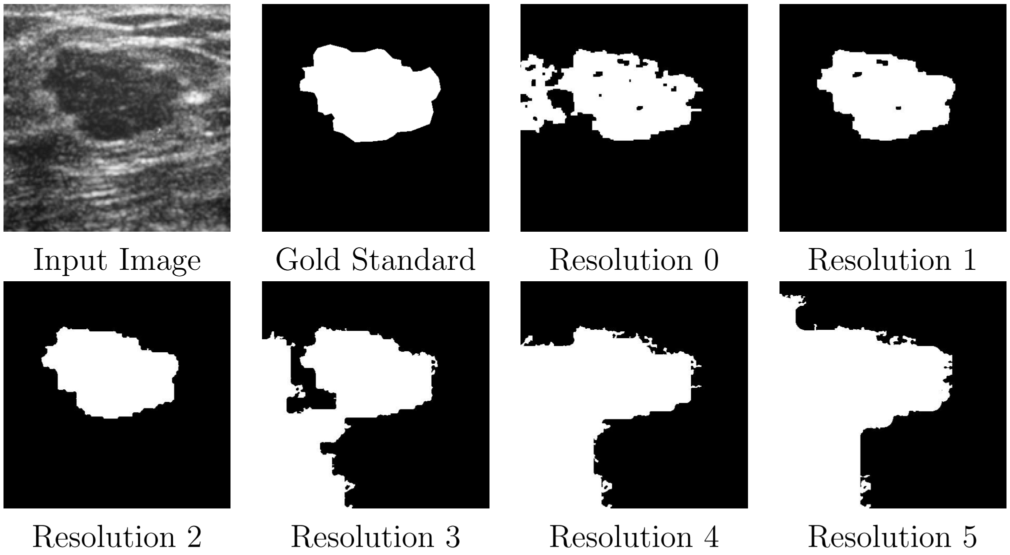
6.1.3 Implementation Environment

The experiments were conducted using a PC with 8 GB of RAM and a CPU speed of 2.20 GHz. The operating system was Windows 7 (64-bit version). The program was written and run with a 64-bit version of Matlab$^{\text{TM}}$.

6.2 Trade-off Measure Performance

6.2.1 Accuracy and Time with Respect to α

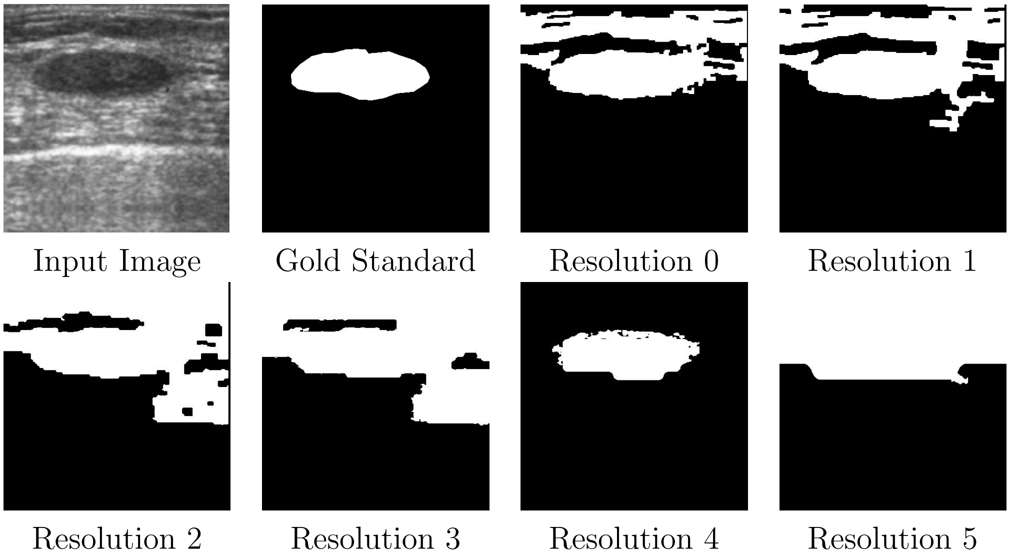
Computing the trade-off measure ω (Eq. 5.1) requires that each image be segmented at all available resolutions, with the accuracy and speed of each resolution being recorded. Figures 6.5-6.12 illustrate the segmentation outputs at all resolutions of the images from the breast ultrasound, Liberty, lung X-ray, and synthetic datasets, respectively. Figure 6.5 reveals that the second resolution obtained the best segmentation: the noise was reduced, and the region of the object became more homogeneous, which results in better segmentation. On the other hand, Figure 6.6 reveals that another image from the same dataset is better segmented at resolution 4. Figures 6.7 and 6.8 show the segmentation results of two images from the Liberty dataset. For the first, resolution 1 has the best segmentation results. It can be seen that the segmentation in resolution 0 is concentrating on local regions, and segmentation in resolution 2 is missing some details (e.g., the crown is trimmed, and the edges are affected by interpolation). The second image in Figure 6.8 is better segmented at resolution 2. The results of segmenting lung X-ray images are presented in Figures 6.9 and 6.10. For the first image, the results are relatively good at all resolutions, but at lower resolutions, where regions tends to be more homogeneous, the lung is segmented as a whole. No postprocessing is therefore needed in the segmentation at the third and fourth resolutions. The output of the fifth resolution shows some lost portions of the segmentation (e.g., the lower part of the right lung). Contrarily, the second image in Figure 6.10 is better segmented at resolution 3. Segmentation at resolution 4 and 5 became worse as the lungs started to merge. Figures 6.11 and 6.12 present the results of segmenting two images from the synthetic dataset. For the first image, the small circle at the top has disappeared in the second resolution, and the square at the bottom is missing at the fourth resolution. At resolution 5, the information seems to have been insufficient for segmentation. The second image in Figure 6.12 contains a high level of noise, which leads to poor segmentation at higher resolutions. It can be noticed that resolution 4 obtained the best segmentation results.



Input Image Gold Standard Resolution 0 Resolution 1

Resolution 2 Resolution 3 Resolution 4 Resolution 5

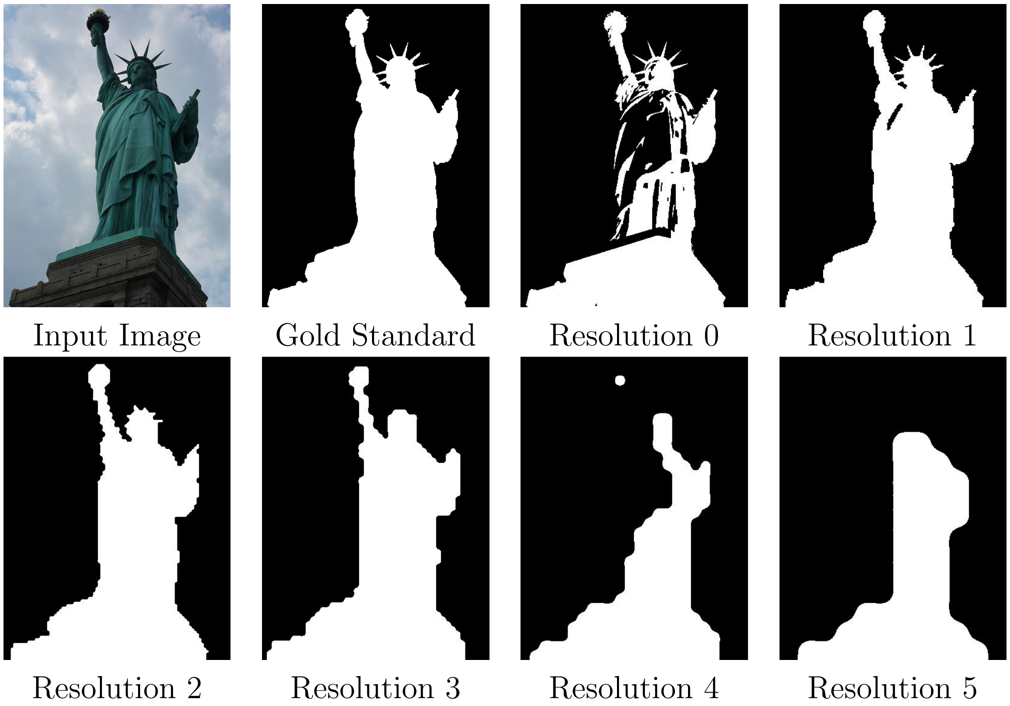
Figure 6.5: Sample outputs for each resolution for the breast ultrasound dataset (image 1).



Input Image Gold Standard Resolution 0 Resolution 1

Resolution 2 Resolution 3 Resolution 4 Resolution 5

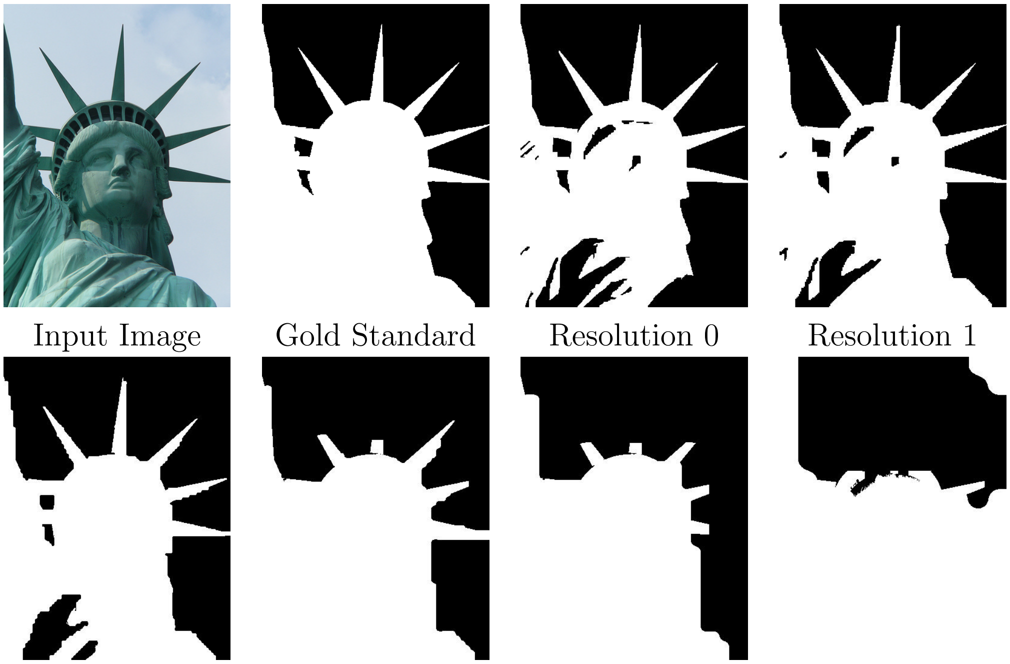
Figure 6.6: Sample outputs for each resolution for the breast ultrasound dataset (image 2).


Input Image Gold Standard Resolution 0 Resolution 1

Resolution 2 Resolution 3 Resolution 4 Resolution 5

Figure 6.7: Sample outputs for each resolution for the Liberty dataset (image 1).

After each image was segmented at all resolutions, and the segmentation accuracies (dice) and times (in seconds) were recorded, ω was calculated as described in Section 5.4. This subsection presents the results of the assessment of the performance, in respect to segmentation accuracy and speed, of the trade-off measure with different choices of α. These values were compared against the accuracies obtained when the images were segmented at the original resolution (level 0), the minimum resolution (level 5), and the **peak resolutions**, which are the resolutions with maximum frequency for being the best resolution for a given dataset (i.e., the mode of a normal or quasi-normal distribution of the best resolutions, see Figures 6.13-6.16). The purpose of the comparison with the original resolution is to compare the difference between the outcomes of using the framework and not using it. Comparing with the minimum resolution answers the question of the applicability of just selecting the lowest possible resolution. The importance of learning can be observed



Input Image Gold Standard Resolution 0 Resolution 1

Resolution 2 Resolution 3 Resolution 4 Resolution 5

Figure 6.8: Sample outputs for each resolution for the Liberty dataset (image 2).

by comparing the results of the framework with the peak resolutions, as they constitute the most frequent best resolution in a given dataset.

Tables 6.1-6.4 present the results of breast ultrasound, Liberty, lung X-ray, and synthetic datasets, respectively. The results show that the degree of accuracy increases as α increases. The change in the accuracy level with respect to α can be large, as with the Liberty and synthetic images, or only minimal, as with the lung images. For all datasets, using higher α values obtained better accuracy results than the original resolution and much better accuracy than the coarsest one. The results listed in the tables confirm that segmentation at resolutions other than the original could enhance the results.

With respect to the running times, it can be seen from the tables that selecting lower values of α results in faster execution times. Compared with the speed at the original resolution, all values of α for the four datasets produces much faster times. For example,

Table 6.1: Accuracy Levels and Processing Times at the Selected, Peak, Original, and Minimum Resolutions for Breast Ultrasound Dataset

	Breast Ultrasound Dataset			
α	Dice at Selected	Dice at Peak	Time at Selected	Time at Peak
0.1	69.88 ± 27.88	65.09 ± 30.01	0.32 ± 0.34	0.82 ± 0.50
0.3	70.73 ± 27.55	65.09 ± 30.01	0.37 ± 0.41	0.82 ± 0.50
0.5	71.14 ± 27.46	65.09 ± 30.01	0.41 ± 0.42	0.82 ± 0.50
0.7	71.65 ± 27.38	65.09 ± 30.01	0.52 ± 0.58	0.82 ± 0.50
0.9	71.93 ± 27.45	65.09 ± 30.01	0.74 ± 0.99	0.82 ± 0.50
Original	63.52 ± 28.92		11.52 ± 7.90	
Minimum	36.65 ± 33.03		0.20 ± 0.09	

Table 6.2: Accuracy Levels and Processing Times at the Selected, Peak, Original, and Minimum Resolutions for Liberty Dataset

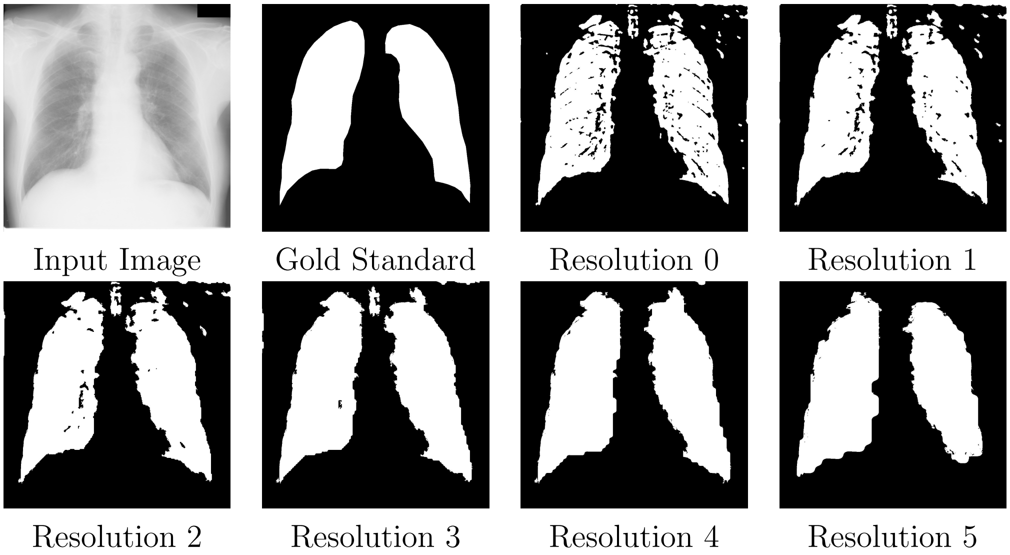
	Liberty Dataset			
α	Dice at Selected	Dice at Peak	Time at Selected	Time at Peak
0.1	72.07 ± 17.92	69.68 ± 19.74	0.33 ± 0.11	0.48 ± 0.18
0.3	75.80 ± 18.16	69.68 ± 19.74	0.48 ± 0.31	0.48 ± 0.18
0.5	76.58 ± 18.45	73.68 ± 20.37	0.57 ± 0.38	1.08 ± 0.33
0.7	77.58 ± 18.69	74.52 ± 20.92	0.84 ± 0.66	3.39 ± 1.20
0.9	78.04 ± 18.71	74.52 ± 20.92	1.05 ± 0.86	3.39 ± 1.20
Original	74.24 ± 18.61		13.54 ± 5.58	
Minimum	62.44 ± 20.25		0.27 ± 0.11	

Table 6.3: Accuracy Levels and Processing Times at the Selected, Peak, Original, and Minimum Resolutions for Lung X-Ray Dataset

	Lung X-Ray Dataset			
α	Dice at Selected	Dice at Peak	Time at Selected	Time at Peak
0.1	85.41 ± 5.75	84.64 ± 6.22	0.61 ± 0.09	1.21 ± 0.18
0.3	85.52 ± 5.73	84.64 ± 6.22	0.89 ± 0.43	1.21 ± 0.18
0.5	85.58 ± 5.68	84.64 ± 6.22	1.11 ± 0.99	1.21 ± 0.18
0.7	85.62 ± 5.65	84.64 ± 6.22	1.44 ± 1.48	1.21 ± 0.18
0.9	85.62 ± 5.65	84.64 ± 6.22	1.59 ± 1.67	1.21 ± 0.18
Original	82.42 ± 5.20		460.84 ± 63.07	
Minimum	80.19 ± 8.27		0.61 ± 0.09	

Table 6.4: Accuracy Levels and Processing Times at the Selected, Peak, Original, and Minimum Resolutions for Synthetic Dataset

	Synthetic Dataset			
α	Dice at Selected	Dice at Peak	Time at Selected	Time at Peak
0.1	90.77 ± 8.18	95.22 ± 6.03	3.01 ± 1.30	4.52 ± 0.09
0.3	93.93 ± 6.07	92.79 ± 12.17	3.45 ± 1.27	15.09 ± 0.14
0.5	95.09 ± 4.79	92.79 ± 12.17	3.89 ± 1.60	15.09 ± 0.14
0.7	95.64 ± 4.23	92.79 ± 12.17	4.29 ± 2.15	15.09 ± 0.14
0.9	96.32 ± 3.69	92.79 ± 12.17	6.03 ± 3.82	15.09 ± 0.14
Original	89.36 ± 0.00		56.79 ± 0.00	
Minimum	45.33 ± 0.00		1.32 ± 0.00	


Input Image

Gold Standard

Resolution 0

Resolution 1

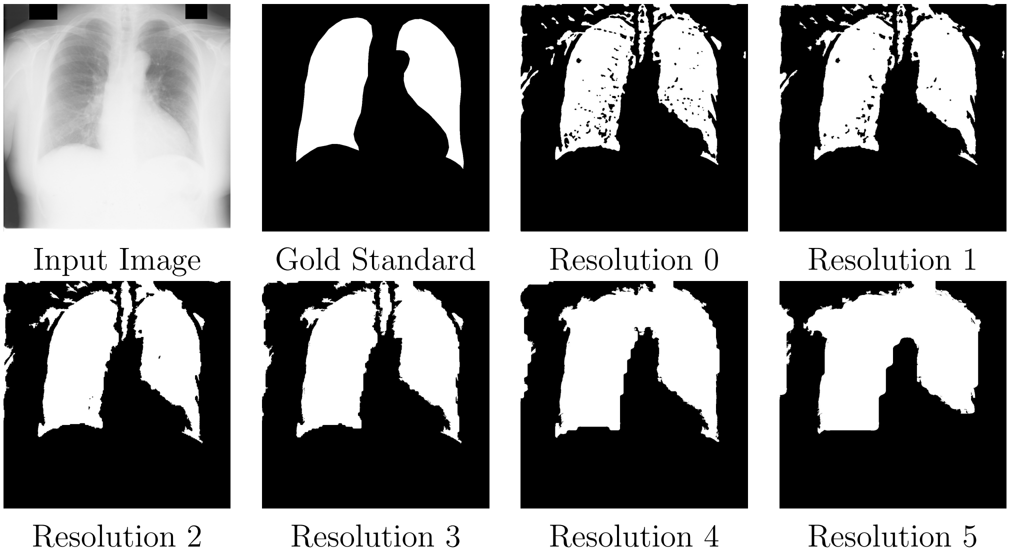
Resolution 2

Resolution 3

Resolution 4

Resolution 5

Figure 6.9: Sample outputs for each resolution for the lung X-ray dataset (image 1).


Input Image

Gold Standard

Resolution 0

Resolution 1

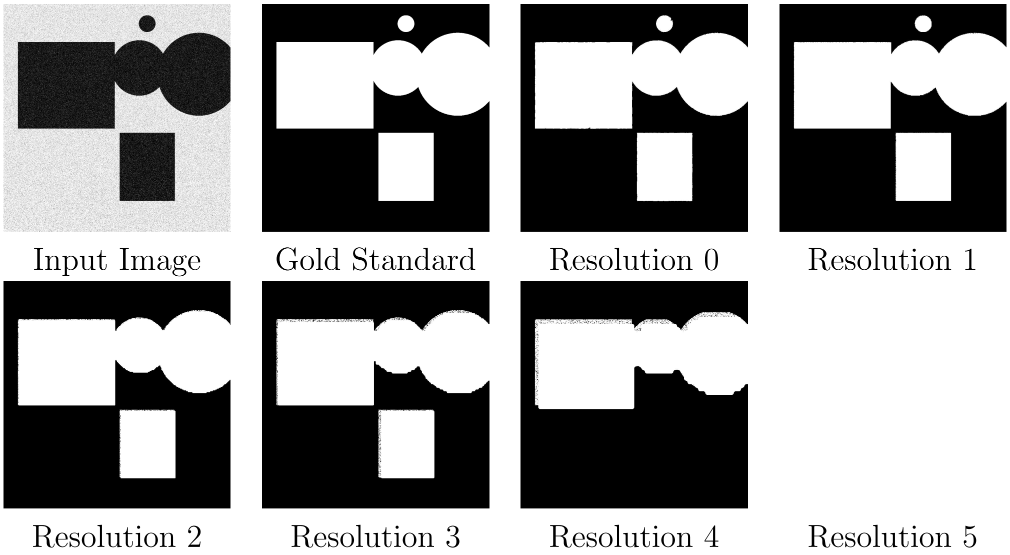
Resolution 2

Resolution 3

Resolution 4

Resolution 5

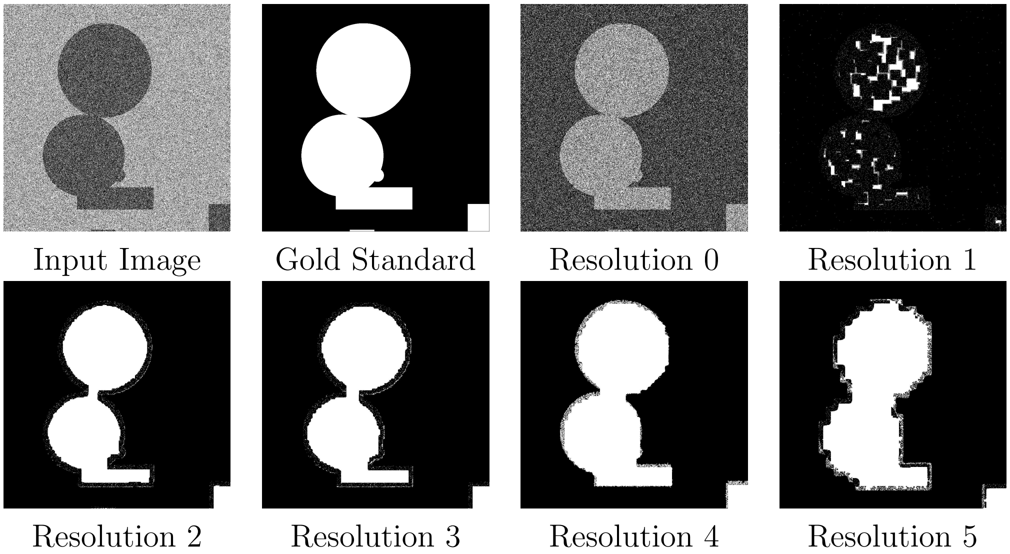
Figure 6.10: Sample outputs for each resolution for the lung X-ray dataset (image 2).



Input Image Gold Standard Resolution 0 Resolution 1

Resolution 2 Resolution 3 Resolution 4 Resolution 5

Figure 6.11: Sample outputs for each resolution for the synthetic dataset (image 1).



Input Image Gold Standard Resolution 0 Resolution 1

Resolution 2 Resolution 3 Resolution 4 Resolution 5

Figure 6.12: Sample outputs for each resolution for the synthetic dataset (image 2).

for the times resulting from α=0.9, which is considered to be the slowest and most accurate, the speed compared with the original resolution increases by 15, 12, 290, and 9 times for the breast ultrasound, Liberty, lung X-ray, and synthetic images, respectively. The acceleration is especially obvious in the case of the lung X-ray dataset, which has large images and was segmented with the inherently slow level set algorithm.

Comparing the selected resolutions with the peak resolutions, except with α=0.1 for the synthetic dataset, accuracies at the selected resolutions with all α values are better than those at the peak ones for the four datasets. With respect to speed, except for α=0.3 for the Liberty dataset and α=0.7 and 0.9 for the lung X-ray dataset, the speed at selected resolutions is faster than that at the peak resolutions. The difference in speed can be up to 2.6 times for the breast ultrasound dataset, 4 times for the Liberty dataset, and 4.4 times for the synthetic dataset.

For breast ultrasound, Liberty, lung X-ray, and synthetic datasets, Figures 6.13, 6.14, 6.15 and 6.16, respectively, illustrate the distributions of the resolutions selected based on different α values for the trade-off measure. It can be observed that the distribution varies significantly from dataset to dataset and for different α values with the same dataset. As can also be seen in many of the figures, the class distribution is imbalanced, which creates difficulties with respect to the learning process as discussed in Section 5.6.1. As well, no single selected resolution is revealed, even for images from the same dataset, and in no dataset is the original resolution selected for any value of α.

The previous results show that selecting resolutions for images in a specific dataset can obtain higher accuracies and faster speeds than fixing one resolution for all images. Segmenting at the original resolution results in lower accuracies and much slower speeds, while segmenting at the lowest resolution will result in much lower accuracies. Even compared with the peak resolutions, resolution selecting based on the trade-off measure obtained better results.

Note that for the lung X-ray dataset, the difference in accuracies among the selected, peak, original, and minimum resolutions is very small. Moreover, the difference in speed between selected and peak resolution is also small, and the peak resolution is faster than the selected ones for α=0.7 and 0.9. The reason for this is the extreme similarity of the objects (the lungs) among all the images in the dataset. This suggests that learning to select best resolutions for image segmentation might not work effectively for such datasets.

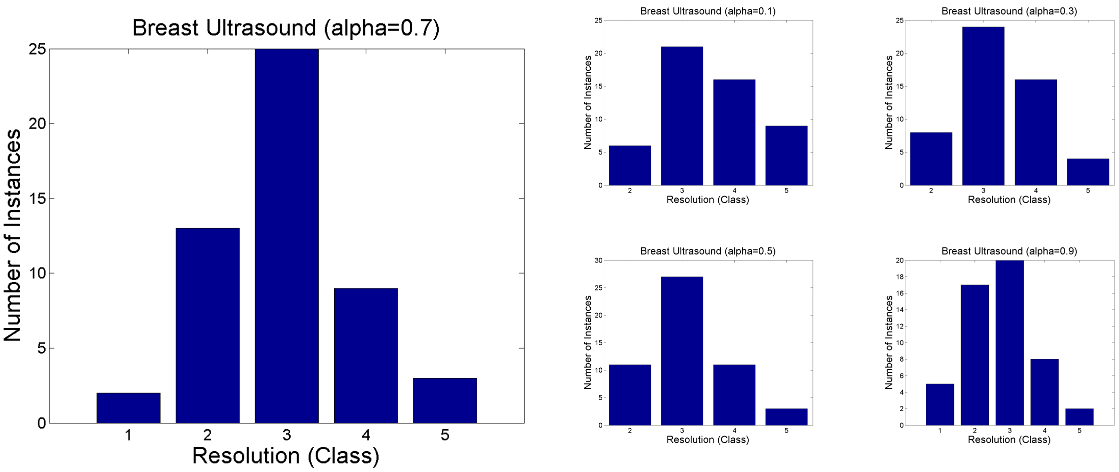


Figure 6.13: Selected resolution (class) distribution with different values of α for the breast ultrasound dataset.

6.2.2 Using Different Segmentation Algorithms

This subsection describes the investigation of the behaviour of the trade-off measure with images from one dataset segmented using three different segmentation algorithms. The purpose of this experiment was to determine whether class distributions are different when different segmentation methods are applied for the same dataset. The Liberty dataset was segmented with SRM and ChanVese level set, in addition to the previous segmentation using PKGraphCuts.

Table 6.5 shows the accuracy results obtained with the three segmentation methods for α=0.1, 0.3, 0.5, 0.7 and 0.9, as well as a comparison of the results with the ones obtained at the original and minimum resolutions. Table 6.6 lists the speed results for the same conditions. It can be observed that the accuracy of the three segmentation methods is very

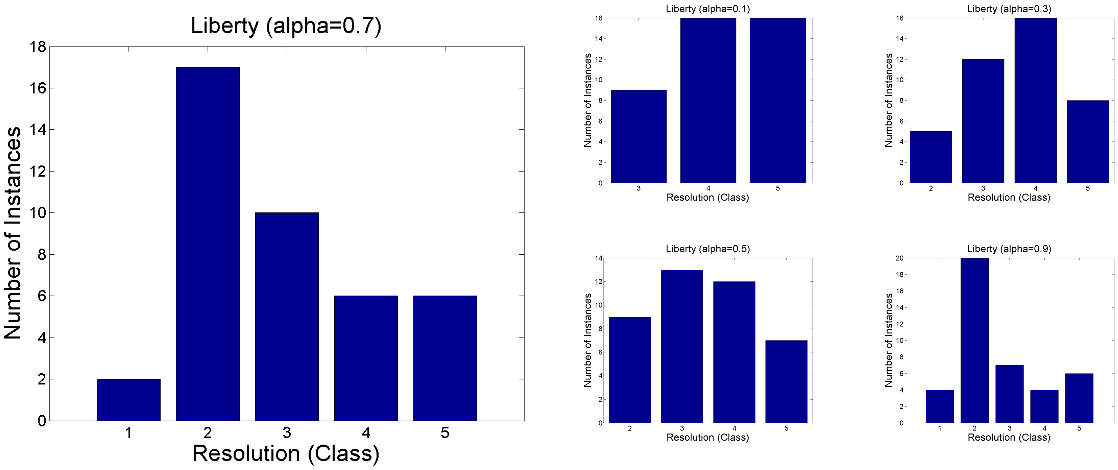


Figure 6.14: Selected resolution (class) distribution with different values of α for the Liberty dataset.

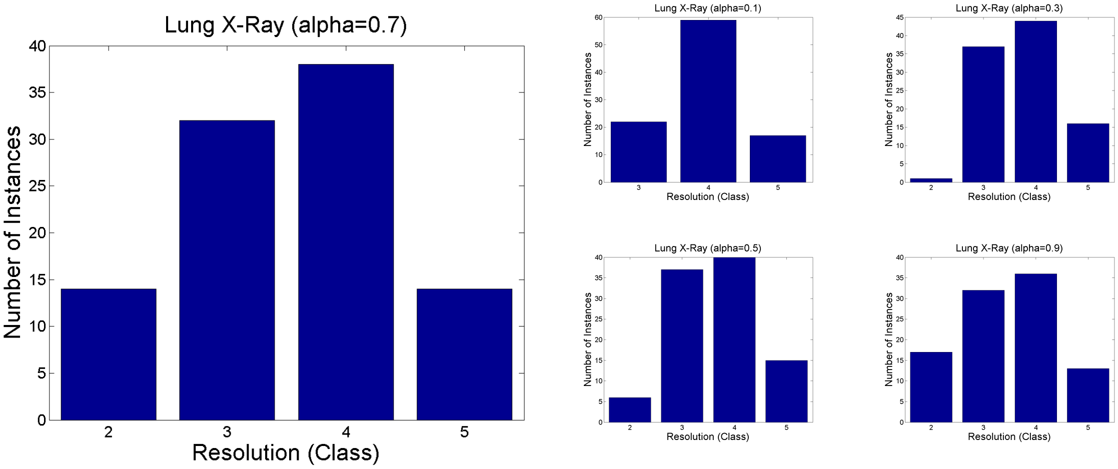


Figure 6.15: Selected resolution (class) distribution with different values of α for lung X-ray dataset.

close and that at α=0.9, the accuracy of PKGraphCuts and ChanVese is better than that at the original resolution. On the other hand, with α=0.9, SRM results in a slightly lower level of accuracy than at the original resolution. Regarding speed, PKGraphCuts provides

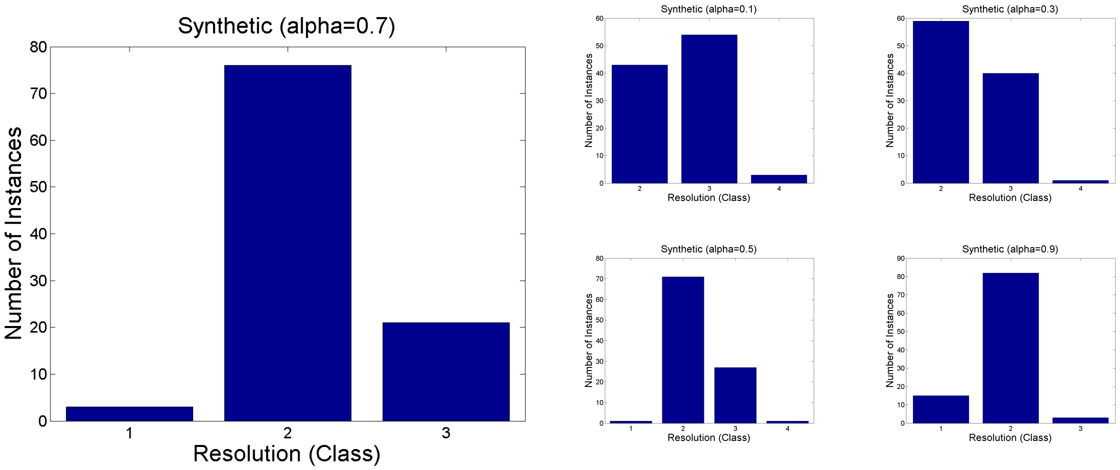


Figure 6.16: Selected resolution (class) distribution with different values of α for the synthetic images dataset.

the fastest performance, closely followed by SRM, then ChanVese. ChanVese shows the greatest increase in speed, because it is very slow at the original resolution.

Figures 6.17 and 6.18 illustrate the class distribution for the selected resolutions of the Liberty dataset segmented with SRM and ChanVese, respectively. A comparison of these figures with Figure 6.14 clearly reveals that, for the same dataset, different segmentation algorithms have different selected resolutions. This observation suggests that, even for the same dataset, learning for resolution selection should be performed for different segmentation algorithms separately.

6.2.3 Maximum Accuracy Resolutions

Interesting question may be asked about what would occur if accuracy were all that was needed. For determining the maximum accuracy resolutions, α should be set to 1. The accuracy and running times for the four datasets (including the Liberty dataset images segmented using the three segmentation methods) with α=1 are presented in Table 6.7, which also shows a comparison with the original resolutions.

Table 6.5: Accuracy at Selected Resolutions with Different Segmentation Algorithms

	Selected Resolutions Accuracy (%)		
α	PKGraphCuts	SRM	ChanVese
0.1	72.07 ± 17.92	70.43 ± 26.19	71.66 ± 18.87
0.3	75.80 ± 18.16	74.02 ± 24.81	75.82 ± 20.45
0.5	76.58 ± 18.45	76.66 ± 23.79	77.69 ± 21.48
0.7	77.58 ± 18.69	78.99 ± 22.81	78.32 ± 21.56
0.9	78.04 ± 18.71	79.45 ± 23.10	78.46 ± 21.53
Original	74.24 ± 18.61	80.74 ± 20.02	76.95 ± 24.05
Minimum	62.44 ± 20.25	41.77 ± 16.51	49.26 ± 17.46

Table 6.6: Times at Selected Resolutions with Different Segmentation Algorithms

	Selected Resolutions Times (seconds)		
α	PKGraphCuts	SRM	ChanVese
0.1	0.33 ± 0.11	0.44 ± 0.13	0.78 ± 0.60
0.3	0.48 ± 0.31	0.62 ± 0.57	1.11 ± 0.74
0.5	0.57 ± 0.38	0.98 ± 1.02	1.36 ± 0.99
0.7	0.84 ± 0.66	1.55 ± 1.25	1.73 ± 1.74
0.9	1.05 ± 0.86	1.74 ± 1.13	2.02 ± 2.09
Original	13.54 ± 5.58	12.47 ± 0.25	33.90 ± 27.77
Minimum	0.27 ± 0.11	0.40 ± 0.04	0.39 ± 0.03

Table 6.7: Maximum Accuracy Resolutions (α=1) Compared with Original Resolution for Different Datasets

Dataset	Sel. Res. Accuracy	Sel. Res. Time	Orig. Res. Accuracy	Orig. Res. Time
Breast Ultrasound	72.30 ± 27.43	3.47 ± 5.44	63.52 ± 28.92	11.52 ± 7.90
X-ray Lung	85.66 ± 5.60	10.40 ± 49.50	82.42 ± 5.20	460.84 ± 63.07
Synthetic	96.65 ± 3.83	12.20 ± 11.50	88.77 ± 7.35	56.77 ± 0.44
Liberty-PKGraphCuts	79.49 ± 18.57	8.20 ± 8.56	74.24 ± 18.6	13.54 ± 5.58
Liberty-SRM	82.79 ± 20.12	9.96 ± 4.48	80.74 ± 20.02	12.47 ± 0.25
Liberty-ChanVese	79.24 ± 21.73	9.97 ± 11.78	76.95 ± 24.05	33.90 ± 27.77

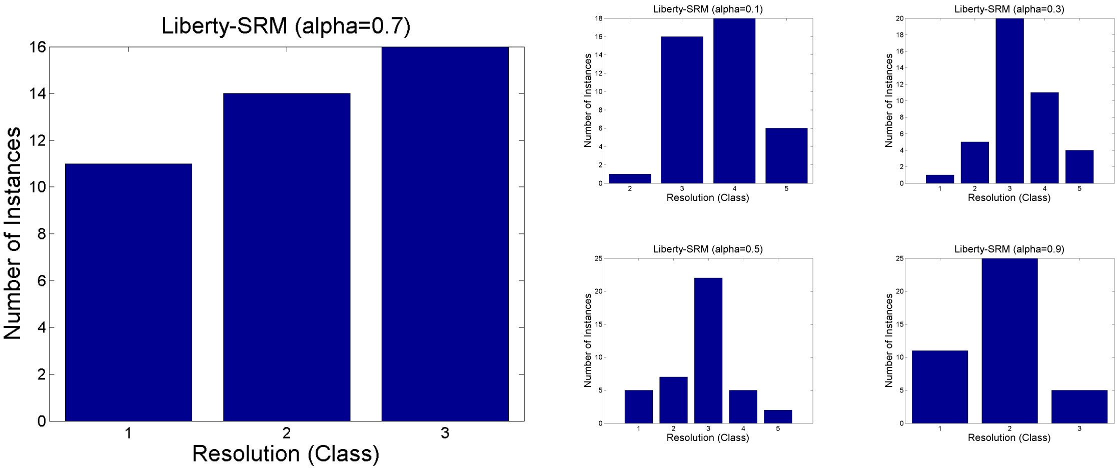


Figure 6.17: Selected resolution (class) distribution with different values of α for the Liberty dataset using the SRM segmentation algorithm.

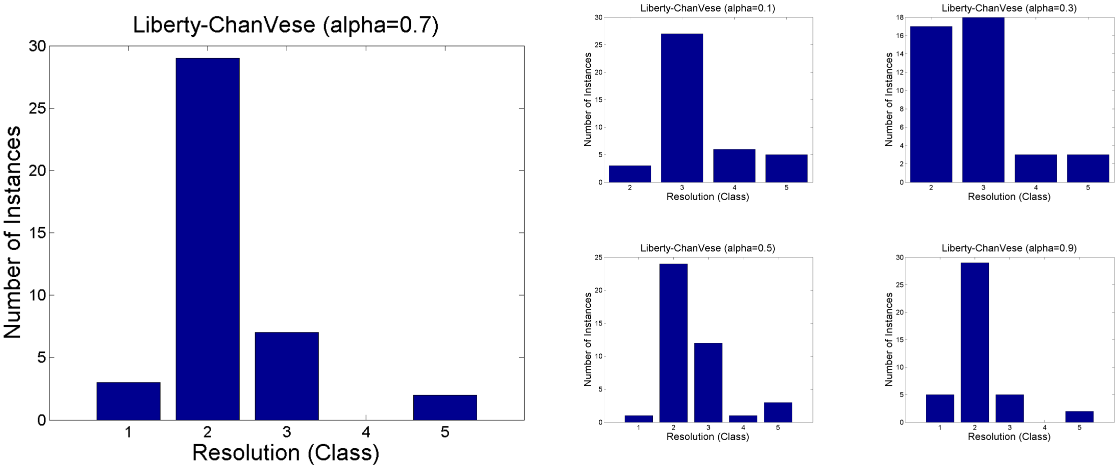


Figure 6.18: Selected resolution (class) distribution with different values of α for the Liberty dataset using the ChanVese level set segmentation algorithm.

It can be observed that the accuracy levels obtained with α=1 are all better than those at the original resolutions. The breast ultrasound and synthetic images datasets exhibit large differences between the accuracy at the selected resolutions and that at the original ones, while the accuracy for the Liberty-SRM images shows little difference. As with Tables 6.1-6.4, the speeds at the selected resolutions are still faster than at the original resolutions, but much slower than at the selected resolutions for lower values of α.

Figure 6.19 illustrates the class distribution. It should be noted that the original resolutions appear among the selected resolutions, while they were absent in the previous results. This change occurs because time is neglected when the selected resolutions are chosen with α=1.

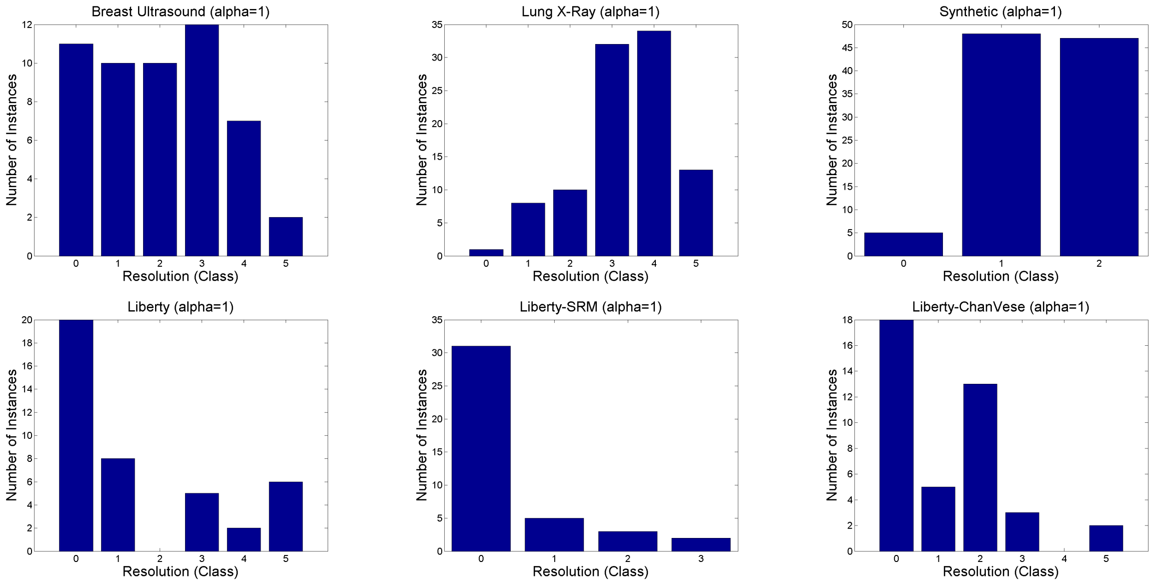


Figure 6.19: Maximum accuracy resolutions (α=1) distribution for, from right to left, (upper row) the breast ultrasound, lung X-ray, and synthetic images datasets, (lower row) Liberty-PKGraphCuts, Liberty-SRM, and Liberty-ChanVese.

6.3 Classifier Performance

This section presents an evaluation of the performance of RAMOBoost for resolution selection. As explained in Section 5.5.1, Local Binary Patterns (LBP) and statistical features

(Stat) were used for the learning process. The results obtained were compared with those produced by AdaBoost, SVM, and SVM with training data re-balanced via ADASYN (SVM-ADASYN). The targets (labels) were defined based on the trade-off measure ω (Eq. 5.1). In these experiments, five values of α were used: 0.1, 0.3, 0.5, 0.7, and 0.9. F_1*-measure* (Eq. 5.7) was used to evaluate the performance per class. The numerical results are listed in Tables C.1, C.2, C.3 and C.4 in Appendix C, and Figures C.1 - C.8 in Appendix C show a summary of them for α=0.1, 0.3, 0.5, and 0.9. Figures 6.20 - 6.27 illustrate the performance for α=0.7 along the confusion matrices. Because the presented classifiers' performances are the average of 10 runs, the mean and standard deviation of 10 confusion matrices are calculated. The values were rounded to the nearest integer for easier interpretation. The number of instances is different for each class. Figures 6.13-6.16 show the number of instances for each class for the four datasets.

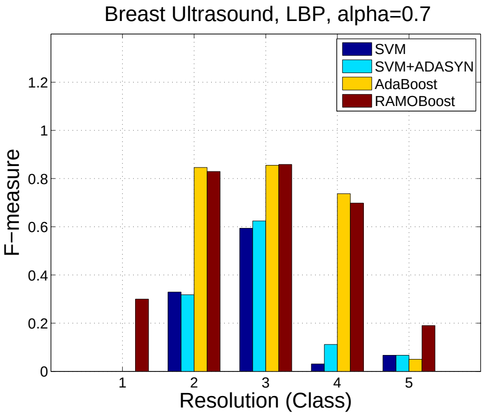
For the four datasets, Figures 6.13, 6.14, 6.15, and 6.16, reveals an imbalance in class distribution. Two of the methods used are designed specifically for imbalanced classification problems: RAMOBoost and ADASYN. The use of these methods was assessed with respect to the effect on the classification of minority class examples.

Breast Ultrasound Dataset

Figures 6.20 and C.1 show a comparison of the values of the F_1*-measure* obtained from the four learning methods for the breast ultrasound dataset using LBP features. It can be observed that RAMOBoost obtained the best overall results, followed closely by AdaBoost. The boosting algorithms significantly outperformed both SVM and SVM-ADASYN. The results reveal that SVM-ADASYN is slightly better than SVM with respect to classifying minority class samples, as in class 2 with α=0.3 and classes 1 and 5 with α=0.9. RAMOBoost is superior to AdaBoost in classifying minority classes, especially for class 5 with α=0.5, classes 1 and 5 with α=0.7 and class 1 with α=0.9.

The results obtained using Stat features are shown in Figures 6.21 and C.2. As with the LBP results, RAMOBoost produced the best classification and AdaBoost, the second best. Both SVM methods performed very poorly, with many classes that could not be correctly estimated, such as class 5 with α=0.3, α=0.5, and α=0.7, and classes 1 and 4 with α=0.9. SVM-ADASYN is slightly better than SVM for estimating minority classes, such as classes 2 and 4 with α=0.5.

Figures D.1, D.2, D.3, and D.4 in Appendix D provide a comparison of the performance of the LBP and Stat features. It can be observed that little difference appears in the performance of the two features when RAMOBoost and AdaBoost are used. However, the LBP features provide performance level that is far superior to that achieved by Stat features for either of the SVM methods.

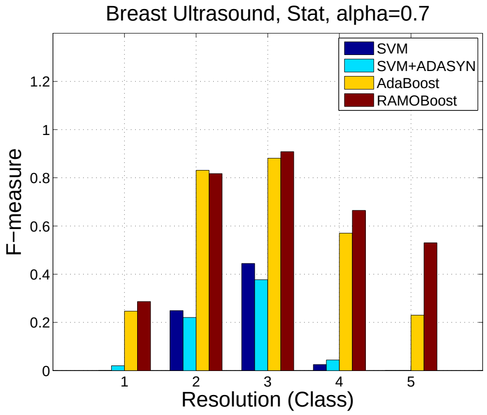


	Estimated				
Actual	1±1	0±0	2±1	0±0	0±0
	0±1	10±1	2±1	0±0	0±0
	0±0	0±1	25±1	0±0	0±0
	0±0	1±1	3±1	5±1	0±0
	0±0	0±0	2±1	1±1	0±1

Figure 6.20: Left: performance of different classifiers with α=0.7 for the breast ultrasound dataset: LBP features; right: confusion matrix of RAMOBoost.

Liberty Dataset

The classification performance for the Liberty dataset using LBP features is presented in Figures 6.22 and C.3. RAMOBoost provided the best overall performance, with AdaBoost producing slightly worse results. Both boosting methods are far superior to SVM and SVM-ADASYN. For example, class 4 with α=0.7 could not be correctly classified by either SVM or SVM-ADASYN, while it was accurately classified by both boosting algorithms. SVM and SVM-ADASYN offer similar performance levels. It should be noted how well



	Estimated				
Actual	1±1	0±0	1±1	0±0	0±0
	0±1	11±1	1±1	0±0	0±0
	0±0	0±0	25±0	0±0	0±0
	0±0	2±1	2±1	5±1	0±0
	0±0	1±1	1±1	0±0	1±0

Figure 6.21: Left: performance of different classifiers with α=0.7 for the breast ultrasound dataset: statistics features; right: confusion matrix of RAMOBoost.

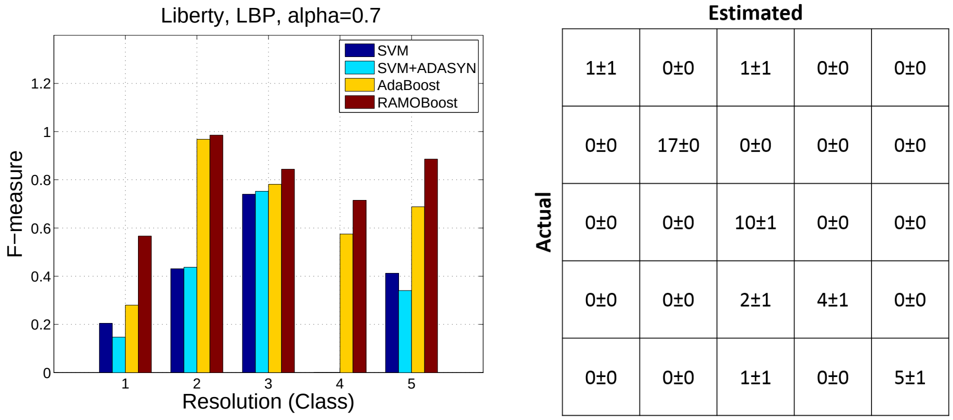
the boosting algorithms perform with respect to the imbalanced class distributions, as with α=0.7 and α=0.9.

Figures 6.23 and C.4 illustrate the performance using the Stat features. As with the LBP features, boosting methods provide much better performance than either of the SVM methods. SVM-ADASYN provides slightly better performance than does SVM for some minority classes, such as class 1 with α=0.7 and classes 1 and 4 with α=0.9.

A comparison of the results obtained with the LBP and Stat features for the four learning methods is presented in Figures D.5, D.6, D.7, and D.8 in Appendix D. It can be observed that the use of LBP and Stat features with RAMOBoost, AdaBoost, and SVM-ADASYN led to comparable performance levels for both features. On the other hand, the use of Stat features resulted in a generally better performance level with SVM.

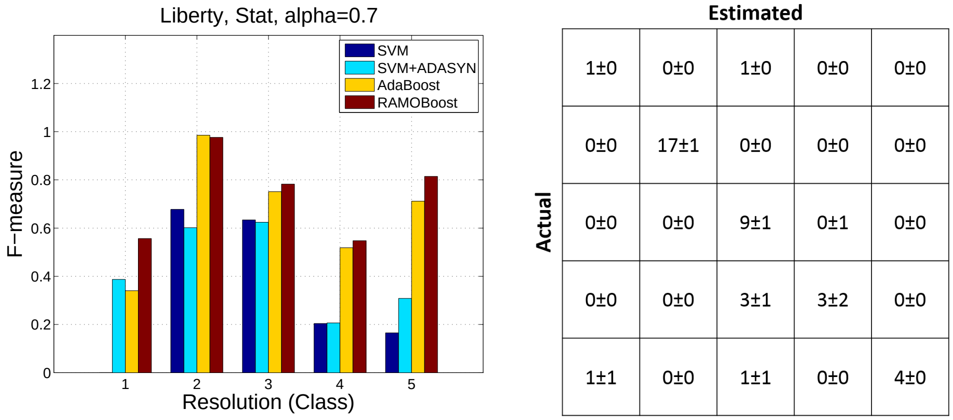
Lung X-Ray Dataset

A comparison of the performance of the different classification algorithms for the lung X-ray dataset using the LBP features is shown in Figures 6.24 and C.5. As with the previous



	Estimated				
Actual	1±1	0±0	1±1	0±0	0±0
	0±0	17±0	0±0	0±0	0±0
	0±0	0±0	10±1	0±0	0±0
	0±0	0±0	2±1	4±1	0±0
	0±0	0±0	1±1	0±0	5±1

Figure 6.22: Left: performance of different classifiers with α=0.7 for the Liberty dataset: LBP features; right: confusion matrix of RAMOBoost.



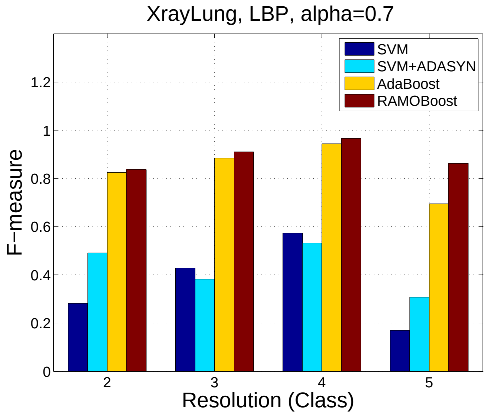
	Estimated				
Actual	1±0	0±0	1±0	0±0	0±0
	0±0	17±1	0±0	0±0	0±0
	0±0	0±0	9±1	0±1	0±0
	0±0	0±0	3±1	3±2	0±0
	1±1	0±0	1±1	0±0	4±0

Figure 6.23: Left: performance of different classifiers with α=0.7 for the Liberty dataset: statistics features; right: confusion matrix of RAMOBoost.

two datasets, RAMOBoost provides the best classification accuracy, followed closely by AdaBoost. SVM and SVM-ADASYN generally produce close accuracy levels. It can be observed how SVM-ADASYN led to better results than SVM for minority classes such as class 5 with α=0.1 and class 2 with α=0.5. As with the other datasets, RAMOBoost and AdaBoost result in far superior performance for minority classes estimation. It should be noted that class 2 with α=0.3 has only one instance, so it is impossible for it to be learned with any method.

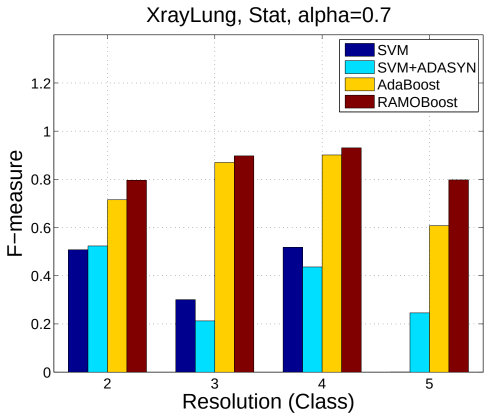
The results obtained using the Stat features are presented in Figures 6.25 and C.6. As with the LBP features, the boosting methods produce the best results, and the results achieved with SVM and SVM-ADASYN are inferior. SVM-ADASYN successfully classified some minority classes examples, such as for class 5 with α=0.1, classes 2 and 5 with α=0.5, and class 5 with α=0.7.

Figures D.9, D.10, D.11, and D.12 in Appendix D present a comparison of the classification results using the LBP and Stat features. It can be seen that with either type of features, the performance is almost the same with RAMOBoost, while the LBP results are generally slightly better with AdaBoost, SVM, and SVM-ADASYN.



	Estimated			
Actual	11±1	3±1	0±0	0±0
	0±0	31±1	0±1	0±0
	0±1	1±1	37±1	0±0
	0±0	2±1	1±1	11±1

Figure 6.24: Left: performance of different classifiers with α=0.7 for the lung X-ray dataset: LBP features; right: confusion matrix of RAMOBoost.



	Estimated			
Actual	11±2	2±1	1±1	0±0
	1±1	31±1	1±1	0±0
	1±1	1±1	36±1	0±0
	1±1	2±0	2±1	9±0

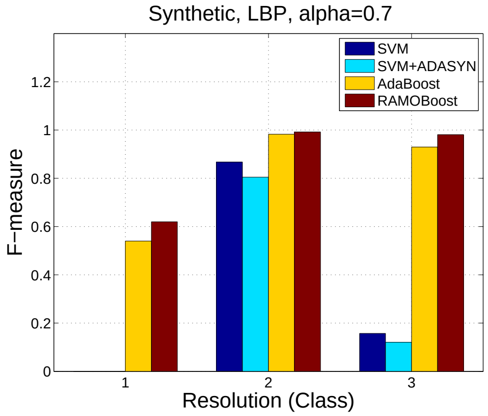
Figure 6.25: Left: performance of different classifiers with α=0.7 for the lung X-ray dataset: statistics features; right: confusion matrix of RAMOBoost.

Synthetic Dataset

Figures 6.26 and C.7 illustrate the results produced by the four classifiers using LBP features for the synthetic dataset. Both RAMOBoost and AdaBoost obtained excellent and comparable results, with SVM and SVM-ADASYN exhibiting worse results. In fact, for extremely skewed distributions (as can be seen in Figure 6.16), the boosting methods provide good results, as with class 1 with α=0.7 and class 3 with α=0.9, whereas neither SVM nor SVM-ADASYN could classify any instance of these classes. It should be noted that class 4 with α=0.3 and classes 1 and 4 with α=0.5 have only one instance, making it impossible for them to be learned.

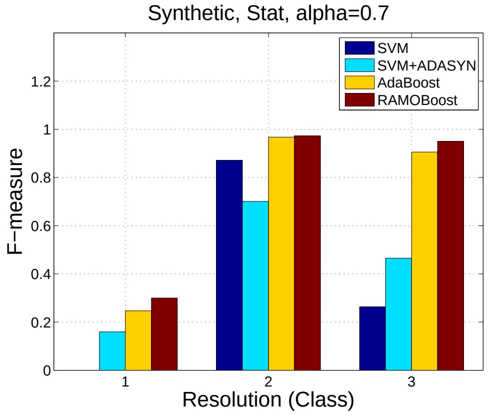
Figures 6.27 and C.8 show the results using the Stat features, which are very close to those obtained with the LBP features, with the exception that SVM-ADASYN could classify some instances from the classes that it was not able to classify using the LBP features.

A comparison of the LBP and Stat features is presented in Figures D.13, D.14, D.15, and D.16 in Appendix D. With RAMOBoost and AdaBoost, the results using the two types of features are very close. The Stat features produce slightly better results with the SVM and SVM-ADASYN methods.



	Estimated		
Actual	1±0	1±0	1±0
	0±0	76±0	0±0
	0±0	0±0	21±0

Figure 6.26: Left: performance of different classifiers with α=0.7 for the synthetic images dataset: LBP features; right: confusion matrix of RAMOBoost.



	Estimated		
Actual	1±0	2±0	0±0
	0±0	76±0	0±0
	0±0	2±1	19±1

Figure 6.27: Left: performance of different classifiers with α=0.7 for the synthetic images dataset: statistics features; right: confusion matrix of RAMOBoost.

Observations

Several points can be observed from the results in this section. The boosting algorithms greatly outperformed both SVM algorithms. As well, RAMOBoost has better classification performance than AdaBoost, especially for minority classes. This is due to the adaptability of RAMOBoost with respect to imbalanced class distribution.

SVM-ADASYN is better than SVM for minority classes classification, but slightly worse for majority ones. This shows that using ADASYN to re-balance class distribution can improve the classification of minority classes, but with the cost of affecting the performance for majority ones.

It can be noted from the confusion matrices that the classification performance is good, and the misclassified instances are usually classified as a resolution next to the actual ones, which decreases the impact of misclassification on accuracy and time as shown in the next section. Furthermore, the low standard deviations in the confusion matrices indicate that the classification results are robust and consistent.

6.4 Impact of Misclassification on Accuracy and Time

An important factor to assess is the impact of resolution misclassification on segmentation accuracy and speed, the investigation of which is described in this section. The estimated resolutions are compared with the original, minimum, and peak resolutions with respect to both accuracy and speed of segmentation. Although the selected resolutions obtained based on the trade-off measure ω are not available in real-life situations, they have been included here for reference. Paired t-tests with a 95% confidence level were performed in order to evaluate the significance of the differences. The effect of the outcomes obtained with RAMOBoost were analyzed.

Breast Ultrasound Dataset

The findings listed in Tables 6.8 and 6.9 indicate the impact of misclassification on accuracy and time for the breast ultrasound dataset. As can be observed, the accuracy at the estimated resolutions using LBP and Stat features is better than that at the original resolutions for all α values, and is statistically significant for α=0.3, 0.5, 0.7, and 0.9.

Table 6.8: Impact of Misclassification of RAMOBoost on Accuracy for the Breast Ultrasound Dataset; Est: estimated, Orig: original, Min: minimum, and Sel: selected.

α	Features	Est/Orig	Est/Min	Est/Peak	Est/Sel
0.1	LBP	1.35	4.32	1.23	0.96
	Statistical	1.31	4.32	1.20	0.95
0.3	LBP	1.38	4.76	1.26	0.98
	Statistical	1.36	4.65	1.24	0.96
0.5	LBP	1.40	4.96	1.28	0.99
	Statistical	1.37	4.82	1.26	0.97
0.7	LBP	1.40	4.78	1.27	0.97
	Statistical	1.39	4.88	1.26	0.96
0.9	LBP	1.39	4.55	1.26	0.96
	Statistical	1.39	4.88	1.26	0.97

The estimated resolutions with RAMOBoost provide substantially better accuracy levels than those at minimum resolutions, and are better than those at the peak resolutions, with α=0.5, 0.7, and 0.9 demonstrating significant differences. With respect to speed, the segmentation at the estimated resolutions is much faster than at the original ones, substantially slower than at the minimum resolutions, and faster than the peak resolutions, with α=0.1, 0.3, 0.5, and 0.7 with LBP and α=0.1, 0.3 and 0.5 with Stat demonstrating significant differences.

Liberty Dataset

The impact of misclassification on accuracy and time for the Liberty dataset is shown in Tables 6.10 and 6.11. It can be seen that the accuracy at the estimated resolutions is almost identical to the accuracy at the original ones, with the estimated resolutions only slightly worse than the original ones at α=0.1, and slightly better at the remaining values of the α settings. The accuracy at the estimated resolutions is significantly higher than at the minimum ones and better than the peak resolutions. The speed at estimated resolutions is much faster than at the original ones, but significantly slower than at the

Table 6.9: Impact of Misclassification of RAMOBoost on Time for the Breast Ultrasound Dataset; Est: estimated, Orig: original, Min: minimum, and Sel: selected.

α	Features	Orig/Est	Min/Est	Peak/Est	Sel/Est
0.1	LBP	22.41	0.42	1.64	0.98
	Statistical	23.39	0.43	1.69	1.03
0.3	LBP	21.19	0.40	1.55	0.99
	Statistical	21.77	0.40	1.57	1.00
0.5	LBP	19.97	0.37	1.45	0.97
	Statistical	19.75	0.36	1.43	0.98
0.7	LBP	18.47	0.34	1.34	1.07
	Statistical	17.78	0.32	1.28	0.98
0.9	LBP	15.47	0.29	1.12	0.97
	Statistical	16.01	0.29	1.16	0.98

minimum resolutions. For α=0.1 and 0.3, the segmentation at the estimated resolutions is slower than the ones at the peak resolutions, but it is much faster for α=0.5, 0.7, and 0.9.

Lung X-Ray Dataset

Tables 6.12 and 6.13 present the impact of misclassification on accuracy and time for the lung X-ray dataset. The accuracy at the estimated resolutions is statistically better than at the original, minimum, and peak resolutions. The running time is substantially faster at the estimated resolutions than at the original resolutions and slower than at the minimum and peak resolutions.

Synthetic Dataset

The impact of resolution misclassification on accuracy and speed for the synthetic dataset is presented in Tables 6.14 and 6.15. It can be observed that at all the estimated resolutions, better accuracy was obtained than at the original resolution, with a significant difference for all α values, except α=0.1. The accuracy at the estimated resolutions is substantially

Table 6.10: Impact of Misclassification of RAMOBoost on Accuracy for the Liberty Dataset; Est: estimated, Orig: original, Min: minimum, and Sel: selected.

α	Features	Est/Orig	Est/Min	Est/Peak	Est/Sel
0.1	LBP	0.99	1.22	1.05	1.00
	Statistical	0.99	1.22	1.06	1.00
0.3	LBP	1.02	1.26	1.09	0.98
	Statistical	1.01	1.26	1.09	0.98
0.5	LBP	1.03	1.29	1.05	0.99
	Statistical	1.04	1.29	1.05	0.99
0.7	LBP	1.04	1.29	1.04	0.98
	Statistical	1.03	1.28	1.03	0.97
0.9	LBP	1.04	1.29	1.04	0.98
	Statistical	1.03	1.29	1.04	0.97

Table 6.11: Impact of Misclassification of RAMOBoost on Time for the Liberty Dataset; Est: estimated, Orig: original, Min: minimum, and Sel: selected.

α	Features	Orig/Est	Min/Est	Peak/Est	Sel/Est
0.1	LBP	23.40	0.45	0.83	0.99
	Statistical	23.08	0.45	0.84	1.00
0.3	LBP	19.44	0.39	0.70	1.02
	Statistical	18.64	0.38	0.70	0.99
0.5	LBP	17.38	0.36	1.42	0.99
	Statistical	17.47	0.37	1.45	0.99
0.7	LBP	14.33	0.31	3.66	1.03
	Statistical	12.58	0.27	3.22	0.94
0.9	LBP	12.00	0.25	3.08	0.98
	Statistical	12.45	0.26	3.15	1.04

Table 6.12: Impact of Misclassification of RAMOBoost on Accuracy for the Lung X-Ray Dataset; Est: estimated, Orig: original, Min: minimum, and Sel: selected.

α	Features	Est/Orig	Est/Min	Est/Peak	Est/Sel
0.1	LBP	1.04	1.07	1.01	1.00
	Statistical	1.04	1.07	1.01	1.00
0.3	LBP	1.04	1.07	1.01	1.00
	Statistical	1.04	1.07	1.01	1.00
0.5	LBP	1.04	1.07	1.01	1.00
	Statistical	1.04	1.07	1.01	1.00
0.7	LBP	1.04	1.08	1.01	1.00
	Statistical	1.04	1.07	1.01	1.00
0.9	LBP	1.04	1.07	1.01	1.00
	Statistical	1.04	1.07	1.01	1.00

Table 6.13: Impact of Misclassification of RAMOBoost on Time for the Lung X-Ray Dataset; Est: estimated, Orig: original, Min: minimum, and Sel: selected.

α	Features	Orig/Est	Min/Est	Peak/Est	Sel/Est
0.1	LBP	298.86	0.40	0.78	1.00
	Statistical	309.05	0.41	0.81	1.00
0.3	LBP	282.12	0.37	0.74	1.01
	Statistical	291.66	0.39	0.76	1.01
0.5	LBP	265.23	0.35	0.70	1.01
	Statistical	274.23	0.37	0.72	1.02
0.7	LBP	248.50	0.33	0.65	1.01
	Statistical	248.27	0.33	0.65	0.98
0.9	LBP	235.63	0.32	0.62	0.98
	Statistical	234.23	0.32	0.61	0.97

Table 6.14: Impact of Misclassification of RAMOBoost on Accuracy for the Synthetic Dataset; Est: estimated, Orig: original, Min: minimum, and Sel: selected.

α	Features	Est/Orig	Est/Min	Est/Peak	Est/Sel
0.1	LBP	1.02	2.47	0.95	1.00
	Statistical	1.02	2.45	0.94	0.99
0.3	LBP	1.06	2.57	1.06	1.00
	Statistical	1.06	2.58	1.06	1.00
0.5	LBP	1.06	2.55	1.06	0.99
	Statistical	1.06	2.55	1.06	0.99
0.7	LBP	1.07	2.61	1.07	0.99
	Statistical	1.07	2.61	1.07	0.99
0.9	LBP	1.09	2.67	1.08	1.00
	Statistical	1.09	2.66	1.08	1.00

better than at the minimum ones. Compared with peak resolutions, the accuracy at the estimated resolutions is worse with α=0.1, but better for the remaining α values. With respect to speed, segmentation at the estimated resolutions is much faster than at the original and peak resolutions, and significantly slower than at the minimum ones.

Observations

Several trends can be observed based on the results:

- Except at α=0.1 for the Liberty dataset, the overall accuracy at the estimated resolutions is always superior to that at the original ones. In some cases, the difference in accuracy can be significant, as in the breast ultrasound, lung X-ray, and synthetic images datasets.

- The accuracy at the estimated resolutions is always significantly better than at the minimum resolutions.

Table 6.15: Impact of Misclassification of RAMOBoost on Time for the Synthetic Dataset; Est: estimated, Orig: original, Min: minimum, and Sel: selected.

α	Features	Orig/Est	Min/Est	Peak/Est	Sel/Est
0.1	LBP	16.57	0.39	1.32	0.99
	Statistical	17.93	0.42	1.43	1.00
0.3	LBP	14.73	0.34	3.92	0.99
	Statistical	15.46	0.36	4.11	0.98
0.5	LBP	13.19	0.31	3.51	1.02
	Statistical	13.96	0.33	3.71	1.03
0.7	LBP	12.61	0.29	3.35	1.03
	Statistical	13.14	0.31	3.49	1.03
0.9	LBP	9.94	0.23	2.64	1.01
	Statistical	10.37	0.24	2.76	1.02

- Except with α=0.1 for the synthetic dataset, the accuracy at the estimated resolutions is always better than that at the peak ones.

- The speed at the estimated resolutions is always much faster than at the original resolutions.

- The speed at the minimum resolutions is always substantially faster than at the estimated resolutions, but it must be remembered that the extreme speed at the minimum resolutions is accompanied by a huge reduction in accuracy.

- Except with the lung X-ray dataset, the speed at the estimated resolutions is almost always faster than that at the peak ones. Many cases are significantly faster, such as α=0.1, 0.3, 0.5, and 0.7 for the breast ultrasound dataset, and α=0.5, 0.7, and 0.9 for the Liberty and synthetic datasets.

When these points are considered, it is obvious that even with misclassification, the worst case is that the estimated resolutions produce the same degree of accuracy as the original resolutions but at significantly faster speeds. For three of the datasets, mostly the accuracy at the estimated resolutions is higher than the peak ones, and the speed is faster. This implies that resolution selection for individual images obtains better results than fixing at the peak resolutions. As discussed in Section 6.2.1, because of the extreme similarity of the images in the lung X-ray dataset, the results in terms of speed compared with the peak resolutions are not satisfactory. This confirms that resolution selection for images of extreme similarity might not be efficient.

6.5 Overhead Time Analysis

The calculation of the overhead time for feature extraction and classification is highlighted in this section. Although this time was included in the calculations for determining the impact of misclassification explained in Section 6.4, it is interesting to assess the contribution of each individual component. Two tasks result in overhead time: feature extraction and classification. Table 6.16 indicates the average times in seconds for these tasks. The time required for extracting the LBP and Stat features were calculated for images from each dataset. The lung X-ray and synthetic datasets required more time than the other two datasets because of their large size (1024×1024). The Stat features were faster to extract from the synthetic images. For the other datasets, the LBP and Stat extraction times are very close.

Although the training of RAMOBoost and AdaBoost differs, the classification time is the same because they use the same voting algorithm. It can be seen that classification time per instance is much shorter than that required for feature extraction.

6.6 Discussions

The primary objective of this research was to design a framework for the automated selection of resolutions for image segmentation. For the best resolution to be appropriately defined, a trade-off measure was developed. The measure allows the user or system to choose a trade-off between accuracy and time through a single parameter α. RAMOBoost was

Table 6.16: Overhead Times for Feature Extraction and for Classification Using RAMOBoost and AdaBoost

Dataset	Task	Time (mean)
Breast Ultrasound	Feature extraction (LBP)	0.1982
	Feature extraction (Stat)	0.1988
Liberty	Feature extraction (LBP)	0.2546
	Feature extraction (Stat)	0.2784
Lug X-Ray	Feature extraction (LBP)	0.7960
	Feature extraction (Stat)	0.7741
Synthetic	Feature extraction (LBP)	0.7941
	Feature extraction (Stat)	0.5779
—	Classification	0.0280

used for learning, using LBP and statistical features. The experimental verification of the new framework based on four different image datasets has been described in this chapter.

Trade-off Measure (ω)

The monotonic increase in accuracy with a shift from low to high α values and the similar decrease in time for the reverse shift suggest that α can indeed be used to control the trade-off between accuracy and time. Significant overall savings in time result when the segmentation is performed at resolutions having maximum ω values, even with α=1. For a segmentation algorithm with fixed parameters, highest accuracy through all resolutions is guaranteed when ω is calculated with α=1. Similarly, α=0 will result in the shortest times, but this effect is meaningless because the lowest resolution can, in fact, be selected for the fastest segmentation regardless of the accuracy.

The selected resolutions obtained with ω differ according to the segmentation algorithm, even for the same dataset. This finding suggests that ω can be used independently with many existing segmentation approaches.

Proposed Framework

The ability to use the trade-off measure ω (Eq. 5.1) with numerous segmentation algorithms indicates that the proposed framework can be used to learn the best resolutions for many segmentation algorithms. The results with different α values are also consistent. Higher α values provide better accuracy, and lower values result in faster segmentation. The selection of the best resolution could therefore be learned based on the user's requirements. For example, for critical systems, such as medical applications, α could be set to 1, which would yield the highest level of accuracy (even higher than at the original resolution). On the other hand, for applications entailing fast and rough segmentation, such as robot navigation or image retrieval, lower values of α could be selected.

Using a machine learning approach for resolution selection yields promising and statistically significant results, as verified based on the outcomes using four different datasets.

Learning Method and Features

The two boosting algorithms, RAMOBoost and AdaBoost, obtained much better results than did SVM or SVM-ADASYN. For the learning tasks in this research, 10 combined decision trees provide far greater accuracy than the strong SVM classifier. In addition, the SVM parameters were selected through cross-validation, while decision trees parameters were chosen arbitrarily and then fixed for all datasets. One reason for the superior performance of boosting is the directed learning: in each subsequent iteration, learning is concentrated on the instances that are difficult to learn. This feature might be the reason for the successful classification of some of the instances relating to minority classes.

ADASYN was used for re-sampling training data for SVM (SVM-ADASYN) and for each AdaBoost (RAMOBoost) iteration. Although the classification of minority classes is slightly better with SVM-ADASYN than with SVM, the estimation of the majority classes was slightly affected, showing decreased accuracy. On the other hand, the RAMOBoost performance was more impressive for all classes: much better than AdaBoost for minority classes classification, and slightly better for the majority classes. The time required for RAMOBoost to classify an input image is fast, which is desirable for fast decisions.

The performance levels produced with RAMOBoost using LBP and statistical features are very similar. These features are relatively fast with promising results, making them a good choice for the problem at hand. Other features can also be valuable, such as Gabor and some colour features, but they are much slower and hence are not suitable for fast tasks.

Accuracy and Time

The accuracy obtained and time required at the estimated resolutions were calculated and compared to those at the original, minimum, and peak resolutions. Except in one case, accuracy is always better at the estimated resolutions than at either the original or the minimum resolutions. The segmentation times at the estimated resolutions are extremely fast compared to those at the original resolutions, and significantly slower than at the minimum resolutions. However, the extreme speed at the lowest resolutions comes at the cost of a severe reduction in accuracy.

Given these observations, it seems that the worst case that can result from the use of the proposed framework is the selection of resolutions that provide the same segmentation accuracy as at the original resolution, but that do so significantly faster.

Comparing the accuracy and speed at the selected resolutions with those at the original, minimum, and peak ones showed that learning for resolution selection provides better overall results. However, it is not effective for images having extremely similar objects, as shown with the lung X-ray dataset.

Time Complexity Analysis

For an image of size $M \times N$, the input size is $n = M \cdot N$. Computing the LBP with an operator of size $k \times k$ requires a time of $O(k^2 n)$. Statistical features can be divided into two groups: ones that use filtering in order to extract the features, such as spatial uniformity, and those that rely on the histogram for the calculation of the features. Calculating the histogram requires a time of $O(n)$, and calculating the features from the histogram takes $O(g)$ time, where g is the intensity levels. Using the Sobel operator of size $k \times k$ for the spatial uniformity filter requires a time of $O(k^2 n)$, and as with the other statistical features, the standard deviation requires a time of $O(n + g)$. The time complexity of the combined statistical features is therefore $O(k^2 n + g)$. It should be noted that the LBP and statical features entail almost the same time complexity, which can be further observed from the examination of the running times reported in Section 6.5.

RAMOBoost involves a time complexity of $O(n^2 I log(n))$ in the training phase [130], where I is the number of iterations. The classification time is $O(I)$ [130], that is, it is dependent on the number of trained hypotheses. It is important to mention that these times are applicable only for RAMOBoost, without the base classifier. Different base classifiers have different time complexities, with the classification time usually being much faster than the training time.

As was determined based on this research, image segmentation at coarser resolutions is much faster than at finer ones because of the reduction in the input size (e.g., n above). A lower input size (number of pixels) reduces the time complexity, resulting in faster segmentation.

It is important to mention that all experiments conducted in this thesis were performed using Matlab$^{\text{TM}}$. Although it is a well-known platform for development and research, it is relatively slow compared with other programming languages. Therefore, the use of faster languages, such as *C++*, will provide much faster segmentation.

This chapter has presented the experiments conducted in order to verify the performance of the proposed framework. The results have also been provided and discussed. The next chapter summarizes the findings, highlights the contributions, and provides suggestions for future research.

Chapter 7

Summary, Conclusions and Future Work

7.1 Summary and Main Findings

In this thesis, a framework for the automated resolution selection for image segmentation was proposed. This framework is applicable for numerous segmentation methods. The pyramid representation was chosen as the multiresolution analysis method, and the multiresolution coarse-to-fine segmentation strategy was adopted. The best resolutions are determined based on the simultaneous consideration of both time and accuracy, which is achieved by using a trade-off measure ω, whose parameter α is controlled by the user to specify the desired trade-off between accuracy and speed. A supervised machine learning approach, RAMOBoost, was used for the selection of the best resolutions. The main findings of this thesis are summarized as follows:

- Segmentation at resolutions other than the original one may provide better accuracy, usually at a much faster speed.
- Images in the same dataset can have different best resolutions for segmentation purposes.

- Different segmentation algorithms can have different best resolutions for the same image.

- The trade-off measure can be used to define and obtain the best resolution based on user requirements: higher α values result in the selection of resolutions with better accuracy, and lower values favour resolutions with faster segmentation.

- The framework introduced in this thesis has been shown to successfully learn and estimate resolutions for input images. Its efficiency was verified with the use of four different image datasets and three segmentation algorithms.

- Boosting algorithms can learn more effectively than single strong classifiers. In this research, 10 iterations of AdaBoost and RAMOBoost with decision trees as base classifiers substantially outperformed the powerful SVM.

- Synthetic oversampling can be used to enhance learning from imbalanced class data, an effect that is especially obvious with boosting methods, such as RAMOBoost. However, oversampling can also affect learning from majority classes, as indicated by the results obtained with SVM-ADASYN.

- The LBP and statistical features used in this research provided promising results with fast performance, characteristics that make them good candidates for fast decision making with respect to the selection of a resolution for image segmentation.

- The worst case of selection performance seems to be providing resolutions that result in segmentation accuracy at least equal to that produced at the original resolutions but at much faster speeds.

7.2 Contributions

The main contributions in this research are as follows:

- A framework for the resolution selection for image segmentation has been proposed. This framework is suitable for use with many segmentation algorithms and is comprised of several components: the definition of best resolutions as labels for learning, feature extraction, and the learning and selection of resolutions based on the features and labels.

- A trade-off measure for defining the best resolution for image segmentation has been proposed. Because such a definition is based on the application at hand, using a single parameter, the trade-off measure provides the user or system with the flexibility to determine an appropriate trade-off between accuracy and speed.

- RAMOBoost has been shown to be effective for learning from imbalanced class distribution data. RAMOBoost combines the strength of a boosting algorithm with synthetic oversampling and is therefore a good choice for the problem at hand. The experimental results confirm the advantage of RAMOBoost over AdaBoost, SVM, and SVM-ADASYN.

Other contributions of the research include:

- The utilization of LBP features for learning has been demonstrated. These features offer good learning performances and are fast to extract, which makes them suitable for fast estimation.

- An analysis of the behaviour of image segmentation at different resolutions with respect to accuracy and speed (Section 4.2)

- Empirical evidence that images of the same dataset can have different best resolutions (Section 6.2.1)

- Verification that images segmented with different segmentation algorithms can have different best resolutions (Section 6.2.2)

- Comprehensive verification of the framework using four different datasets and three segmentation algorithms

7.3 Future Work

The following are suggestions for extending the research conducted for this thesis. Other multiresolution approaches could be investigated. An important criterion would be that the images be subsampled at lower resolutions, excluding methods such as scale-space. Examples of suggested choices are irregular pyramids [80] and adaptive pyramids [82].

An approach involving an adaptive parameter setting per resolution for the coarse-to-fine strategy for image segmentation could be studied. It is logical to assume that using the same parameters of a specific image segmentation algorithm at all resolutions would not produce the best results at each resolution: the parameters could be the best for one resolution, but be very poor for others.

A natural extension suitable for real applications is the ability to learn continually. Online learning methods could be utilized for this purpose. For example, in multi-core processors, the decision making could be performed in one core, while other cores segment the input image at all resolutions. The user could then determine the best choice, or the results could be automatically compared with existing gold standard images.

This approach could be used for automated resolution selection for video tracking. Fast tracking in high-definition video may be impossible, giving rise to a need for automated decisions about the best resolution for tracking.

The application of the proposed framework in combination with other image processing applications could be examined. For example, it could be applied for image registration, in which case the trade-off measure could be used as is, because accuracy and time are the two main concerns in this area.

References

[1] R. Patel and A. Greig, “Segmentation of 3d acoustic images for object recognition purposes,” in *OCEANS ’98 Conference Proceedings*, vol. 1, pp. 577–581, 1998.

[2] R. Lu, P. Marziliano, and C. H. Thng, “Liver tumor volume estimation by semi-automatic segmentation method,” in *27th Annual International Conference of the Engineering in Medicine and Biology Society*, pp. 3296–3299, 2005.

[3] T.-W. Chen, Y.-L. Chen, and S.-Y. Chien, “Fast image segmentation and texture feature extraction for image retrieval,” in *IEEE 12th International Conference on Computer Vision Workshops*, pp. 854–861, 2009.

[4] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[5] R. Nock and F. Nielsen, “Statistical region merging,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1452–1458, 2004.

[6] J. N. Kapur, P. K. Sahoo, and A. K. C. Wong, “A new method for gray-level picture thresholding using the entropy of the histogram,” *Graph. Models Image Process*, vol. 29, pp. 273–285, 1985.

[7] Y. Boykov and O. Veksler, “Graph cuts in vision and graphics: Theories and applications,” in *Handbook of Mathematical Models in Computer Vision* (N. Paragios, Y. Chen, and O. Faugeras, eds.), pp. 79–96, Springer US, 2006.

[8] R. J. Brunner, S. G. Djorgovski, T. A. Prince, and A. S. Szalay, "Handbook of massive data sets," ch. Massive datasets in astronomy, pp. 931–979, Norwell, MA, USA: Kluwer Academic Publishers, 2002.

[9] M. Merino and J. Nunez, "Super-resolution of remotely sensed images with variable-pixel linear reconstruction," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 5, pp. 1446–1457, 2007.

[10] I. Scholl, T. Aach, T. Deserno, and T. Kuhlen, "Challenges of medical image processing," *Computer Science - Research and Development*, vol. 26, pp. 5–13, 2011.

[11] B. Leroy, I. Herlin, and L. Cohen, "Multi-resolution algorithms for active contour models," in *ICAOS '96*, vol. 219 of *Lecture Notes in Control and Information Sciences*, pp. 58–65, Springer Berlin / Heidelberg, 1996.

[12] F. Al-Qunaieer, H. Tizhoosh, and S. Rahnamayan, "Multi-resolution level set image segmentation using wavelets," in *18th IEEE International Conference on Image Processing (ICIP)*, pp. 269–272, 2011.

[13] H. Lombaert, Y. Sun, L. Grady, and C. Xu, "A multilevel banded graph cuts method for fast image segmentation," in *Tenth IEEE International Conference on Computer Vision*, vol. 1, pp. 259–265, 2005.

[14] J. Maerker, W. Grob, W. Middelmann, and A. Ebert, "Hyperspectral band selection using statistical models," 2011.

[15] A. P. S. V. Mamta Bhojne, Abhishek Chakravarti, "High performance computing for satellite image processing and analyzing a review," *International Journal of Computer Applications Technology and Research*, vol. 2, no. 4, pp. 424–430, 2013.

[16] P.-Y. Tsang, "Multi-resolution image segmentation using geometric active contours," Master's thesis, University of Waterloo, Waterloo, Ontario, Canada, 2004.

[17] R. Fernandez-Gonzalez, T. Deschamps, A. Idica, R. Malladi, and C. O. de Solorzano, "Automatic segmentation of histological structures in mammary gland tissue sections," *Journal of Biomedical Optics*, vol. 9, no. 3, pp. 444–453, 2004.

[18] A. Joulin, F. Bach, and J. Ponce, "Discriminative clustering for image co-segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1943–1950, 2010.

[19] C. Wang, S. Guo, J. Wu, Q. Liu, and X. Wu, "Lung region segmentation based on multi-resolution active shape model," in *7th Asian-Pacific Conference on Medical and Biological Engineering*, vol. 19, pp. 260–263, 2008.

[20] D. Bruce, "Object oriented classification: case studies using different image types with different spatial resolutions," in *21st International Society for Photogrammetry and Remote Sensing (ISPRS) World Congress*, vol. 37, pp. 515–520, 2008.

[21] H. Huiping, W. Bingfang, and F. Jinlong, "Analysis to the relationship of classification accuracy, segmentation scale, image resolution," in *IEEE International Geoscience and Remote Sensing Symposium, IGARSS*, vol. 6, pp. 3671–3673, 2003.

[22] R. Gonzalez and R. Woods, *Digital Image Processing*. Prentice Hall, third ed., 2008.

[23] N. Otsu, "Threshold selection method from gray-level histogram," *IEEE Trans. Sys. Man Cybern.*, vol. SMC-9, no. 1, pp. 62–66, 1979.

[24] L. Sobel, *Camera Models and Machine Perception*. Stanford University, 1970.

[25] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 1, pp. 679–699, 1986.

[26] W. Tan, G. Coatrieux, B. Solaiman, and R. Besar, "A region based segmentation using pixel block fuzzy similarity," in *2nd Information and Communication Technologies, ICTTA '06*, pp. 1516–1521, 2006.

[27] R. Malladi, J. Sethian, and B. Vemuri, "Shape modeling with front propagation: a level set approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 2, pp. 158–175, 1995.

[28] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 146–168, 2004.

[29] J. Kittler and J. Illingworth, "Minimum error thresholding," *Pattern Recognition*, vol. 19, no. 1, pp. 41–47, 1985.

[30] W. Tao, H. Jin, and L. Liu, "A new image thresholding method based on graph cuts," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. I–605–I–608, 2007.

[31] Y. Yang, C. Zheng, and P. Lin, "Image thresholding based on spatially weighted fuzzy c-means clustering," in *The Fourth International Conference on Computer and Information Technology, CIT*, pp. 184–189, 2004.

[32] S. Rahnamayan and H. Tizhoosh, "Image thresholding using micro opposition-based differential evolution (micro-ode)," in *IEEE Congress on Evolutionary Computation, CEC 2008*, pp. 1409–1416, 2008.

[33] S. Rahnamayan, H. Tizhoosh, and M. Salama, "Weighted voting-based robust image thresholding," in *IEEE International Conference on Image Processing*, pp. 1129–1132, 2006.

[34] F. Al-Qunaieer, H. Tizhoosh, and S. Rahnamayan, "Oppositional fuzzy image thresholding," in *IEEE International Conference on Fuzzy Systems (FUZZ)*, pp. 1–7, 2010.

[35] J. Kang and W. Zhang, "An approach for image thresholding using cnn associated with histogram analysis," in *International Conference on Measuring Technology and Mechatronics Automation, ICMTMA*, pp. 421–424, 2009.

[36] X. Ren, “An optimal image thresholding using genetic algorithm,” in *International Forum on Computer Science-Technology and Applications, IFCSTA*, pp. 169–172, 2009.

[37] Z. Lin, Z. Wang, and Y. Zhang, “Image thresholding using particle swarm optimization,” in *International Conference on MultiMedia and Information Technology, MMIT*, pp. 245–248, 2008.

[38] D. Bradley and G. Roth, “Adaptive thresholding using the integral image,” *Journal of Graphics Tools*, vol. 12, no. 2, pp. 13–21, 2007.

[39] L. Roberts, *Optical and Electro-Optimal Information Processing*, ch. Machine Perception of Three-Dimension Solids. MIT Press, Cambridge, 1965.

[40] J. Prewitt, *Picture Process*, ch. Enhancement and Extraction. Academic Press, New York, 1970.

[41] D. Marr and E. Hildreth, “Theory of edge detection,” *Proceedings of the Royal Society of London. Series B, Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.

[42] T. Law, H. Itoh, and H. Seki, “Image filtering, edge detection, and edge tracing using fuzzy reasoning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 5, pp. 481–491, 1996.

[43] M. Gonzalez-Hidalgo, A. M. Torres, and J. T. Sastre, “Noisy image edge detection using an uninorm fuzzy morphological gradient,” in *Ninth International Conference on Intelligent Systems Design and Applications, ISDA*, pp. 1335–1340, 2009.

[44] W. Li, C. Wang, Q. Wang, and G. Chen, “An edge detection method based on optimized bp neural network,” in *International Symposium on Information Science and Engieering, ISISE*, pp. 40–44, 2008.

[45] H. Mehrara, M. Zahedinejad, and A. Pourmohammad, “Novel edge detection using bp neural network based on threshold binarization,” in *Second International Conference on Computer and Electrical Engineering, ICCEE*, pp. 408–412, 2009.

[46] P. Terry and D. Vu, "Edge detection using neural networks," in *The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, pp. 391–395, 1993.

[47] S. Bhandarkar, Y. Zhang, and W. Potter, "A genetic algorithm-based edge detection technique," in *Proceedings of 1993 International Joint Conference on Neural Networks, IJCNN*, pp. 2995–2999, 1993.

[48] W. Cui, Z. Guan, and Z. Zhang, "An improved region growing algorithm for image segmentation," in *International Conference on Computer Science and Software Engineering*, vol. 6, pp. 93–96, 2008.

[49] D. Kelkar and S. Gupta, "Improved quadtree method for split merge image segmentation," in *First International Conference on Emerging Trends in Engineering and Technology*, pp. 44–47, 2008.

[50] H. Malki, "Image segmentation using multilayer neural network," in *International Joint Conference on Neural Networks, IJCNN*, pp. 354–360, 1992.

[51] A. Visa, "A genetic algorithm based method to improve image segmentation," in *Fourteenth International Conference on Pattern Recognition*, pp. 1015–1017, 1998.

[52] S. Bhandarkar and H. Zhang, "Image segmentation using evolutionary computation," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 1, pp. 1–21, 1999.

[53] D. Cohen, "On active contour models and balloons," in *CVGIP: Image Understanding*, 1991.

[54] S. Rahnamayan, H. Tizhoosh, and M. Salama, "Automated snake initialization for the segmentation of the prostate in ultrasound images," in *Image Analysis and Recognition* (M. Kamel and A. Campilho, eds.), vol. 3656 of *Lecture Notes in Computer Science*, pp. 930–937, Springer Berlin Heidelberg.

[55] T. McInerney and D. Terzopoulos, "T-snakes: Topology adaptive snakes," *Medical Image Analysis*, vol. 4, no. 2, pp. 73–91, 2000.

[56] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations," *Journal of Computational Physics*, vol. 79, no. 1, pp. 12–49, 1988.

[57] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw, "Two-way coupled sph and particle level set fluid simulation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 4, pp. 797–804, 2008.

[58] A. Brodersen, K. Museth, S. Porumbescu, and B. Budge, "Geometric texturing using level sets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 277–288, 2008.

[59] Y. S. Kim, J. K. Byun, and I. H. Park, "A level set method for shape optimization of electromagnetic systems," *IEEE Transactions on Magnetics*, vol. 45, no. 3, pp. 1466–1469, 2009.

[60] A. Ghodrati, F. Calderero, D. Brooks, G. Tadmor, and R. MacLeod, "A level sets algorithm for the inverse problem of electrocardiography," in *Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1590–1594, 2004.

[61] V. Caselles, F. Catté, T. Coll, and F. Dibos, "A geometric model for active contours in image processing," *Numerische Mathematik*, vol. 66, pp. 1–31, 1993.

[62] R. Malladi, J. Sethian, and B. Vemuri, *Evolutionary fronts for topology-independent shape modelling and recovery*, vol. 800 of *Lecture Notes in Computer Science*, pp. 1–13. Springer Berlin / Heidelberg, 1994.

[63] T. Chan and L. Vese, "Active contours without edges," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 266–277, 2001.

[64] J. Yezzi, A., A. Tsai, and A. Willsky, "A statistical approach to snakes for bimodal and trimodal imagery," in *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 898–903, 1999.

[65] A. Tsai, J. Yezzi, A., and A. Willsky, “Curve evolution implementation of the mumford-shah functional for image segmentation, denoising, interpolation, and magnification,” *IEEE Transactions on Image Processing*, vol. 10, no. 8, pp. 1169–1186, 2001.

[66] D. Mumford and J. Shah, “Optimal approximations by piecewise smooth functions and associated variational problems,” *Communications on Pure and Applied Mathematics*, vol. 42, no. 5, pp. 577–685, 1989.

[67] L. A. Vese and T. F. Chan, “A multiphase level set framework for image segmentation using the mumford and shah model,” *International Journal of Computer Vision*, vol. 50, no. 3, pp. 271–293, 2002.

[68] J. B. Roerdink and A. Meijster, “The watershed transform: Definitions, algorithms and parallelization strategies,” *Fundam. Inf.*, vol. 41, no. 1,2, pp. 187–228, 2000.

[69] J. Xuan, T. Adali, and Y. Wang, “Segmentatin of magnetic resonance brain image: Integrating region growing and edge detection,” in *International Conference on Image Processing*, pp. 544–547, 1995.

[70] Y.-W. Yu and J.-H. Wang, “Image segmentation on region growing and edge detection,” in *IEEE International Conference on Systems, Man, and Cybernetics, IEEE SMC ’99*, pp. 798–803, 1999.

[71] J. L. Moigne and J. Tilton, “Refining image segmentation by integration of edge and region data,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 33, no. 3, pp. 605–615, 1995.

[72] M. Salah, A. Mitiche, and I. Ayed, “Multiregion image segmentation by parametric kernel graph cuts,” *IEEE Transactions on Image Processing*, vol. 20, no. 2, pp. 545–557, 2011.

[73] D. Park, D. Ramanan, and C. Fowlkes, “Multiresolution models for object detection,” in *Computer Vision ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios,

eds.), vol. 6314 of *Lecture Notes in Computer Science*, pp. 241–254, Springer Berlin Heidelberg, 2010.

[74] P. Burt and E. Adelson, "The laplacian pyramid as a compact image code," *IEEE Transactions on Communications*, vol. 31, pp. 532–540, apr 1983.

[75] S. P. Raja and A. Suruliandi, "Analysis of efficient wavelet based image compression techniques," in *International Conference on Computing Communication and Networking Technologies*, pp. 1–6, 2010.

[76] C. E. Jacobs, A. Finkelstein, and D. H. Salesin, "Fast multiresolution image querying," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pp. 277–286, ACM, 1995.

[77] M. Rezaee, P. M. J. Van der Zwet, B. P. F. Lelieveldt, R. van der Geest, and J. Reiber, "A multiresolution image segmentation technique based on pyramidal segmentation and fuzzy clustering," *IEEE Transactions on Image Processing*, vol. 9, no. 7, pp. 1238–1248, 2000.

[78] M. Saeed, W. Karl, T. Nguyen, and H. Rabiee, "A new multiresolution algorithm for image segmentation," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 5, pp. 2753–2756, 1998.

[79] J. Yang and T. Huang, *Image super-resolution: historical overview and future challenges.* from the book: Super-Resolution Imaging (edited by Peyman Milanfar), CRC Press (Taylor & and amp and Francis Group), 2011.

[80] R. Marfil, L. Molina-Tanco, A. Bandera, J. Rodrguez, and F. Sandoval, "Pyramid segmentation algorithms revisited," *Pattern Recognition*, vol. 39, no. 8, pp. 1430–1451, 2006.

[81] P. Meer, "Stochastic image pyramids," *Computer Vision, Graphics, and Image Processing*, vol. 45, no. 3, pp. 269–294, 1989.

[82] J. Jolion and A. Montanvert, "The adaptive pyramid: A framework for 2d image analysis," *CVGIP: Image Understanding*, vol. 55, no. 3, pp. 339–348, 1992.

[83] A. P. Witkin, "Scale-space filtering," in *Proceedings of the Eighth international joint conference on Artificial intelligence*, pp. 1019–1022, 1983.

[84] T. Lindeberg, *Scale-Space Theory in Computer Vision.* Kluwer international series in engineering and computer science, Springer, 1993.

[85] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, 1990.

[86] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674–693, 1989.

[87] J.-L. Starck and J. Bobin, "Astronomical data analysis and sparsity: From wavelets to compressed sensing," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1021–1030, 2010.

[88] W. Huang and M. Solorzano, "Wavelet preprocessing of acoustic signals," in *Conference Record of the Twenty-Fifth Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1114–1118, 1991.

[89] Y. Li, H. Szu, Y. Sheng, and H. Caulfield, "Wavelet processing and optics," *Proceedings of the IEEE*, vol. 84, no. 5, pp. 720–732, 1996.

[90] X. Huang, A. Madoc, and M. Wagner, "Noises removal for images by wavelet-based bayesian estimator via levy process analysis," in *IEEE International Conference on Multimedia and Expo*, vol. 1, pp. 327–330, 2004.

[91] Y. Hao, L. Changshun, and P. Lei, "An improved method of image edge detection based on wavelet transform," in *IEEE International Conference on Computer Science and Automation Engineering*, vol. 3, pp. 678–681, 2011.

[92] Y. Lu and M. N. Do, "Crisp-contourlet: a critically sampled directional multiresolution representation," in *Proc. SPIE Conf. on Wavelets X*, pp. 655–665, 2003.

[93] R. Gaetano, G. Scarpa, and G. Poggi, "A hierarchical segmentation algorithm for multiresolution satellite images," in *IEEE International Geoscience and Remote Sensing Symposium*, pp. 1885–1888, 2007.

[94] R. Gaetano, G. Scarpa, and G. Poggi, "Hierarchical texture-based segmentation of multiresolution remote-sensing images," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 47, no. 7, pp. 2129–2141, 2009.

[95] L. Qingsheng and L. Guoying, "Multi-resolution markov random field model with variable potentials in wavelet domain for texture image segmentation," in *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, vol. 9, pp. 342–346, 2010.

[96] G. Liu and G. Tao, "Multiresolution algorithm for image segmentation using mrmrf with edge information," in *Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on*, pp. 1–4, 2010.

[97] D. Boukerroui, O. Basset, N. Gurin, and A. Baskurt, "Multiresolution texture based adaptive clustering algorithm for breast lesion segmentation," *European Journal of Ultrasound*, vol. 8, no. 2, pp. 135–144, 1998.

[98] P. Yan, S. Xu, B. Turkbey, and J. Kruecker, "Discrete deformable model guided by partial active shape model for trus image segmentation," *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 5, pp. 1158–1166, 2010.

[99] Y. Akgul and C. Kambhamettu, "A coarse-to-fine deformable contour optimization framework," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 174–186, 2003.

[100] J. Dehmeshki, M. Siddique, W. Wong, and I. Ster, "Multiresolution active contour model applied on non lung and colon images," in *SPIE International Symposium on Medical Imaging*, vol. 5370, pp. 1685–1694, 2004.

[101] S. Yoon, C. Lee, J. Kim, and M. Lee, "Wavelet-based multi-resolution deformation for medical endoscopic image segmentation," *Journal of Medical Systems*, vol. 32, pp. 207–214, 2008.

[102] N. Lin, W. Yu, and J. S. Duncan, "Combinative multi-scale level set framework for echocardiographic image segmentation," in *Proceedings of the 5th International Conference on Medical Image Computing and Computer-Assisted Intervention-Part I*, MICCAI, pp. 682–689, 2002.

[103] N. Lin, W. Yu, and J. S. Duncan, "Combinative multi-scale level set framework for echocardiographic image segmentation," *Medical Image Analysis*, vol. 7, no. 4, pp. 529–537, 2003.

[104] J. M. Hao Shan, "Curvelet-based geodesic snakes for image segmentation with multiple objects," *Pattern Recognition Letters*, vol. 31, no. 5, pp. 355–360, 2010.

[105] V. Roullier, V. T. Ta, O. Lezoray, and A. Elmoataz, "Graph-based multi-resolution segmentation of histological whole slide images," in *IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pp. 153–156, 2010.

[106] C. Bouman and B. Liu, "Multiple resolution segmentation of textured images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 2, pp. 99–113, 1991.

[107] X. Munoz, J. Marti, X. Cufi, and J. Freixenet, "Unsupervised active regions for multiresolution image segmentation," in *16th International Conference on Pattern Recognition*, vol. 2, pp. 905–908, 2002.

[108] A. Rosenfeld, *Multiresolution Image Processing and Analysis.* Springer Series in Information Sciences, Springer London, Limited, 1984.

[109] T. Lindeberg and J. Garding, "Shape from texture from a multi-scale perspective," in *Fourth International Conference on Computer Vision*, pp. 683–691, 1993.

[110] T. Lindeberg, “Feature detection with automatic scale selection,” *Int. J. Comput. Vision*, vol. 30, no. 2, pp. 79–116, 1998.

[111] T. Lindeberg, “Junction detection with automatic selection of detection scales and localization scales,” in *IEEE International Conference Image Processing*, pp. 924–928, 1994.

[112] T. Lindeberg, “Edge detection and ridge detection with automatic scale selection,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 465–470, 1996.

[113] H. Jeong and C. Kim, “Adaptive determination of filter scales for edge detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 5, pp. 579–585, 1992.

[114] J. Elder and S. Zucker, “Local scale control for edge detection and blur estimation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 7, pp. 699–716, 1998.

[115] M. Jagersand, “Saliency maps and attention selection in scale and spatial coordinates: an information theoretic approach,” in *Fifth International Conference on Computer Vision*, pp. 195–202, 1995.

[116] A. Renyi, *Some fundamental questions of information theory*, pp. 526–552. Budapest, Hungary: Akademiai Kiado, 1976.

[117] J. Sporring and J. Weickert, “Information measures in scale-spaces,” *IEEE Transactions on Information Theory*, vol. 45, no. 3, pp. 1051–1058, 1999.

[118] E. Hadjidemetriou, M. D. Grossberg, and S. K. Nayar, “Resolution selection using generalized entropies of multiresolution histograms,” in *Proceedings of the 7th European Conference on Computer Vision-Part I*, pp. 220–235, 2002.

[119] M. Tanaka, T. Watanabe, and T. Mishima, “Tsallis entropy in scale spaces,” in *Proc. SPIE*, vol. 3811, pp. 273–283, 1999.

[120] T. Kadir and M. Brady, "Saliency, scale and image description," *Int. J. Comput. Vision*, vol. 45, pp. 83–105, Nov. 2001.

[121] H. Mirzaalian and G. Hamarneh, "Vessel scale-selection using mrf optimization," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3273–3279, 2010.

[122] D. Marimont and Y. Rubner, "A probabilistic framework for edge detection and scale selection," in *Sixth International Conference on Computer Vision*, pp. 207–214, 1998.

[123] K. S. Pedersen, M. Loog, and B. Markussen, "Generic maximum likely scale selection," in *Proceedings of the 1st international conference on Scale space and variational methods in computer vision*, pp. 362–373, 2007.

[124] K. Pedersen, "Properties of brownian image models in scale-space," in *Scale Space Methods in Computer Vision* (L. Griffin and M. Lillholm, eds.), vol. 2695 of *Lecture Notes in Computer Science*, pp. 281–296, Springer Berlin Heidelberg, 2003.

[125] G. Gomez, J. L. Marroquin, and L. Sucar, "Probabilistic estimation of local scale," in *15th International Conference on Pattern Recognition*, vol. 3, pp. 790–793, 2000.

[126] E. Bayram, C. L. Wyatt, and Y. Ge, "Automatic scale selection for medical image segmentation," in *Proc. SPIE*, vol. 4322, pp. 1399–1410, 2001.

[127] J. Piovano and T. Papadopoulo, "Local statistic based region segmentation with automatic scale selection," in *Computer Vision* (D. Forsyth, P. Torr, and A. Zisserman, eds.), vol. 5303 of *Lecture Notes in Computer Science*, pp. 486–499, Springer Berlin Heidelberg, 2008.

[128] Y. Li, D. M. Tax, and M. Loog, "Scale selection for supervised image segmentation," *Image and Vision Computing*, vol. 30, no. 12, pp. 991–1003, 2012.

[129] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[130] S. Chen, H. He, and E. Garcia, "Ramoboost: Ranked minority oversampling in boosting," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1624–1642, 2010.

[131] G. Beliakov, T. Calvo, and S. James, "Aggregation of preferences in recommender systems," in *Recommender Systems Handbook* (F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, eds.), pp. 705–734, Springer US, 2011.

[132] P. J. Walsh and W. Wheeler, "Water quality index aggregation and cost benefit analysis," NCEE Working Paper Series 201205, National Center for Environmental Economics, U.S. Environmental Protection Agency, 2012.

[133] M. J. Scott and E. K. Antonsson, "Aggregation functions for engineering design trade-offs," *Fuzzy Sets and Systems*, vol. 99, no. 3, pp. 253–264, 1998.

[134] I. Fogel and D. Sagi, "Gabor filters as texture discriminator," *Biological Cybernetics*, vol. 61, no. 2, pp. 103–113, 1989.

[135] L. Vincent, "Fast granulometric methods for the extraction of global image information," *The proceedings of PRASA*, pp. 119–140, 2000.

[136] G. Zhao and M. Pietikäinen, "Dynamic texture recognition using local binary patterns with an application to facial expressions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 915–928, 2007.

[137] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face recognition with local binary patterns," in *8th European Conference on Computer Vision*, pp. 469–481, 2004.

[138] T. Ojala, M. Pietikäinen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[139] T. Ahonen, J. Matas, C. He, and M. Pietikäinen, "Rotation invariant image description with local binary pattern histogram fourier features," in *Proceedings of the 16th Scandinavian Conference on Image Analysis*, pp. 61–70, 2009.

[140] T. Menp and M. Pietikinen, "Multi-scale binary patterns for texture analysis," in *Image Analysis* (J. Bigun and T. Gustavsson, eds.), vol. 2749 of *Lecture Notes in Computer Science*, pp. 885–892, Springer Berlin Heidelberg, 2003.

[141] S. Theodoridis and K. Koutroumbas, *Pattern Recognition.* Academic Press, 4th ed., 2008.

[142] P. Correia and F. Pereira, "Objective evaluation of video segmentation quality," *IEEE Transactions on Image Processing*, vol. 12, no. 2, pp. 186–200, 2003.

[143] P. Radivojac, N. V. Chawla, A. K. Dunker, and Z. Obradovic, "Classification and knowledge discovery in protein databases," *Journal of Biomedical Informatics*, vol. 37, no. 4, pp. 224–239, 2004.

[144] D. D. Lewis and J. Catlett, "Heterogeneous Uncertainty Sampling for Supervised Learning," in *In Proceedings of the Eleventh International Conference on Machine Learning*, pp. 148–156, 1994.

[145] Y. Liu, E. Shriberg, A. Stolcke, and M. P. Harper, "Using machine learning to cope with imbalanced classes in natural speech: evidence from sentence boundary and disfluency detection," in *INTERSPEECH*, 2004.

[146] V. Engen, J. Vincent, and K. Phalp, "Enhancing network based intrusion detection for imbalanced data," *Int. J. Know.-Based Intell. Eng. Syst.*, vol. 12, no. 5,6, pp. 357–367, 2008.

[147] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Mach. Learn.*, vol. 30, no. 2-3, pp. 195–215, 1998.

[148] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[149] H. He and E. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[150] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, “Handling imbalanced datasets: A review,” *GESTS International Transactions on Computer Science and Engineering*, 2006.

[151] N. Japkowicz, “The class imbalance problem: Significance and strategies,” in *Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI*, pp. 111–117, 2000.

[152] X.-Y. Liu, J. Wu, and Z.-H. Zhou, “Exploratory undersampling for class-imbalance learning,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 39, no. 2, pp. 539–550, 2009.

[153] H. He, Y. Bai, E. Garcia, and S. Li, “Adasyn: Adaptive synthetic sampling approach for imbalanced learning,” in *IEEE International Joint Conference on Neural Networks, IJCNN 2008*, pp. 1322–1328, 2008.

[154] X.-Y. Liu and Z.-H. Zhou, “The influence of class imbalance on cost-sensitive learning: An empirical study,” in *Sixth International Conference on Data Mining*, pp. 970–974, 2006.

[155] S. Ertekin, J. Huang, and C. L. Giles, “Active learning for class imbalance problem,” in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 823–824, 2007.

[156] G. Wu and E. Chang, “Class-Boundary Alignment for Imbalanced Dataset Learning,” in *Workshop on Learning from Imbalanced Datasets in ICML*, pp. 49–56, 2003.

[157] G. Wu and E. Chang, “Aligning boundary in kernel space for learning imbalanced dataset,” in *IEEE International Conference on Data Mining*, pp. 265–272, 2004.

[158] G. Wu and E. Chang, “Kba: kernel boundary alignment considering imbalanced data distribution,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 786–795, 2005.

[159] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[160] Y. Freund and R. E. Schapire, "Experiments with a New Boosting Algorithm," in *International Conference on Machine Learning*, pp. 148–156, 1996.

[161] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.

[162] H. Guo and H. L. Viktor, "Learning from imbalanced data sets with boosting and data generation: the databoost-im approach," *SIGKDD Explor. Newsl.*, vol. 6, no. 1, pp. 30–39, 2004.

[163] N. Chawla, A. Lazarevic, L. Hall, and K. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *Knowledge Discovery in Databases: PKDD* (N. Lavra, D. Gamberger, L. Todorovski, and H. Blockeel, eds.), vol. 2838 of *Lecture Notes in Computer Science*, pp. 107–119, Springer Berlin Heidelberg, 2003.

[164] Y. Freund and R. Schapire, "A short introduction to boosting," *Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771–780, 1999.

[165] R. E. Schapire and Y. Freund, *Boosting: Foundations and Algorithms (Adaptive Computation and Machine Learning series).* The MIT Press, 2012.

[166] J. Zhu, S. Rosset, H. Zou, and T. Hastie, "Multi-class AdaBoost," tech. rep., 2006.

[167] P. Natesan, P. Balasubramanie, and G. Gowrison, "Performance Comparison of AdaBoost Based Weak Classifiers in Network Intrusion Detection," *Journal of Information Systems and Communication*, vol. 3, no. 1, pp. 295–299, 2012.

[168] N. Japkowicz and M. Shah, *Evaluating Learning Algorithms: A Classification Perspective.* New York, NY, USA: Cambridge University Press, 2011.

[169] Y. Sun, M. Kamel, and Y. Wang, "Boosting for learning multiple classes with imbalanced class distribution," in *Sixth International Conference on Data Mining, ICDM*, pp. 592–602, 2006.

[170] D. J. Hand and R. J. Till, "A simple generalisation of the area under the roc curve for multiple class classification problems," *Machine Learning*, vol. 45, no. 2, pp. 171–186, 2001.

[171] T. Fawcett, "An introduction to roc analysis," *Pattern Recogn. Lett.*, vol. 27, no. 8, pp. 861–874, 2006.

[172] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen, "Interactively co-segmentating topically related images with intelligent scribble guidance," *Int. J. Comput. Vision*, vol. 93, no. 3, pp. 273–292, 2011.

[173] Japanese Society of Radiological Technology, "Lung X-Ray Dataset." available at: http://www.jsrt.or.jp/jsrt-db/eng.php.

[174] K. Zuiderveld, *Contrast limited adaptive histogram equalization*, pp. 474–485. San Diego, CA, USA: Academic Press Professional, Inc., 1994.

[175] I. B. Ayed, "Parametric Kernel Graph Cuts Code." available at: http://http://www.mathworks.com/matlabcentral/fileexchange/38555-kernel-graph-cut-image-segmentation.

[176] S. Boltz, "Statistical Region Merging Code." available at: http://www.mathworks.com/matlabcentral/fileexchange/25619-image-segmentation-using-statistical-region-merging.

[177] L. R. Dice, "Measures of the amount of ecologic association between species," *Ecology*, vol. 26, no. 3, pp. 297–302, 1945.

[178] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 1–27, 2011.

[179] C.-C. Chang and C.-J. Lin, "LIBSVM - A Library for Support Vector Machines." available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[180] H. H. Yang and J. Moody, “Data visualization and feature selection: New algorithms for nongaussian data,” in *Advances in Neural Information Processing Systems*, pp. 687–693, MIT Press, 1999.

[181] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” in *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144–152, 1992.

[182] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[183] S. Ben-Yacoub, Y. Abdeljaoued, and E. Mayoraz, “Fusion of face and speech data for person identity verification,” *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1065–1074, 1999.

[184] H. Drucker, S. Wu, and V. Vapnik, “Support vector machines for spam categorization,” *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1048–1054, 1999.

[185] I. N. Flaounas, D. K. Iakovidis, and D. E. Maroulis, “Cascading svms as a tool for medical diagnosis using multi-class gene expression data,” *International Journal on Artificial Intelligence Tools*, 2006.

[186] Y. Freund and R. E. Schapire, “Experiments with a New Boosting Algorithm,” in *International Conference on Machine Learning*, pp. 148–156, 1996.

[187] G. Brown, A. Pocock, M.-J. Zhao, and M. Luján, “Conditional likelihood maximisation: A unifying framework for information theoretic feature selection,” *J. Mach. Learn. Res.*, vol. 13, pp. 27–66, 2012.

[188] F. Tab, G. Naghdy, and A. Mertins, “Scalable multiresolution color image segmentation with smoothness constraint,” in *IEEE International Conference on Electro Information Technology*, pp. 1670–1687, 2005.

[189] J. Wang, J. Li, R. Gray, and G. Wiederhold, "Unsupervised multiresolution segmentation for images with low depth of field," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 1, pp. 85–90, 2001.

[190] G. Liu and G. Tao, "Multiresolution algorithm for image segmentation using mrmrf with edge information," in *6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*, pp. 1–4, 2010.

[191] R. Gaetano, G. Scarpa, and G. Poggi, "Hierarchical texture-based segmentation of multiresolution remote-sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 7, pp. 2129–2141, 2009.

[192] K. Okada, D. Comaniciu, and A. Krishnan, "Scale selection for anisotropic scale-space: application to volumetric tumor characterization," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. I–594–I–601, 2004.

[193] S. Banerji, A. Verma, and C. Liu, "Lbp and color descriptors for image classification," in *Cross Disciplinary Biometric Systems*, vol. 37 of *Intelligent Systems Reference Library*, Springer Berlin Heidelberg, 2012.

[194] "The support vector machine under test," *Neurocomputing*, vol. 55, no. 12, pp. 169–186, 2003.

[195] X. Hong, S. Chen, and C. Harris, "A kernel-based two-class classifier for imbalanced data sets," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 28–41, 2007.

[196] M. Shahriar, T. Ahsan, and U. Chong, "Fault diagnosis of induction motors utilizing local binary pattern-based texture analysis," *EURASIP Journal on Image and Video Processing*, vol. 2013, no. 1, pp. 1–11, 2013.

APPENDICES

Appendix A

Algorithms

A.1 Support Vector Machines

Support Vector Machines (SVM) [181] is a well-known learning method in machine learning. It was originally introduced for binary classification problems, but was later extended for multi-class problems. The objective in SVM is to search for the optimal hyperplane that separates two classes, a goal that is achieved by finding parameters that maximize the marginal distance between the classes. Since its introduction, SVM has become a popular choice in many fields, such as optical character recognition [182], personal identification [183], spam filtering and categorization [184], and medical imaging [185].

For a binary classification problem, let x_i be the feature vectors (input) of the n instances training set X, and y_i be the targets, where i = 1,2,$\ldots$, n. The hyperplane separating two classes linearly can be defined as

$$w^T x + b = 0, \tag{A.1}$$

where w is a weight vector and b is a bias. In this way, we have

$$\begin{aligned} w^T x + b &\geq 0 \quad for\, y_i = +1, \\ w^T x + b &< 0 \quad for\, y_i = -1. \end{aligned} \tag{A.2}$$

The separating hyperplane defined in Eq. A.1 is not unique. Different hyperplanes can be defined with different weights w and biases b. The objective of SVM is to find the optimal hyperplane with the maximum separation between classes by searching for optimal values of w and b. Separation can be represented by a margin, which is the distance between the hyperplane and the closest data point of either class (support vectors). This margin is defined by

$$m = \frac{1}{||w||}, \tag{A.3}$$

The margin of separation between the two classes is therefore

$$\frac{2}{||w||}. \tag{A.4}$$

Maximizing the margin is equivalent to minimizing the Euclidean norm of w. The optimization problem can thus be stated as follows:

$$\begin{aligned} &\text{minimize} \quad && \frac{1}{2}||w||^2 \\ &\text{subject to} \quad && y_i \left(w^T x_i + b \right) \geq 1, i = 1, 2, \ldots, n. \end{aligned} \tag{A.5}$$

This solution is applicable only to strictly separable classes. For nonseparable classes, the problem must be relaxed, which can be accomplished with slack variables ξ_i. The optimization problem will then be

$$\begin{aligned} &\text{minimize} \quad && \frac{1}{2}||w||^2 + C\Sigma_{i=1}^{n}\xi_i \\ &\text{subject to} \quad && y_i \left(w^T x_i + b \right) \geq 1 - \xi_i, \xi_i \geq 0, \end{aligned} \tag{A.6}$$

where $C > 0$ is a constant for controlling the relative influence of the two terms: maximizing the margin and minimizing the amount of slack. Figure A.1 illustrates the cases of linear and non-linear separable classes.

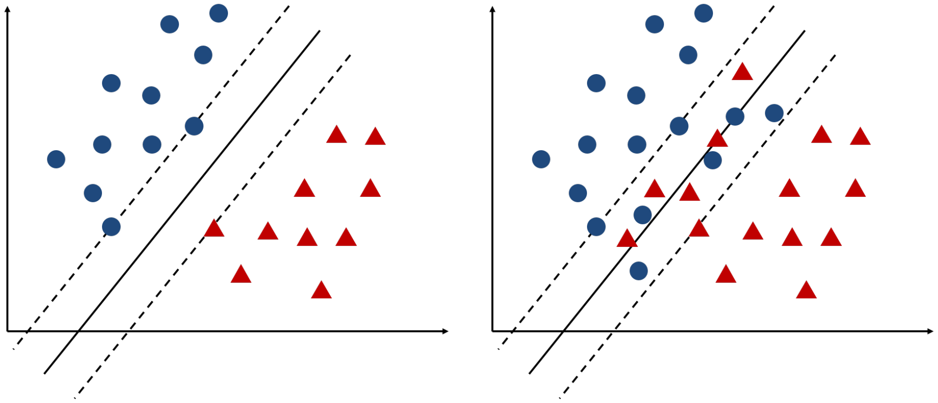


Figure A.1: Left: linear separable classes; right: linear non-separable classes.

An interesting aspect of SVM is the ability to linearly solve a non-linear problem by transforming the non-linear problem into higher dimensional space, and then finding the optimal linear hyperplane there. The linear hyperplane in the transformed space is equivalent to a non-linear one in the input space as shown in Figure A.2. Computing the dot product in very high-dimension space can be prohibitively expensive. Fortunately, this computation can be performed implicitly from the input space through the use of kernels. For the non-linear mapping function $\phi(.)$, kernel K is therefore defined as

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j). \tag{A.7}$$

There are many types of kernels; Table A.1 lists the most common three.

Up to this point, this discussion has referred to binary classification problems. SVM can also be extended for multiclass problems: the most popular methods are one versus all and one versus one [141]. For an N class problem, the one-versus-all strategy constructs N classifiers, whereby each class is discriminated from the rest of the classes.

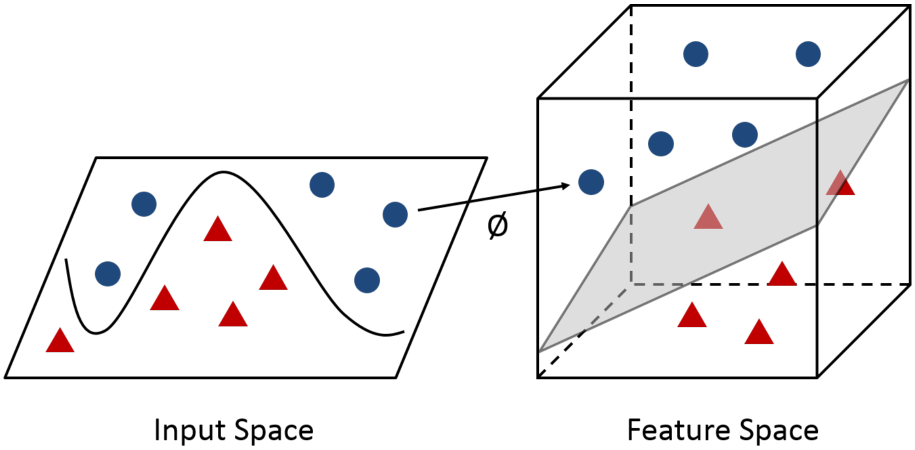


Figure A.2: Mapping from input space to higher dimensional space.

Table A.1: Common kernel types for SVM

Kernel	Equation $K(x_i, x_j)$
Polynomial	$\left(x_i^T x_j + 1\right)^p$, $p > 0$
Gaussian	$exp\left(-\frac{\lvert\lvert x_i - x_j \rvert\rvert^2}{\sigma^2}\right)$
Sigmoid	$tanh\left(k x_i^T x_j - \delta\right)$

Classification is performed by selecting the classifier that provides the maximum separability. In the one-versus-one approach, $N(N-1)/2$ binary classifiers are constructed, whereby each classifier discriminates between two pairs of classes. The classification is performed via majority vote.

A.2 Adaptive Synthetic Sampling (ADASYN)

The Adaptive Synthetic Sampling (ADASYN) method, proposed by He et al. [153], is an adaptive oversampling approach. Instances of minority classes are synthetically generated based on their distributions. In this way, more instances are generated for difficult-to-learn minority class instances than for those that are easier to learn. ADASYN can be straightforwardly extended for multiclass problems, as the authors described. The ADASYN method is described in Algorithm 5.

A.3 AdaBoost

The Adaptive Boosting (AdaBoost) method used in this research is described in Algorithm 6. This version of AdaBoost is called AdaBoost.M2 [186] and is designed for multi-class classification problems. The usual strict assumption in AdaBoost for the base learning method is to have an error less than 50%. This assumption is relaxed in AdaBoost.M2.

A.4 Rank Minority Oversampling in Boosting (RAMOBoost)

RAMOBoost [130] is a variant of the AdaBoost learning algorithm that has been designed specifically for learning from imbalanced data. The power of RAMOBoost stems from its two combined components: boosting and sampling. Boosting trains several base classifiers consecutively while adaptively adjusting the weights of the training instances. In

Algorithm 5 Adaptive Synthetic Sampling (ADASYN)

```
Inputs
$m_l$: number of majority class instances
$m_s$: number of minority class instances
$\beta \in [0,1]$: specify balancing level, where $\beta = 1$ creates fully balanced dataset
$d_{thresh}$: threshold of the maximum tolerated ratio of class imbalance
$K$: number of nearest neighbours in the K-nearest neighbours algorithm

$d = m_s/m_l$ [degree of class imbalance]
if $d < d_{thresh}$ then
  $G = (m_l - m_s) \times \beta$ [number of synthetic data to be generated for minority class]
  for all $x_i \in$ minority class do
    $knn_i = findKNN(x_i)$ [find K-nearest neighbors of $x_i$]
    $\Delta_i = numMajorityInstances(knn_i)$ [number of majority class instances in $knn_i$]
    $r_i = \Delta_i/K, i = 1, 2, \ldots, m_s$
    $\hat{r}_i = r_i / \sum_{i=1}^{m_s} r_i$ [normalize $r_i$]
    $g_i = \hat{r}_i \times G$ [number of synthetic instances to generate for minority instance $x_i$]
  end for
  for all $x_i \in$ minority class do
    for 1 to $g_i$ do
      $x_{zi} = rand(knn_i, minority)$ [randomly choose a minority class instance from $knn$
      of $x_i$]
      $s_i = x_i + (x_{zi} - x_i) \times \lambda$ [generate synthetic instance, $s_i$]
      [$\lambda \in [0,1]$ is a randomly generated number]
    end for
  end for
end if
```

this manner, the decision boundary is adaptively shifted during each boosting iteration so that the focus is on instances that are difficult to learn. In RAMOBoost the sampling is based on adaptive adjustment to the sampling weights of minority class samples according to their distribution. Greater emphasis is thus placed on rare examples that are inherently difficult to learn.

Algorithm 6 AdaBoost

```
Inputs
$m$ Samples, $(x_i, y_i)$, $i = 1, \ldots, m$, with $y_i \in Y = 1, \ldots, k$
$BaseLearn$: the base learning algorithm
$B = (i, y) : i \in 1, \ldots, m, y \neq y_i$
$T$: number of iterations

Initialize: $D_1(i, y) = 1/|B|$ for $(i, y) \in B$
for t = 1,2,...,T do
    call $BaseLearn$, with mislabel distribution $D_t$
    obtain hypothesis $h_t$: $X \times Y \to [0, 1]$
    $\epsilon_t = \frac{1}{2} \sum_{(i,y) \in B} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$ [calculate the pseudo-loss of $h_t$]
    $\beta_t = \epsilon_t/(1 - \epsilon_t)$
    update the sampling distribution $D_t$:
    $D_t$: $D_{t+1}(i, y) = \frac{D_t(i,y)}{Z_t} \cdot \beta_t^{(1+h_t(x_i,y_i)-h_t(x_i,y))}$, where $Z_t$ is a normalization constant
end for
Output
$h_{final}(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} (log \frac{1}{\beta_t}) h_t(x, y)$ [final hypothesis]
```

The RAMOBoost technique is presented in Algorithm 7, which reveals that it is a direct extension of the AdaBoost.M2 algorithm (see Algorithm 6), making it inherently applicable for multi-class problems.

A.5 Joint Mutual Information (JMI)

Joint Mutual Information (JMI) [180] is a well-known method for selecting the most relevant features from features sets. Given random variables X and Y representing features and targets, their mutual information is calculated by

$$I(X, Y) = \sum_x \sum_y P(X, Y) log_2 \frac{P(X, Y)}{P(X)P(Y)}, \tag{A.8}$$

where $P(.)$ is the probability density function.

Algorithm 7 RAMOBoost

```
**Inputs**
$m$ Samples, $(x_i, y_i)$, $i = 1, \ldots, m$, with $y_i \in Y = 1, \ldots, k$
$BaseLearn$: the base learning algorithm
$B = (i, y) : i \in 1, \ldots, m$, $y \neq y_i$
$T$: number of iterations
$N$: number of synthetic data samples to generate at each iteration
$k_1$: number of nearest neighbours for adjusting the sampling probability of the minority samples
$k_2$: number of nearest neighbours used to generate the synthetic data instances
$\alpha$: scaling coefficient

**Initialize:** $D_1(i, y) = 1/|B|$ for $(i, y) \in B$
**for** t=1,2,...,T **do**
  $S_e$: mislabeled training data sampled with $D_t$
  $e_1$: majority class samples in $S_e$ (with size $m_{lt}$)
  $e_2$: minority class samples in $S_e$ (with size $m_{st}$)
  **for** each sample $x_i \in e_2$ **do**
    find its $k_1$ nearest neighbours in $S_e$
    $r_i = \frac{1}{1+exp(-\alpha \cdot \delta_i)}$, $i = 1, \ldots, m_{st}$ [$\delta_i$: number of majority classes in $k1$ examples]
  **end for**
  $\hat{r_i} = \frac{r_i}{\sum^{m_{st}}}_{i=1} r_i$, [normalize $r_i$]
  $d_t = r_i$
  $g_t = e_2$ sampled with $d_t$
  **for** each sample $x_i \in g_t$ **do**
    find its $k_2$ nearest neighbours in $S_e$
    generate $N$ synthetic data samples using linear interpolation
  **end for**
  call $BaseLearn$, with the sampled dataset $S_e$ and the $N$ synthetic data samples
  obtain hypothesis $h_t$: $X \times Y \rightarrow [0, 1]$
  $\epsilon_t = \frac{1}{2} \sum_{(i,y) \in B} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$ [calculate the pseudo-loss of $h_t$]
  $\beta_t = \epsilon_t/(1 - \epsilon_t)$
  update the sampling distribution $D_t$:
  $D_{t+1}(i, y) = \frac{D_t(i,y)}{Z_t} \cdot \beta_t^{(1+h_t(x_i,y_i)-h_t(x_i,y))}$, where $Z_t$ is a normalization constant
**end for**
**Output**
$h_{final}(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} (log \frac{1}{\beta_t}) h_t(x, y)$ [final hypothesis]
```

The JMI of feature X_i is calculated by [187]

$$JMI(X_i) = \sum_{X_j \in S} I(X_i X_j; Y), \tag{A.9}$$

where S is the set of currently selected features.

Using JMI for feature selection returns a subset of features X that are the most relevant to targets Y and at the same time reduce the redundancy among the selected features.

Appendix B

Trade-off Measure Alternatives

Several methods can be used to measure the trade-off between two values. The measure adopted in this research is the weighted geometric mean, defined in Eq. 5.1. In this Appendix, two other measures are discussed, namely, weighted arithmetic mean and weighted division of accuracy and time.

Weighted arithmetic mean is a well-known measure used as aggregation function, and is usually compared with the weighted geometric mean [132]. For accuracy A and time T, weighted arithmetic mean is defined as

$$\omega = A \times \alpha + (1 - T) \times (1 - \alpha), \tag{B.1}$$

where α is a parameter selected to control the trade-off between accuracy and time.

Another possible measure can be defined by dividing accuracy over time as follows

$$\omega = \frac{A^{\alpha}}{T^{1-\alpha}}. \tag{B.2}$$

A comparison of the trade-off values of the three measures for α=0.1, 0.5 and 0.9 is presented in Figures B.1, B.2 and B.3, respectively. For a better view, the weighted arithmetic mean plots in the three figures were rotated.

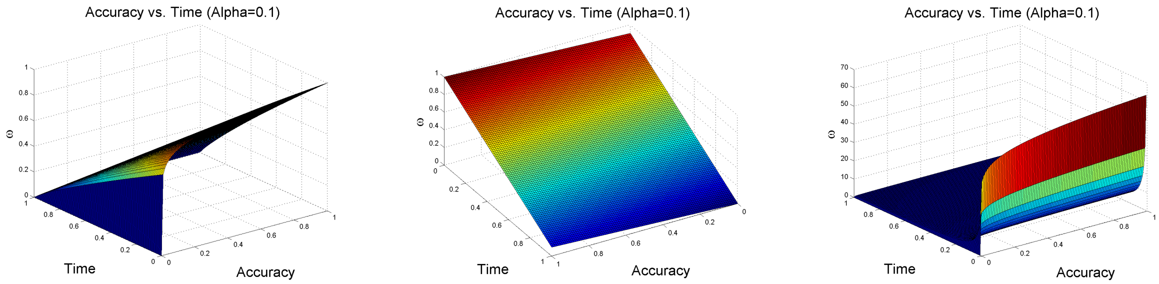


Figure B.1: Comparison of three trade-off measures with α=0.1. From left to right: weighted geometric mean, weighted arithmetic mean, and weighted accuracy over time.

Comparing the weighted geometric mean with the weighted arithmetic mean, it can be noticed that the former changes non-linearly, while the latter changes linearly. The weighted arithmetic mean lacked sensitivity to low values [132], therefore it has less penalty for low values of accuracy and time. At extreme points (e.g., A=0 or T=1), the weighted arithmetic mean obtains values greater than 0. For example, for α=0.5, even when accuracy is equal to 0, the weighted arithmetic mean obtained values up to 0.5. Unless α=1 or α=0, the trade-off measure should be 0 for those values, as obtained by the weighted geometric mean.

The values obtained by dividing accuracy over time rapidly change with lower values of time. This is especially noticeable with lower α values as in Figures B.1 and B.2. Furthermore, the values of ω are not bounded, which makes it difficult to understand the results.

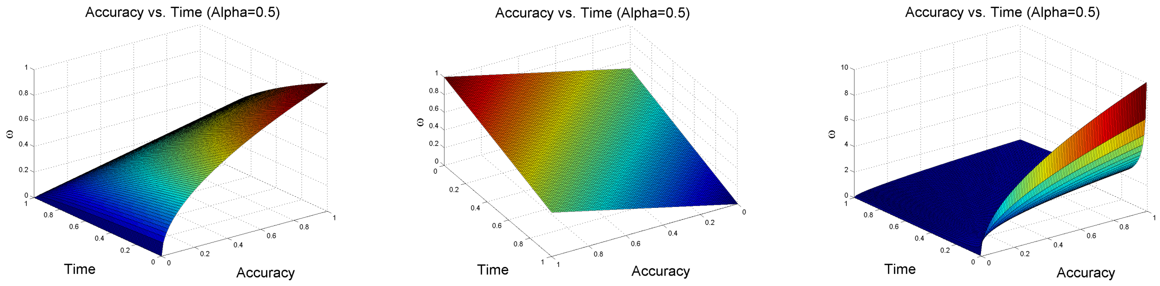


Figure B.2: Comparison of three trade-off measures with α=0.5. From left to right: weighted geometric mean, weighted arithmetic mean, and weighted accuracy over time.

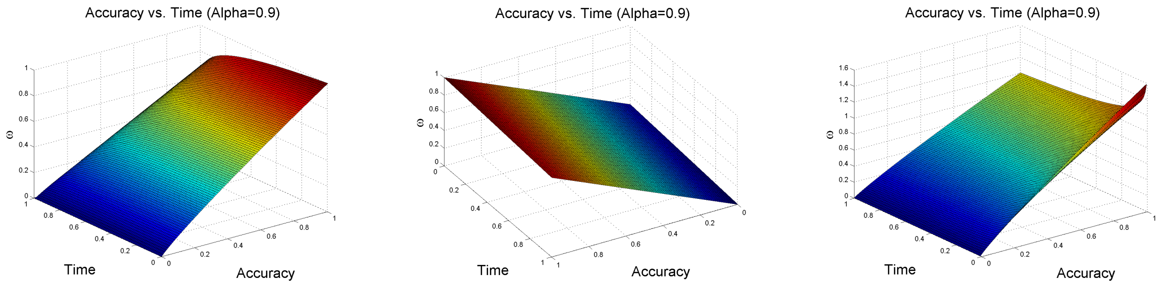


Figure B.3: Comparison of three trade-off measures with α=0.9. From left to right: weighted geometric mean, weighted arithmetic mean, and weighted accuracy over time.

Appendix C

Classifiers Performance Results

This Appendix presents the numerical results of the classifiers' performance and the comparative figures of F_1-*measure* with α=0.1, 0.3, 0.5, and 0.9.

C.1 Numerical Results

Table C.1: Performance of Different Classifiers with Different Values of α for the Breast Ultrasound Dataset

			F-measure					
α	Features	Learning Alg.	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5
0.1	LBP	SVM	—	—	0.2760	0.6110	0.2310	0.2430
		SVM+ADASYN	—	—	0.2928	0.6145	0.2712	0.1753
		AdaBoost	—	—	0.5549	0.8445	0.8560	0.6277
		RAMOBoost	—	—	0.6488	0.8628	0.8842	0.7530
	Statistical	SVM	—	—	0.1415	0.5689	0.0775	0.0382
		SVM+ADASYN	—	—	0.0934	0.5909	0.1056	0.0778
		AdaBoost	—	—	0.5241	0.8687	0.8835	0.7795
		RAMOBoost	—	—	0.5021	0.8822	0.9044	0.8549
0.3	LBP	SVM	—	—	0.0000	0.6926	0.4108	0.0000
		SVM+ADASYN	—	—	0.2337	0.6565	0.3213	0.0000
		AdaBoost	—	—	0.6589	0.8666	0.8863	0.1990
		RAMOBoost	—	—	0.7718	0.8870	0.8836	0.5067
	Statistical	SVM	—	—	0.0250	0.5741	0.3922	0.0000
		SVM+ADASYN	—	—	0.0959	0.4860	0.3364	0.0000
		AdaBoost	—	—	0.6278	0.8917	0.8895	0.2105
		RAMOBoost	—	—	0.6804	0.8955	0.8661	0.7429
0.5	LBP	SVM	—	—	0.1896	0.6805	0.1304	0.1000
		SVM+ADASYN	—	—	0.3214	0.6230	0.2525	0.0000
		AdaBoost	—	—	0.8268	0.8797	0.8187	0.0500
		RAMOBoost	—	—	0.8809	0.9200	0.8282	0.5000
	Statistical	SVM	—	—	0.0000	0.6853	0.0000	0.0000
		SVM+ADASYN	—	—	0.0518	0.4243	0.0944	0.0000
		AdaBoost	—	—	0.7663	0.9161	0.7215	0.2000
		RAMOBoost	—	—	0.7887	0.9002	0.7794	0.5200
0.7	LBP	SVM	—	0.0000	0.3292	0.5938	0.0308	0.0667
		SVM+ADASYN	—	0.0000	0.3179	0.6240	0.1115	0.0667
		AdaBoost	—	0.0000	0.8461	0.8550	0.7373	0.0500
		RAMOBoost	—	0.3000	0.8295	0.8585	0.6984	0.1900
	Statistical	SVM	—	0.0000	0.2490	0.4447	0.0250	0.0000
		SVM+ADASYN	—	0.0200	0.2203	0.3770	0.0442	0.0000
		AdaBoost	—	0.2467	0.8312	0.8813	0.5703	0.2300
		RAMOBoost	—	0.2867	0.8173	0.9086	0.6648	0.5300
0.9	LBP	SVM	—	0.0000	0.4772	0.5317	0.2053	0.0000
		SVM+ADASYN	—	0.0606	0.4956	0.4488	0.2823	0.1308
		AdaBoost	—	0.1905	0.8923	0.8182	0.6097	0.0000
		RAMOBoost	—	0.6749	0.8622	0.8390	0.5209	0.0667
	Statistical	SVM	—	0.0000	0.0154	0.4201	0.0000	0.0000
		SVM+ADASYN	—	0.0000	0.0345	0.4341	0.0000	0.0000
		AdaBoost	—	0.4441	0.9114	0.8321	0.6296	0.0000
		RAMOBoost	—	0.6405	0.8733	0.8588	0.5794	0.1333

Table C.2: Performance of Different Classifiers with Different Values of α for the Liberty Dataset

			F-measure					
α	Features	Learning Alg.	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5
0.1	LBP	SVM	—	—	—	0.7020	0.5797	0.6213
		SVM+ADASYN	—	—	—	0.6899	0.6428	0.7117
		AdaBoost	—	—	—	0.9175	0.9643	0.9665
		RAMOBoost	—	—	—	0.9531	0.9707	1.0000
	Statistical	SVM	—	—	—	0.5028	0.5665	0.7487
		SVM+ADASYN	—	—	—	0.5344	0.5402	0.6832
		AdaBoost	—	—	—	0.8626	0.9496	0.9602
		RAMOBoost	—	—	—	0.9895	0.9935	1.0000
0.3	LBP	SVM	—	—	0.3153	0.1450	0.5342	0.1239
		SVM+ADASYN	—	—	0.3460	0.1601	0.5103	0.0903
		AdaBoost	—	—	0.5790	0.8547	0.9687	0.7325
		RAMOBoost	—	—	0.5805	0.8576	0.9744	0.9241
	Statistical	SVM	—	—	0.2500	0.3923	0.6512	0.0000
		SVM+ADASYN	—	—	0.3884	0.4473	0.6871	0.0479
		AdaBoost	—	—	0.5259	0.8326	0.9216	0.7522
		RAMOBoost	—	—	0.6652	0.8782	0.9677	0.8934
0.5	LBP	SVM	—	—	0.6434	0.3231	0.2479	0.0879
		SVM+ADASYN	—	—	0.6793	0.3422	0.2460	0.2237
		AdaBoost	—	—	0.8963	0.8589	0.9041	0.6574
		RAMOBoost	—	—	0.9489	0.9292	0.9251	0.8774
	Statistical	SVM	—	—	0.8176	0.4430	0.3951	0.3951
		SVM+ADASYN	—	—	0.6768	0.3893	0.4165	0.0582
		AdaBoost	—	—	0.8330	0.8586	0.9069	0.5467
		RAMOBoost	—	—	0.9385	0.9155	0.9436	0.8726
0.7	LBP	SVM	—	0.2043	0.4310	0.7401	0.0000	0.4123
		SVM+ADASYN	—	0.1472	0.4371	0.7523	0.0000	0.3403
		AdaBoost	—	0.2800	0.9680	0.7812	0.5751	0.6880
		RAMOBoost	—	0.5667	0.9854	0.8440	0.7151	0.8858
	Statistical	SVM	—	0.0000	0.6777	0.6337	0.2040	0.1653
		SVM+ADASYN	—	0.3871	0.6015	0.6243	0.2063	0.3082
		AdaBoost	—	0.3400	0.9855	0.7512	0.5183	0.7115
		RAMOBoost	—	0.5567	0.9763	0.7824	0.5478	0.8145
0.9	LBP	SVM	—	0.2547	0.3628	0.6872	0.0000	0.3671
		SVM+ADASYN	—	0.2495	0.4323	0.7564	0.0644	0.4559
		AdaBoost	—	0.6401	0.9950	0.7379	0.4267	0.6871
		RAMOBoost	—	0.7389	0.9974	0.7880	0.4810	0.9273
	Statistical	SVM	—	0.0000	0.6897	0.7253	0.0000	0.2389
		SVM+ADASYN	—	0.2281	0.4751	0.5391	0.0944	0.3402
		AdaBoost	—	0.5908	0.9797	0.6643	0.4019	0.6047
		RAMOBoost	—	0.4579	0.9752	0.7292	0.3962	0.8887

Table C.3: Performance of Different Classifiers with Different Values of α for the Lung X-Ray Dataset

			F-measure					
α	Features	Learning Alg.	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5
0.1	LBP	SVM	—	—	—	0.2399	0.7345	0.0000
		SVM+ADASYN	—	—	—	0.4086	0.5703	0.2388
		AdaBoost	—	—	—	0.8812	0.9532	0.8266
		RAMOBoost	—	—	—	0.9070	0.9658	0.9314
	Statistical	SVM	—	—	—	0.2310	0.7336	0.0000
		SVM+ADASYN	—	—	—	0.4032	0.5699	0.3493
		AdaBoost	—	—	—	0.8440	0.9339	0.8041
		RAMOBoost	—	—	—	0.8938	0.9562	0.9175
0.3	LBP	SVM	—	—	0.0000	0.4142	0.5805	0.0000
		SVM+ADASYN	—	—	0.0000	0.4149	0.5854	0.0206
		AdaBoost	—	—	0.0000	0.9191	0.9552	0.7743
		RAMOBoost	—	—	0.0000	0.9416	0.9709	0.8733
	Statistical	SVM	—	—	0.0000	0.1571	0.5975	0.0000
		SVM+ADASYN	—	—	0.0000	0.1445	0.5912	0.0281
		AdaBoost	—	—	0.0000	0.9239	0.9561	0.7496
		RAMOBoost	—	—	0.0000	0.9433	0.9712	0.9030
0.5	LBP	SVM	—	—	0.0000	0.5554	0.5752	0.0250
		SVM+ADASYN	—	—	0.2594	0.3922	0.5694	0.2498
		AdaBoost	—	—	0.6713	0.9236	0.9457	0.7999
		RAMOBoost	—	—	0.5708	0.9419	0.9501	0.8634
	Statistical	SVM	—	—	0.0000	0.4769	0.5175	0.0000
		SVM+ADASYN	—	—	0.2105	0.4028	0.4990	0.0586
		AdaBoost	—	—	0.5244	0.9016	0.9234	0.6369
		RAMOBoost	—	—	0.5938	0.9258	0.9530	0.8354
0.7	LBP	SVM	—	—	0.2819	0.4281	0.5732	0.1682
		SVM+ADASYN	—	—	0.4908	0.3824	0.5321	0.3076
		AdaBoost	—	—	0.8245	0.8845	0.9437	0.6946
		RAMOBoost	—	—	0.8363	0.9102	0.9650	0.8625
	Statistical	SVM	—	—	0.5078	0.3009	0.5180	0.0000
		SVM+ADASYN	—	—	0.5238	0.2127	0.4366	0.2460
		AdaBoost	—	—	0.7154	0.8700	0.9013	0.6078
		RAMOBoost	—	—	0.7961	0.8975	0.9310	0.7978
0.9	LBP	SVM	—	—	0.4603	0.5188	0.5077	0.1453
		SVM+ADASYN	—	—	0.4159	0.3847	0.4435	0.1901
		AdaBoost	—	—	0.8691	0.8905	0.9454	0.6450
		RAMOBoost	—	—	0.8716	0.9075	0.9531	0.7737
	Statistical	SVM	—	—	0.5516	0.4290	0.4663	0.0222
		SVM+ADASYN	—	—	0.5291	0.3292	0.4274	0.2692
		AdaBoost	—	—	0.7960	0.8891	0.9310	0.5742
		RAMOBoost	—	—	0.8234	0.8955	0.9409	0.7717

Table C.4: Performance of Different Classifiers with Different Values of α for the Synthetic Images Dataset

			F-measure					
α	Features	Learning Alg.	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5
0.1	LBP	SVM	—	—	0.4199	0.7131	0.0000	—
		SVM+ADASYN	—	—	0.4571	0.7142	0.0000	—
		AdaBoost	—	—	0.9847	0.9659	0.0500	—
		RAMOBoost	—	—	0.9848	0.9737	0.5000	—
	Statistical	SVM	—	—	0.7289	0.7090	0.0000	—
		SVM+ADASYN	—	—	0.7373	0.4830	0.0914	—
		AdaBoost	—	—	0.9717	0.9596	0.3300	—
		RAMOBoost	—	—	0.9762	0.9720	0.7157	—
0.3	LBP	SVM	—	—	0.7332	0.5553	0.0000	—
		SVM+ADASYN	—	—	0.7278	0.5631	0.0000	—
		AdaBoost	—	—	0.9808	0.9583	0.0000	—
		RAMOBoost	—	—	0.9975	0.9826	0.0000	—
	Statistical	SVM	—	—	0.8239	0.6584	0.0000	—
		SVM+ADASYN	—	—	0.8128	0.6637	0.0000	—
		AdaBoost	—	—	0.9619	0.9285	0.0000	—
		RAMOBoost	—	—	0.9841	0.9646	0.0000	—
0.5	LBP	SVM	—	0.0000	0.8381	0.4599	0.0000	—
		SVM+ADASYN	—	0.0000	0.7419	0.3928	0.0000	—
		AdaBoost	—	0.0000	0.9861	0.9241	0.0000	—
		RAMOBoost	—	0.0000	0.9855	0.9213	0.0000	—
	Statistical	SVM	—	0.0000	0.8323	0.4664	0.0000	—
		SVM+ADASYN	—	0.0000	0.7777	0.4662	0.0000	—
		AdaBoost	—	0.0000	0.9834	0.9162	0.0000	—
		RAMOBoost	—	0.0000	0.9799	0.9105	0.0000	—
0.7	LBP	SVM	—	0.0000	0.8674	0.1573	—	—
		SVM+ADASYN	—	0.0000	0.8047	0.1203	—	—
		AdaBoost	—	0.5400	0.9826	0.9294	—	—
		RAMOBoost	—	0.6200	0.9922	0.9812	—	—
	Statistical	SVM	—	0.0000	0.8715	0.2636	—	—
		SVM+ADASYN	—	0.1596	0.7007	0.4653	—	—
		AdaBoost	—	0.2467	0.9675	0.9056	—	—
		RAMOBoost	—	0.3000	0.9731	0.9505	—	—
0.9	LBP	SVM	—	0.2013	0.9040	0.0000	—	—
		SVM+ADASYN	—	0.1318	0.8341	0.0000	—	—
		AdaBoost	—	0.9314	0.9844	0.7567	—	—
		RAMOBoost	—	0.9650	0.9940	1.0000	—	—
	Statistical	SVM	—	0.0694	0.8893	0.0000	—	—
		SVM+ADASYN	—	0.2883	0.7230	0.0892	—	—
		AdaBoost	—	0.8948	0.9798	0.7933	—	—
		RAMOBoost	—	0.8903	0.9833	0.9514	—	—

C.2 F_1-*measure* Figures

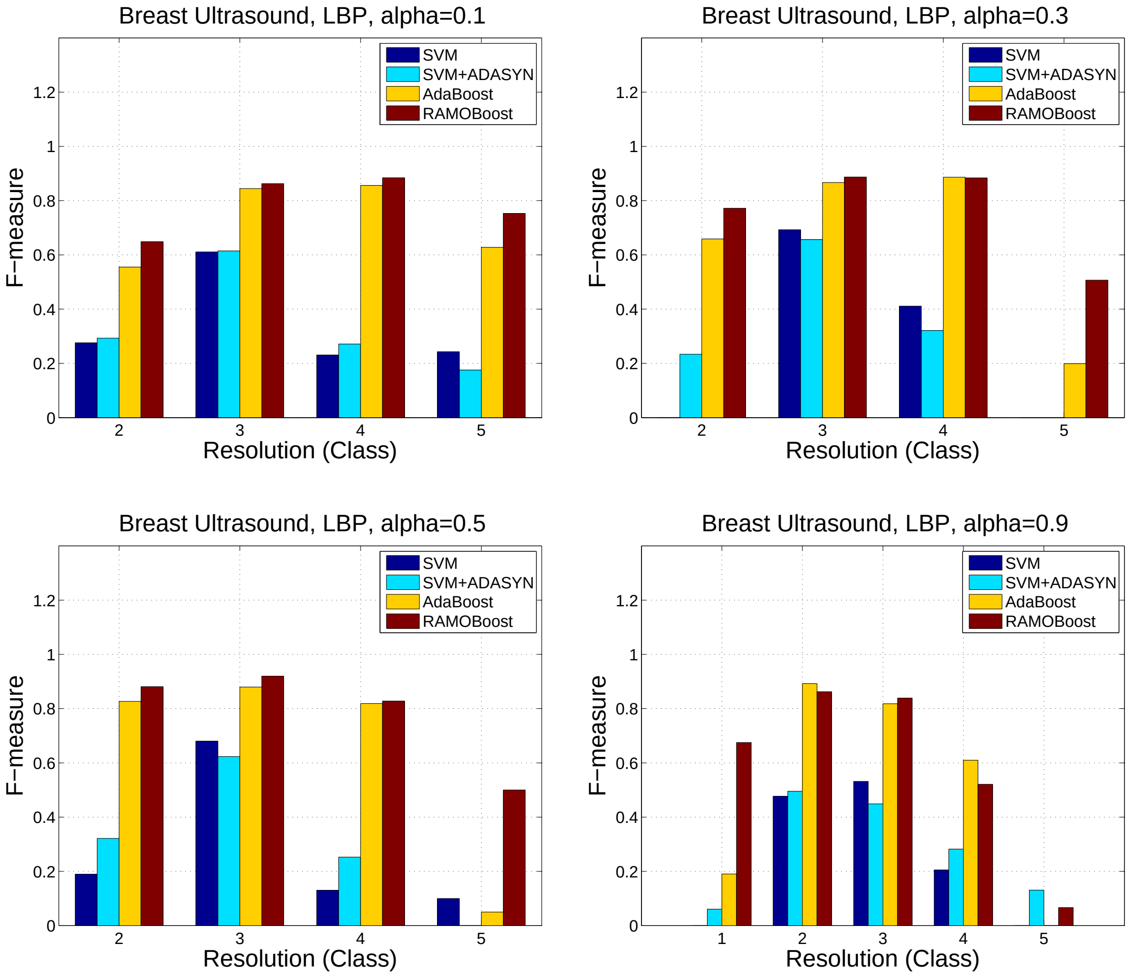


Figure C.1: Performance of different classifiers with α=0.1, 0.3, 0.5, and 0.9 for the breast ultrasound dataset: LBP features.

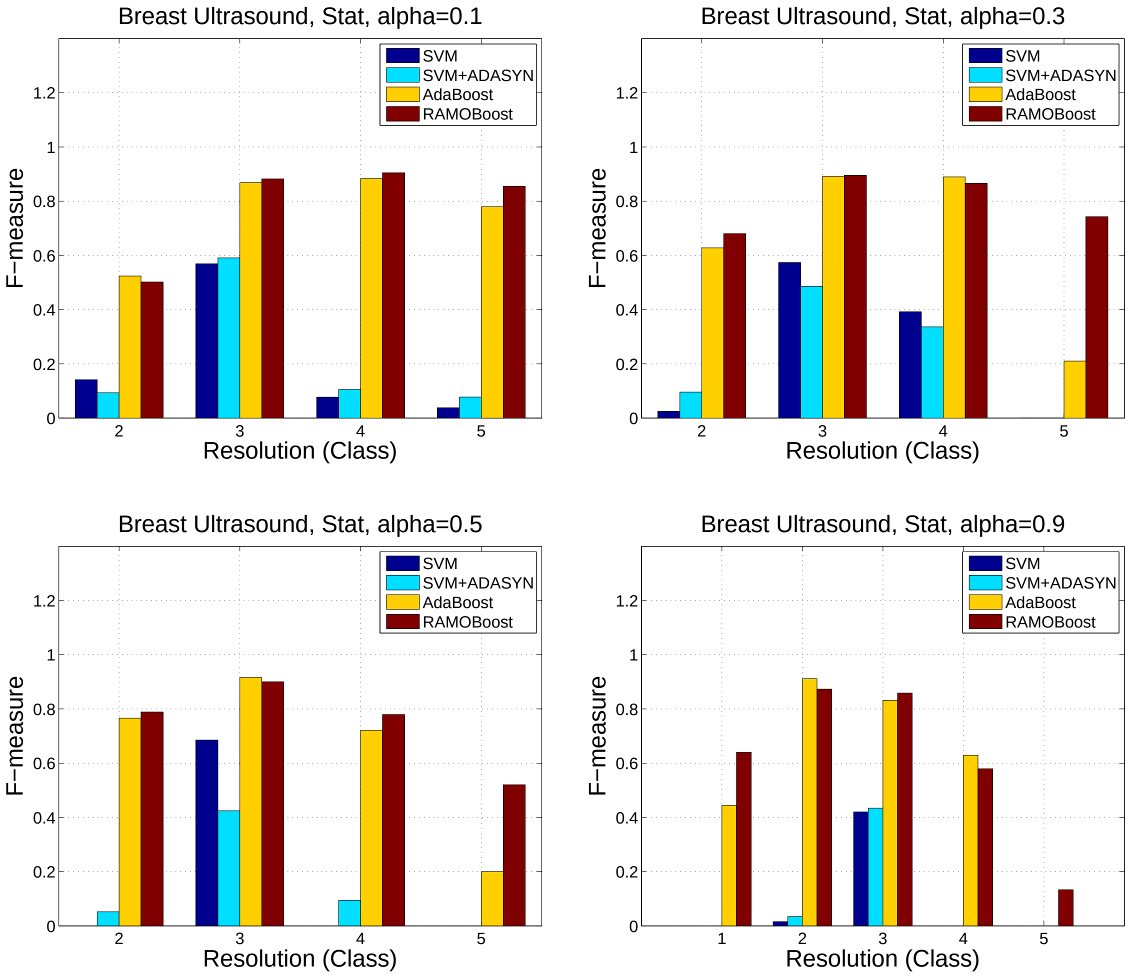


Figure C.2: Performance of different classifiers with α=0.1, 0.3, 0.5, and 0.9 for the breast ultrasound dataset: statistics features.

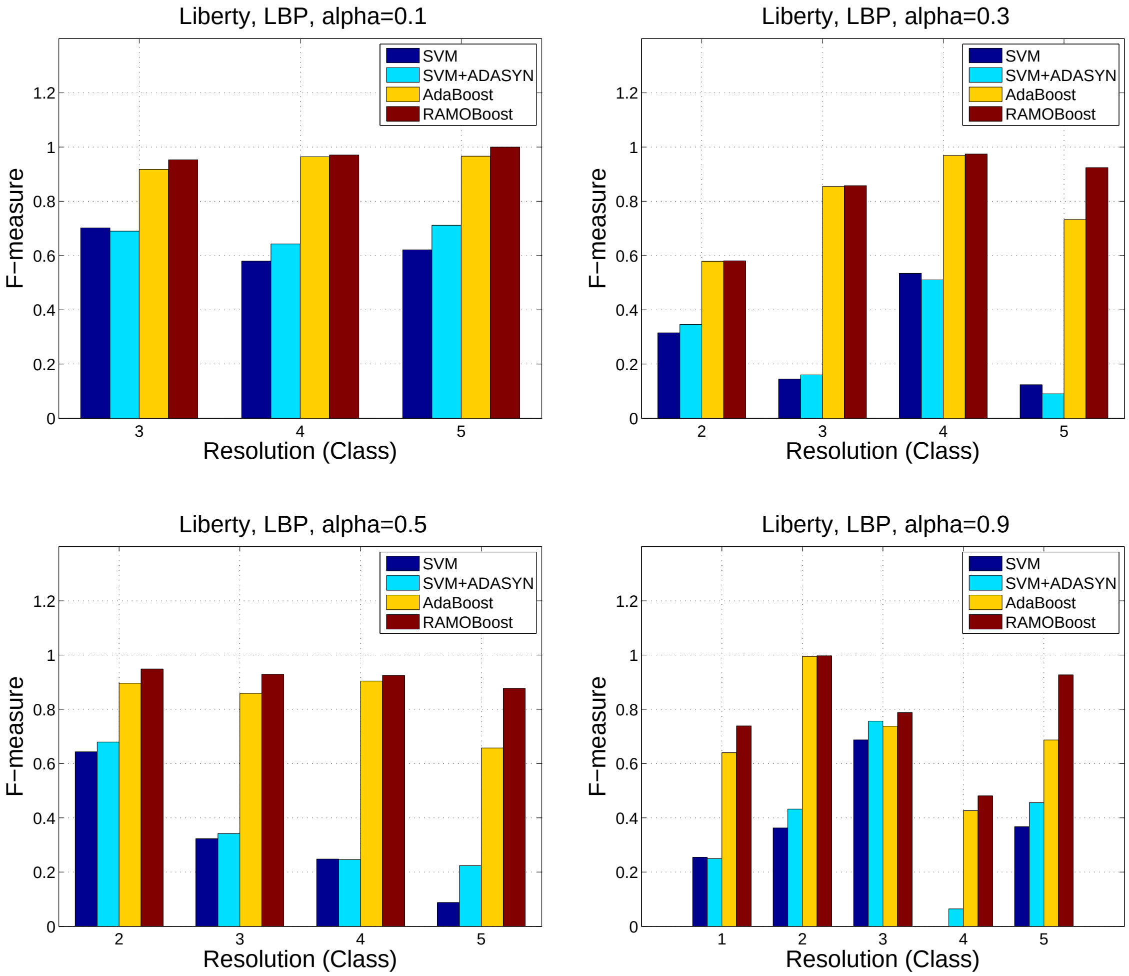


Figure C.3: Performance of different classifiers with α=0.1, 0.3, 0.5, and 0.9 for the Liberty dataset: LBP features.

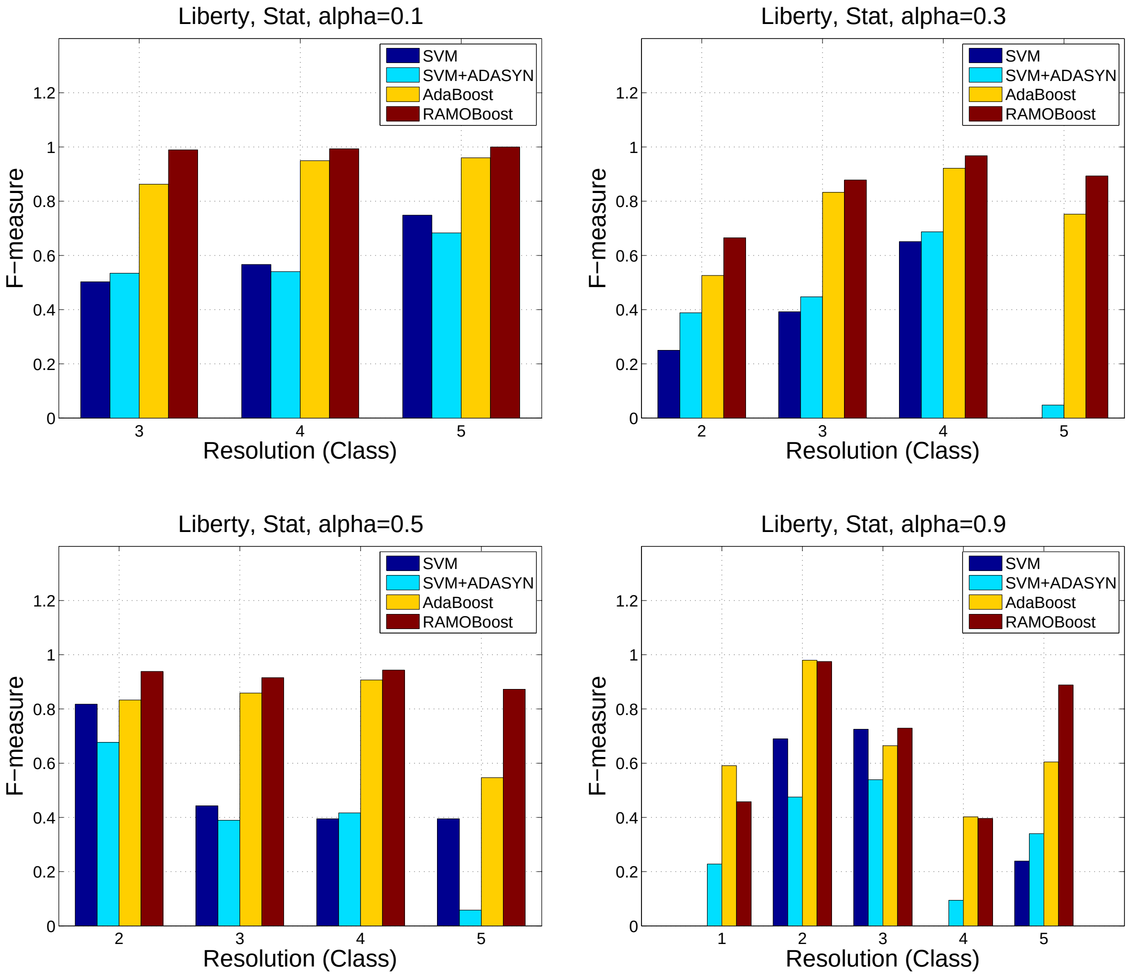


Figure C.4: Performance of different classifiers with α=0.1, 0.3, 0.5, and 0.9 for the Liberty dataset: statistics features.

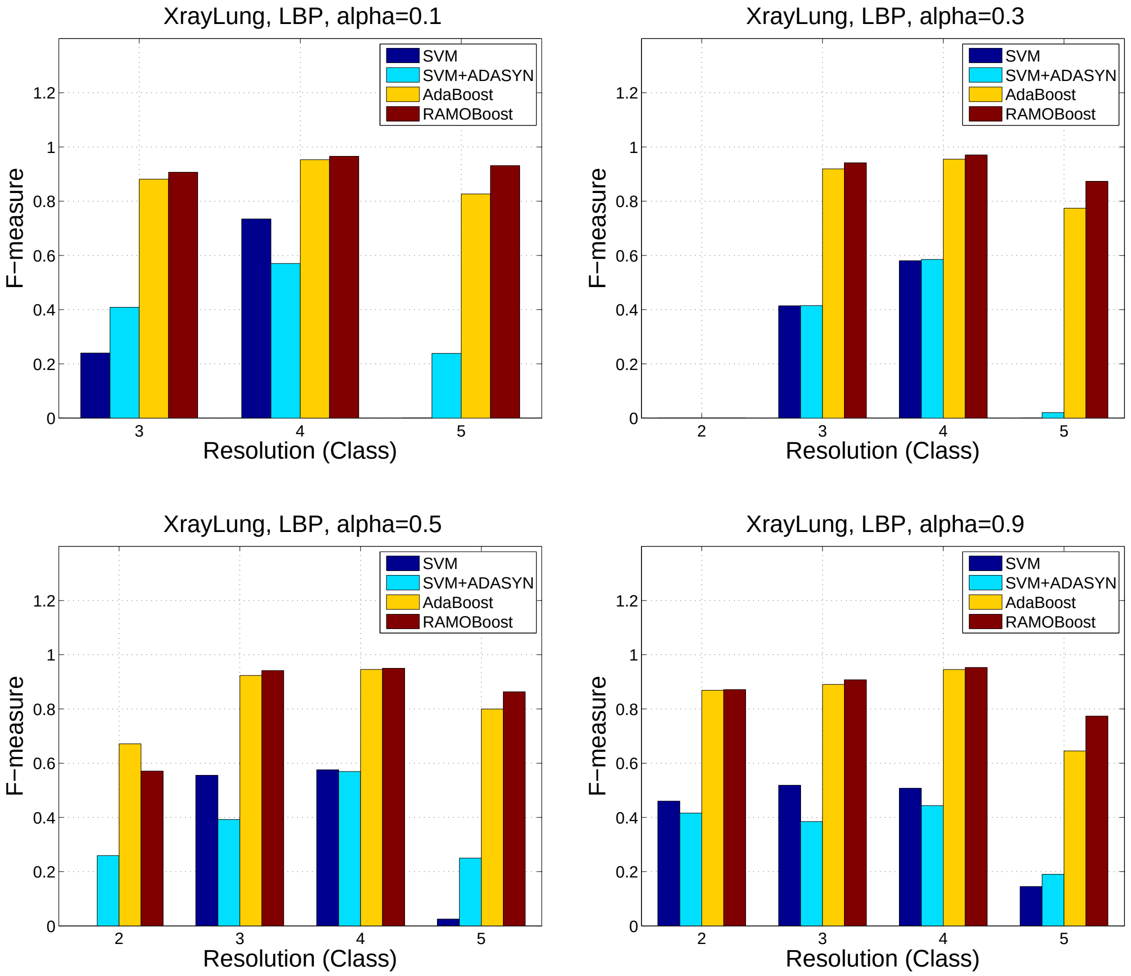


Figure C.5: Performance of different classifiers with α=0.1, 0.3, 0.5, and 0.9 for the lung X-ray dataset: LBP features.

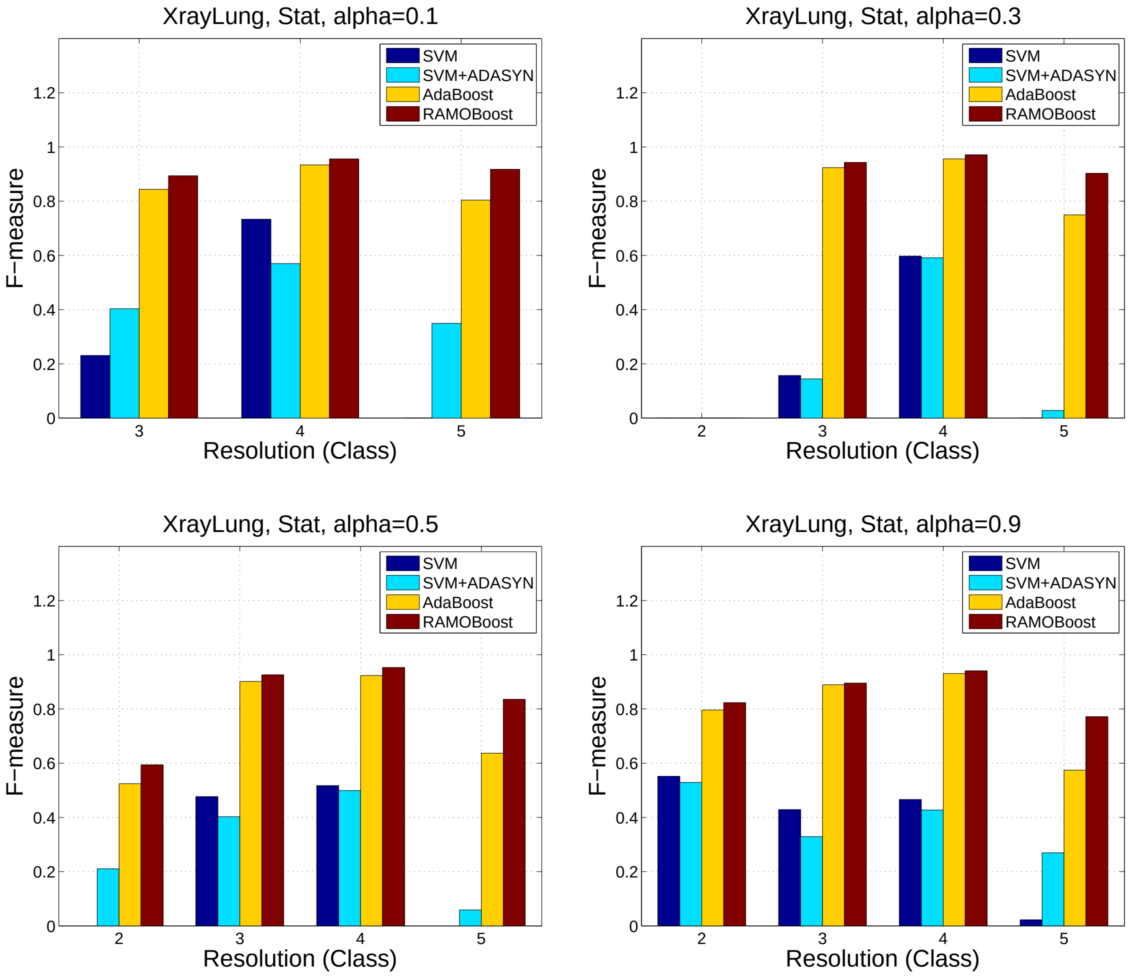


Figure C.6: Performance of different classifiers with α=0.1, 0.3, 0.5, and 0.9 for the lung X-ray dataset: statistics features.

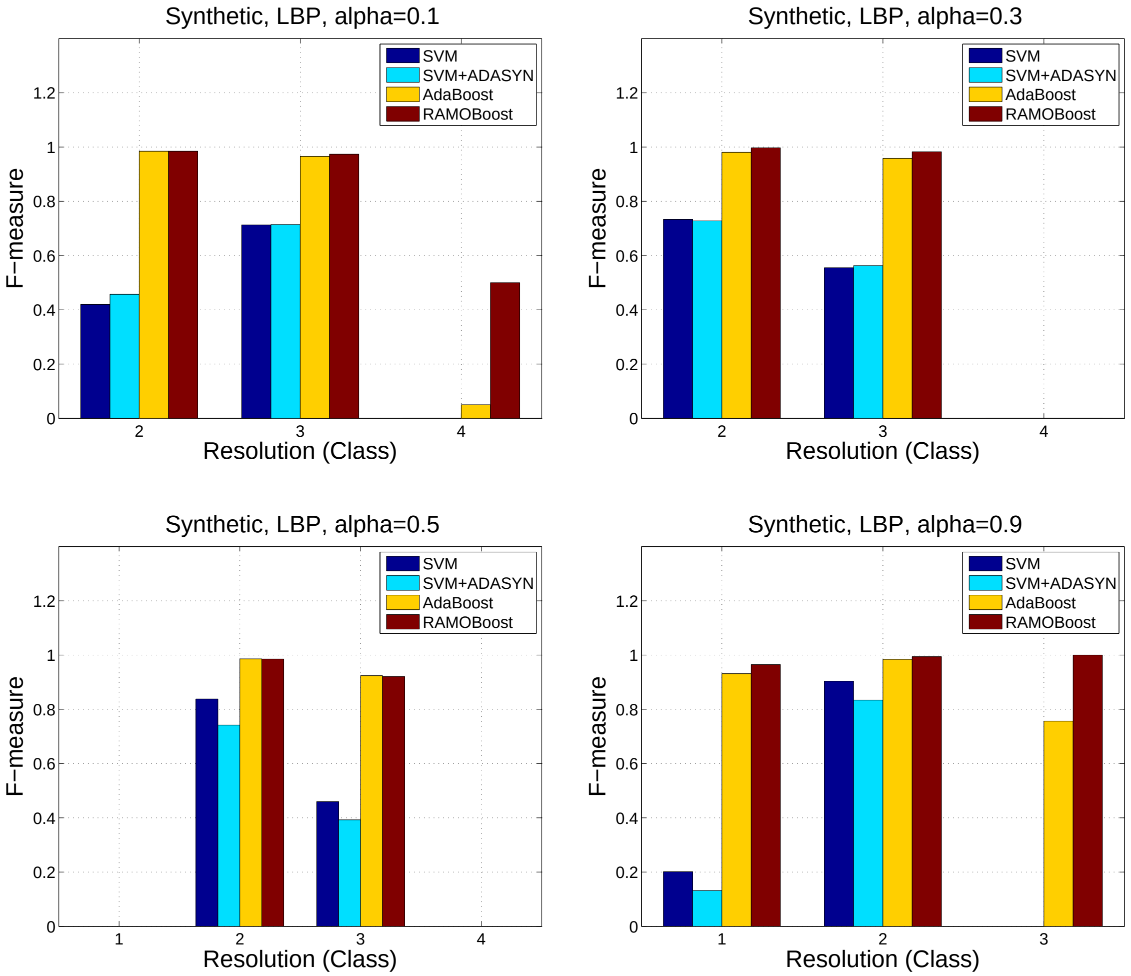


Figure C.7: Performance of different classifiers with α=0.1, 0.3, 0.5, and 0.9 for the synthetic images dataset: LBP features.

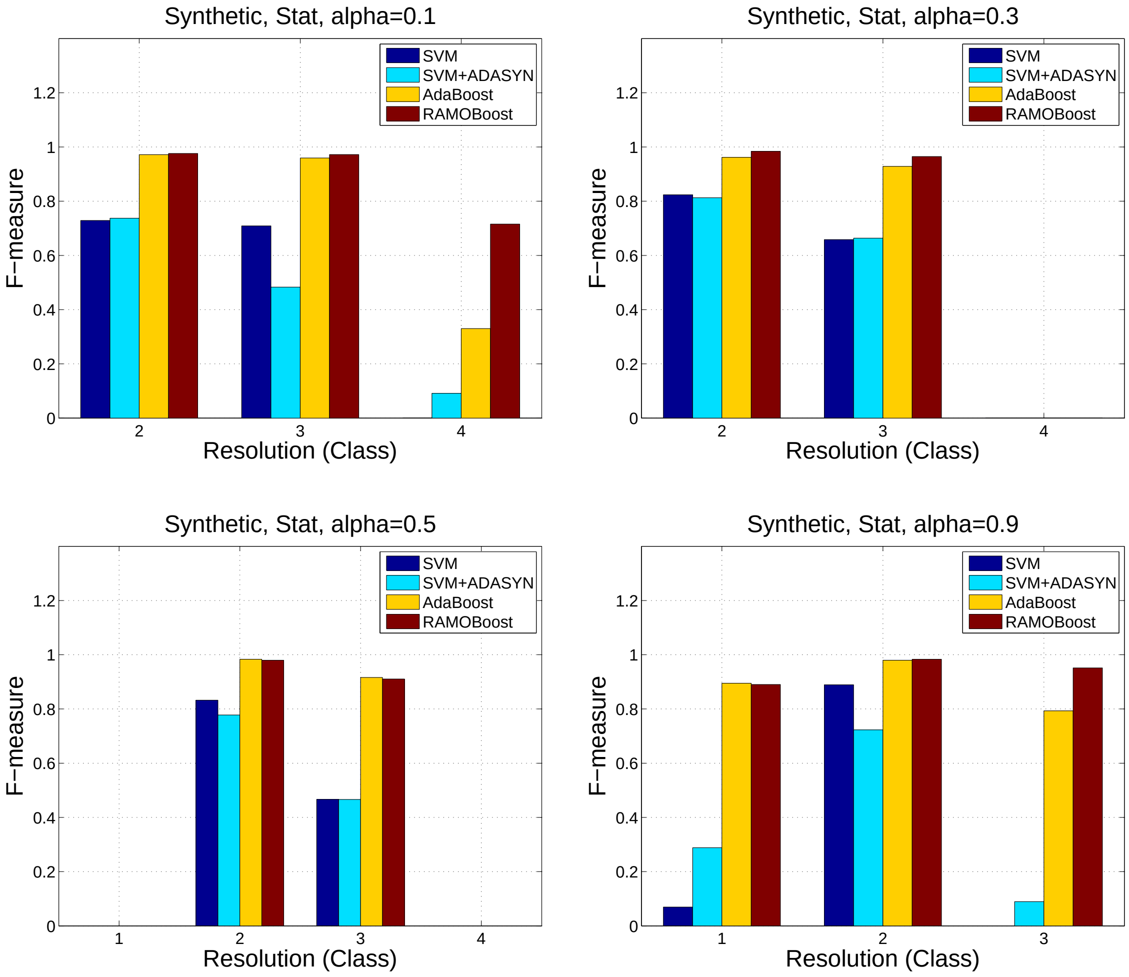


Figure C.8: Performance of different classifiers with α=0.1, 0.3, 0.5, and 0.9 for the synthetic images dataset: statistics features.

Appendix D

LBP and Statistical Features Results Comparison

Comparisons of the learning performance using LBP and Statistical features are presented in this Appendix.

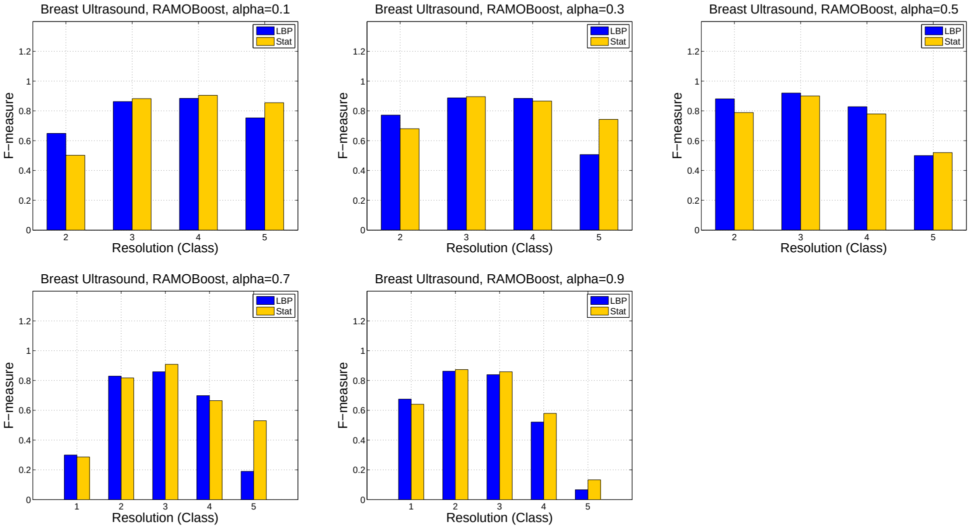


Figure D.1: Features comparison for RAMOBoost classifier with different values of α for the breast ultrasound dataset.

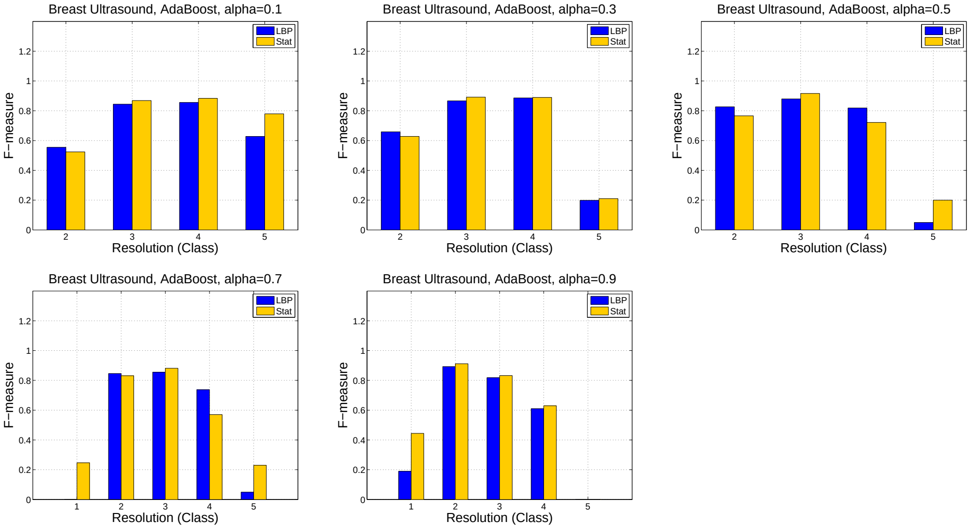


Figure D.2: Features comparison for AdaBoost classifier with different values of α for the breast ultrasound dataset.

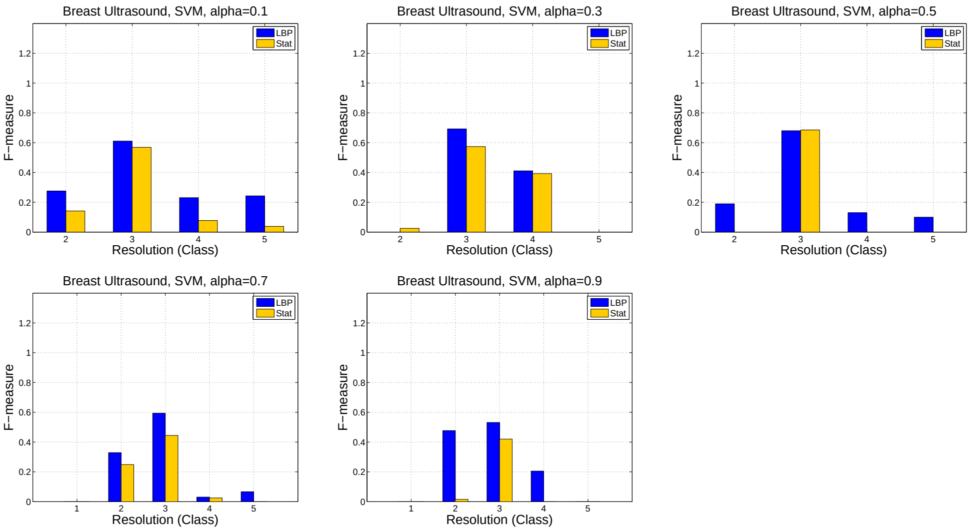


Figure D.3: Features comparison for SVM classifier with different values of α for the breast ultrasound dataset.

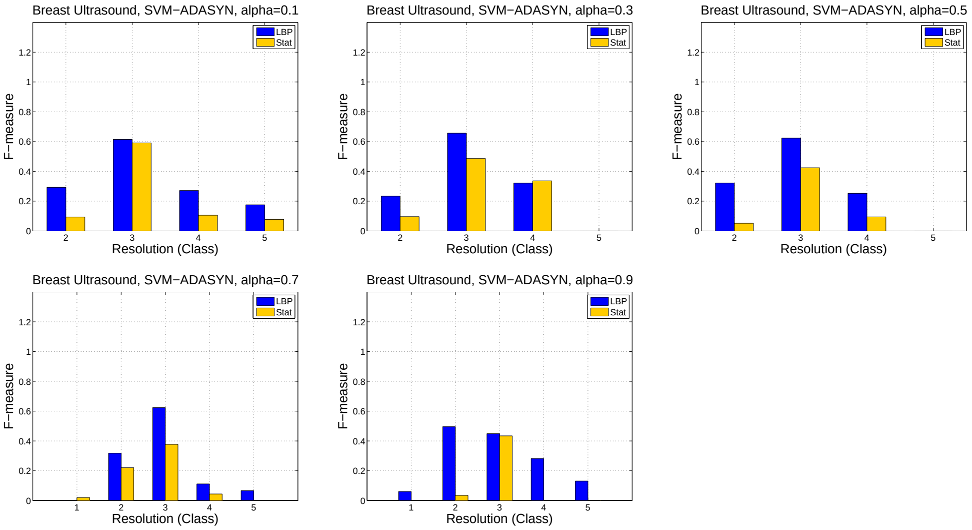


Figure D.4: Features comparison for SVM-ADASYN classifier with different values of α for the breast ultrasound dataset.

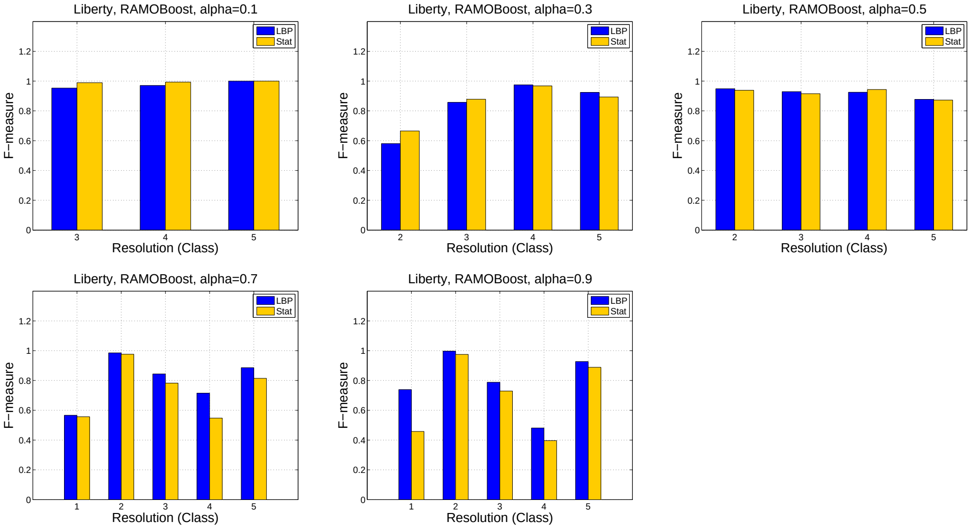


Figure D.5: Features comparison for RAMOBoost classifier with different values of α for the Liberty dataset.

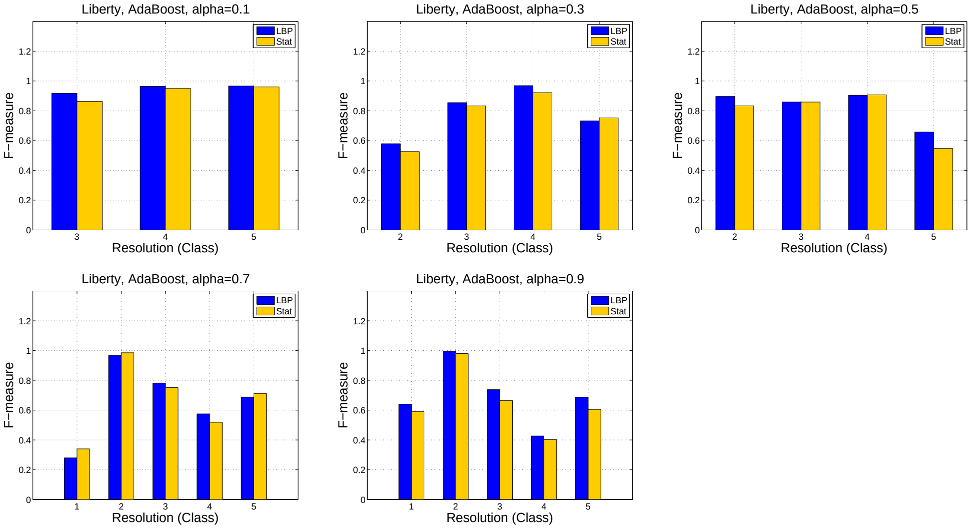


Figure D.6: Features comparison for AdaBoost classifier with different values of α for the Liberty dataset.

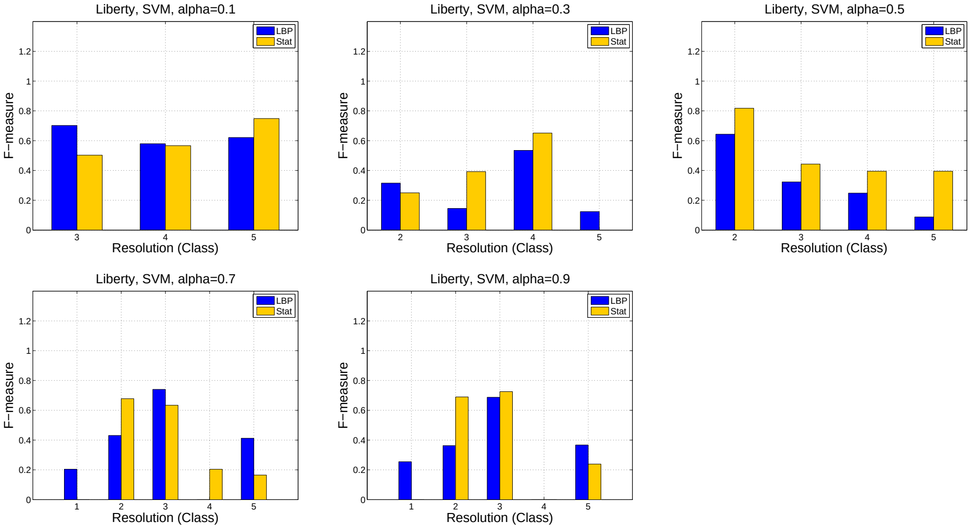


Figure D.7: Features comparison for SVM classifier with different values of α for the Liberty dataset.

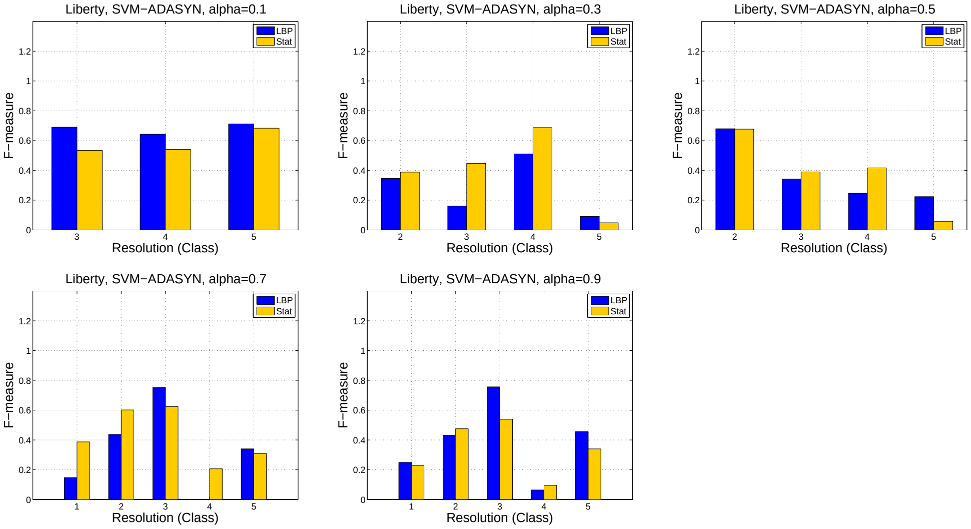


Figure D.8: Features comparison for SVM-ADASYN classifier with different values of α for the Liberty dataset.

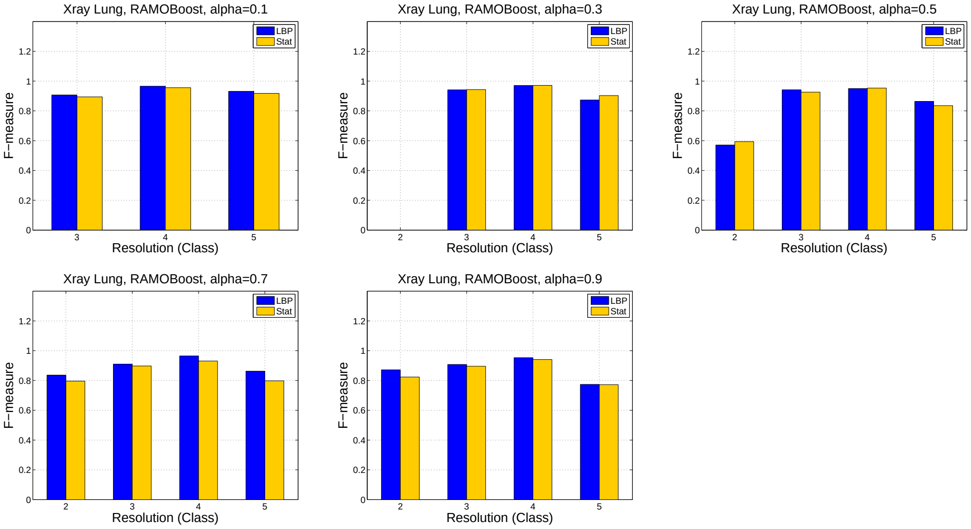


Figure D.9: Features comparison for RAMOBoost classifier with different values of α for the lung X-ray dataset.

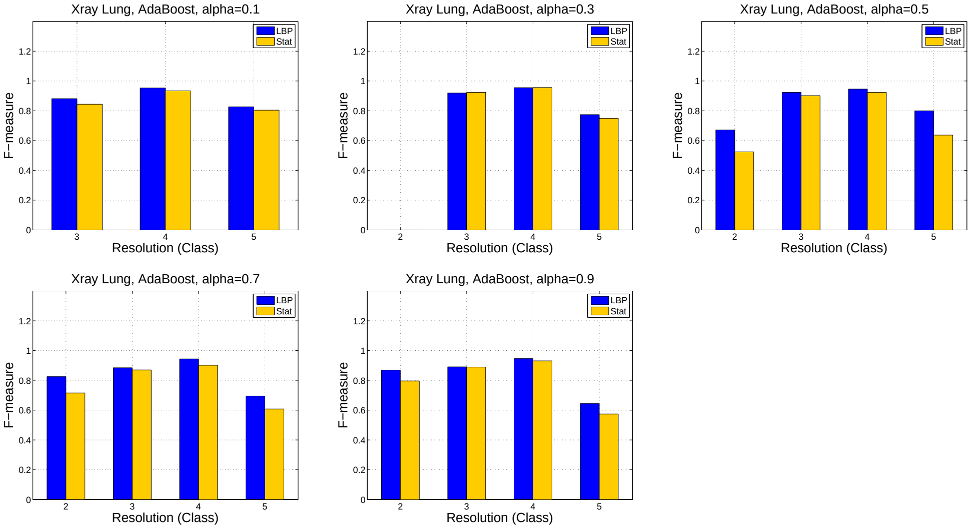


Figure D.10: Features comparison for AdaBoost classifier with different values of α for the lung X-ray dataset.

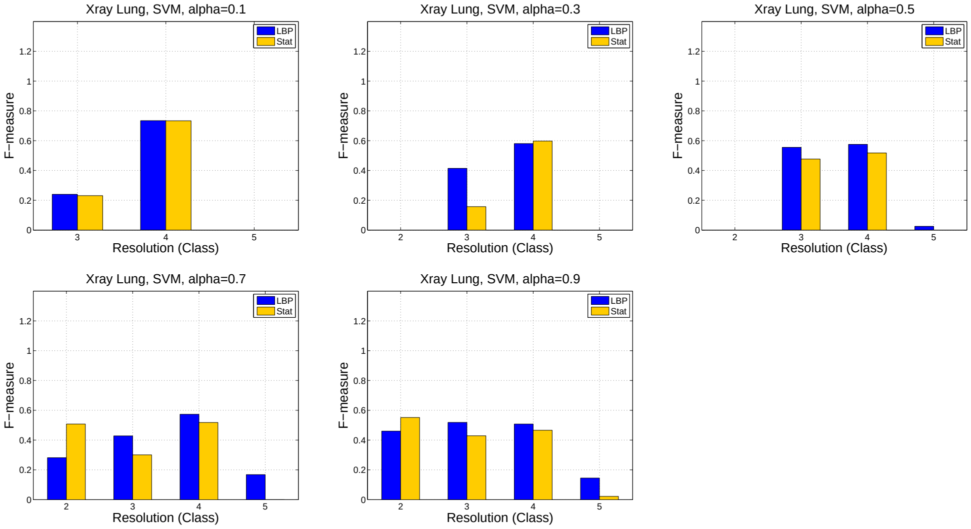


Figure D.11: Features comparison for SVM classifier with different values of α for the lung X-ray dataset.

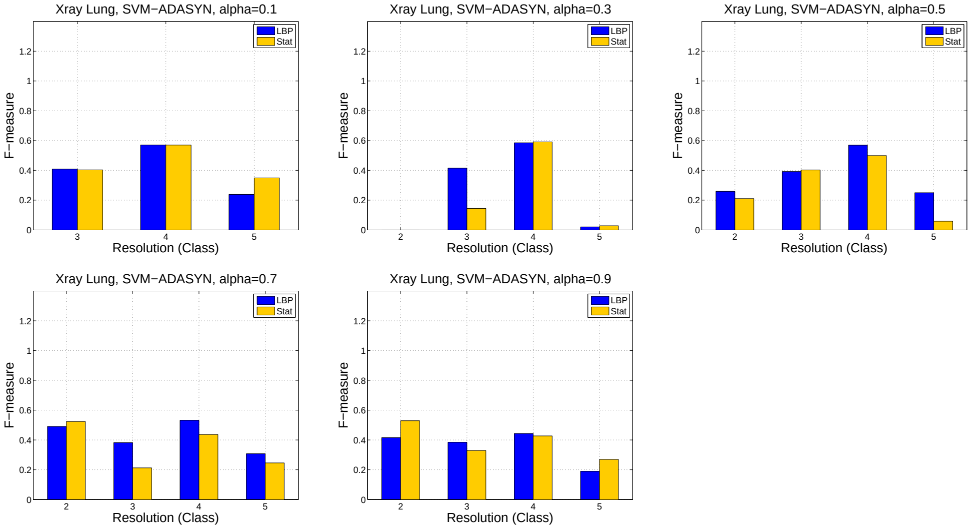


Figure D.12: Features comparison for SVM-ADASYN classifier with different values of α for the lung X-ray dataset.

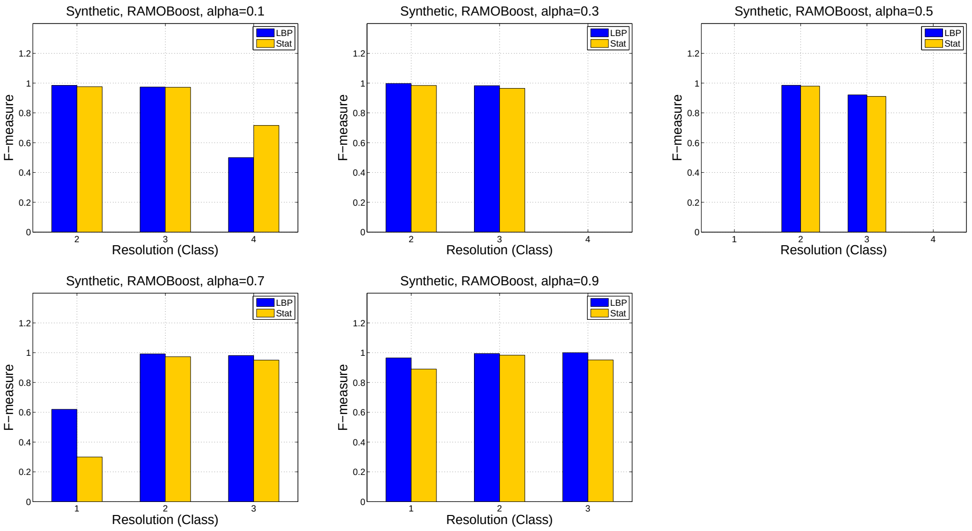


Figure D.13: Features comparison for RAMOBoost classifier with different values of α for the synthetic dataset.

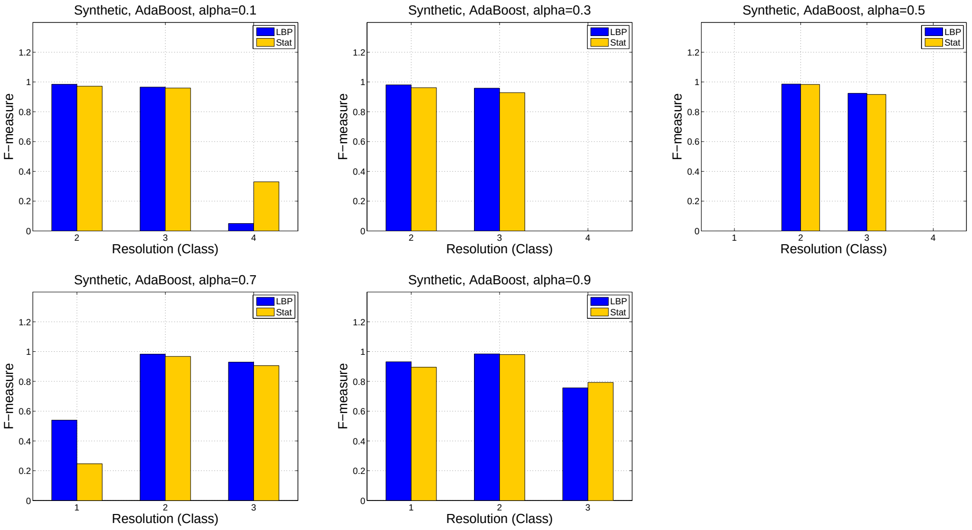


Figure D.14: Features comparison for AdaBoost classifier with different values of α for the synthetic dataset.

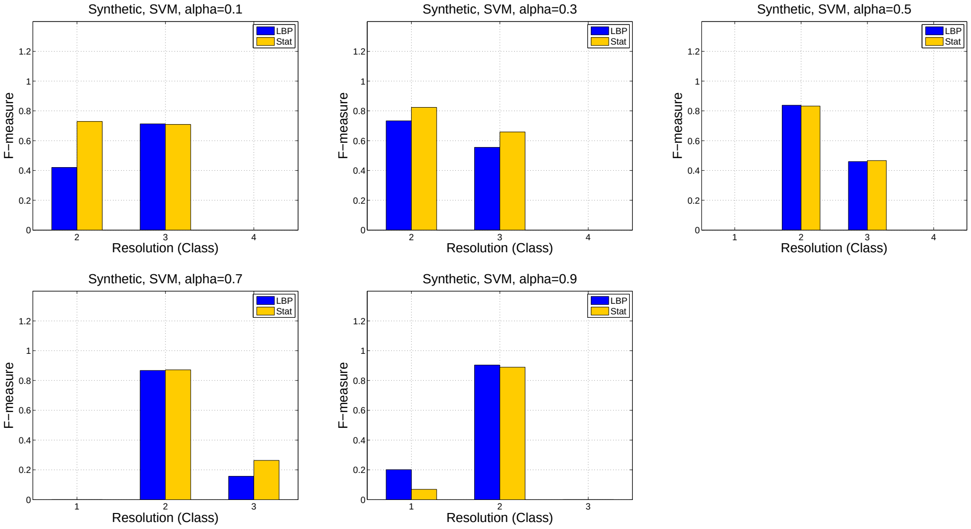


Figure D.15: Features comparison for SVM classifier with different values of α for the synthetic dataset.

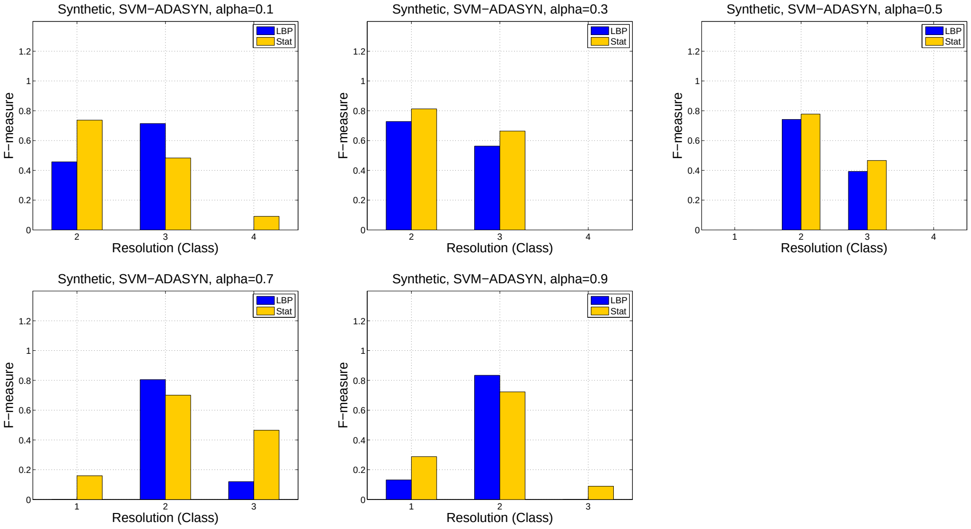


Figure D.16: Features comparison for SVM-ADASYN classifier with different values of α for the synthetic dataset.