COMPUTER AIDED DETECTION OF SPINA BIFIDA USING FEATURES DERIVED FROM CURVATURE SCALE SPACE AND ZERNIKE MOMENTS

by

Umut Konur

B.S., Computer Engineering, Boğaziçi University, 2003

M.S., Computer Engineering, Boğaziçi University, 2006

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2015

ACKNOWLEDGEMENTS

The efforts that I have carried out to arrive at this thesis would not have resulted in a completion unless the supports and assistances of many people had not pushed me towards the goal. I would like to express my sincere thanks to my supervisor Prof. Fikret S. Gürgen for guidance, support and patience of all sorts. The work was initiated with the suggestion of Prof. Füsun G. Varol of the Obstetrics and Gynecology Department of the Medical Faculty of Trakya University, to whom I return my thanks for medical assistance, hospitality and generosity. She has actually been the supervising agent that helped me in supervised learning. I feel gratitude for Prof. Lale Akarun, who has always fed me with useful and sober comments to help improve the work. Our interdisciplinary work, combining engineering and medicine, has also been supported by the Obstetrics and Gynecology Department of the Medical Faculty of İstanbul University. I would like to thank Prof. Atıl Yüksel and especially Assoc. Prof. İbrahim H. Kalelioğlu for their understanding, friendly treatment and providing fetal ultrasound image samples. I would like to express my gratitude to Prof. Cengizhan Öztürk and Assist. Prof. Arzucan Özgür for participating in my thesis jury.

Each successful work emerges in a peaceful environment, most of which has been supplied within the Computer Engineering Department of Boğaziçi University. I thank all my instructors, colleagues and friends taking part throughout my academic life.

Finally, I can not do without mentioning my parents' endless love and support. Without such contributions, it would honestly be impossible to resist and stand against very hard and grueling processes.

The work of this thesis is supported by the Scientific Research Projects fund (BAP 10A01D5, BAP 14A01P2) of Boğaziçi University and the Turkish Ministry of Development under the TAM Project, number 2007K120610.

ABSTRACT

COMPUTER AIDED DETECTION OF SPINA BIFIDA USING FEATURES DERIVED FROM CURVATURE SCALE SPACE AND ZERNIKE MOMENTS

The work of this dissertation focuses on a specific *computer aided diagnosis* (*CAD*) problem, although the main concept can be generalized to similar problems. Our aim is to detect the presence of the *spina bifida* (open spine) neural tube defect that is evident for a physician when the fetal skull image of a subject is examined. The objective of applications performing automatic anomaly detection can be set in their original contexts. Such systems, as a second observer, may help avoid false diagnoses. Fetal skull shapes possess markers that signal the presence of spina bifida. That is why this thesis concentrates on exploiting features extracted from skull shapes obtained via *ultrasound* (*US*). Among the variety of shape representation and feature extraction schemes, we have implemented and experimented with two. Both the *curvature scale space* (*CSS*) and moment-based (i.e. *Zernike* moments) representations have proved to be robust in that the extracted features provide classification invariant under the similarity transformations of translation, rotation and scaling. Classification of shapes is commonly coupled with the problem of segmentation. Since the fully-automatic segmentation of US images is practically difficult, we have attempted to achieve segmentation semi-automatically after the manual marking of a small number of points on images, based on simple heuristics and the *Active Shape Models* (*ASM*). Our experiments use *k-nearest neighbor* (*kNN*) and *Support Vector Machines* (*SVM*) classifiers. The inherent problem of rarity of medical data sets is tackled with methods of undersampling and oversampling. The results, reported for ground truth segmentations, reveal the availability of optimal operating points serving particular objectives.

ÖZET

EĞRİLİK ÖLÇEK UZAYINDAN VE ZERNİKE MOMENTLERİNDEN TÜRETİLEN ÖZNİTELİKLERLE SPİNA BİFİDANIN BİLGİSAYAR DESTEKLİ TANISI

Bu tezdeki yaklaşım ve yöntemler başka problemlere genelleştirilebilir olmakla birlikte, özelde iyi belirlenmiş bir bilgisayar destekli tanı problemi üzerinde uygulanmıştır. Amacımız, fetal kafatası imgeleri uzmanlarca incelendiğinde görülebilen spina bifida (açık omurga) sinir tüpü hasarının saptanmasıdır. Otomatik hastalık tespiti yapan uygulamaların amacı özgün çerçevelerde belirlenebilir. Bu tip sistemler, yanlış teşhisleri önlemek amaçlı alternatif gözlemciler olarak kullanılabilir. Spina bifidanın varlığı fetal kafatası biçiminin taşıdığı işaretlerden anlaşılmakta ve bu yüzden, okuyacağınız tezde, ultrason (US) ile edinilen fetal kafatası imgelerinden çıkarılan biçim öznitelikleri kullanılmaktadır. Literatürdeki biçim gösterimi ve öznitelik edinim teknikleri çeşitlilik gösterirken, bunlardan ikisi gerçeklenmiştir. Eğrilik ölçek uzayı ve momentlere (Zernike momentleri) dayalı gösterimler, özniteliklerin ötelenme, dönme ve ölçeklenme dönüşümleri altında değişimsiz olması veya yapılandırılabilmeleri itibariyle, gürbüz gösterimler olarak değerlendirilmektedir. Biçimlerle sınıflandırma, genellikle, bölütleme sorunu ile beraber ortaya çıkmaktadır. US imgelerinin tam-otomatik olarak bölütlenmesi uygulamada zor olduğundan, tezimizde, az sayıda nokta işaretlenerek yarı-otomatik bölütleme hedeflenmiştir. Kullanılan yöntemler basit sezgisellere ve aktif görünüm modellerine dayanmaktadır. Deneylerde en yakın komşu ve destek vektör makineleri sınıflandırıcıları kullanılmakta ve tıbbi verilerin doğasındaki azlık sorunu yüzünden veri kümeleri alt-örnekleme ve üst-örneklemeyle işlenmektedir. Temelde doğru bölütlemelerle bildirilen sonuçlar, belirli amaçları gözeten en iyi işletim noktalarının belirlenebileceğini göstermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A_{pqk}	Coefficients of real-valued radial Zernike polynomials
$\mathbf{b}$	Vector of shape model parameters
b_i	i^{th} parameter of the shape model
$\mathbf{C}_i$	i^{th} class or cluster
$C_{p,q}$	Complex moments of order $p+q$
$f(x,y)$	Image function defined on regional positions (x,y)
$f(\mathrm{r},\theta)$	Image function defined on polar coordinates (r,θ)
$\hat{f}$	Reconstructed image function
$\mathbf{F}(\mathbf{b},\mathbf{p})$	Fit function of model instance with model parameters $\mathbf{b}$ and pose parameters $\mathbf{p}$
$\mathrm{g}(u,\sigma)$	Gaussian kernel used to smooth a curve
$\mathbf{g}_i$	Vector of sampled points along normal profiles in i^{th} image
$G_{p,q}$	Geometric moment of order $p+q$
$\mathbf{H}(x)=c$	Hyperplane equation
$\mathrm{H}(I_1,I_2)$	Hamming distance of binary images I_1 and I_2
$\mathrm{K}(x,x^t)$	Kernel function, between support vectors and input in the original space, to replace inner product of basis functions $\bar{\Phi}$
mP_i	Model point in DMLM matching
mV	Feature vector of model in DMLM matching
moP_i	Model output point in DMLM matching
moV	Feature vector of model output in DMLM matching
m_{pq}	General moment of order (p,q)
$\mathbf{p}$	Vector of pose parameters
qP_i	Query point in DMLM matching
qV	Feature vector of query in DMLM matching
r	Parametric curve
$\hat{r}$	Normalized parametric curve
r^t	Desired output (class label) of input x^t
R	Smoothed parametric curve

$\hat{R}$	Smoothed and normalized parametric curve
$\hat{R}_g$	Normalized, smoothed and gain controlled parametric curve
R_{ref}	Reflected contour (curve)
R_{pq}	Real-valued radial Zernike polynomial with order p and repetition q
$\mathbf{S}$	Covariance matrix
$\mathrm{t}(u)$	Tangent vector at position u
T	Shift parameter of CSS matching
u	Positional curve parameter
V_{pq}	Complex Zernike polynomial with order p and repetition q
w	Width attribute of CSS arcs
$\mathbf{x}$	Shape vector of nd elements
$\mathbf{x}_i$	i^{th} shape vector
$\bar{\mathbf{x}}$	Mean shape vector
Z_{pq}	Zernike moment of order p and repetition q
$\delta(u)$	Differential turning angle function
$\delta(u, \sigma)$	Differential turning angle scale space function
ϵ	Hamming distance threshold
θ	Turning angle
κ	Curvature
ν	Targeted number of pixels in scale-normalized images
σ	Width of Gaussian kernel used in smoothing
ϕ	Differential turning angle
$\bar{\Phi}$	Basis function to map input to higher dimension
$\mathbf{\Phi}$	Eigenvectors matrix
Φ_i	i^{th} eigenvector in $\mathbf{\Phi}$
Ω_i	Output score of the 20NN classifier for the i^{th} instance

LIST OF ACRONYMS/ABBREVIATIONS

1D	One dimensional
2D	Two dimensional
AAM	Active appearance models
AR	Autoregressive
ASM	Active shape models
AUC	Area under the ROC curve
BPD	Biparietal diameter
BW	Black and white
CAD	Computer aided diagnosis/detection
CART	Classification and regression trees
CBA	Classification-based association
CBIR	Content-based image retrieval
CSS	Curvature scale space
CT	Computed tomography
DBSCAN	Density-based spatial clustering of applications with noise
DMLM	Distance matrix-like matrix
dTA	Differential turning angle
dTASS	Differential turning angle scale space
EM	Expectation maximization
FD	Fourier descriptor
FN	False negative
FP	False positive
GFD	Generic Fourier descriptor
GMRP	Geometric mean of recall and precision
KDD	Knowledge discovery from databases
kNN	k-nearest neighbor
LoG	Laplacian of Gaussian
MAT	Medial axis transform

MLP	Multilayer perceptron
MR	Magnetic resonance
MRI	Magnetic resonance imaging
NTD	Neural tube defect
OFD	Occipitofrontal diameter
PA	Posterior-anterior
PACS	Picture archiving and communications system
PCA	Principal component analysis
PRANSAC	Pseudo-random sample consensus
RANSAC	Random sample consensus
RBF	Radial basis functions
ROC	Receiver operating characteristics
ROI	Region of interest
RU	Random undersampling
US	Ultrasound
SMOTE	Synthetic minority oversampling technique
SVM	Support vector machines
TA	Turning angle
TN	True negative
TP	True positive
WD	Wavelet descriptor

1. INTRODUCTION

1.1. Motivation

Computer aided diagnosis/detection (*CADx/CADe*) aims at assisting humans in the interpretation of medical data. Most commonly, the term refers to automatized procedures that produce outputs from radiological images of body structures. Each particular CAD application has a scope of usage in a well-defined problem domain and due to the ethical constraints arising in medicine, the produced outputs can not be fully-trusted and humans are responsible for the final decision. Nevertheless, CAD systems can be employed to help specialists evaluate the huge information available within images in a short time, or may be used as agents to prevent wrong decisions.

As the name implies, CAD systems are typically designed to detect conspicuous sections/structures and identify the presence of particular defects/pathologies. The treatment plan for particular cases may be organized followed by the interpretation of medical images, both by specialists and as suggested by CAD outputs. CAD is a relatively new interdisciplinary field combining technologies from artificial intelligence and radiological and digital image processing whose primary objective is to play a supporting role in medical decision making processes. Automatic pattern recognition is essential for CAD; since the problems of detection and recognition of body structures in medical images, are studied within this area of computer science.

Medical imaging techniques; that is, the sources of inputs to CAD systems, include magnetic resonance imaging (MRI), computed tomography (CT), X-ray and ultrasound (US). Interpreting the images is the task of doctors (i.e. radiologists or experts of specific fields) and the objective is to arrive at decisions to guide any following treatment. Computer aided diagnosis/detection is the interdisciplinary field targeting to ease and improve this task. In this dissertation, we handle a specific CAD problem, spina bifida detection, attempting to solve it using US images of fetal skulls as inputs.

Pattern recognition, in general, requires representing samples with a descriptive set of features and running classification routines to assign labels to samples. For instance, the support vector machines (SVM) technique models separating hyperplanes between instances of different classes and classification is performed based on which side of a hyperplane an instance (i.e. the features) is. On the other hand, a lazy approach such as the k-nearest neighbor (kNN) scheme decides based on distances of instances to one another. The distance computation may be between equal-sized feature vectors. There may also be cases where feature vectors are of different sizes or explicit feature representation is hard/impractical. An approach to follow is to perform classification using sound representations and distance measures where feature vectors of equal size are not necessary. In the kNN method, no model fitting is required and the classifier decides based on the state of neighborhoods defined with respect to each sample.

Classification modules that operate on image inputs, such as those of CAD systems, must be supported by tools which present inputs to the module in a suitable form. Images, in their raw state, can often not serve as the input (i.e. set of features) for classification. First of all, the entire body of an image (i.e. all pixels and their attributes such as spatial coordinates and intensities) is not of interest and the portions for which the classification problem is defined and from which descriptive features or proper representations are derived must be isolated. This is the problem of segmentation to be a priori solved. In the CAD domain, the image portions that need to be transferred to the further stages of signal processing are referred to as *regions of interest (ROI)*. Following ROI detection (i.e. segmentation), features can be extracted using one of a wealthy set of techniques. The selection of the technique to use is a design decision and some may work better than others.

Figure 1.1 shows the block diagram of a pattern recognition system for computer aided detection of spina bifida. There are two possible paths to follow after input images are described as skull shapes: in (i), features extracted from a skull shape are evaluated by a classifier built on a set of model shapes (e.g. SVM) and a class label is assigned; whereas in (ii), the input shape is matched to the models in the database using the extracted features and similarity scores are computed for an appropriate

interpretation. (ii) works with the principle of similarity matching such as the lazy kNN classifier and many retrieval systems.

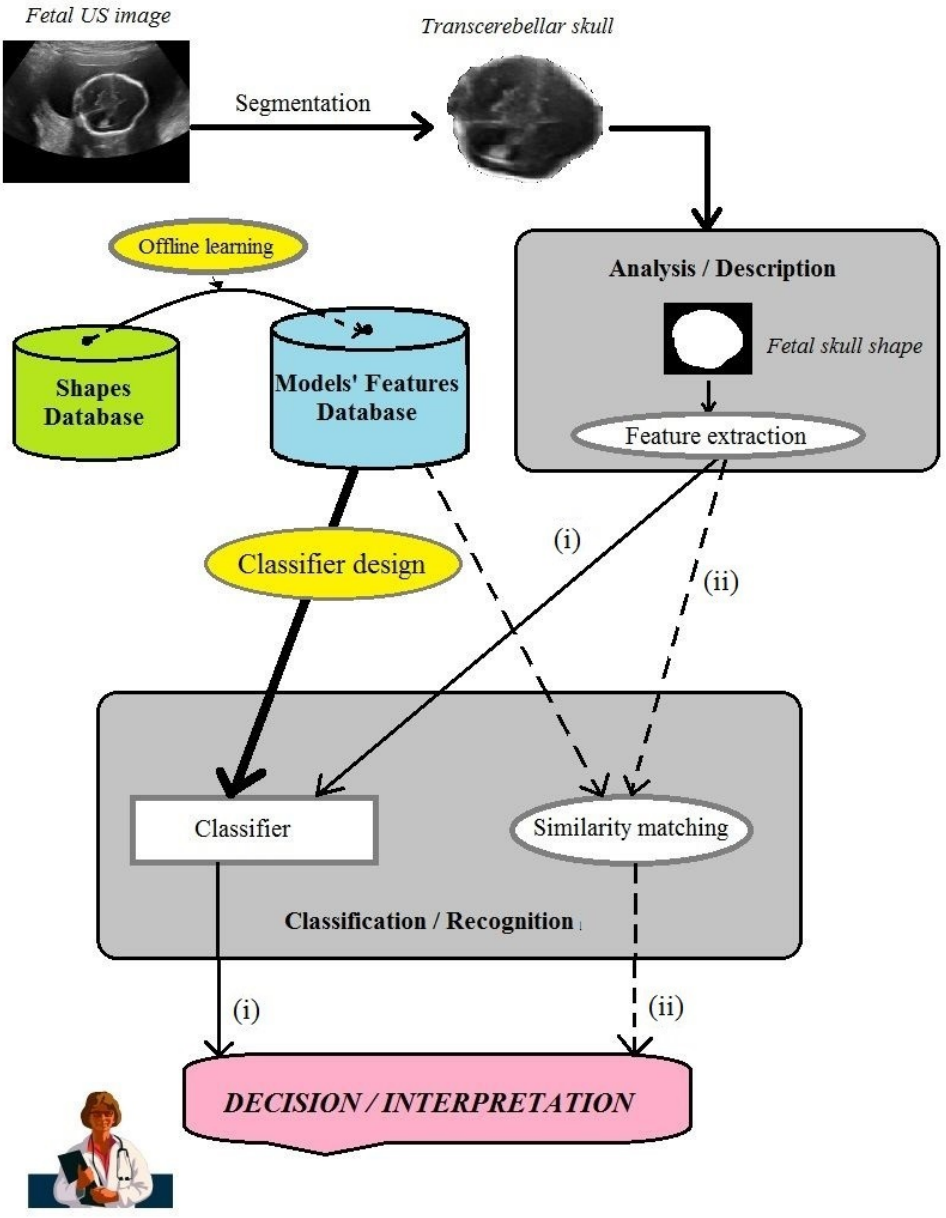

Figure 1.1. Pattern recognition (CAD) system for spina bifida detection.

Machine learning problems may be accompanied with the additional issues of rarity and class imbalance. Credit card fraud detection, detection of oil spills from satellite images, computer aided diagnosis of rare diseases are examples of such problems. Rarity may be absolute and hence the number of learning samples insufficient. It

may also be relative, standing for cases where the ratios of samples with different class labels are considerably different, to give rise to the class imbalance problem. When a machine learning system has to be designed under the rarity and class imbalance constraints, special care must be taken to handle learning data for the realization of a robust system. Various approaches for this task are available in the literature. Assessing classifier performance, especially for those designed with rarity and imbalance burdens, is demanding and must be carefully done suiting the needs of target applications. There exist popular performance measures such as precision, recall, GMRP, F-measure, etc. which can be employed when the classical measure of classification accuracy is unsatisfactory. *Receiver Operating Characteristics (ROC)* analysis is a commonly-followed procedure to evaluate classifiers at a number of operating points and to show their superiorities over others.

In summary, the realization of a successful CAD system depends on careful considerations for identifying a particular problem in a well-defined domain, selecting the input modality (type of input), the methodology used in segmentation, how features are extracted out of ROIs or how those ROIs are compactly and descriptively represented and finally the classification algorithm running with correctly-chosen parameters. Deciding how well classification performs with all the considered selections and design decisions, is perhaps the most challenging work among all. This is especially true for CAD systems which are designed using data sets that possess rarities and imbalances. As a result, one must be highly cautious when generalizing outcomes of such systems.

1.2. Spina Bifida

In this dissertation, the goal is the automatized detection of the common neural pathology called *spina bifida* (open/split spine) from ultrasound (US) images of fetal skulls acquired in prenatal terms. As the name implies, spina bifida is one of a group of defects known as *neural tube defects* (*NTD*) related to the spine and spinal cord. In neural development of embryos, a tissue called the neural plate folds and forms a tube, which then folds into the spinal cord. Incorrect folding of the neural plate causes the spina bifida defect [1], whose result is an abnormally formed section of the spinal

column. Abnormality occurs at some vertebral column location, referred to as the lesion level. People suffering from spina bifida experience loss of body control below the lesion. The higher the lesion level (the more cranial or the closer to the brain), the more severe the defect. Loss of body control may appear as problems in bladder control, sensation loss and paralysis. Figure 1.2 and Figure 1.3 show the axial and sagittal sections of defective fetal spines viewed with US. The prevalence of spina bifida is one-two cases per 1000 births worldwide and the incidence is observed to vary up to three-four cases per 1000 in some populations. Although what causes the injury is not well known; the consumption of folic acid, a type of vitamin B, by pregnant women shows to prevent up to 70% of neural tube defects including spina bifida. Spina bifida can be grouped

Figure 1.2. Axial sonogram of a fetus with spina bifida.

Figure 1.3. Sagittal sonogram of a fetus with spina bifida.

in three main types, two of which can be readily diagnosed at birth. The third type appears later in life. The least common type is called *meningocele* identified by a sac protruding from the back which contains fluid tissues that cover the spinal cord. A

more common form is *myelomeningocele* with the sac not only containing fluid but also nerves of the spinal cord. The third type, *spina bifida occulta*, is not apparent at birth and discovered much later in life. This type of the defect rarely causes a problem for 5-10% of the people who have it. Figure 1.4 shows schematic illustrations for the structures of a healthy spine (i) together with those having meningocele (ii) and myelomeningocele (iii). US examination is a convenient tool to discover neural

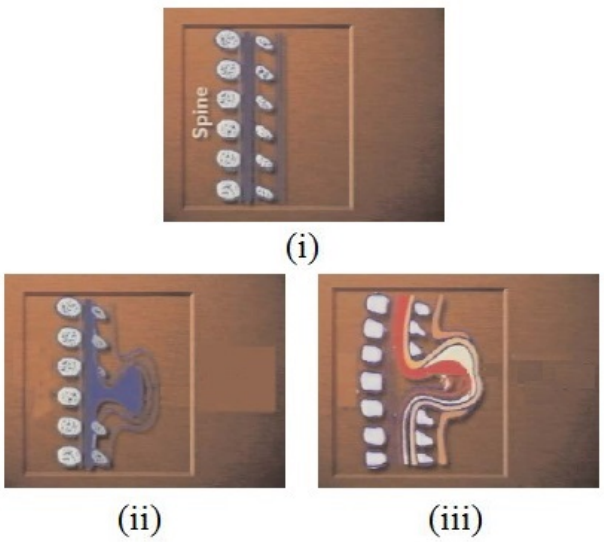

Figure 1.4. Types of spina bifida: (i) normal (ii) meningocele (iii) myelomeningocele.

tube defects in the prenatal stage. Detection of the defect before birth leads to careful planning and effective remedy. Surgical treatment after birth and fetal surgery may be possible, however, most pregnancies with neural tube defects are terminated because of the poor future quality of life of newborns having such defects. Observing the spine itself for detection is not a necessity because fetal heads contain markers indicating the presence of spina bifida. The so-called *lemon sign* [2] that appears when the frontal bones of the skull look flattened and inwardly bent, is a very typical marker. Figure 1.5 shows the transcerebellar section of a malformed skull of a defective fetus with lemon sign. The two main dimensions of a transcerebellar skull are called *occipitofrontal* diameter (*OFD*) and *biparietal* diameter (*BPD*) as shown in Figure 1.6.

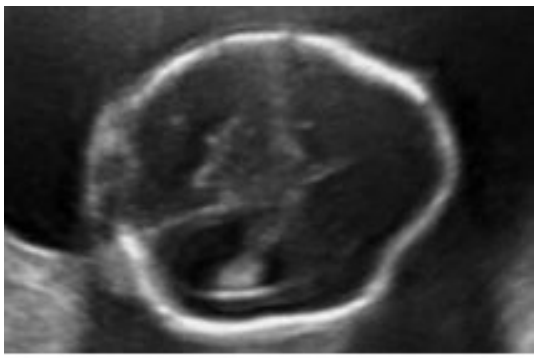

Figure 1.5. Fetal head with lemon sign.

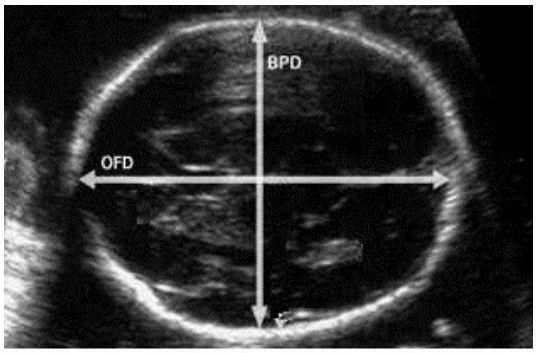

Figure 1.6. Dimensions of a skull.

1.3. Contributions

Solutions for automatically detecting spina bifida in prenatal terms have been proposed using inputs acquired by ultrasound viewing and classification has been performed via both parametric [3–5] and non-parametric [6] techniques. Although the well-defined and quite specific problem seems to be attacking at a tight-fitting target, the applications using similar constructs can be extended for many other problems and the conclusions drawn remain valid. Particularly, the related sub-problems of segmentation, feature representation and extraction, handling rare and unbalanced data together with rational quality assessment are all considered.

Our first solution approach employs the CSS representation [5,6] of contours and

the kNN (k-nearest neighbor) algorithm [6] run on those. Distance computation in kNN does not require feature vectors of equal sizes and is performed using a matching procedure (e.g. CSS matching) on appropriate representations of skulls contours. This is a similarity-based classification approach learning from neighbors and accepted as it is. It may be applicable in many contexts as well as the specific problem of the dissertation. The CSS representation has been used in image retrieval before, however our study benefits CSS in classification. The class imbalance problem is addressed with a simple oversampling trick that counts instances of the rare class more than those of the frequent class. In another variant, the balance of data sets is achieved by discarding most of the majority class samples and equalizing the number of rare class samples to that of the majority class.

In the second solution scheme, magnitudes of Zernike moments computed on black-and-white (BW (or *binary*) images [4] are used as inputs to an SVM-based classifier [3, 4]. In the BW images of fetal skulls, internal regions of skull contours are filled in with white pixels and then the moment computation is carried on. Since the invariance properties under similitude transformations of images are essential for robust recognition, the Zernike moments computations are performed on normalized images. The number of Zernike moments required in the particular classification task is determined by a simple heuristic that observes reconstruction quality (i.e. accuracy) with different numbers of moments.

Our experiments with SVMs are performed after resampling the learning data with combinations of oversampling and undersampling. We perform oversampling by creating synthetic rare-class samples and undersample the frequent class, both in the feature space with different rates. We have implemented and experimented with a particular extension of the *synthetic minority oversampling technique* (*SMOTE*) (i.e. borderline-SMOTE).

The segmentation problem is handled independently from feature extraction and classification by the proposed *semi-automatic* methods of an intensity-based averaging (a very personal heuristic) and the Active Shape Models (ASM) [7] technique adapted

to our specific problem and the type of data our CAD systems work on.

Finally, the results obtained within each setting (i.e. configuration) of experiments (either due to the methods used, different parameters for a specific method or differences in resampling) are comparatively analyzed and discussed.

1.4. Organization of the Dissertation

Background and literature survey towards solutions are provided in Chapter 2. In Chapter 3, the CSS representation of contours, in particular contours of fetal skulls, and the proposed solution employing this representation scheme are described. Chapter 4 is an elaboration on Zernike moments out of which features for parametric classification techniques are derived and whose exact computation invariant under translation, rotation and scaling is described. The details of feature acquisition for fetal skull shapes are also presented in Chapter 4. The support vector machines (SVM) classifier used in parametric classification is summarized in Chapter 5. Chapter 6 brings out the prior problem of segmentation with classifiers initiated with raw image inputs and describes proposed solutions. In Chapter 7, performance evaluation of classifiers, the rare/imbalanced data problem are considered and the strategies utilized to handle spina bifida detection with relevant customizations are explained. Experimental results with the implemented classifiers obtained for inputs of ground truth segmentations are presented in Chapter 8. Chapter 9 concludes the dissertation with discussions and future work. For comparative reference, Appendix A presents results when the inputs are acquired with the proposed semi-automatic segmentation procedures.

2. BACKGROUND

Essential basics and literature review on computer aided diagnosis and related problems are presented in this chapter. The presentation includes CAD in general; shape representation, description, feature extraction techniques; classification and segmentation. Not required from a general perspective but dealt with in the scope of our problem of spina bifida detection, we also devote a section to the rarity of classes/cases and the class imbalance problems.

2.1. Computer Aided Diagnosis

CAD has become a major area of research with computer outputs serving as *second opinion* and helping experts in the interpretation of medical images. What to expect from a CAD system is not the same as that from a physician. The performance of computers does not have to be better than or comparable to that of physicians, but is supposed to play supporting and complementary roles. Computerized analysis of medical images started in the 1960s [8–13], however, the concept of *automated computer diagnosis* evolved to *computer aided diagnosis* and systematic research began in the 1980s. The implication of the current trend of CAD is embedding successful applications in the Picture Archiving and Communications System (PACS) environment [14].

There has been an extensive amount of literature on CAD and particular applications, which is quite difficult to keep up with. Different groupings may include detections of malignant pulmonary nodules, various cancer types such as lung, breast and colon; coronary artery disease, mammographic masses, peripheral soft tissue masses, vertebral deformities, intracranial aneurysms, interval changes in bone scans and potentially many more.

Chest radiographs (X-ray) are generally used to distinguish malignant and benign pulmonary nodules. Nakamura *et al.* [15] utilize artificial neural networks in order to

estimate the likelihood of solitary pulmonary nodules and perform a computerized analysis. Aoyama *et al.* [16] and Shiraishi *et al.* [17] propose computerized schemes for distinguishing the two types of nodules. Katsuragawa *et al.* [18,19], Ishida *et al.* [20] and Ashizawa *et al.* [21,22] target diagnosing, characterizing and analyzing interstitial lung disease. Nakamori *et al.* [23, 24] analyze features of digital chest images and perform detection of anomalies based on sizes of lung and heart (i.e. cardiomegaly). Sanada *et al.* [25] handle automated detection of pneumothorax. Kano *et al.* [26], Difazio *et al.* [27], Ishida *et al.* [28], Li *et al.* [29] and Kakeda *et al.* [30, 31] use chest images and deal with detection of interval changes. Shiraishi *et al.* [32] introduce detection of lung nodules in lateral views to improve the performance of the CAD scheme using only PA (posterior-anterior) views.

Fraioli *et al.* [33] evaluate the performance of a CAD algorithm for lung cancer screening using CT images of chests on a homogeneous population having a large number of members and compare system performance to that of radiologists. Investigations on detection of breast cancer by means of mammographic images have also been considered. Freer and Ulissey [34] report the results of a prospective study of 12,860 patients in a breast center with mammography screening and computer aided detection of breast cancer. Gur *et al.* [35] show the changes in breast cancer detection and mammography recall rate after the introduction of a CAD system. Birdwell *et al.* [36], Cupples *et al.* [37], Morton *et al.* [38] and Dean and Ilvento [39] provide prospective evaluations and case studies. Rangayyan *et al.* [40] present a review of computer aided diagnosis of breast cancer. Linguraru *et al.* [41] demonstrate a CAD system for colon cancer detection from CT colonography.

Arnoldi *et al.* [42], Halpern and Halpern [43] and Kang *et al.* [44] tackle with coronary artery disease detection and offer solutions. Dominguez and Nandi [45] propose a scheme to detect masses in mammograms. Chen *et al.* [46] exploit geometric and texture features in detection of peripheral soft tissue tumors. Kasai *et al.* [47, 48] develop a method for detection of vertebral fractures on lateral chest radiographs to help physicians in early diagnosis of osteoporosis. Arimura *et al.* [49, 50] realize automated detection of intracranial aneurysms from MR angiography. Shiraishi *et al.* [51] uti-

lize a temporal-subtraction image obtained with non-linear image warping and detect interval changes in successive whole-body bone scans.

To sum up, many examples of CAD applications exist and the potential of new ones to be introduced is huge and very promising. An intuitive review of CAD systems including the history, current status and future expectations is available in the compilation of Doi [52].

2.2. Shape Description

An important visual property of objects is *shape*, out of which useful features to identify, describe and recognize objects can be derived. The spina bifida detection problem manifests itself as a shape recognition task in that shapes of transcerebellar skulls are examined to detect presence of the lemon sign marker and hence of spina bifida. In computer vision and associated recognition tasks, shape features of objects of interest in digital images are in common use to arrive at classification decisions. There exist many 2D shape representation and description techniques among which one can choose from to get an appropriate set of features to be utilized in particular recognition or retrieval (e.g. *content-based image retrieval* (*CBIR*)) applications.

In the outermost scope, shape representation schemes can be identified as either *contour-based* or *region-based*. Contour-based techniques exploit the shape boundary information, whereas region-based methods employ all the information available through all the image pixels within a shape region. Each of the two groups of techniques can further be discriminated as being *global* or *structural*. Global representation considers shapes as a whole and structural methods describe them as segments/sections. An implicit representation of entire shapes is available within global descriptions, however, for a full specification with structural descriptions, the features of the shape must be known for all segments (perhaps, mutually exclusive). It should also be noted that, whether the representation is global or structural, shape features can only be partially retrieved in cases when digital images contain only some portion of objects of interest and do not display entire shape regions. Any shape description technique derives fea-

tures from the *space* domain or a *transform* domain. The sections that follow give a brief overview of shape description approaches. Zhang and Lu [53] present a review in a richer and much broader sense.

2.2.1. Global Contour-Based Methods

Simple global shape descriptors include those such as *area*, *circularity*, *eccentricity* and *major axis orientation*. These descriptors do not have the ability to discriminate similar shapes but can only be used as filters to avoid false hits and combined with other descriptors to describe shapes.

A common practice in shape retrieval is to compute matching scores between shapes to measure how similar two shapes are. In *correspondence-based matching*, points along contours of objects to be compared are sampled and the comparison is done using the two sets of points $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$. The *Hausdorff distance* [54] between A and B is defined as

$$\text{hsdf}(A, B) = \max(\max_{a \in A} \min_{b \in B} \|a - b\|, \max_{b \in B} \min_{a \in A} \|b - a\|) \tag{2.1}$$

where $\|a - b\|$ is the norm of a and b. The disadvantages of Hausdorff distance include its sensitivity to noise and outliers in addition to the lack of invariance properties under scale, translation and rotation changes. For a matching between a model shape and another shape in an image to take place, the model shape has to be overlapped on the image with different positions, orientations and scales. On the other hand, partial matching of shapes can be performed using Hausdorff distance.

Another correspondence-based shape matching scheme proposed by Belongie *et al.* [55] uses *shape contexts*. A shape context is a global feature extracted from the set of points that identify a particular shape. To construct a shape context, all points are considered one by one and the vectors from any point P_i to all other points of the shape are found. The lengths l and orientations θ are quantized and a histogram map representing P_i is constructed. This process is repeated for all points of the shape,

the vectors are put in *log-polar* space, all the histograms are flattened, and they are concatenated to form the context of the shape. Shape matching using shape contexts is performed by a matrix-wise matching procedure and the method is an improvement over matching using Hausdorff distance.

A *shape signature* is a shape representation by a one dimensional function derived from shape boundary points. *Centroidal profile*, *complex coordinates*, *centroid distance*, *tangent angle*, *cumulative angle*, *curvature*, *area* and *chord-length* are examples of shape signature. Normalization achieves translation and scale invariance of shape signatures. Compensating for orientation changes in shape matching requires shift matching of signatures in either 1D or 2D. Due to the high matching cost and sensitivity of shape signatures to noise and slight changes in the boundary, large matching errors can arise. Direct representation of shapes using shape signatures is inappropriate and further processing is required to enhance robustness and minimize matching load. Reducing the dimensions of a signature-based boundary representation is obtainable by using *moments* [56]. Although the implementation of boundary moments is easy, the physical interpretation of high-order moments is quite difficult and intractable.

Elastic matching, where a deformed template is generated as the sum of an original template and a warping deformation, is proposed by Bimbo and Pala [57]. Shape similarity between the original template shape and an image shape is measured by minimizing a compound function consisting of strain energy, bend energy and degree of overlapping terms. Shape complexity of the template shape and correlation between the template curvature and that of the image shape are also taken into account to compute a similarity measure. The computed parameters are then classified by a back-propagation neural network. Computation and matching complexities associated with elastic matching make the technique impractical for online image retrieval. Other disadvantages of the method are the lack of rotation invariance of shape descriptions and the arbitrary nature of warping.

Describing shapes in the spectral domain is preferred to overcome the problems of noise sensitivity and boundary variations. Spectral transforms such as *Fourier* and

wavelet provide shape descriptors derived from 1D shape signatures. Fourier descriptors (FD) are in common use for closed contours [56, 58]. The variations include FD for partial boundaries [59] and affine-invariant representations [60, 61]. Fourier invariants that describe the rotational symmetry of shapes is introduced by Granlund [62]. An FD scheme that can describe disjoint or articulated contour shapes is presented by Rauber [63]. Instead of conventional distance measurement techniques for computing the distance of two sets of FDs, Richard and Hemami [64] introduce a complex distance measurement (i.e. true distance). True distance measurement requires two Fourier transforms for a matching and spends considerably more time than an ordinary distance computation. FDs, particularly used for character recognition and object classification, possess several advantages such as simple computation, each descriptor having a specific physical meaning, normalization and matching simplicity and the ability to capture both global and local features. Wavelet descriptors (WD) used in shape representation [65–67] are both spatially and spectrally multi-resolution. This is an advantage over FDs, however, the increase in spatial resolution sacrifices frequency resolution. WDs are impractical for online shape retrieval because of complicated matching consisting of a large number of operations and the fact that matching is also dependent on shape complexity.

Time-series models (especially *autoregressive* (AR) modeling [68]), based on stochastic modeling of a 1D function obtained from the shape, are used to obtain shape descriptors. When complex boundaries are involved, few AR parameters are insufficient for adequate descriptions and an empirical decision concerning the number of parameters has to be taken. Furthermore, the associated physical meanings of AR model coefficients are not clear.

The scale space method produces shape signatures by tracking the positions of *inflection* points in shape boundaries at successively decreasing scales, where the boundary becomes smoother as the width (σ) of a Gaussian low pass filter increases. Spatial domain methods are usually accompanied with noise sensitivity and boundary variation problems and thus the use of the scale space method is justified. Inflection points that remain present for a large number of scales in the representation are supposed to

reflect significant object characteristics. The end result of a scale space description is known by several names such as *interval tree*, *fingerprint* and *scale space image*. Interval trees and their associated branches are first interpreted by Asada and Brady [69,70]. The interpretations are mainly based on detecting the peaks of the tree branches from coarse scales towards fine scales. Mokhtarian and Mackworth [71] present the application of scale space for shoreline registration and call the acquired signature *curvature scale space* (*CSS*) contour image. The detected peaks are generally not considered as realizations of primitive events, but are used in the matching of shapes. Mokhtarian *et al.* [71–73] and Abbasi *et al.* [74] later extend the method to shape retrieval.

2.2.2. Structural Contour-Based Methods

Structural shape description methods work by breaking the shape boundary to a number of primitives or segments whose combination provides a full description of the whole boundary. Conceptually, each segment is represented as a string symbol which includes attributes of the associated segment. The concatenation of the symbols s_i for all segments forms the string $\dot{S}$ that is the description of the boundary:

$$\dot{S} = s_1 s_2 \dots s_n \tag{2.2}$$

Chain code, introduced by Freeman [75], describes an object by a sequence of unit-size line segments with a given orientation. The method allows encoding arbitrary geometric configurations where curves are represented as a sequence of unit-size vectors each with one of a limited set of directions. A general chain code [76] is one with $N = 2^k$ possible vector directions. Chain codes are normalized with respect to the selection of the first boundary pixel when they are used for matching. Rotational invariance of chain codes through cyclic permutations can be achieved by representing the boundary using differences of successive directions instead of relative directions and selecting the permutation with the smallest number. Although scaling objects to the same size is a straightforward matter, scale invariance of chain codes is a troublesome issue since scaling changes code length and makes comparison of two shapes impractical. Chain

codes have high dimensionality and noise sensitivity and are generally not used as a feature of last stage but fed as input to a higher-level analysis.

Polygon approximation [77,78] breaks down a shape boundary into line segments and the vertices of the polygon are used as primitives. Each primitive is associated with a 4-element feature vector containing internal angle, distance from the next vertex, x and y coordinates. Similarity of two shapes is measured as the *editing distance* between two feature strings. Mehrotra and Gary [79] represent a shape as a chain of vectors, where a set of interest points for a shape are detected from the polygonal approximation of the boundary. Polygonal approximation lacks translation, rotation and scale invariance and its application to natural objects is impractical. Berretti *et al.* [80] propose another type of approximation using a set of tokens retrieved by detecting the curvature zero-crossings of a smoothed version of the object boundary. Each token is a curve segment and represented with its maximum curvature and orientation. The similarity of tokens is measured by a weighted Euclidian distance. Since orientation of tokens are included in feature sets, the representation is not rotation invariant.

Dudek and Tsotsos [81] obtain shape primitives from a curvature-tuned smoothing technique, analyze shapes in scale space and use a model-by-model matching scheme. Each segment is described by its length, an integer-valued position and a curvature-tuning parameter. Shape description is through the concatenation of all string descriptors belonging to the shape. A model-by-model matching of shapes is performed with dynamic programming. Shape features are treated in a curvature scale space so as to perform matching at different scales. The inclusion of segment lengths make the representation scale-variant. The matching algorithm uses three empirical or adhoc parameters and limits the applicability of the approach.

In syntactic analysis of shapes [82], the inspiration is the similarity of natural shape compositions to language compositions. Languages are formed by sentences built from phrases which are formed by words that consist of characters in an alphabet. Syntactic analysis of shapes, similarly, uses a set of primitives for shape description. The set of predefined primitives is called a *codebook* and each primitive is a *codeword*.

Matching of shapes employs string matching rules. The disadvantage with the syntactic approach is the dependency of alphabets to applications and shape databases in consideration.

Although the desired invariance properties of scale, translation and rotation can be attained with other description techniques, they depend on viewpoint [83]. Techniques that employ *shape invariants* aim to acquire boundary properties that remain unchanged under a more general class of transformations. Various examples of invariants include geometric invariants such as *cross-ratio*, *length ratio*, *distance ratio*, *angle*, *area* [84], *triangle* [85], *invariants from coplanar points* [83]; algebraic invariants such as *determinant*, *eigenvalues* [86], *trace* [83]; differential invariants such as *curvature*, *torsion* and *Gaussian curvature*. Geometric and algebraic invariants are useful in applications where boundary segments can be represented by lines or algebraic curves. Differential invariants, that are local and large in number, can be formed where geometric and algebraic invariants are not applicable.

2.2.3. Global Region-Based Methods

One main class of region-based shape description methods contains moment invariants. Hu [87] presents the first work on image moment invariants to be used in 2D pattern recognition. The general form of a regional (i.e. 2D) moment is

$$m_{pq} = \sum_{x} \sum_{y} x^p y^q f(x, y) \tag{2.3}$$

where $p, q = 0, 1, 2, \ldots$ are moment orders for x and y dimensions. $f(x, y)$ is a 2D function defined on regional positions (x, y) (e.g. the image intensity function). *Geometric moment* invariants, having the desired properties of invariance with respect to scale, translation and orientation, are derived by using non-linear combinations of lower order moments. Using higher order moments has not been addressed in pattern analysis and recognition. Small values of computed moments generally require a normalization prior to their use in applications. Only a few moment invariants from lower order moments

can be acquired and they are not sufficient to describe shapes effectively. The derivation of higher order moments is difficult. Algebraic moment invariants introduced by Taubin and Cooper [88,89] are computed from the first m central moments. They can be computed up to an arbitrary order, are invariant under affine transformations and work either very well or very poorly on different objects. When pixel distributions of objects is important rather than their outline, the performance is high; however, poor performance is obtained when outline configurations are important.

The idea of replacing $x^p y^q$ in the algebraic moment transform of Equation 2.3 by general kernels $P_p(x)$ and $P_q(y)$ is used by Teague [90] to introduce *orthogonal moments* of *Legendre* and *Zernike*. Legendre and Zernike polynomials used for replacing $x^p y^q$ are both complete sets of an orthogonal basis and that is why Legendre and Zernike moments are called orthogonal moments. *Pseudo-Zernike moments* are another type of orthogonal moments obtained by using real-valued radial polynomials in Zernike polynomials as the moment transform kernel. Optimal utilization of shape information and accurate reconstruction of shapes is achieved by orthogonal moment transforms. In the detailed study of Teh and Chin [91]; concerning orthogonal Legendre, Zernike, pseudo-Zernike moments and non-orthogonal geometric, complex, rotation moments; it is shown that geometric moments, complex moments and pseudo-Zernike moments are affected less by noise than Legendre moments which are severely affected. The reconstruction power of Zernike and pseudo-Zernike moments is more than that of Legendre moments both for noisy and noiseless images. The reconstruction error for noisy images reaches a minimum value and starts to increase as the number of used moments increases, indicating the lower reliability of higher order moments in noisy environments. Liao and Pawlak [92] show that coarser quantization (i.e. less resolution) of images produces more accurate moments. Shape description using moments is generally concise and robust. Computations of and shape matching with moments are easy, however, associating precise physical meanings for higher order moments is difficult.

Zhang and Lu [93] propose a *generic Fourier descriptor* (*GFD*) for shapes, acquired by applying a 2D Fourier transform on polar-raster sampled shape images.

Shape similarity is measured using the *city block* distance of GFDs (i.e. normalized coefficients of the GFD transform). GFD features are pure spectral and their retrieval performance is better than that of Zernike moments due to the multi-resolution analysis involved.

Lu and Sajjanhar [94] introduce a grid-based shape descriptor obtained by overlaying a grid of cells on a shape, scanning the cells from left to right and top to bottom and marking the cells with 1 or 0 depending on whether a cell is occupied or not. The resulting bitmaps are then used in measuring the similarity of two shapes with a metric such as *Hamming distance*. The grid descriptor is simple and intuitive but rotation normalization performed based on the major axis orientation is sensitive to noise and unreliable. Goshtasby [95] uses a *shape matrix* derived from a circular raster sampling. Concentric circles and corresponding radial lines are overlaid on a shape and the intersection points of circles and lines constitute shape matrix entries.

2.2.4. Structural Region-Based Methods

For any two points P_1 and P_2 in a region $\tilde{R}$, if the line segment P_1P_2 is inside $\tilde{R}$, $\tilde{R}$ is called a *convex region*. The *convex hull* of $\tilde{R}$ is the smallest convex region $\tilde{H}$ satisfying the condition $\tilde{R} \subseteq \tilde{H}$. The difference region $\tilde{H} - \tilde{R}$ is called the *convex deficiency* of $\tilde{R}$. To obtain a convex hull representation of a region, the region boundary is first smoothed to eliminate undesired effects of digitization, noise, segmentation variations; its convex hull is extracted [56, 83] and the convex hull is recursively partitioned to detect deficiencies and convex hulls at lower levels. Each recursive step is composed of finding the convex hull of an extracted deficiency at a previous step and further detecting the deficiencies along the boundary defined by the extracted deficiency and its corresponding convex hull. Recursion ends when no convex deficiencies for some recursive step can be detected. The shape representation is by means of a concavity tree, whose root is the shape region $\tilde{R}$ and whose all leaves correspond to subregions that can no longer produce convex deficiencies (i.e. no more concavity can be detected). Concavities can be described by their areas, bridge (line cut of the concavity) lengths, maximum curvature values, distances between maximum curvature points to bridges,

etc. Shape matching can be handled as a string matching or graph matching problem.

Methods to represent shape regions as *skeletons*, defined as connected sets of medial lines along branches of regions, aim to eliminate redundancy and use only the topological information related to object structure. Blum [96] discusses skeleton methods as the *medial axis transform* (MAT).

2.3. Classification

The target of a classifier application is identifying input data as a member of one of a set of categories (i.e. *classes*/*labels*/*concepts*). This classification problem is examined in the more general scope of *machine learning*. Machine learning is actually programming computers to perform actions that humans can do with or without knowing the exact biological procedure carried out by the brain. To make computers imitate the decision processes of human beings is through optimizing some performance criterion using example data or past experience [97]. Alternatively, machine learning is the scientific discipline studying how the data we observe can be explained. Although human beings (i.e. experts of some field) are supposed to be more talented than computers to arrive at decisions, such processes may contain extensive amount of data to handle and be impractical. It is useful to extract knowledge from big data collections (also called *data mining* and *knowledge discovery from databases* (KDD)) and represent this knowledge by simple and explanatory models. The theory of statistics serves as the core component utilized in most machine learning applications to construct mathematical models that explain the observed data. There are many fields such as forensics, psychology, agriculture, multimedia, biometrics, medical imaging, robotics, etc. where machine learning applications are benefited from. *Pattern recognition* applications are those which operate by analyzing data obtained from the members of a category, capturing the pattern specific to those members and recognizing objects of interest by checking this pattern. When there are two or more classes of data to identify in some sample space, the machine learning problem is referred to as *classification* and the associated computer program is a *classifier*. Machine learning is not only composed of fitting mathematical models to data but also contains components

of artificial intelligence. The systems must be intelligent (i.e. must learn) to adapt to changing environments. The mathematical models defined up to some parameters are optimized using *training* data. The applications produced by the trained models may be used as either *descriptive* or *predictive*, depending on whether the objective is to gain knowledge from data or to make predictions with new data.

There exist a variety of approaches that have been applied in the literature for constructing classifiers. From a top view, all of them are evaluated based on the criteria of *accuracy*, construction and classification complexities, *robustness*, *scalability* and *interpretability*. Accuracy is the ability of a classifier to make correct predictions on data. The time and space used in generating classifiers and using them to predict labels of samples is an issue related to the computational cost of these processes. The ability of producing correct predictions with noisy or missing data indicates the robustness of classifiers. Efficient classifier construction and prediction using large volumes of data are considered as the scalability property. Finally, interpretability is the level of understanding and insight provided by classifiers. Interpretability is a rather subjective issue and may be difficult to assess. In the following flow, we shortly mention "some" of the popular classification approaches.

Class-labeled training data may be used to learn *decision trees*. Decision trees are tree structures where internal nodes denote some test on a data attribute, branches are paths followed according to the result of the test and leaf nodes are class labels. Building decision trees does not require any domain knowledge and the learned concepts (i.e. rules) are appropriate for exploratory knowledge discovery. During each step of tree construction, the attribute that best partitions the input data arriving at a node to distinct classes is selected and the test is performed on that attribute. Quinlan develops and presents the decision tree algorithms of ID3 [98] and C4.5 [99] as benchmarks to which *supervised* algorithms designed later are compared. Breiman *et al.* describe the generation of binary decision trees in their book *Classification and Regression Trees* (*CART*) [100]. Decision trees are used for rule extraction by traversing paths from the root node to each leaf node. The extracted IF-THEN rules consist of *antecedent* parts (i.e. IF) formed by logically ANDing the splitting criteria along given paths and

consequent parts (i.e. THEN) holding class predictions in leaf nodes.

Interesting relationships between attribute (input data or features) conditions and class labels can be characterized by association rules. With frequent patterns, these association rules can be used for effective classification. An association rule is of the form X $\rightarrow$ Y, meaning that the realization of X implies the realization of Y. In *associative classification*, associative rules are generated and analyzed for use in classification. Strong associations between frequent patterns (i.e. conjunctions of attribute-value pairs) and class labels can be discovered and classification rules can be generated. Since highly-confident associations among multiple attributes are explored by association rules, some constraints of decision tree induction such as considering only one attribute at a time, are overcome with this approach. Liu *et al.* [101] present the first and simple algorithm (*Classification-Based Association* (*CBA*)) to perform associative classification.

Bayesian classifiers are statistical classifiers that can predict class membership probabilities. Bayesian classification is based on *Bayes' theorem* given in Equation 2.4,

$$\Pr(\mathrm{H}|\mathrm{X}) = \frac{\Pr(\mathrm{X}|\mathrm{H})\Pr(\mathrm{H})}{\Pr(\mathrm{X})} \tag{2.4}$$

where Pr(H|X) is the *posterior* probability of H conditioned on X. Similarly, Pr(X|H) is the *posterior* probability of X conditioned on H; Pr(H) and Pr(X) are the *prior* probabilities of H and X, respectively. If H is considered to be the realization of a particular class membership, Pr(H|X) can be computed using estimated values of Pr(X|H), Pr(H) and Pr(X). In the Bayesian classification context where the posterior probability Pr(H|X) is to be computed with Pr(H) being the prior probability of H not conditioned on any other event, Pr(X|H) is called the class *likelihood* of input data X when the belonged class is indicated by H and Pr(X) is called the *evidence* standing for the marginal probability of the occurrence of X. When X is multidimensional, dependencies among components of X (i.e. x_i) may exist. Performing sound classification using the Bayes's rule in such cases relies on constructing graphical models showing the dependencies/interactions among input variables and training the model with appro-

priate algorithms. These graphical models are referred to as *Bayesian networks*, *belief networks* or *probabilistic networks* [102]. In *naive Bayes classification*, all input variables are assumed to be conditionally independent and the multivariate classification problem can be handled as a group of univariate problems.

Artificial *neural networks* [103], whose underlying concept has originally risen in the fields of psychology and neurobiology, have been used in computer science for classification tasks. A neural network is a set of connected input/output units where each connection has a weight associated with it. Learning of neural networks is accomplished by adjusting the weights in order to correctly predict the labels of input data. In the simplest case of a *perceptron* model, the units of the network are either *input* units or the *output* unit. In the general case, a set of perceptrons can be connected in serial or parallel fashion to form a neural network of some topology. The inputs of some perceptrons may be the outputs of some other perceptrons arranged to perform a specific task (in particular, classification). The general network topology is known as a *multilayer perceptron* (MLP), which has *hidden* or *intermediate* layers in addition to input and output layers. An MLP is able to implement nonlinear *discriminant*s used for classification. Neural networks, trained using the *backpropagation* algorithm, require long learning times and therefore feasibility is a concern prior to their use. Although having been criticized for their poor interpretability, their high tolerance on noisy data and skill to classify previously unseen patterns are the advantages of neural networks.

Support vector machines (SVM) classification is a popular method that uses a linear discriminant to separate the instances of two classes. The method, although linear in nature, works both for linear and nonlinear data. This is accomplished by using a nonlinear mapping to transform the original training data to a higher-dimensional space where separation is performed by means of a linear optimal separating hyperplane. Nonlinear mappings of input data to a sufficiently high dimension ensures that separation can be performed linearly. Optimal separating hyperplanes (i.e. decision boundaries) are found using *support vectors*, which are those training samples of either class closest to the hyperplane from both sides. The distance between the closest in-

stance to the hyperplane at one side and the closest instance to it at the other side is called *margin*, which one wants to maximize for better generalization ability. SVMs are highly accurate due to their ability to model complex nonlinear decision boundaries and support vectors provide a compact description of the learned model. Historically, Boser *et al.* [104] present the first paper on support vector machines in 1992. Nevertheless, the groundwork on statistical learning theory by Vapnik and Chervonenkis [105] has been around for more time. SVMs have been applied in various fields such as handwritten digit recognition, speaker identification, object recognition, etc.

In contrast to the *eager* classification techniques presented so far in this section, the *k-nearest neighbor* (*kNN*) classifier is a *lazy* approach that does not construct a model for classification. A database of training instances is stored and nothing is done for generalization until a new instance (i.e. test tuple) to be classified arrives. Nearest neighbor classifiers learn based on similarity of new instances to those in the training set. The kNN classifier returns the most similar (i.e. nearest) k training tuples to the test tuple and the decision of which class the test instance belongs to is given based on a score such as the majority vote or weighted counts of the neighbors (e.g. closer training instances are counted more than farther ones). The closeness of two tuples can be defined in terms of a distance metric such as the Euclidian distance. The Euclidian distance $\text{dist}(\tilde{X}_1, \tilde{X}_2)$ of tuples $\tilde{X}_1 = (\tilde{x}_{11}, \tilde{x}_{12}, \ldots, \tilde{x}_{1n})$ and $\tilde{X}_2 = (\tilde{x}_{21}, \tilde{x}_{22}, \ldots, \tilde{x}_{2n})$ with n attributes each is given in Equation 2.5:

$$\text{dist}(\tilde{X}_1, \tilde{X}_2) = \sqrt{\sum_{i=1}^{n} (\tilde{x}_{1i} - \tilde{x}_{2i})^2} \tag{2.5}$$

Of course, the selection of the k parameter is usually empirical and sometimes may require additional insight such as domain knowledge, etc. Considering the number of training samples, the kNN classifier (generally all lazy classifiers) are labor-intensive and their popularity may be attributed to the availability of high computing power. Although such classifiers offer little explanation for the structure of data, they support incremental learning and model complex decision surfaces not as easily describable by other learning algorithms.

2.4. Segmentation

Isolating image pixels that are of particular interest with the purposes set by predefined tasks and generally in the context of specific automatized applications is known as the problem of *image segmentation*. In fact, there is an awful lot of data contained in images acquired in raw state that are either redundant, useless or even task-complicating when dealt with. To obtain compact representations describing only the "interesting" portions of images (i.e. objects or regions of interest (ROI)) out of which descriptively useful features are extracted, the segmentation step must be carried prior to further steps of image processing, feature extraction, pattern recognition, classification, etc. Segmentation subdivides an image into its constituent regions or objects, where the definition of a region (or object) is strictly dependent on problem specification which also includes the level to which subdivision is carried. The general principle is stopping segmentation when objects of interest have been isolated from their surroundings.

Image segmentation [56, 106–108], excluding simple cases when image structure can be well-modeled, is one of the most difficult tasks in computer vision that remains unsolved. Measuring the success of a segmentation procedure is not a trivial issue in general, may be taken as subjective to human perception and may also be associated with the performance of computer analysis tasks that follow. Due to the subjectivity of success measures and the dimness of having a single segmentation routine working well for all possible problems, it is natural to conclude that no formal theory for segmentation exists. Classifying segmentation algorithms based on some criteria is yet another uneasy and impractical issue. From a broad perspective, segmentation procedures attempt to exploit discontinuities and similarities of image intensity values. We now present a summary on methodologies utilized in a variety of image processing and computer vision applications. The methods apply to monochrome images, however generalization to color images is usually possible.

2.4.1. Discontinuity Detection

Since segmenting images aims at locating interesting objects which are usually marked by the presence of simple geometric entities; the detection of primitives such as points, lines and edges appears as a non-avoidable step in many segmentation tasks. The simplest way to follow for detecting intensity discontinuities at a single pixel is to run a mask on the image window that encompasses the pixel of interest. The response of a mask is usually defined with respect to its center which is positioned on the particular image pixel whose response is sought. Equation 2.6 depicts the response $\dot{R}$ of a particular image pixel to a 3x3 mask which is centered on the pixel,

$$\dot{R} = \sum_{i=1}^{9} c_i v_i = c_1 v_1 + c_2 v_2 + \ldots + c_9 v_9 \tag{2.6}$$

where v_i is the intensity of the pixel associated with mask coefficient c_i. Decisions regarding the value of $\dot{R}$ are taken after comparison against a threshold τ.

Figure 2.1 displays 3x3 masks to detect isolated points (i.e. pixels) and lines of one-pixel thickness oriented horizontally, vertically, diagonally at 45° and −45°. It should be noted that the weights of each mask sum to zero to guarantee that $\dot{R}$ values will be 0 for regions of constant intensity and larger weights are used for preferred directions of lines. Being more general than points and lines, edges are the essential structures that signal meaningful discontinuities in images. Edges are detected using the first and second order derivatives of the image intensity function $f(x, y)$ where x and y stand for image point coordinates (i.e. pixels). The first order derivative used in image processing is called the *gradient*. It is the vector $\nabla \mathbf{f}$ defined as

$$\nabla \mathbf{f} = [G_x \, G_y]^T = \left[\frac{\delta f}{\delta x} \frac{\delta f}{\delta y}\right]^T \tag{2.7}$$

with magnitude

$$\nabla f = \sqrt{{G_x}^2 + {G_y}^2} \tag{2.8}$$

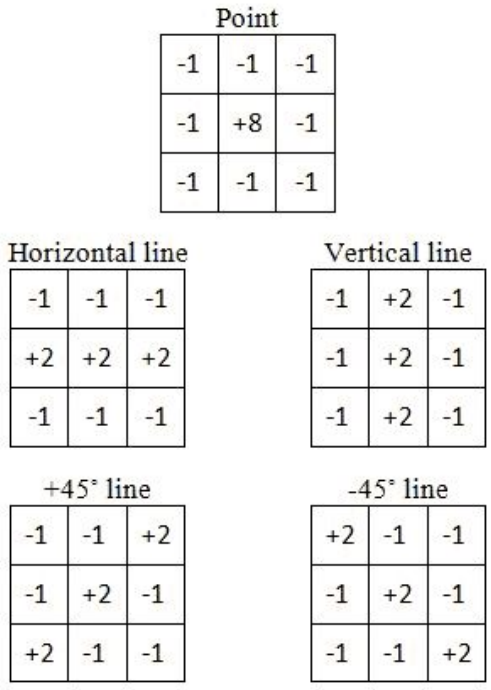

Figure 2.1. Masks for point and line detection.

which sometimes is approximated by omitting the square root operation or by adding the absolute values of the vector components. The approximations still act as derivatives, because their values are zero in constant-intensity regions and proportional to the degree of intensity change in areas of varying intensity. The gradient vector points in the direction of maximum rate of change of f. The angle α of this rate of change is

$$\alpha = \tan^{-1}(G_y/G_x) \tag{2.9}$$

and the magnitude of the gradient vector is simply referred to as "the gradient". Although the second-order derivatives of f can also be used in edge detection, it has disadvantages such as noise sensitivity, detecting double edges and inability to detect edge directions which make their direct utilization in edge detection unsuitable. It is often the case that locations where the magnitude of the first order derivative (gradient) is greater than a specified threshold are found and marked as edge pixels.

Estimating the first order derivatives G_x and G_y of the image function $f(x, y)$ digitally is achieved via the sets of masks shown in Figure 2.2. That is, the gradient value at the center pixel of a neighborhood is computed by running the masks on the neighborhood. Sobel [109] and Prewitt [110] edge detectors use 3x3 masks whereas Roberts [111] detector works on 2x2 windows. The *Laplacian of Gaussian* (*LoG*)

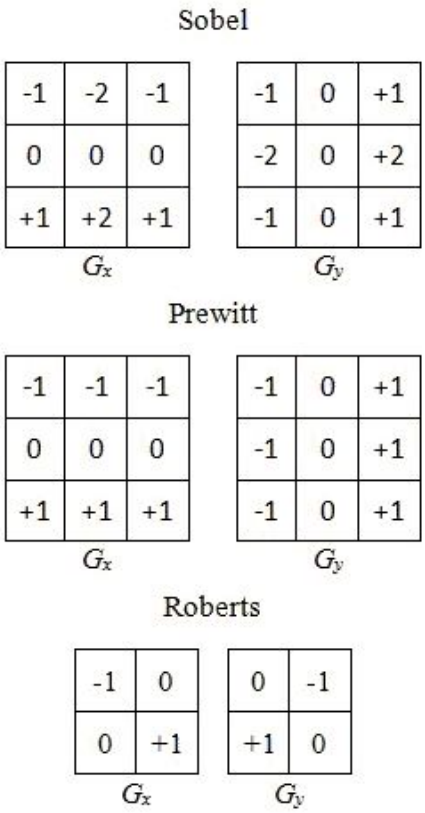

Figure 2.2. Masks for edge detection.

edge detector works by first convolving the image intensity function with the second derivative (i.e. Laplacian) of a Gaussian function in order to smooth it and yield a double-edged image, and then by detecting the zero-crossings between double edges to locate the edges. Alternatively, a zero-crossings edge detector is based on the same concept as a LoG detector but uses a possibly different filter for smoothing (i.e. it is the generalization of the LoG detector using a non-Gaussian filter).

Another popular and effective edge detector is proposed by Canny [112] in 1986.

An input image is smoothed with a Gaussian filter to reduce noise. The gradient magnitude and direction at each pixel of the smoothed image is computed using one of the masks of Figure 2.2. The pixels which have locally maximum gradient magnitudes in the direction of associated gradient vectors are identified as edge pixels. The gradient magnitude image of the edge pixels contains ridges. The algorithm tracks along the top values of these ridges, those pixels that are not actually on ridge tops are set to zero (i.e. *nonmaxima suppression*) and thin lines in the output image are obtained. Thresholding this image using two threshold values classifies ridge pixels as either strong or weak edge pixels or not an edge pixel at all. Finally, edge linking is performed by incorporating weak edge pixels that are 8-connected to strong edge pixels.

2.4.2. Similarity-Based Techniques

Testing intensities of image pixels against particular values and acting based on whether this value is exceeded or not is known as *thresholding* and can be considered as the simplest approach to obtain two groups having similar pixels in each. Selecting threshold values can be performed by detecting the modes in histograms visualizing intensity distributions. Mode selection is achieved either visually by inspecting histograms or by an automatic procedure such as that of Otsu [113]. Having L discrete intensity values in the range $\{0, 1, \ldots, L-2, L-1\}$, Otsu's method exhaustedly searches for the threshold value τ that minimizes the weighted within-class variance ${\sigma_W}^2$, equivalently that maximizes the between-class variance ${\sigma_B}^2$ of the two groups of pixels. Pixels having intensity values in $\{0, \ldots, \tau-1\}$ form one group and those in $\{\tau, \tau+1, \ldots, L-1\}$ form the other group. σ_W^2 and σ_B^2 are defined as

$$\sigma_W^2 = \Pr_1(\tau)\sigma_1^2 + \Pr_2(\tau)\sigma_2^2 \tag{2.10}$$

$$\sigma_B^2 = \sigma^2 - \sigma_W^2 \tag{2.11}$$

where $\Pr_1(\tau)$ and $\Pr_2(\tau)$ are the probabilities of the two groups obtained with τ, σ_1^2 and σ_2^2 are their variances, and σ^2 is the variance of all pixels. *Global thresholding* fixes τ for all pixels whereas *local thresholding* lets τ vary within image regions.

Clustering refers to partitioning a set of vectors into groups having similar values. As applicable to image segmentation, the vector components may be intensities, color values, texture measurements or any property derived from the former. Once vectors in the measurement space have been obtained and grouping performed with appropriate clustering techniques, connected regions can be obtained via *connected components labeling*. In *K-means clustering*, there are K clusters, $\mathbf{C}_1, \ldots, \mathbf{C}_K$, with their associated means, $\mu_1, \ldots, \mu_K$ and assigning cluster indices to vectors tries to minimize an error function E defined as

$$E = \sum_{k=1}^{K} \sum_{x_i \in \mathbf{C}_k} ||x_i - \mu_k|| \tag{2.12}$$

where x_i are the measurement space vectors. One of the natural algorithms for clustering is divisive clustering that starts with a single cluster and splits them. The other is agglomerative clustering initiated with each vector being a separate cluster and merging them iteratively at successive steps. Perhaps the most popular clustering algorithm is *iterative K-means clustering*, which uses Equation 2.12 explicitly at each iterative step to refine the assignment of vectors to clusters and update the means until convergence. *Isodata clustering* is another iterative technique assuming the existence of K clusters employing a split-and-merge strategy. Histogram-based clustering techniques operate on the data as a single pass and the retrieved modes manifest themselves as clusters. Ohlander *et al.* [114] propose a histogram-based clustering technique, where mode seeking is applied first on the entire image and then recursively on the obtained clusters until the splitting of clusters can no more be carried. Shi and Malik [115] formulate image segmentation as a graph partitioning problem that reduces to an eigenvector/eigenvalue problem.

The objective of segmentation can be stated as partitioning an image into regions.

Region growing is the term used for the process of starting with one or more pixels determined manually or by an automatic procedure (i.e. seeds) and appending to each seed those neighboring pixels that are similar with respect to some criteria. Different from general clustering, region growing takes into account the connectivity relations of pixels during its operation. A stopping rule for growth must also be specified. Basically, region growing should stop when no more adjacent pixels to a region satisfy the determined growth criteria. Alternatives to growing regions starting with seed points include allowing *splitting* and *merging* of regions. If at any point two connected regions satisfy the common criteria used for growing, then it is sensible to merge these two regions into a single region. On the contrary, conditions that necessitate splitting regions could hold.

Other techniques based on similarity work using various ideas. *Background subtraction* is an approach that can be used when objects of interest lie on a stable background. Excluding the background pixels from the image would then isolate sought objects/regions. Computing distance between corresponding images/frames could be performed via adhoc methods of frame differencing, histogram-based comparison, block comparison, edge differencing, etc. All these methods compare pixel features such as gray-level/color intensities or higher level structures such as edge maps from two images. The spatial insensitivity of histograms is treated with placing grids on images, organizing them as a set of blocks much smaller in size and reducing the degree of insensitivity using a block-by-block comparison.

2.4.3. Fitting Models

Groups of pixels may belong to a simple family of components and it may be desired to detect such components for further analysis. Lines, circles, ellipses, etc. might be of interest and mathematical models could be exploited to either reveal their existence or locate them accurately in images. Vital image characteristics are usually available by means of edges which can rarely be detected completely because of noise, nonlinear illumination and other effects that introduce random intensity discontinuities. Similarly, isolated pixels that belong to components described by particular

mathematical models may be at hand but exact structures including some pixels and excluding others may not be visible. Such situations require explicit representations of the underlying structures. Estimating the parameters of a model for a group of pixels, or identifying those pixels that belong together for some model is called *fitting*.

The *Hough transform* technique records all structures on which points can lie and selects the structures getting high votes. This is by considering the fitting problem not in the pixel coordinate space but in the parameter space. Each image pixel in consideration has the potential to lie on infinitely many structures defined by parameter values used for model description. For all points and a subdivision of the parameter space to accumulator cells each standing for a tuple of parameter values, those cells that define the structure on which the point can lie are determined and the vote for the corresponding cell (initialized to zero at the beginning) is incremented. When this procedure ends, the accumulator cells having high votes manifest themselves as the structures to be selected. The quite general technique can be applied for detecting lines, circles and to any type of curves having analytic equations of the form $\mathbf{f}(\dot{\mathbf{x}}, \mathbf{a}) = 0$ where $\dot{\mathbf{x}}$ are vectors denoting image points on the curve and $\mathbf{a}$ is the vector of parameter values. A range of $\mathbf{a}$ values with proper quantization is determined and the accumulator array $A[\mathbf{a}]$ is initialized to zero. Each $\dot{\mathbf{x}}$ is considered, the corresponding set of parameter vectors $\mathbf{a}$ satisfying $\mathbf{f}(\dot{\mathbf{x}}, \mathbf{a}) = 0$ are determined and the accumulator array values $A[\mathbf{a}]$ are incremented. Local maxima in A (parameters having high votes) correspond to the curve function $\mathbf{f}$.

Fitting has practical limitations such as missing data points, outliers, high dimensionality of data and parameters, error definition, additional constraints for fitting, etc. These factors may occasionally turn fitting worthless and the robustness may be questionable. The target of using probabilistic methods in curve fitting and introducing a dimension of uncertainty to the process is to improve the quality of fits by avoiding the undesired effects of the aforementioned problems, to some extent. Assuming that fitting must be done with missing (i.e. incomplete) data, the fitting process (e.g. fitting a line) to an incomplete set of points can follow the lines of the *Expectation-Maximization* (*EM*) [116, 117] algorithm, where at each iterative step an expectation for complete

data using incomplete data and current parameter values is computed followed by re-estimating parameter values to maximize the likelihood of the incomplete data. The algorithm terminates when convergence is reached and fitting is carried with the best estimate of the complete data. Although the method does not guarantee to arrive at a global maximum, local maxima which are candidates for a good solution are found. In the *RANdom SAmple Consensus RANSAC* fitting of models, few points are randomly selected and fitting is performed with the selected points. The distances of the points not used in fitting to the output are tested against a threshold and close but unused points are detected. If the number of close points is large enough, it is decided that a good fit is found and fitting is repeated with all points (random subset of points and close points) terminating the algorithm. Obviously, this is an iterative method that runs until a good fit is found. The algorithm requires a number of parameters including the number of points in random selections, the distance threshold used for testing closeness and a criterion to decide if a fit is good (usually the number of points close to the fit).

2.4.4. Miscellaneous

The widespread phenomenon of texture is related to the spatial arrangement of intensities or colors. Texture measures providing vales for each pixel describing the texture around the pixel neighborhood can be used to segment images into regions of similar texture. Texture measures may either be used to measure similarities of pixels used in region-based segmentation procedures such as region growing and clustering, or to detect texture discontinuities (i.e. texture edges) for separating image areas with different texture distributions. Belongie *et al.* [118] present a segmentation method exploiting color and texture information with the EM algorithm whose results are presented for natural scenes.

Model-based approaches for segmentation turn out to be successful for images containing objects which fit particularly well to defined models of rigid shapes. In most segmentation problems including the localization of medical structures such as internal body organs, the sought objects possess a degree of variability and they do not

fit to models of rigid shapes. A *deformable template model* is one that deforms a shape to match a known object in a given image under an implicit or explicit optimization criterion. Following its frontiers [119,120], Cootes *et al.* [121] propose the *Active Shape Models* (*ASM*) which learns shape variability through observation. *Active Appearance Models* (*AAM*) [122] extends ASM to incorporate both shape and textual information across objects (i.e. pixel intensities) in deformable template models.

2.5. Rarity and Imbalance

The inductive bias of most data mining systems is towards favoring generalization over specialization in order to minimize overall misclassification. Although such a generality bias is satisfactory for common cases/classes, it prevents successful classification of *rare* class samples. To handle the problems involved in classification problems utilizing training sets which contain few samples of a class/case in an *absolute* sense due to lack of data or when compared to instances of other classes/cases (i.e. *relative rarity*), appropriate actions during the learning phase modifying used datasets must be taken. This preprocessing is supposed to improve the detection rate of *positive* instances, for which classifiers are actually intended, and lead to realistic (i.e. fair) performance assessment of classifiers. *Rare classes* and *rare cases* differ in that rarity of a class refers to uneven class distributions (also known as *class imbalance*) and rare cases correspond to relatively small but meaningful subsets of the data space that may contain samples of any class [123]. In spite of this conceptual difference, the associated problems of and the solution approaches for both are similar.

Basically, assessing classification performance is by means of comparing classifier decisions on samples and true labels of those samples. Any metric employed to report performance is formulated as functions of these comparisons. The two distinct classes are called *positive* and *negative*. In general, the class whose detection is deemed more vital is designated as the positive class and the other as the negative class. When the decided label of any sample is the same as its true label, this contributes to success and the opposite accounts for failure. In a diagnostic system with two types of actual *event*s (possibility of two distinct labels for any sample) and correspondingly two types

of possible decisions, the classification of a sample can result in one of four situations, either contributing to success or causing failure. *TP* (true positive) refers to positive events correctly classified as positive, *FP* (false positive) to negative events incorrectly classified as positive, *FN* (false negative) to positive events incorrectly classified as negative and *TN* (true negative) to negative events correctly classified as negative. The counts of occurrences of these situations is displayed as a *contingency table* [124] (or a contingency matrix). Table 2.1 displays the 2x2 contingency table of a two-class problem. The columns of the contingency matrix are associated with the actual events (i.e. whether an instance is positive or negative) and the rows with the label assignments of the classifier.

Table 2.1. 2x2 contingency table.

		Event	
		Positive	Negative
Decision	Positive	TP	FP
	Negative	FN	TN

One significant problem of data mining with rarity is the employment of improper evaluation metrics. Classification accuracy, being the most commonly used evaluation metric that computes the fraction of correctly-classified samples, does not value rare classes as much as common classes and thus is not a true indicator of classification performance for rare classes. *Receiver Operating Characteristics* (*ROC*) analysis [125] and the associated *area under the ROC curve* (*AUC*) [126] assessing the overall classification performance have come into common use both to guide data mining and produce a numeric performance figure. ROC curves can be used to visualize and assess tradeoffs of classification rules such as increasing the detection rate of positive samples at the expense of introducing more negative samples misclassified as positive. AUC is not biased against the rare class since it does not put more emphasis on one class than the other. *Precision* and *recall*, originating in information retrieval, are metrics indicating the percentage of a classification rule correctly predicting a class and the fraction of class instances covered by the rule. Some metrics using variations of precision and

recall such as *Geometric Mean of Recall and Precision* (*GMRP*) and *F-measure* [127] are also popular. Adjusting F-measure parameters lets weigh and specify relative importances of precision and recall. Joshi [128] puts the metrics defined by precision and recall of the target rare class into a common analytical context to allow for objective comparisons and judges their suitability for rare class problems over variations of learning difficulty and rarity levels.

Class imbalance causes problems in the form of difficulty to detect regularities within rare cases/classes, hence causes insufficient or misleading learning and wrong generalization. Learning algorithms employing divide-and-conquer strategies with the rarity problem lead to data fragmentation and regularities can only be found in individual partitions that contain less data. As a result, data mining problems involving rarity should not be attacked with divide-and-conquer algorithms. Another source of problem in learners is associated with their inductive bias. Most learning systems use a general bias to avoid overfitting but this may adversely affect the performance on rare classes. Noise, being a problem in any context, has a greater impact when learning with rarity is considered because fewer noisy examples would affect learned subconcepts and avoiding overfits would be more critical.

Sampling is the most common technique used to address problems associated with rarity. It is a preprocessing step taken to modify the data distributions in the training sets so as to make the representation of rare classes equally-well to those of the majority classes. Simple sampling techniques include *undersampling*, which randomly discards majority class examples and *oversampling*, which randomly duplicates rare class examples. The objective of sampling is to modify the distributions of the classes to make them well-represented in the training set. Random undersampling may potentially lose valuable information whereas random oversampling adds exact copies of rare class samples to the training set adding no new information and increasing the risk of overfitting. Since the basic versions of sampling do not work well in practice, some heuristic sampling methods have come out. Kubat and Matwin [129] propose *OneSidedSelection* that eliminates special majority class examples (noisy, redundant or those close to the boundary separating the two classes) and keep all minority class

examples. Chawla *et al.* [130] propose *Synthetic Minority Oversampling TEchnique* (*SMOTE*) which performs oversampling of the rare class samples by producing synthetic samples using each sample and its k nearest neighbors. The experiments show that the accuracy of classifiers is improved, and combining SMOTE with undersampling performs better than plain undersampling. Gaining optimal performance from sampling methods that depends on class distributions based on sampling ratios is another consideration. Although intuition suggests that 1:1 class distributions would produce optimal performance, experimental studies show that 2:1 or even 3:1 ratios in favor of the majority class yield better results.

Boosting, effectively altering the distribution of training data and which can be viewed as a generalized sampling method, is a sequential ensemble learning algorithm that can improve the performance of weak base learners. For a series of basic classifiers, the weights of training samples are adaptively changed such that the samples misclassified in the previous iteration are assigned more weight than the others. Since rare class samples are more error-prone than majority class samples, it is reasonable to believe that boosting would assign more weight to members of the rare class and improve their classification. Standard boosting [131] treats false positives and false negatives equally, hence the majority class may still dominate the training set after successive iterations. *RareBoost* [132] focuses on both precision and recall equally and updates the weights of positive samples and negative samples differently. Chawla *et al.* [133] propose *SMOTEBoost* where SMOTE is applied in boosting iterations.

Other approaches to handle rarity include cost-sensitive learning, learning only the rare class, selecting more appropriate inductive bias, incorporating knowledge/human interaction in data mining, etc. Weiss [123] and Han *et al.* [134] present detailed discussions and surveys on rare class mining.

3. CURVATURE SCALE SPACE REPRESENTATION

The curvature scale space [71, 135] is a contour-based global representation technique in the spatial domain for planar shapes. CSS exploits the point locations where curvature drops to zero along a contour (curve) at multiple scales. When sampling of points is required, they are picked at equal distances to ensure robustness. To represent the curve completely, points are marked along the whole contour at a sufficient resolution (sampling rate) and no contour section (arc) is left unmarked. If some section of a represented curve is missing due to occlusion or other constraints, the representation is considered to be partial.

The CSS representation has very desirable properties for robust and invariant description of shapes. Since the location of a shape in a scene is not considered, the CSS representation is translation invariant. It is rotation invariant because any change in the orientation of a shape results only in circular shifts of the underlying CSS image. In other words, changing the starting point for a set of sampled contour points corresponds to rotated versions of the same contour. All such representations are essentially the same and rotation effects can easily be removed when the selection of starting points is considered. Scale invariance of CSS is trivially observed knowing that all shapes are represented using the same number of evenly spaced contour points and at a continuum of scale values corresponding to possible scales at which the shape is viewed.

Two planar curves are said to have the same shape if there exists a transformation consisting of uniform scaling, translation and rotation such that when this transformation is applied on one of the curves, the result is identical to the other curve. CSS allows the representations of curves having the same shape to be the *same*. This is important for robust classification of shapes since changes in any of scaling, translation and rotation must not affect the recognition result. The CSS representation is invariant in that two curves with the same shape have the same representation. It is unique because two curves with different shapes have different representations. Finally, the

stability property asserts that small shape differences of two curves correspond to small differences in their representations and small differences of representations correspond to small shape differences.

Let $r(u) = (x(u), y(u))$ be a parametric vector equation for the curve r, where $x(u)$ and $y(u)$ are the parametric representations of the x and y coordinates of r. Provided that a set of coordinates (x_i, y_i) of r is given and functional models are found for $x(u)$ and $y(u)$, curvature values $\kappa(u)$ can be computed using Equation 3.1, where $\dot{x}(u)$, $\ddot{x}(u)$, $\dot{y}(u)$, $\ddot{y}(u)$ denote the first and second derivatives of the twice-differentiable functions $x(u)$ and $y(u)$, respectively:

$$\kappa(u) = \frac{\dot{x}(u).\ddot{y}(u) - \ddot{x}(u).\dot{y}(u)}{(\dot{x}^2(u) + \dot{y}^2(u))^{3/2}} \tag{3.1}$$

Different parameterizations u for $x(u)$ and $y(u)$ exist. The most natural parameterization uses *arc length*. In this approach, the curve is successively sampled at equi-distant points along the contour where the distance measure is the arc length of curve segments between two adjacent points. Instead of using arc length as a parameter, one may try different parameterizations that are believed to sample closed curves better. Such an approach is called *iso-area normalization* [136], where any two adjacent curve points and the centroid of the contour constitute slices of the shape having equal areas. Let P_i and P_{i+1} denote adjacent sampled points along a contour and M denote the centroid of the contour. Furthermore, let sampling be done for N points on the curve and A(.) be the area function. The parameterization U of curves with iso-area normalization is

$$U = \{u_i \mid i \in \mathbb{Z}^+, 1 \leq i \leq N \text{ and } \forall i \text{ A}(P_i P_{i\%N+1} M) = \Delta\} \tag{3.2}$$

where Δ is a constant. When M is inside the shape boundaries and the total area of the shape is A_{tot}, $\Delta = A_{tot}/N$. The values $x(u_i)$ and $y(u_i)$ are the x and y coordinates of the sampled curve points. The exact values of u_i are of no concern as long as the

condition in Equation 3.2 is satisfied and exact functional forms of $x(u)$ and $y(u)$ are not required in curvature computations. Each of the values in Equation 3.3, Equation 3.4 and Equation 3.5 can be assigned to u_i:

$$u_i = i \mid 1 \leq i \leq N \text{ hence } u_i \in [1, N] \tag{3.3}$$

$$u_i = i/N \mid 1 \leq i \leq N \text{ hence } u_i \in (0, 1] \tag{3.4}$$

$$u_i = (i-1).(2\pi/N) \mid 1 \leq i \leq N \text{ hence } u_i \in [0, 2\pi) \tag{3.5}$$

Figure 3.1 illustrates iso-area normalization for four points of a sampled curve:

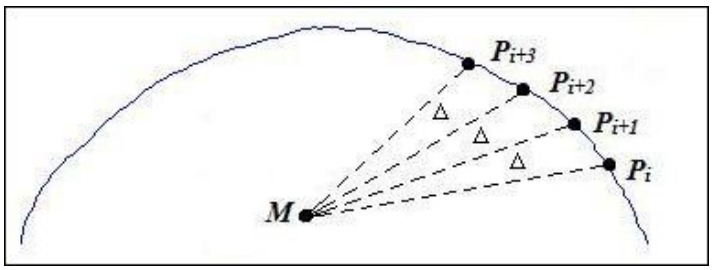

Figure 3.1. Iso-area normalization.

The curvature of a curve, $\kappa(u)$ is defined as

$$\kappa(u) = \lim_{h \to 0} \frac{\phi}{h} \tag{3.6}$$

where ϕ is the angle between tangent vectors $\mathrm{t}(u)$ and $\mathrm{t}(u+h)$, u being the parameter of the curve. Computing the curvature of a curve consists of estimating the values at all sampled points. The invariance of the CSS representation with respect to scale arises from the fact that curvature values are obtained for a range of scales, from fine to coarse. Before the curvature computation is performed for a scale, the original curve

$r(u)$ is held subject to a smoothing process and the curve points are resampled from the smoothed curve. The iterations proceed with coarser scales, one at a time. The iterative procedure of obtaining smoother curves for each of the scales is known as *curve evolution*. To evolve a curve $r(u) = (x(u), y(u))$, a 1D Gaussian kernel with standard deviation σ, $\mathrm{g}(u, \sigma)$ is used. The resulting curve $R(u) = (X(u), Y(u))$ is computed by convolving $x(u)$ and $y(u)$ with $\mathrm{g}(u, \sigma)$ as stated in Equation 3.7 and Equation 3.8:

$$X(u, \sigma) = x(u) * \mathrm{g}(u, \sigma) \tag{3.7}$$

$$Y(u, \sigma) = y(u) * \mathrm{g}(u, \sigma) \tag{3.8}$$

The selection of σ in the iterative curve evolution process is towards increasing its value at successive iterations. Given the standard deviation of the Gaussian kernel in the first iteration as σ_0 , the standard deviation for iteration n can be computed as

$$\sigma_n = \sigma_0 \sqrt{n}. \tag{3.9}$$

Other types of selections are possible. In our experiments, we have found the rule of Equation 3.9 working and curve features well-captured at decreasing levels of detail.

The *turning angle* (TA) and *differential turning angle* (dTA) [137–139] concepts are useful in the computation of curvature values at sampled points when the sampling rate (curve resolution) is sufficiently high. In the limit case where the number of curve points is infinite, the differential turning angle is exactly identical to curvature. Turning angles are appropriate to establish curvature values at sampled points.

A CSS image of a curve is a representation of the curve where the sampled points with zero curvature values are marked. The CSS image of a curve at a particular scale

σ is defined as

$$\kappa(u, \sigma) = 0. \tag{3.10}$$

The CSS image is the collection of points satisfying Equation 3.10 for all values of u and σ. That is, the CSS image is defined for all values of the u parameter and all scales σ. A CSS image is actually a 2D plot where the horizontal and vertical axes list u and σ values respectively. Curvature zero-crossings are marked for (u, σ) pairs.

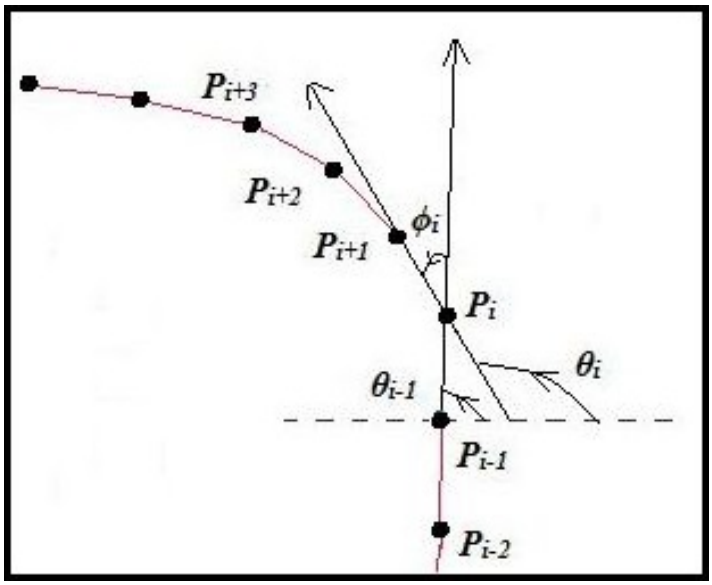

Figure 3.2. Differential turning angle.

3.1. Turning Angle Based Representation

Given three adjacent points P_{i-1}, P_i, P_{i+1} located on a curve, the turning angle associated with the vector $\overrightarrow{P_{i-1}P_i}$ is the angle between the vector and the x-axis in the positive direction. The differential turning angle at point P_i, is defined as the difference between the turning angles of $\overrightarrow{P_iP_{i+1}}$ and $\overrightarrow{P_{i-1}P_i}$. Denoting the turning angle of $\overrightarrow{P_{i-1}P_i}$ with θ_{i-1} and that of $\overrightarrow{P_iP_{i+1}}$ with θ_i, the differential turning angle ϕ_i at P_i is

$$\phi_i = \theta_i - \theta_{i-1}. \tag{3.11}$$

Figure 3.2 is an illustration of turning angle and differential turning angle. Differential turning angles are descriptive of contour characteristics at various scales. In dTA

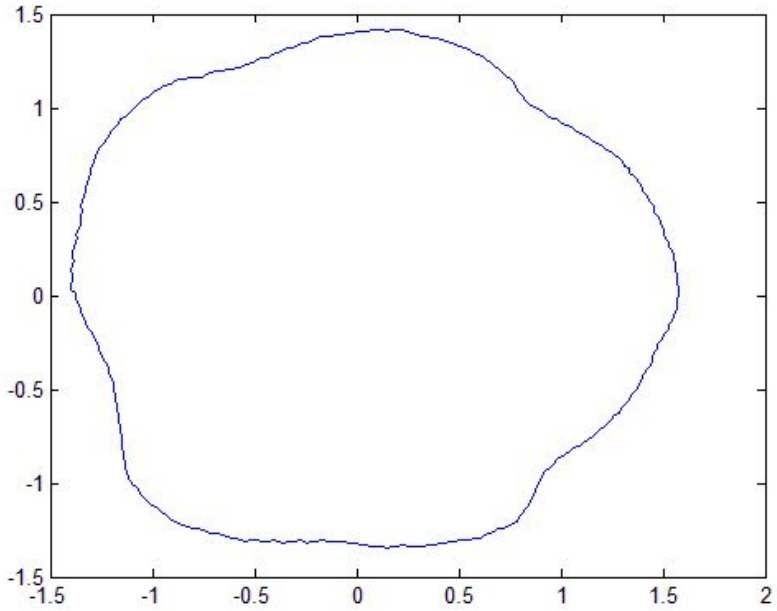

Figure 3.3. Skull contour sampled with 360 points.

computation for closed contours, the convention is to select a starting point P_0 and order the points in counterclockwise direction. That is P_{i+1} is the point located counterclockwise with respect to P_i for each i. The differences of turning angles between pairs of adjacent points are computed to obtain the dTA function $\delta(u)$. Figure 3.4 shows $\delta(u)$ of the contour in Figure 3.3 at scale $\sigma = 0.1246$. Figure 3.5 shows $\delta(u)$ for all considered scales σ. The plot of $\delta(u, \sigma)$ is a depth-image where low intensities correspond to small dTA values (i.e. negative) and high intensities correspond to large dTA values (i.e. positive). $\delta(u, \sigma)$ is called the dTASS (*differential turning angle scale space*) function (scalogram or map) of the contour under consideration.

Positive and negative values of differential turning angles may correspond to local maxima and local minima of the dTA function $\delta(u)$, respectively. All zero values are the zero-crossings of $\delta(u)$. Local minima of dTA functions are known as *alpha(α)* points and local maxima are called *beta(β)* points. Zero values in the dTA function are referred to as *gamma(γ)* points. An *essential points image* shows all α, β and γ points (i.e.

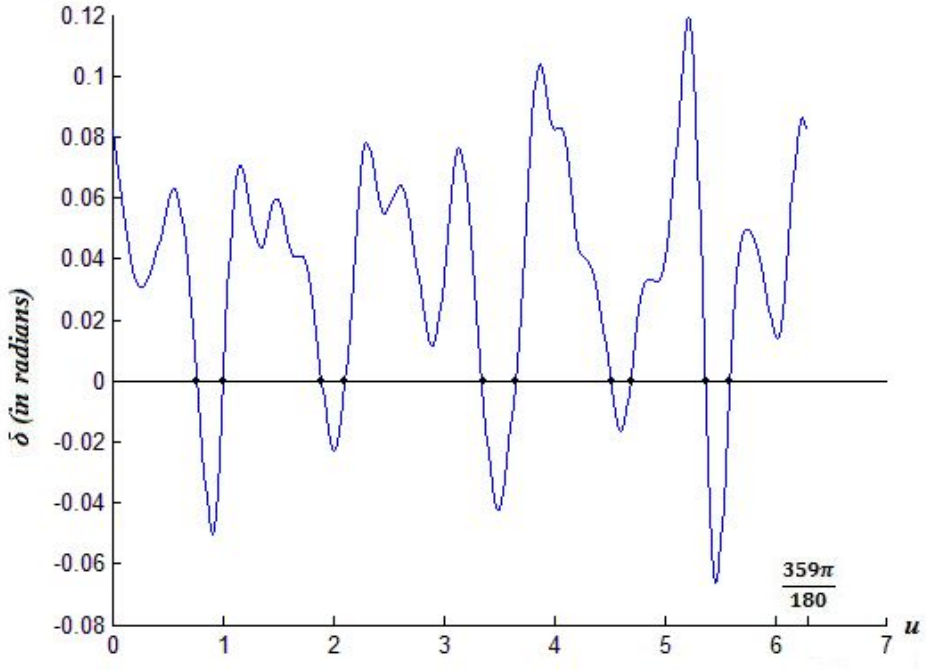

Figure 3.4. dTA function $\delta(u)$ at scale $\sigma = 0.1246$ for the contour of Figure 3.3.

Figure 3.5. dTASS scalogram of the contour of Figure 3.3.

extreme points) of the dTASS map and no others. Figure 3.6 illustrates the essential points image of the contour of Figure 3.3. Comparing Figure 3.5 and Figure 3.6 helps observe the relationship between differential turning angles and dTASS scalograms.

A CSS image of a curve and its dTASS scalogram which shows only γ points are identical. The contour points with zero differential turning angles are those points where the curvature is zero. With $\delta(u, \sigma)$ denoting the differential turning angle function of $r(u)$ at scale σ, $\kappa(u, \sigma)$ denoting the curvature function and N being the number of points of $r(u)$, the relationship between differential turning angles and curvatures of points $P_i = (x(u_i), y(u_i))$ is stated as

$$\lim_{1 \leq i \leq N,\ N \to \infty} \delta(u_i, \sigma) = \kappa(u_i, \sigma). \tag{3.12}$$

Figure 3.7 shows the CSS image of the contour in Figure 3.3. CSS images are convenient representations for curves, features can be extracted from them or two such representations can be compared to measure distance/similarity.

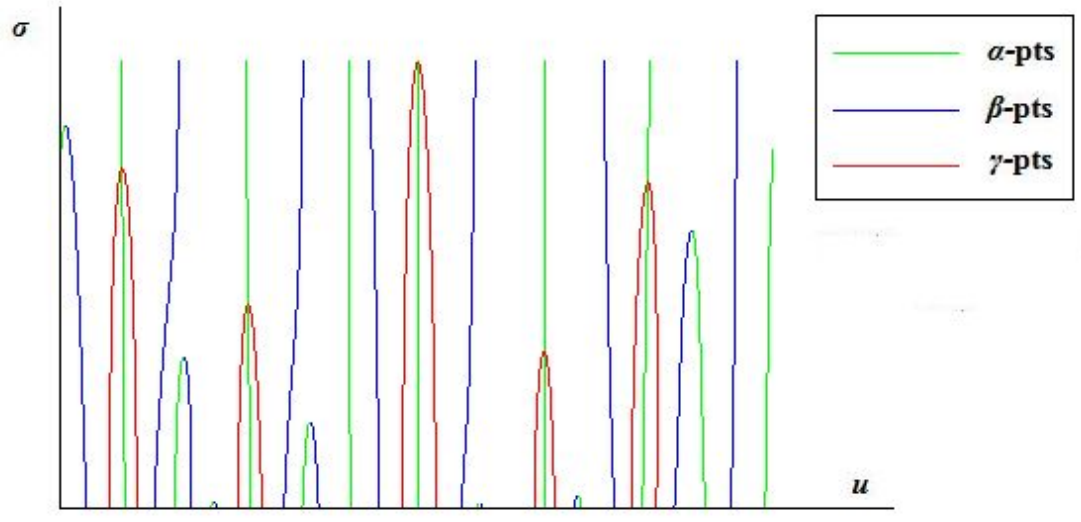

Figure 3.6. Essential points image of the contour in Figure 3.3.

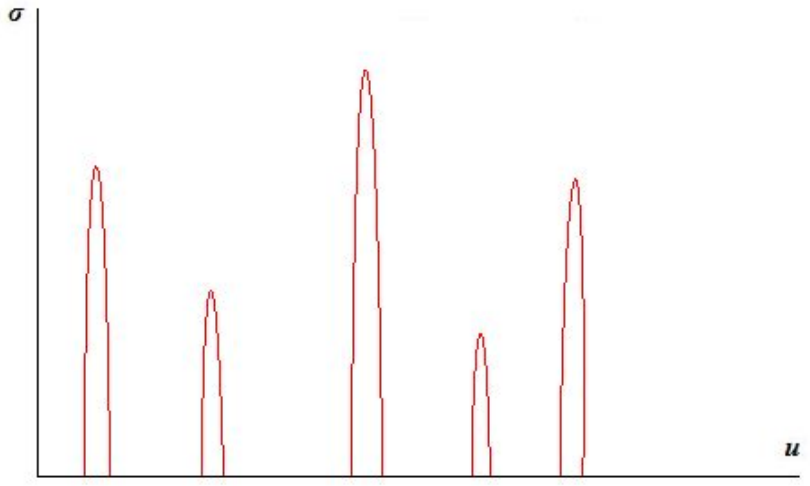

Figure 3.7. CSS image for the contour of Figure 3.3.

3.2. CSS Images Computation

The contour of concern $r(u)$ is represented by two parametric functions $x(u)$ and $y(u)$. Cubic *B-splines* are used to model these functions with $N = 360$ points. As a result, the integer-valued pixels of the curve are replaced by real-valued points. The curve is then normalized using the procedure outlined by Avrithis *et al.* [140]. The orthogonalization procedure takes the center of gravity of $r(u)$ as the origin M of the coordinate system. Furthermore, the moments of inertia m_{01}, m_{10}, m_{11} are set to zero and m_{02}, m_{20} are set to one. The 360 points sampled after B-spline modeling are ordered counterclockwise and finally applying iso-area normalization fine-tunes the coordinates of sampled points in order to arrive at the point coordinates where each pair of adjacent points is separated by shape slices of equal area originating at M. Representing the curve of this stage with $\hat{r}(u)$, the CSS image computation begins.

Given the points $P_i = \hat{r}(\hat{x}(u_i), \hat{y}(u_i))$ of $\hat{r}$, $i = 1, 2, 3, \ldots, 360$, where P_1 is the point located at $\theta = 0$ angle with respect to the XY coordinate system whose origin is M, the differential turning angles for all points are computed. The computation for differential turning angles is an iterative procedure. At iteration n, $\hat{r} = (\hat{x}(u), \hat{y}(u))$

is smoothed with a Gaussian kernel of standard deviation $\sigma_n = \sigma$ and a smoothed version $\hat{R} = (\hat{X}(u,\sigma), \hat{Y}(u,\sigma))$ of $\hat{r}$ is obtained. $\hat{R}$ is subjected to a gain control [141] process to bring it to approximately the same size as $\hat{r}$. The gain-controlled curve $\hat{R}_g = (\hat{X}_g(u,\sigma), \hat{Y}_g(u,\sigma))$ is computed by Equation 3.13 and Equation 3.14,

$$\hat{X}_g(u,\sigma) = S_x[\hat{X}(u,\sigma) - \hat{X}_M(\sigma)] + \hat{X}_M(\sigma) \tag{3.13}$$

$$\hat{Y}_g(u,\sigma) = S_y[\hat{Y}(u,\sigma) - \hat{Y}_M(\sigma)] + \hat{Y}_M(\sigma) \tag{3.14}$$

where $\hat{X}_M(\sigma)$ and $\hat{Y}_M(\sigma)$ are the coordinates of the center of gravity of the smoothed contour $\hat{R}$. The coefficients S_x and S_y are defined by Equation 3.15 and Equation 3.16,

$$S_x = \frac{\sum_{i=1}^{N} |\hat{x}(u_i) - \hat{x}_M|}{\sum_{i=1}^{N} |\hat{X}(u_i,\sigma) - \hat{X}_M(\sigma)|} \tag{3.15}$$

$$S_y = \frac{\sum_{i=1}^{N} |\hat{y}(u_i) - \hat{y}_M|}{\sum_{i=1}^{N} |\hat{Y}(u_i,\sigma) - \hat{Y}_M(\sigma)|} \tag{3.16}$$

where $\hat{x}_M$ and $\hat{y}_M$ are the coordinates of the center of gravity of $\hat{r}$.

The differential turning angles of all N points of $\hat{R}_g$ are computed and recorded. The outcome of each iteration is the dTA function δ of $\hat{R}_g$ at scale σ_n. The δ function values are examined to detect zeroes. In practice, it is not possible to find absolute zeroes because of finite sampling; however, transitions from negative values to positive and from positive values to negative are the locations where a curvature zero can be marked. Given $\delta_i > 0$ *and* $\delta_{i+1} < 0$ *OR* $\delta_i < 0$ *and* $\delta_{i+1} > 0$, the index z of the point of zero curvature is decided by

$$z = \begin{cases} i & \text{if } |\delta_i| \leq |\delta_{i+1}|, \\ i+1 & \text{otherwise.} \end{cases} \tag{3.17}$$

The iterations of CSS computation can be terminated when no zero curvature points can be detected for iteration n. This is because encountering zeroes at coarser scales is not possible. Starting the computation process by letting $n = 1$ is possible, but very fine scales of curves are often considered to contain noise. To remove this noise content in further stages of signal processing, zeroes of curvature at a number of fine scales can be discarded. In our implementation, we set $\sigma_0 = \pi/180$ and start the iterations letting $n = 51$. We do not take the first 50 scales into consideration. For each n, σ_n is computed with Equation 3.9 and the computations of the iteration are performed. We stop iterating when no more curvature zeroes are found. For normalization purposes (i.e. for displaying rotation invariance), the CSS image is circularly-translated such that the maximum σ coordinate of γ points is located at $u = \pi$. This corresponds to a horizontal shift of the CSS image. What leaves the constant-sized image at the right or the left edge enters the image at the left or the right edge. The CSS representation remains the same no matter how the CSS image is horizontally-shifted.

3.3. Enhancing CSS Features

Curvature zeroes are present for curves that possess concave segments. CSS representation is poor for convex curve segments. Zero values of curvature occur at points where a transition from a concave curve portion to a convex portion takes place. Such points are called *inflection* points. A curve which has no concavity (i.e. which is entirely convex) has no inflection points having zero curvature, hence the CSS representation is *null* and can not produce any descriptive features.

Kopf *et al.* [142] present a method to enhance CSS features for convex curve segments. In this method, the actual contour is mirrored using a circular mirror centered in the center of gravity M of the shape enclosed by the actual (i.e. real) contour. The reflected contour produced as the outcome of this process is treated with the same procedure as the actual contour was subjected to. As a result, two CSS images are obtained to represent a single contour and the set of features is enriched.

The circle to serve as the mirror and the shape itself are both centered at $M =$

(M_x, M_y). The contour must entirely be within the circle of radius D. $\hat{R}_g(u) = (\hat{X}_g(u), \hat{Y}_g(u))$ being the points of the actual contour, the formulation to obtain the points of the reflected contour $R_{ref}(u) = (X_{ref}(u), Y_{ref}(u))$ is given as

$$X_{ref}(u) = (\hat{X}_g(u) - M_x)\frac{2D - \text{dist}(\hat{R}_g(u), M)}{\text{dist}(\hat{R}_g(u), M)} + M_x \tag{3.18}$$

$$Y_{ref}(u) = (\hat{Y}_g(u) - M_y)\frac{2D - \text{dist}(\hat{R}_g(u), M)}{\text{dist}(\hat{R}_g(u), M)} + M_y \tag{3.19}$$

where dist(.) is the Euclidian distance function. Figure 3.8 shows the contour of Figure 3.3 and its reflection on the circular mirror. The CSS images of the actual and reflected contours of Figure 3.3 are displayed in Figure 3.9. The positive and negative values are for the actual and the reflected contours, respectively.

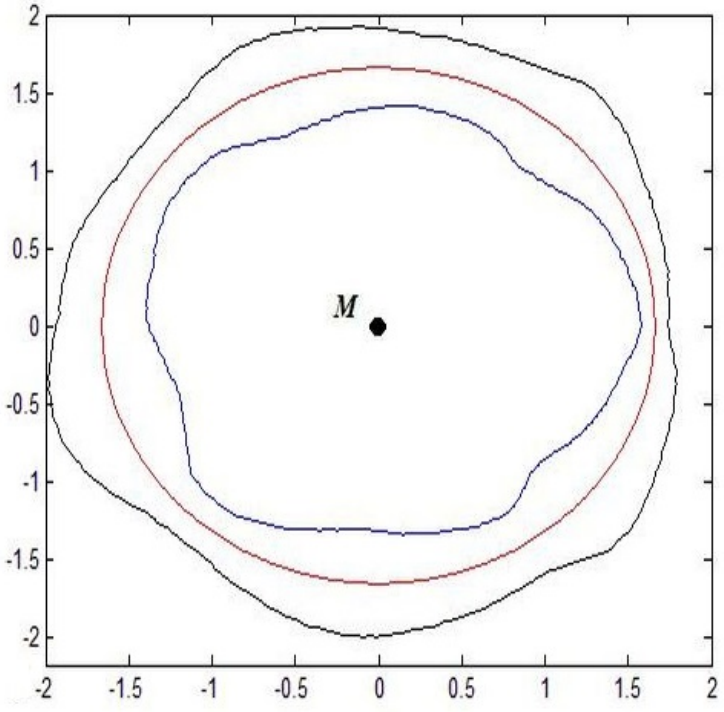

Figure 3.8. Contour of Figure 3.3 and its reflection.

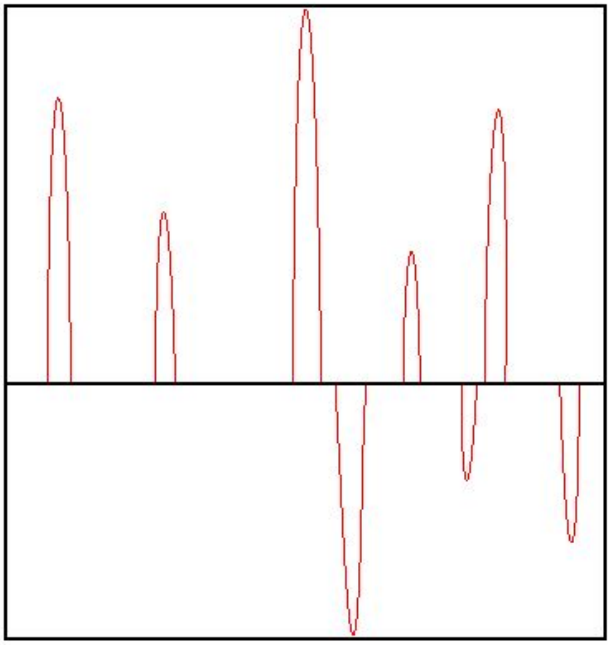

Figure 3.9. CSS images of actual and reflected contours of Figure 3.3.

3.4. CSS Matching

Two CSS images that are representations of two different contours can be compared in order to produce a matching score corresponding to the similarity of the two contours or a difference figure to indicate how different from each other the contours are. In the matching procedure, one need not deal with normalizations to compensate for translation, rotation and scale invariance because the CSS representation itself has the invariance property with respect to all these transformations.

The entities to use in a comparison/matching process of two CSS images could be the positions of peaks of γ points (i.e. the u coordinate at which arc-shaped curves in the CSS image reach their maximum σ values), the heights of those peaks (σ) and the widths of the arc-shaped curves at the bottomline of the image. Besides these three types of basic features, it may be possible to identify others. Figure 3.10 illustrates the position, height and width properties of an arc-shaped curve of a CSS image. Using positions and heights of peaks of γ points as descriptive entities is rather easy to justify.

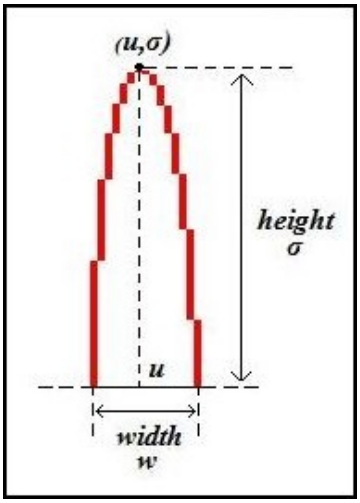

Figure 3.10. Entities of CSS arcs: position u, height σ, width w.

The differences of position and of height for arcs of γ points, of course, do matter in discrimination. In order to improve the description of arcs, one may also consider the width attribute. The width w actually corresponds to the normalized length of curve segments the CSS image arc is associated with. As mentioned before, CSS arcs correspond to concave curve segments and the associated height σ is the degree at which concavity disappears (vanishes). When heights σ of CSS arcs is the same, different widths w are associated with *deep* or *shallow* concavities. Large w values signal deep concavities where the length of the associated curve segment is relatively large (i.e. concavity is deep) and small w values signal shorter curve segments (i.e. concavity is shallow). In the CSS matching algorithm of Richter *et al.* [143], the width attribute is used to decide whether two CSS arcs can ever be matched. If the widths of two arcs from two CSS images differ much, a matching of the two arcs is considered not to be possible. In our versions of the CSS matching algorithm of Abbasi *et al.* [74], we either completely discard the width attribute or use it in the computation of the match cost of two CSS arcs. In fact, fetal skull images do not differ much in their nature of concave curve segments. In our extended version of CSS matching [74], we use heights σ and widths w of arcs to measure how well the patterns (i.e. shapes) of CSS arcs match. In the plain version, only the Euclidian distances between γ-peaks are used.

3.4.1. CSS Matching Algorithm

The CSS matching algorithm compares two sets of γ-peaks, one for the *image* and one for the *model*. The outcome of the algorithm is the matching score of the image and the model. The matching score is an accumulation of costs for pairs of peaks, again one from the image and one from the model and considered as the minimum distance between the two. An outline of the matching algorithm [74] is as follows:

(i) Create nodes to use during the matching process. In each node, one peak from the image and one peak from the model take place. The node creation process is described in the following list:
   - Select the largest scale peak from the image and the largest scale peak from the model.
   - If model peaks that are close to the largest scale peak of the image (within 80% of the σ coordinate of the image peak) exist, create nodes corresponding to the image peak and each model peak satisfying the closeness criterion.
   - Repeat steps a and b for the second largest scale peak of the image and all the peaks of the model.

(ii) For each node created in (i), compute a shift parameter T:

$$T = u_m - u_i \tag{3.20}$$

where u_m and u_i stand for the u coordinates of the model and image peaks of the node and T is used to translate the image to equalize the u coordinate of the image peak to that of the model peak. Initialize the match cost of the node as the absolute difference of the σ coordinates of the image and the model.

(iii) For each node created in (i), create two lists *I_list* and *M_list*. *I_list* will accommodate the image peaks matched within that node and *M_list* will do the same for the model peaks within the same node. Initialize *I_list* and *M_list* to contain the image peak and the model peak of the node, respectively.

(iv) Expand each node and update the lists and costs associated with the nodes. Node

expansion is described in Section 3.4.1.1.

(v) Select the node with minimum cost. If there are no image or model peaks that remain unmatched, return that node as the minimum cost node. Otherwise, go back to (iv) and expand the minimum cost node.

(vi) Reverse the roles of the image and the model so that the image becomes the model and the model becomes the image. Repeat (i) through (v) to compute the match cost for this case. Take the minimum cost found in (i) through (vi) as the match cost of the two curves.

3.4.1.1. Node Expansion. To expand a node, execute the following steps:

(i) Select the largest scale image peak that is not in *I_list*. Apply the shift parameter T of the node to map the selected image peak to the model image. Take the nearest model peak not in *M_list*.

(ii) If the selected image peak and model peak are close enough (within 20% of the maximum possible horizontal distance), find the cost of the match (to be described in Section 3.4.1.2). Otherwise, define the cost of the match as the σ value of the selected image peak.

(iii) This step applies to cases when the number of image peaks and that of model peaks are different. If there is no more image peak to match, define the cost of the match as the σ value of the largest scale model peak. Similarly, if there is no more model peak to match, define the cost of the match as the σ value of the selected image peak.

(iv) Update the overall match cost of the node by adding the cost of the match to it. Also update the two lists *I_list* and *M_list* of the node accordingly.

3.4.1.2. Computing Match Costs. Computing the cost of a particular match of two peaks is done in either of one of two ways, depending on what the design decision is. This decision can be taken as subjective.

Let $F_1 = (u_1, \sigma_1, w_1)$ and $F_2 = (u_2, \sigma_2, w_2)$ be the attributes of the first and second

CSS peaks and their associated arcs which are matched. In addition, let $mx = 1$ or $mx = 2$ depending on whether $\sigma_1 < \sigma_2$ or $\sigma_2 < \sigma_1$, similarly let $Mx = 1$ or $Mx = 2$ depending on whether $\sigma_1 > \sigma_2$ or $\sigma_2 > \sigma_1$. When $\sigma_1 = \sigma_2$, both mx and Mx can take either of the two values. In our implementation, the cost of the match is computed in either of the two ways as C_1 or C_2 as given by Equation 3.21 and by Equation 3.22,

$$C_1 = \sqrt{(\text{circ\_diff}(u_1, u_2))^2 + (\sigma_1 - \sigma_2)^2} \tag{3.21}$$

$$C_2 = \sqrt{(\text{circ\_diff}(u_1, u_2))^2 + (\sigma_1 - \sigma_2)^2} + |\frac{max(\sigma_1, \sigma_2)}{min(\sigma_1, \sigma_2)} w_{mx} - w_{Mx}| \tag{3.22}$$

where circ_diff(u_1, u_2) is the function that measures the distance between u_1 and u_2 when positioning is circular. Circular difference is defined by Equation 3.23,

$$\text{circ\_diff}(u_1, u_2) = \begin{cases} |u_1 - u_2|/N & \text{if } |u_1 - u_2| < (N/2), \\ (N - |u_1 - u_2|)/N & \text{otherwise.} \end{cases} \tag{3.23}$$

where $1 \leq i \leq N$ for all $u_i \in \{1, 2, 3, \ldots, N-2, N-1, N\}$ and $i \in Z^+$. N is actually the number of sampled points on a curve. As can be seen, circ_diff$(u_1, u_2) \in [0, 1)$ is a normalized distance measure. w_{mx} and w_{Mx} are also normalized distances because they are computed using the circ_diff(.) function. u_L and u_R being the u coordinates of the two endpoints of a CSS arc, the associated width w of the arc is computed as

$$w = \text{circ\_diff}(u_L, u_R). \tag{3.24}$$

The first match cost C_1 is simply the straight-line distance between two peaks. On the other hand, the second match cost C_2 also includes the difference of widths, as an additive term, when the width of the arc of the peak with smaller σ has been amplified with a factor to equalize its height to that of the peak with larger σ. This term, in a way, measures the dissimilarity of the two arc patterns (shapes).

3.4.2. CSS Matching Example

For an illustration of CSS matching and the involved features, we consider the CSS images in Figure 3.13 of the normalized skull contours in Figure 3.12, which are associated with the fetal skulls viewed in Figure 3.11. The CSS image of the first contour (extracted from the first US image) contains four arcs and the CSS image of the second contour has three arcs. For any CSS arc, there are three features u, σ and w as shown in Figure 3.10. For any CSS image, the features are those for all arcs of the image. Table 3.1 displays the features of the two CSS images of Figure 3.13. The rows of the table correspond to the features of individual CSS arcs. The ordering of the arcs from top to bottom is from the highest σ value to the lowest in accordance with the fact that γ peaks with higher σ are considered before those with lower σ in CSS matching. The number of arcs for any two CSS images and thus the number of features of the two may be different, as in the example. In Figure 3.14, the three pairs of image and model peaks that result in the lowest match cost in CSS matching (considering Equation 3.21) are shown. The blue lines and dots correspond to the image peaks and the red are for the model peaks. Comparing the CSS images of Figure 3.13 and the plot of peaks in Figure 3.14 may produce a perception such that the heights of peaks that correspond in the two figures are different. This is because each row of pixels in the CSS images is associated with a particular σ and the differences of σ between consecutive rows are not the same for all pairs of adjacent rows, however the plot of Figure 3.14 reflects actual values of σ and the scaling is absolutely correct. All in all, the CSS images of Figure 3.13 and the peaks in Figure 3.14 actually correspond. Figure 3.14 shows the pairs of matching peaks inside different ellipses. One of the peaks of the model does not match to any image peak. Clearly, the CSS matching procedure of [74] does not require equal-sized feature vectors. The costs of matching the CSS images in Figure 3.13 computed with Equation 3.21 and Equation 3.22 turn out to be $C_1 = 0,6473$ and $C_2 = 0,6925$, respectively. C_2 must always be greater than or equal to C_1 because of the additive term in Equation 3.22.

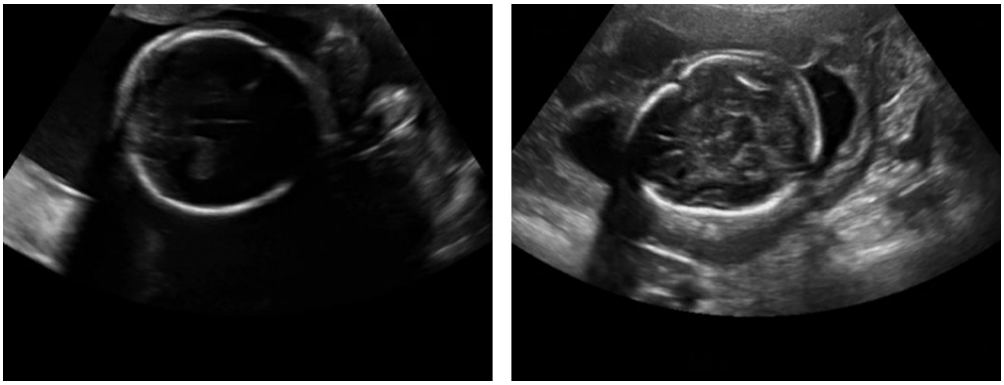

Figure 3.11. US image samples.

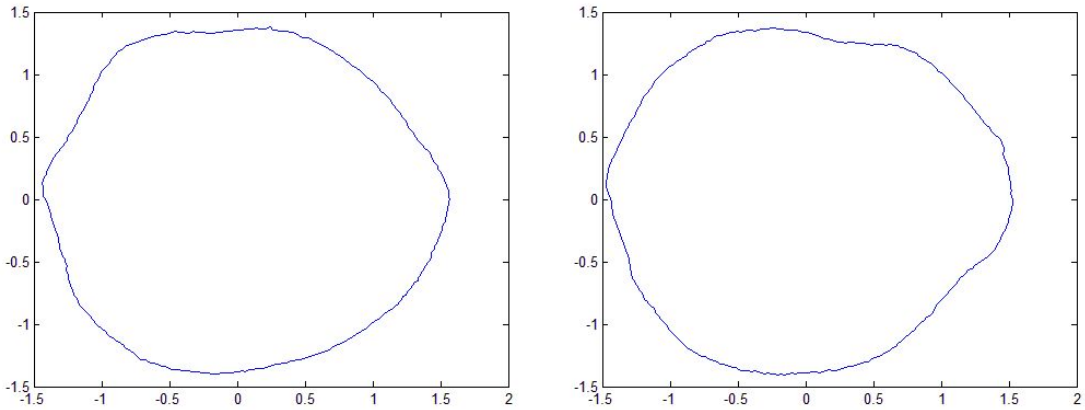

Figure 3.12. Normalized skull contours associated with the images of Figure 3.11.

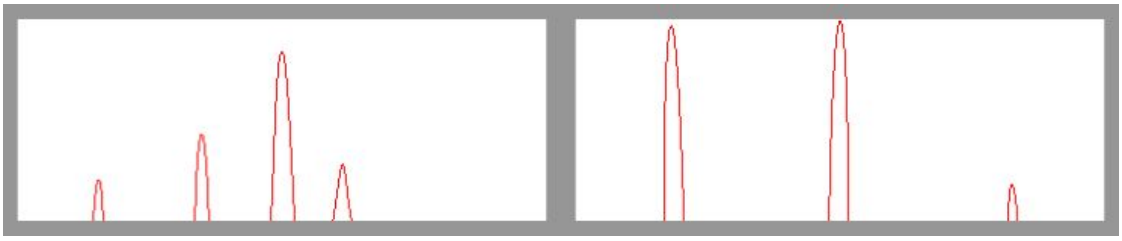

Figure 3.13. CSS images of the contours in Figure 3.12.

Table 3.1. Features of the CSS images of Figure 3.13.

CSS image on the left of Figure 3.13			CSS image on the right of Figure 3.13		
u_1	σ_1	w_1	u_2	σ_2	w_2
0,5000	0,2857	0,0444	0,5000	0,2553	0,0361
0,3472	0,2541	0,0278	0,1833	0,2535	0,0361
0,6167	0,2418	0,0361	0,8250	0,1771	0,0167
0,1528	0,2348	0,0194			

3.5. Distance Matrix-Like Matrix Matching

Measuring the similarity of two CSS images can also be done using the γ-peak points matching procedure of Kpalma *et al.* [138]. This method employs the *Distance Matrix-Like Matrix* (*DMLM*), whose entries are defined as the distances between pairs of peak points in two CSS images. There exists a distance for every pair of peak points, one point from the *query* CSS image and the other from the *model* CSS image. The ij^{th} entry of the DMLM associated with two images is the distance between peak point i in the query image and peak point j in the model image.

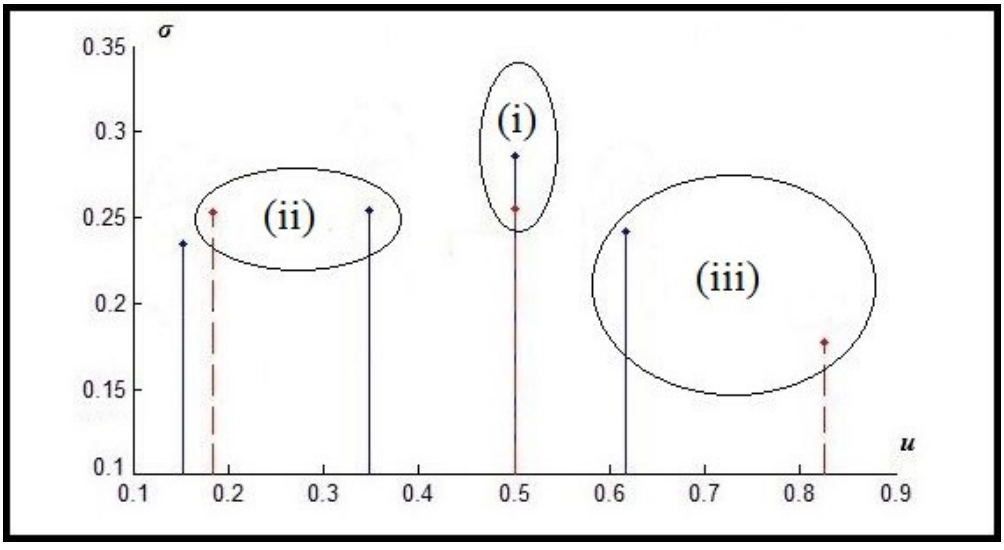

Figure 3.14. CSS matching: peaks of Figure 3.13 (blue: image, red: model).

3.5.1. DMLM Peak Points Matching Procedure

All peaks in the query are matched with at most one peak of the model. When the number of peaks of the query image is more than that of the model peaks that can be used in matching, the output model is zero-padded. On the other hand, when the number of model peaks is more, some model peaks are discarded. For a matching in a pair of peak points, one from each image, to take place; the distance between them (such as Euclidian or Manhattan) must be less than some threshold distance *distMAX*. In our implementation, we use the straight-line (Euclidian) distance measure between points. At the end, when matching is complete, two equal-sized feature vectors qV and moV are obtained and their similarity is an indication of how similar the two CSS images are. In the sequel, q is a prefix to mean a query point, m is the same for a model point and mo is a prefix that will be used to identify model output points. For clarification, we distinguish query, model and model output points as follows: *The query points (q) are matched to the model points (m), the result of matching is the model output points (mo).* Although the number of query points and model points do not have to be the same, there are as many query points as model output points at the end of matching and before similarity computation.

Given two CSS images (i.e. γ-peaks of the CSS images), the work of Kpalma *et al.* [138] proposes to assign the role of query (q) to one image, assign the role of model (m) to the other image, match the two images, extract the model output (mo) peaks, form the equal-sized feature vectors qV and moV of the query and the model output, and finally compute the similarity of qV and moV. Next, the roles of query and model are exchanged and the same procedure is applied once more to compute another similarity score. The maximum of the two similarities is taken as the similarity of the images of γ-peaks, hence of the CSS images and hence of the contours (curves). In our implementation, we prefer to assign the query role to the image of γ peaks with larger number of peaks and the model role to that with smaller number of peaks. Only when the number of peaks for both CSS images is the same, we measure similarity in two directions. Our intuition is that assimilating a richer image to a poorer image is more sensible than assimilating a poorer image to a richer image.

To exploit rotation invariance implicit for CSS images and find the best position to match two images, we execute DMLM peak points matching for a set of horizontal shifts of the u parameter. This set of horizontal shifts is computed by considering the heights σ of peaks in the query image which are at least as high as half of the maximum peak height of the same image. For peaks of the query image satisfying this criterion, if the difference of the height of any peak of the model image and the height of the considered query peak is not more than half the height of the taller of two peaks, a horizontal shift value (the difference of u values of the two peaks) is added to a list of shift parameters. We execute the matching procedure for all values in the shift list and select the largest similarity.

Before computing the DMLM for two sets of γ-peaks, the peaks of both the query and the model (a shifted version of it) are sorted in decreasing order of σ values. This may be viewed as a preprocessing action to consider larger maxima first during DMLM computation. The distances in the DMLM of the query and model images of γ-peaks and a distance threshold $distMAX$ are used to match the peaks in the query image and the peaks in the model image. For the specific application of matching fetal skull contours, we take $distMAX = 0.2$. (u is normalized to [0, 1)). If no model point mP_j that can be matched to a query point qP_i exists, the query point is matched to the point $moP_i = (qu_i, 0)$. In other words, the abscissa u (i.e. position) remains the same as that of the query point but the scale value is set to zero (i.e. zero-padded). An example of a DMLM for two sets of γ-peaks (q and m) of Figure 3.15, corresponding to the CSS images of Figure 3.13 is given in Table 3.2. The ellipses show the pairwise matches of the query points to the model points. Note that the second and third query points (qP_2 and qP_3) are not close enough to any model point, that is why the scale values of the corresponding model output points are set to zero. A procedure to match all query points to model points is given in the algorithm of Figure 3.16. The five model output points for the query points of Figure 3.15 are $moP_1 = (mu_1, m\sigma_1), moP_2 = (qu_2, 0), moP_3 = (qu_3, 0)$ and $moP_4 = (mu_2, m\sigma_2)$.

After all query points are matched to model points and model output points (mo) are found, the feature vectors can be defined by concatenating the coordinates of each

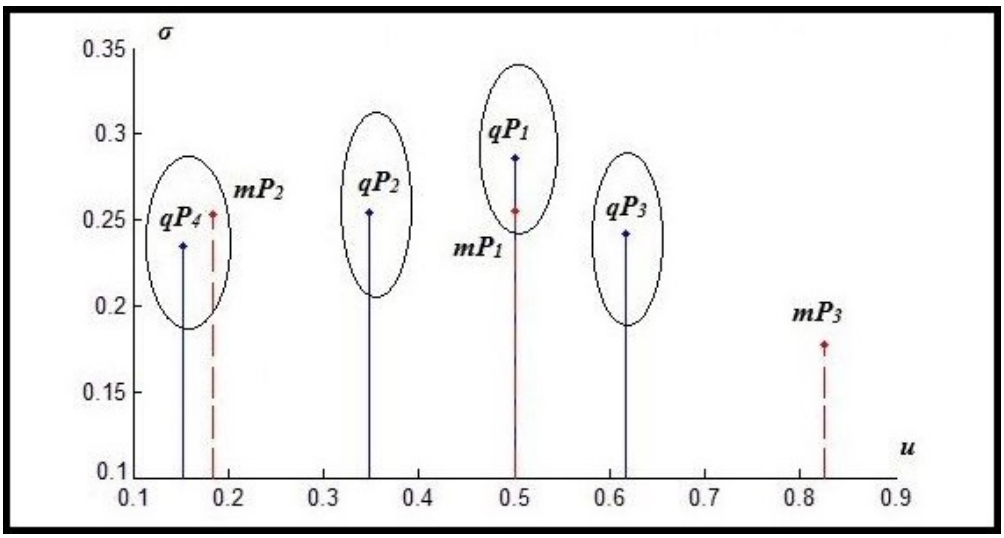

Figure 3.15. DMLM matching: peaks of Figure 3.13 (blue: query, red: model).

set of points (q and mo). Denoting the number of query points with qD, the feature vectors qV and moV are given by

$$\begin{cases} qV = (qu_1, qu_2, \ldots, qu_{q_D}, q\sigma_1, q\sigma_2, \ldots, q\sigma_{q_D}) \\ moV = (mou_1, mou_2, \ldots, mou_{q_D}, mo\sigma_1, mo\sigma_2, \ldots, mo\sigma_{q_D}). \end{cases} \tag{3.25}$$

Table 3.2. DMLM example for γ peaks of Figure 3.12.

DMLM		**Model points** $(mP_1 \ldots mP_3)$		
		1	**2**	**3**
Query points $(qP_1 \ldots qP_4)$	**1**	0.0304	0.3183	0.3427
	2	0.1528	0.1639	0.4839
	3	0.1174	0.4335	0.2182
	4	0.3478	0.0358	0.6747

```
1: for i ← 1, no_query_points do
2:     retrieve model point mP_j closest to query point qP_i
3:     if dist(qPi, mPj) < distMAX then
4:         moP_i ← (mu_j, mσ_j)
5:     else
6:         moP_i ← (qu_i, 0)
7: return moP_i for all i
```

Figure 3.16. Pseudo-code for DMLM matching.

3.5.2. Similarity Computation

α being the angle between two feature vectors qV and moV, the similarity of the vectors is computed by

$$\text{sim}(qV, moV) = 50(1 + cos(\alpha))\frac{min(||qV||, ||moV||)}{max(||qV||, ||moV||)} \tag{3.26}$$

where $\text{sim}(.) \in [0\%, 100\%]$. For the sake of completing the example of Figure 3.15 (Table 3.2); the qV, moV and $\text{sim}(qV, moV)$ values are shown in Table 3.3.

Table 3.3. qV, moV and $\text{sim}(qV, moV)$ values for the example of Figure 3.15.

qV =	(0.5000, 0.3472, 0.6167, 0.1528, 0.2857, 0.2541, 0.2418, 0.2348)
moV =	(0.5000, 0.1833, 0.6167, 0.1528, 0.2553, 0.2535, 0, 0)
$\text{sim}(qV,moV)$ =	47.8215%

4. ZERNIKE MOMENTS

Orthogonal moments such as Zernike [90, 144] have become popular in applications including pattern recognition. Orthogonality of Zernike moments [91] asserts that they can be used to represent images with minimum redundancy. In addition, the rotational invariance property of their magnitudes with easy scale and translation normalization makes them attractive for image representation.

This chapter provides a theoretical background on Zernike moments, a method for their exact computation devised by Hosny [145]; schemes for acquiring translation and scale invariance and for deciding the maximum order of moments to use in classification, all adopted from the work of Khotanzad and Hong [146]; and presents the overall methodology to compute invariant features (i.e. magnitudes of Zernike moments) of fetal skull shapes in the context of spina bifida detection.

4.1. Image Representation with Zernike Moments

A complex Zernike polynomial V_{pq} with order p and repetition q, p and q satisfying $p \in Z, p \geq 0$, $q \in Z$ and $p - q$ is even, is defined inside the unit circle ($x^2 + y^2 = 1$) as

$$V_{pq}(x, y) = V_{pq}(\mathrm{r}, \theta) = R_{pq}(\mathrm{r}) e^{jq\theta} \tag{4.1}$$

where $j = \sqrt{-1}$, r is the length of the vector $\overrightarrow{P}$ from the origin to the pixel (x, y) and θ is the angle between $\overrightarrow{P}$ and the x-axis in counterclockwise direction. The real-valued radial Zernike polynomial R_{pq} is given by

$$R_{pq}(\mathrm{r}) = \sum_{k=0}^{\frac{p-|q|}{2}} (-1)^k \frac{(p-k)!}{k!(\frac{p+|q|}{2}-k)!(\frac{p-|q|}{2}-k)!} \mathrm{r}^{(p-2k)} \tag{4.2}$$

with $R_{pq} = R_{p,-q}$. Zernike polynomials form a complete orthogonal set where their orthogonality relation is defined as

$$\begin{aligned} & \int\int_{x^2+y^2=1} V_{nm}(x,y)V_{pq}^*(x,y)dxdy \\ = & \int_0^{2\pi}\int_0^1 V_{nm}(\mathrm{r},\theta)V_{pq}^*(\mathrm{r},\theta)\mathrm{r}d\mathrm{r}d\theta \\ = & \frac{\pi}{n+1}\eta_{np}\eta_{mq} \end{aligned} \tag{4.3}$$

with

$$\eta_{ab} = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

Zernike moments are the projections of the image function $f(x,y)$ onto the orthogonal basis. For $f(x,y)$ that vanishes outside the unit circle, the Zernike moment Z_{pq} is defined as

$$Z_{pq} = \frac{p+1}{\pi}\int\int_{x^2+y^2\leq 1} V_{pq}^*(\mathrm{r},\theta)f(x,y)dxdy. \tag{4.5}$$

In polar coordinates, Z_{pq} is expressed as

$$Z_{pq} = \frac{p+1}{\pi}\int_0^{2\pi}\int_0^1 V_{pq}^*(\mathrm{r},\theta)f(\mathrm{r},\theta)\mathrm{r}d\mathrm{r}d\theta. \tag{4.6}$$

Replacing the integrals in Equation 4.5 with summations, the Zernike moments of discrete image functions can be computed as

$$Z_{pq} = \frac{p+1}{\pi}\sum_x\sum_y V_{pq}^*(\mathrm{r},\theta)f(x,y), \quad x^2+y^2 \leq 1. \tag{4.7}$$

Magnitudes of Zernike moments are naturally rotational invariants. Given $f(\mathrm{r},\theta)$ and $f^{rot}(\mathrm{r},\theta-\alpha)$ as the image functions of the original image and its rotated version by an angle α in the counterclockwise direction respectively, $Z_{pq}^{rot} = Z_{pq}e^{-jq\alpha}$. Since

$|e^{-jq\alpha}| = 1$, the magnitude values before and after rotation are identical.

Summation to infinite order p and repetition q is impossible. Image reconstruction can be performed using Zernike moments up to order max as shown in Equation 4.8:

$$\hat{f}(x,y) = \sum_{p=0}^{max} \sum_{q} Z_{pq} V_{pq}(\mathrm{r},\theta) \tag{4.8}$$

q in Equation 4.8 are both positive and negative. That is,

$$\hat{f}(x,y) = \sum_{p=0}^{max} \sum_{q<0} Z_{pq} V_{pq}(\mathrm{r},\theta) + \sum_{p=0}^{max} \sum_{q\geq 0} Z_{pq} V_{pq}(\mathrm{r},\theta). \tag{4.9}$$

Z_{pq} and V_{pq} are both complex quantities. Since working with real-valued functions is easier; noting that $V_{pq}^{*} = V_{p,-q}$, Equation 4.8 is expanded as

$$\begin{aligned}
\hat{f}(x,y) &= \sum_{p=0}^{max} \sum_{q>0} Z_{p,-q} V_{p,-q}(\mathrm{r},\theta) + \sum_{p=0}^{max} \sum_{q\geq 0} Z_{pq} V_{pq}(\mathrm{r},\theta) \\
&= \sum_{p=0}^{max} \sum_{q>0} Z_{p,q}^{*} V_{p,q}^{*}(\mathrm{r},\theta) + \sum_{p=0}^{max} \sum_{q\geq 0} Z_{pq} V_{pq}(\mathrm{r},\theta) \\
&= \left[\sum_{p=0}^{max} \sum_{q>0} \left[Z_{p,q}^{*} V_{p,q}^{*}(\mathrm{r},\theta) + Z_{pq} V_{pq}(\mathrm{r},\theta) \right] \right] + Z_{p0} V_{p0} \\
&= \left[\sum_{p=0}^{max} \sum_{q>0} \left\{ \Big(\mathrm{Re}(Z_{pq}) - j\ \mathrm{Im}(Z_{pq}) \Big) R_{pq}(\mathrm{r})(\cos p\theta - j \sin p\theta) \right.\right. \\
&+ \left.\left. \Big(\mathrm{Re}(Z_{pq}) + j\ \mathrm{Im}(Z_{pq}) \Big) R_{pq}(\mathrm{r})(\cos p\theta + j \sin p\theta) \right\} \right] \\
&+ \Big(\mathrm{Re}(Z_{p0}) + j\ \mathrm{Im}(Z_{p0}) \Big) R_{p0}(\mathrm{r})
\end{aligned} \tag{4.10}$$

and the reconstruction formula using moments up to order max appears in Equation 4.11,

$$\hat{f}_{max}(x,y) = \frac{C_{p0}}{2} R_{p0}(\mathrm{r}) + \sum_{p=0}^{max} \sum_{\substack{q=0 \\ p-q \text{ even}}}^{p} (C_{pq} \cos q\theta + S_{pq} \sin q\theta) R_{pq}(\mathrm{r}) \tag{4.11}$$

where C_{pq} and S_{pq} are given by

$$C_{pq} = 2\,\mathrm{Re}(Z_{pq}) = \frac{2p+2}{\pi} \int\int_{x^2+y^2\leq 1} f(x,y) R_{pq}(\mathrm{r}) \cos q\theta \; dxdy \tag{4.12}$$

$$S_{pq} = -2\,\mathrm{Im}(Z_{pq}) = \frac{-2p-2}{\pi} \int\int_{x^2+y^2\leq 1} f(x,y) R_{pq}(\mathrm{r}) \sin q\theta \; dxdy. \tag{4.13}$$

The total number of Zernike moments used to reconstruct an image is

$$N_{total} = \begin{cases} \left(\frac{max+2}{2}\right)^2 & max \text{ is even} \\ \left(\frac{max+1}{2}\right)^2 + \left(\frac{max+1}{2}\right) & max \text{ is odd.} \end{cases} \tag{4.14}$$

During image reconstruction, the individual components of the orthogonal Zernike moments are added to obtain the reconstructed image. Low order moments represent low-frequency image content while high order moments describe fine detail.

4.2. Computation of Exact Zernike Moments

Hosny [145] proposes a method for exact computation of full and subsets of Zernike moments. The centers of pixels of an NxM digital image are denoted by (x_i, y_j) at which the image function f is defined. Before the computations are carried, a transformed image defined in $[-1/\sqrt{2}, 1/\sqrt{2}]$ x $[-1/\sqrt{2}, 1/\sqrt{2}]$ is obtained by a square to circular mapping shown in Figure 4.1. The transformed image coordinates are given by

$$x_i = \frac{2i-N-1}{N\sqrt{2}}, \quad y_j = \frac{2j-M-1}{M\sqrt{2}} \tag{4.15}$$

$$\mathrm{r}_{ij} = \sqrt{x_i^2 + y_j^2}, \quad \theta_{ij} = \tan^{-1}(y_j/x_i) \tag{4.16}$$

with $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, M$. The sampling intervals, Δx and Δy, for the transformed image (i.e. horizontal and vertical sizes of pixels) are

$$\Delta x = \sqrt{2}/N, \quad \Delta y = \sqrt{2}/M. \tag{4.17}$$

Real-valued radial Zernike polynomials defined in Equation 4.2 can be rewritten as

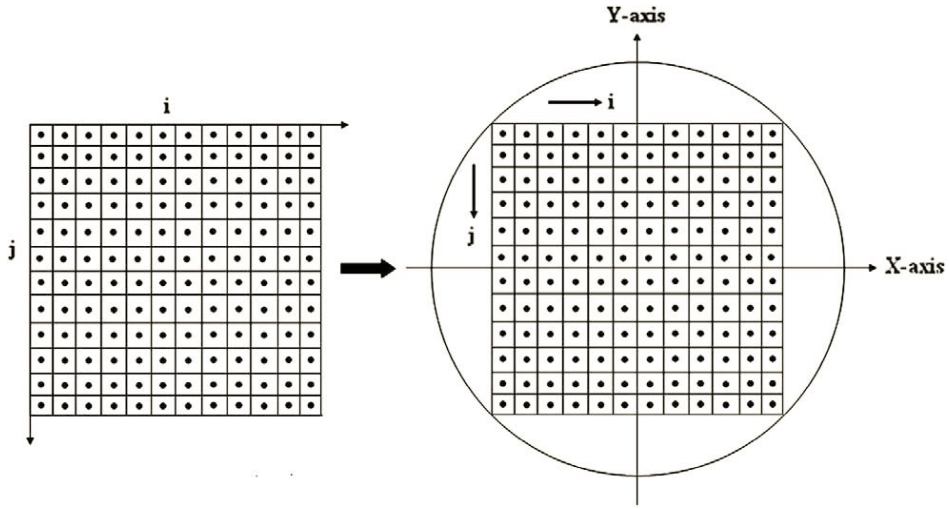

Figure 4.1. Square to circular mapping: reprinted from Hosny [145].

$$R_{pq}(\mathrm{r}) = \sum_{\substack{k=q \\ p-k \text{ even}}}^{p} A_{pqk}\mathrm{r}^k \tag{4.18}$$

where A_{pqk} are the coefficients of those polynomials defined as

$$A_{pqk} = \frac{(-1)^{((p-k)/2)}\left(\frac{p+k}{2}\right)!}{\left(\frac{p-k}{2}\right)!\left(\frac{k+q}{2}\right)!\left(\frac{k-q}{2}\right)!}. \tag{4.19}$$

Computation of A_{pqk} is a time consuming process. Instead, the recurrence relations in Equation 4.20, Equation 4.21 and Equation 4.22 are used to efficiently compute these coefficients:

$$A_{ppp} = 1 \tag{4.20}$$

$$A_{p(q-2)p} = \frac{p+q}{p-q+2} A_{pqp} \tag{4.21}$$

$$A_{pq(k-2)} = -\frac{(k+q)(k-q)}{(p+k)(p-k+2)} A_{pqk} \tag{4.22}$$

The image-independent A_{pqk} coefficients can be precomputed and stored for future use.

The relation between Zernike moments and complex moments is given by

$$Z_{pq} = \frac{p+1}{\pi} \sum_{\substack{k=|q| \\ p-k \text{ even}}} A_{pqk} C_{\frac{k-q}{2}, \frac{k+q}{2}} \tag{4.23}$$

where A_{pqk} are the coefficients of radial Zernike polynomials and $C_{p,q}$ are the complex moments of order $p+q$ defined, making use of the binomial theorem, as a linear combination of geometric moments G of the same order or less:

$$C_{p,q} = \sum_{a=0}^{p} \sum_{b=0}^{q} \binom{p}{a} \binom{q}{b} (-1)^b j^{a+b} G_{p+q-a-b,a+b} \tag{4.24}$$

Geometric moments of order $p+q$ is defined as

$$G_{p,q} = \int_{\frac{-1}{\sqrt{2}}}^{\frac{1}{\sqrt{2}}} \int_{\frac{-1}{\sqrt{2}}}^{\frac{1}{\sqrt{2}}} x^p y^q f(x,y) dx dy. \tag{4.25}$$

Exact computation of complex moments is based on exact computation of geometric moments. A procedure to compute exact values of geometric moments is introduced also by Hosny [147].

Zernike moments can easily be computed according to the difference m of moment order p and repetition q as illustrated in Figure 4.2. The formulae for computing Zernike

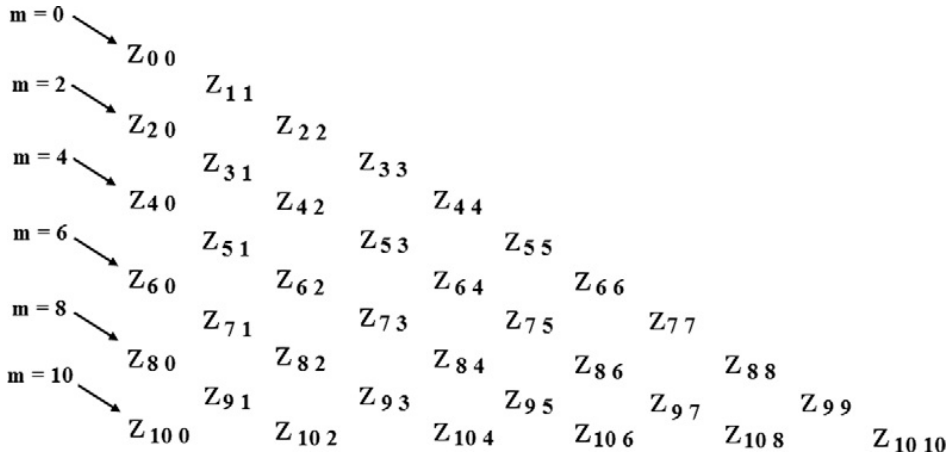

Figure 4.2. Zernike moments computation according to the difference m between moment order p and repetition q: reprinted from Hosny [145].

moments are given in Equation 4.26 and Equation 4.27,

$$Z_{p,p} = \frac{p+1}{\pi} C_{0,p} \tag{4.26}$$

$$Z_{p,p-m} = \sum_{a=0}^{\frac{m}{2}} A_{p,p-m,p-2a} C_{\frac{m}{2}-a,p-\frac{m}{2}-a} \tag{4.27}$$

where $p = 0, 1, 2, \ldots, max$, m is an integer that starts at $m = 2$ and incremented by two at each iteration, $m \leq p$ and $p - m$ is even. The pseudocode of the algorithm for computing the full set of Zernike moments up to order max is shown in Figure 4.3.

```
1: for p ← 0 : 1 : max do
2:     Z_{p,p} ← ((p + 1)/π) * C_{0,p}
3: for m ← 2 : 2 : max do
4:     for p ← m : 1 : max do
5:         Compute Z_{p,p−m} using the formula of Equation 4.27
```

Figure 4.3. Pseudo-code for computation of full set of Zernike moments.

4.3. Invariant Computation with Translation and Scale Normalization

The rotational invariance of the magnitudes of Zernike moments makes it sufficient to transform the set of all processed images to a frame where their translation and scale parameters are the same (i.e. translation and scale uniformity) prior to moments computation in order to obtain robust results from automatized processes that utilize these magnitudes. We adopt the methodology of Khotanzad and Hong [146] for translation and scale normalization.

Translation and scale uniformity are achieved by utilizing general (regular) moments m_{pq} of images. An image is transformed into a new one whose first order moments m_{01} and m_{10} are both zero in order to obtain translation invariance. This is equivalent to saying that the centroid of the original image $f(x, y)$ (i.e. that of the ROI) is shifted such that the centroid of the new image is the coordinate origin of the rectangular image frame. Notationally, $f(x, y)$ becomes $f(x + \bar{x}, y + \bar{y})$ where $\bar{x}$ and $\bar{y}$ are

$$\bar{x} = \frac{m_{10}}{m_{00}},\ \bar{y} = \frac{m_{01}}{m_{00}}. \tag{4.28}$$

Scale invariance is accomplished by adjusting the value of the general moment m_{00} to a fixed value ν for all images. In the case of binary images, it should be noted that m_{00} is the total number of ROI pixels. Denoting with $f(x/a, y/a)$ a scaled version of the original image function $f(x, y)$, m_{pq} of $f(x, y)$ and m'_{pq} of $f(x/a, y/a)$ are related by

$$\begin{aligned} m'_{pq} &= \sum_x \sum_y x^p y^q f(x/a, y/a) \\ &= \sum_x \sum_y a^p x^p a^q y^q f(x, y) a^2 \\ &= \sum_x \sum_y a^{p+q+2} x^p y^q f(x, y) \\ &= a^{p+q+2} \sum_x \sum_y x^p y^q f(x, y) \\ &= a^{p+q+2} m_{pq}. \end{aligned} \tag{4.29}$$

The target $m'_{00} = \nu$ requires the substitution $a = \sqrt{\nu/m_{00}}$. Scale invariance is achieved

by transforming the image $f(x, y)$ to $f(x/a, y/a)$ with $a = \sqrt{\nu/m_{00}}$.

To sum up, both scale and translation invariance are obtained by transforming the original image function $f(x, y)$ into a new image function $f_{norm}(x, y)$ where

$$f_{norm} = f(x/a + \bar{x} + y/a + \bar{y}) \tag{4.30}$$

with $(\bar{x}, \bar{y})$ the centroid of $f(x, y)$, $a = \sqrt{\nu/m_{00}}$ and ν a predetermined value. If the coordinate $(x/a + \bar{x}, y/a + \bar{y})$ does not correspond to a pixel location, the associated function value is interpolated from neighboring pixels.

4.4. Number of Zernike Features to Use in Classification

The inherent rotational invariance property of the magnitudes of Zernike moments accompanied with translation and scale normalization applied on input images (as outlined in Section 4.3) before moment computation suggest their use in classification as descriptive features. It is also noteworthy to state that, the scale and translation uniformity obtained with the procedure of Section 4.3 makes two of the Zernike features valueless. These two features are $|Z_{00}|$ that is the same for all images and $|Z_{11}|$ that is zero. To see why this is so, the observations shown in Equation 4.31, Equation 4.32, Equation 4.33 and Equation 4.34 regarding the values of C_{00}, S_{00}, C_{11} and S_{11} using which $|Z_{00}|$ and $|Z_{11}|$ can be derived are made:

$$\begin{aligned} C_{00} &= \frac{2}{\pi} \int\int_{x^2+y^2 \leq 1} f_{norm}(x, y) R_{00}(\mathrm{r}) dx dy \\ &= \frac{2}{\pi} \int\int_{x^2+y^2 \leq 1} f_{norm}(x, y) dx dy \\ &= \frac{2}{\pi} m_{00} \end{aligned} \tag{4.31}$$

$$S_{00} = 0 \tag{4.32}$$

$$\begin{aligned} C_{11} &= \frac{4}{\pi}\int\int_{x^2+y^2\leq 1} f_{norm}(x,y)R_{11}(\mathrm{r})\cos\theta\ dxdy \\ &= \frac{4}{\pi}\int\int_{x^2+y^2\leq 1} f_{norm}(x,y)\mathrm{r}\cos\theta\ dxdy \\ &= \frac{4}{\pi}\int\int_{x^2+y^2\leq 1} f_{norm}(x,y)xdxdy \\ &= \frac{4}{\pi}m_{10} \end{aligned} \tag{4.33}$$

$$\begin{aligned} S_{11} &= \frac{4}{\pi}\int\int_{x^2+y^2\leq 1} f_{norm}(x,y)R_{11}(\mathrm{r})\sin\theta\ dxdy \\ &= \frac{4}{\pi}\int\int_{x^2+y^2\leq 1} f_{norm}(x,y)\mathrm{r}\sin\theta\ dxdy \\ &= \frac{4}{\pi}\int\int_{x^2+y^2\leq 1} f_{norm}(x,y)ydxdy = \frac{4}{\pi}m_{01} \end{aligned} \tag{4.34}$$

Since $m_{00} = \nu$, $|Z_{00}| = |(C_{00}/2) - j(S_{00}/2)| = \nu/\pi$ and since $m_{10} = m_{01} = 0$, $|Z_{11}| = |(C_{11}/2) - j(S_{11}/2)| = 0$ for all normalized images.

Automatically selecting the maximum number of Zernike features (i.e. the maximum order p^* up to which absolute values of Zernike moments are used) necessary to represent images for classification is inspired by the idea of Khotanzad and Hong [146]. It is soberly argued that a good set of features for a given image is one which can characterize and represent the image well. The difference between an image and its reconstructed version using a finite set of its moments is a good indicator of how well the image is represented with those moments. The less this difference is, the better the representation. Feature selection can be based on an iterative process that reconstructs a given image with a set of moments (of order $p = 0$ through $p = max$), measures the difference between the original image and its reconstruction and decides on the representation power of the set of used moments (i.e. whether the used moments are sufficient or not) with respect to a threshold ϵ; where max starts at zero and is incremented at each successive iteration. In other words, the maximum order p^* of moments used to represent an image can be found by reconstructing the image with all moments having maximum order max and checking if the difference of the two images is less than the distance threshold ϵ. In the case of binary images, the reconstruction

$\hat{f}(x,y)$ of $f(x,y)$ using moments of order $p = 0$ through $p = max$ computed from $f(x,y)$ is given by

$$\hat{f}(x,y) = \mathrm{M}\left(\left|\sum_{p=0}^{max}\sum_{q} Z_{pq}V_{pq}(\mathrm{r},\theta)\right|\right) \forall x \forall y \tag{4.35}$$

where M(.) is a function that consists of a mapping to $[0, 255]$ gray-level range and thresholding at 128. Generalizing Equation 4.35 to gray-level images is straightforward. Finally, the absolute difference between $\hat{f}$ and f is measured using Hamming distance $\mathrm{H}(f,\hat{f})$, which is the total number of pixels that differ in f and $\hat{f}$. The value of p^* is found using a procedure where reconstructions are performed using moments of order $p = 0$ through $p = max$ and checking $\mathrm{H}(f,\hat{f})$ until $\mathrm{H}(f,\hat{f}) < \epsilon$, where $max = 0, 1, \ldots, maxp$. With the assumption that the moments up to maximum order $p \leq maxp$ output a reconstructed image $\hat{f}$ from the moments of f where $\mathrm{H}(f,\hat{f}) < \epsilon$ for some p, the maximum order required for a good representation of f is p. The value of $maxp$ is determined by trial and error. For a given database of N images, p^{max} is the maximum of all p values (i.e. $p_i,\ i = 1, 2, \ldots, N$) acquired for all images.

4.5. Feature Computation for Fetal Skull Images

The steps followed in the entire process of Zernike moments computation, including normalization with respect to translation and scale, determining the maximum order of moments required for a good classification of the images (i.e. shapes) in the images database are presented in this section.

The shapes database used in description is obtained by manually marking 24 points along the boundaries of fetal skulls each present in one US image of the images database and placing nine more points between each two consecutive marked points with cubic B-splines to obtain 24 + 24x9 = 240 points in total. For each image, the 240 pairs of 240 consecutive points are linearly connected consistent with their order on the boundary. After the closed boundary corresponding to the contour of the skull in the image is obtained, the internal region enclosed by the boundary is filled in with white

pixels to obtain a binary image where white pixels are those of the skull region and black pixels are outside. The image at this stage is cropped along the four sides of the rectangular image frame so that the white skull region is displayed in a compact frame containing fewer pixels than the initial image. Provided that point-marking is done with maximum accuracy expected from a medical expert, the described process produces a shapes database consisting of ground truth shapes of skulls contained in the images of the images database, one skull in each image. In our collection of transcerebellar skull images, 358 fetal skulls and hence 358 skull shapes in the shapes database exist. Figure 4.4 illustrates two examples from the images database and the corresponding two shapes from the shapes database obtained manually. All skull shapes from which

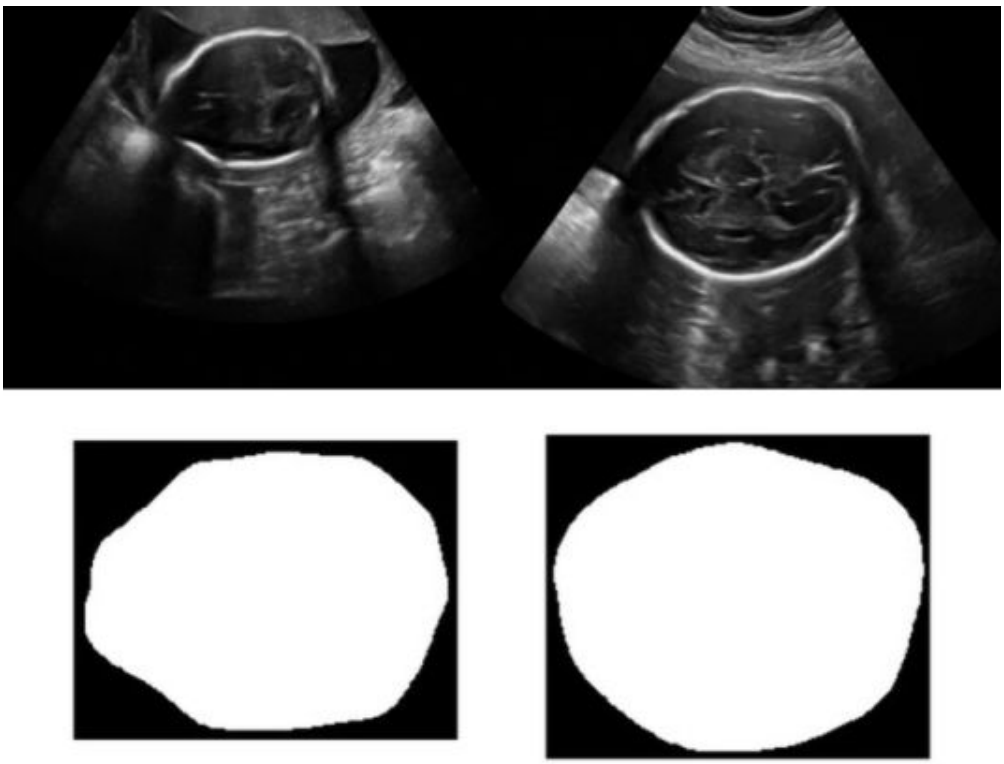

Figure 4.4. Two examples from the images database and corresponding shapes.

the corresponding moments are computed are included in 256x256 images. Prior to moments computation, each of these square images is supposed to contain one skull shape normalized with respect to translation and scale. A 256x256 image is made up of 65,536 pixels. The ν parameter that we select for scale uniformity is 12,800. That is, the skull shapes after scale normalization are intended to contain 12,800 white pixels

in the 256x256 image. Given that a shape before scale normalization consists of m_{00} white pixels, the shape must be stretched/shrank in both x and y dimensions with a factor of

$$k = \sqrt{\frac{\nu}{m_{00}}} \tag{4.36}$$

so that the shape consists of ν pixels after normalization. Due to discretization (quantization) involved in the stretching/shrinking process, it is probable that the scale-normalized shape does not contain exactly ν pixels but as many as the best approximation of this number. More precisely, let $|m|$ and $|n|$ be the integer-valued difference between the maximum and minimum x coordinates of shape pixels and that of y coordinates defined as

$$|m| = x_{max} - x_{min} + 1 \text{ and } |n| = y_{max} - y_{min} + 1. \tag{4.37}$$

$|m|$ and $|n|$ can be referred to as perpendicular (with respect to x or y axes) lengths of the shape as appearing in the initial shape image. The perpendicular lengths of the scale-normalized shape $|M|$ and $|N|$ are then defined as

$$|M| = \text{round}(k.|m|) \text{ and } |N| = \text{round}(k.|n|). \tag{4.38}$$

where round(.) quantizes its argument to the nearest integer. Having retrieved $|m|$, $|n|$ and having computed $|M|$, $|N|$; the $|m|$x$|n|$ rectangular image obtained from the input shape image where the shape pixels touch all the four sides of the image frame, is resized to a new image of size $|M|$x$|N|$ which serves as the scale-normalized shape image I_s. What remains to complete the normalization process is to place I_s in a 256x256 blank image I_{blank} such that the center of gravity of the shape I_s (obtained via regular moments m_{01}, m_{10} and m_{00}) coincides with pixel $(128, 128)$ of I_{blank}. As a result, the translation normalization is also achieved. The output is the final shape image I_{st} after scale and translation normalization. When the described procedure is applied on all shape images in the shapes database, all output images possess translation and scale

uniformity. It should be once more noted that no rotation is involved during normalization since magnitudes of Zernike moments are invariant under rotation. Figure 4.5 displays the normalized versions of the shapes in Figure 4.4 with respect to scale and translation. The maximum order of moments required in classification is determined

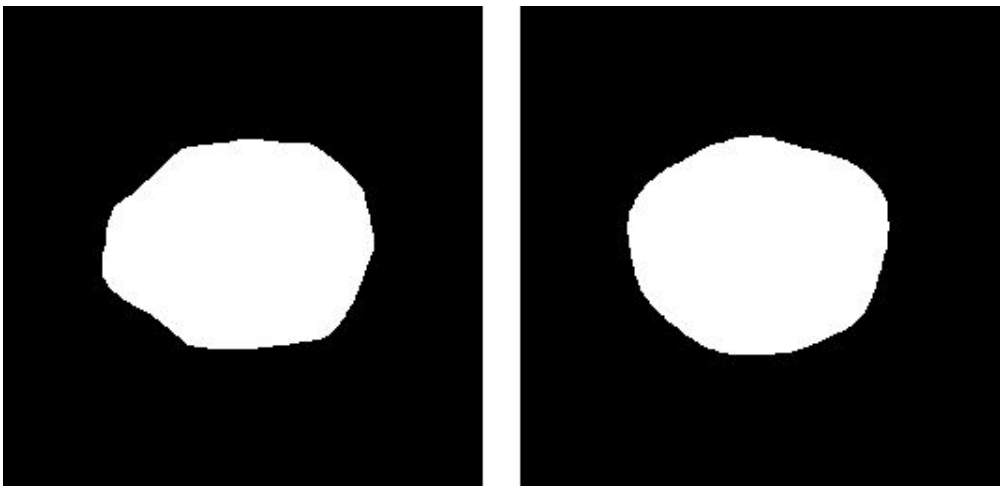

Figure 4.5. Shapes of Figure 4.4 normalized with respect to scale and translation.

with the moments computed from I_{st} images of all shapes in the database and the reconstructions of I_{st} (i.e. I_{rec}) using the computed moments up to the order with a reconstruction error $\mathrm{H}(I_{rec}, I_{st}) \leq \epsilon = 150$. $\nu = 12,800$ and $\epsilon = 150$ imply that the used moments must provide reconstructions with an error of at most 1.17%. Figure 4.6 presents the algorithm that runs on all shapes in the shapes database and outputs the maximum order of Zernike moments used in classification. For reference, Figure 4.7 shows the reconstructions of the normalized shapes I_{st} of Figure 4.5 using Zernike moments of sufficient order ($\mathrm{H}(I_{st}, I_{rec}) \leq 150$).

The p^{max} value (maximum order of moments whose magnitudes are to be used in classification) determined by the algorithm of Figure 4.6 turns out to be 43 for the 358 skull shapes of our collection. The number of moments up to order 43 is $((43+1)/2)^2 + ((43+1)/2) = 506$ (Refer to Equation 4.14. Excluding $|Z_{00}|$ and $|Z_{11}|$, that are not distinguishing when the normalization of Section 4.3 is applied before moments computation, results in feature vectors of size 504 for each skull shape.

```
1: p^max ← 0                                          ▷ maximum order of Zernike moments
2: for all shapes I in the shapes database do
3:     Normalize I with the procedure of Section 4.3 to obtain the corresponding I_st
4:     Compute moments Z of I_st up to order 50 with the procedure of Section 4.2
5:     p ← 0
6:     while p < 51 do                                ▷ Assume while loop breaks with p ≤ 50
7:         Reconstruct I_st using Z of order 0 through p with Equation 4.11 to get I_rec
8:         if H(I_st, I_rec) > 150 then               ▷ Hamming distance test
9:             p ← p + 1
10:        else
11:            break
12:    if p > p^max then
13:        p^max ← p
14: return p^max
15: return |Z| for all shapes
```

Figure 4.6. Pseudo-code for determining the maximum order of Zernike moments whose magnitudes are used as features in classification.

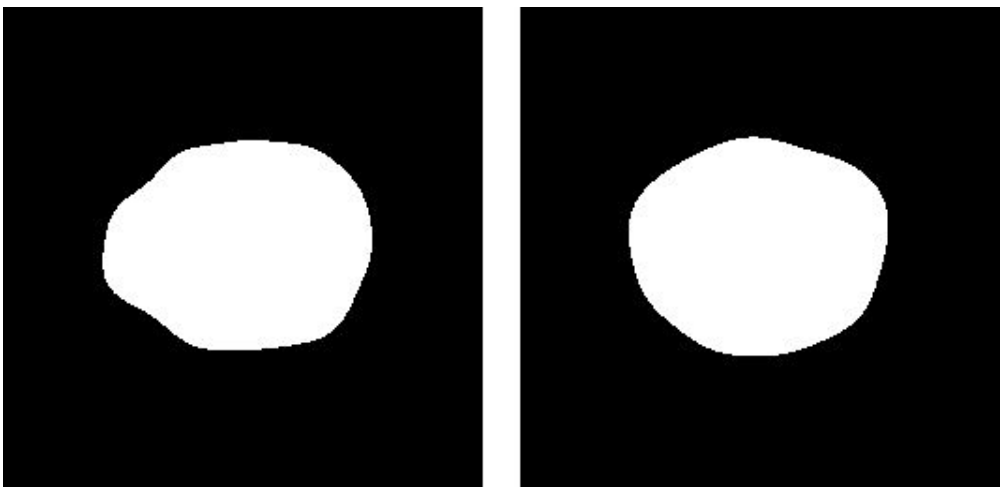

Figure 4.7. Reconstructions of the normalized shapes in Figure 4.5 (left: $\mathrm{H}(I_{st}, I_{rec}) = 135$), right: $\mathrm{H}(I_{st}, I_{rec}) = 147$).

5. SUPPORT VECTOR MACHINES CLASSIFICATION

Support Vector Machines (SVM) is a popular method that learns the linear discriminant of two classes by modeling a separating hyperplane. Assuming a linearly-separable distribution, the principal idea in the design of an SVM classifier is to maximize the distance of instances to the separating hyperplane on either side of it belonging to different class memberships. The key characteristics of SVM learning can be conceived by this simple approach, however most real data sets are not linearly separable because of noise and in the general nonseparable case the inductive bias is maximizing generality as well as minimizing training error. The learning strategy is introduced by Vapnik [104, 105, 148–150] and has been used in many machine learning applications due to its theoretical strength and performance. In this thesis, SVMs are utilized for classifying fetal skulls as either *defective* or *healthy* using 504-dimensional feature vectors (magnitudes of Zernike moments of skull shapes).

5.1. Separable Case: Maximal Margin

$\mathbf{C}_1$ and $\mathbf{C}_2$ being the classes in a two-class problem, let $X = \{x^t, r^t\}$ denote training samples where x^t are input column vectors and r^t are corresponding class labels being either +1 or -1 (i.e. $r^t = +1$ if $x^t \in \mathbf{C}_1$ and $r^t = -1$ if $x^t \in \mathbf{C}_2$). The aim is to find the weight vector $\mathbf{w}$ and constant w_0 such that the conditions

$$\mathbf{w}^T x^t + w_0 \geq +1 \text{ for } r^t = +1 \tag{5.1}$$

$$\mathbf{w}^T x^t + w_0 \leq -1 \text{ for } r^t = -1 \tag{5.2}$$

are satisfied. The inequalities in Equation 5.1 and Equation 5.2 can be rewritten as

$$r^t(\mathbf{w}^T x^t + w_0) \geq +1. \tag{5.3}$$

What is required is not simply

$$r^t(\mathbf{w}^T x^t + w_0) \geq 0, \tag{5.4}$$

which holds when an instance is on the correct side of the separating hyperplane. It is also desired that the instances are some distance away. The distance between the closest instances on either side of the hyperplane is called the *margin* which is to be maximized to attain best generalization. The separating hyperplane that maximizes the margin is called the *optimal-separating hyperplane.*

The distance of x^t to the discriminant is

$$\frac{|\mathbf{w}^T x^t + w_0|}{||\mathbf{w}||}. \tag{5.5}$$

With $r^t \in \{-1, +1\}$, the distance of Equation 5.5 can be written as

$$\frac{r^t(\mathbf{w}^T x^t + w_0)}{||\mathbf{w}||} \tag{5.6}$$

and this distance must be at least some value ρ as in

$$\frac{r^t(\mathbf{w}^T x^t + w_0)}{||\mathbf{w}||} \geq \rho, \forall t. \tag{5.7}$$

Infinitely many solutions for ρ exist if $\mathbf{w}$ is scaled. For a unique solution, $\rho||\mathbf{w}||$ is fixed to one and maximizing the margin corresponds to minimizing $||\mathbf{w}||$. This quadratic optimization problem can be defined as

$$\text{minimize } \frac{||\mathbf{w}||^2}{2} \text{ subject to } r^t(\mathbf{w}^T x^t + w_0) \geq +1, \forall t \tag{5.8}$$

and solved directly to find $\mathbf{w}$ and w_0. The solution complexity depends on the input dimensionality d. As a result, there are instances on both sides of the hyperplane which are $1/||\mathbf{w}||$ away from the hyperplane. The total margin is then $2/||\mathbf{w}||$. Figure

5.1 illustrates the linearly-separable case with related concepts of optimal separating hyperplane ($\mathbf{H}(x) = 0$) and margin ($2/||\mathbf{w}||$). The instances that fall on one of the two hyperplanes ($\mathbf{H}(x) = -1$ and $\mathbf{H}(x) = +1$ in Figure 5.1) are called *support vectors*. If the instances are not linearly separable, the input vectors can be mapped to a

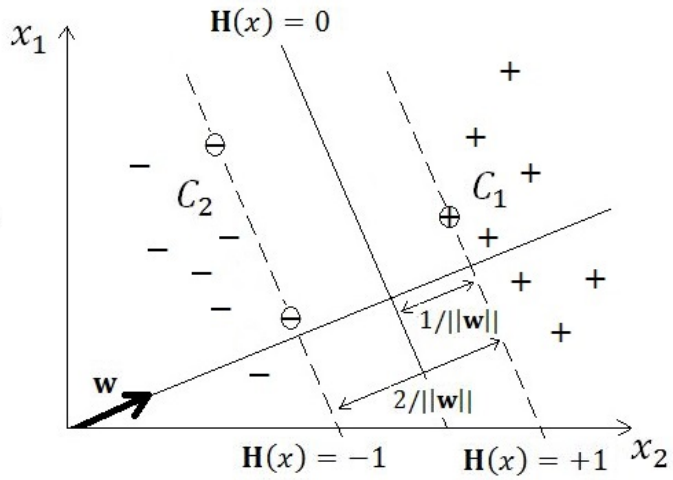

Figure 5.1. Linearly separable inputs (support vectors circled).

higher dimensional space, where linear separation is possible, by using nonlinear basis functions. Linear separability in the new space corresponds to separability by non-linear functions in the original input space. Since the dimensionality d of the new space is generally much higher, it is desired to make the complexity of the optimization problem depend not on d but on the number of training instances N. In the new formulation, Equation 5.8 is written as an unconstrained Lagrangian problem,

$$\begin{aligned} L_p(\mathbf{w}, w_0, \alpha) &= \frac{1}{2}||\mathbf{w}||^2 - \sum_{t=1}^{N} \alpha^t [r^t(\mathbf{w}^T x^t + w_0) - 1] \\ &= \frac{1}{2}||\mathbf{w}||^2 - \sum_{t} \alpha^t r^t(\mathbf{w}^T x^t + w_0) + \sum_{t} \alpha^t, \end{aligned} \tag{5.9}$$

where α^t are the Lagrange multipliers. L_p should be minimized with respect to $\mathbf{w}$ and w_0 and maximized with respect to $\alpha^t \geq 0$. Consequently, the saddle point is the solution. The dual problem of this convex quadratic optimization problem can equivalently be solved making use of the Karush-Kuhn-Tucker conditions. In the dual

problem, L_p is maximized with respect to α^t subject to the constraints that the partial derivatives of L_p with respect to $\mathbf{w}$ and w_0 are zero as in Equation 5.10 and Equation 5.11 together with $\alpha^t \geq 0$:

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_t \alpha^t r^t x^t \tag{5.10}$$

$$\frac{\partial L_p}{\partial w_0} = 0 \Rightarrow \sum_t \alpha^t r^t = 0 \tag{5.11}$$

The dual of L_p in Equation 5.9 is stated as

$$\begin{aligned} L_d(\alpha) &= \frac{1}{2}(\mathbf{w}^T\mathbf{w}) - \mathbf{w}^T \sum_t \alpha^t r^t x^t - w_0 \sum_t \alpha^t r^t + \sum_t \alpha^t \\ &= -\frac{1}{2}(\mathbf{w}^T\mathbf{w}) + \sum_t \alpha^t \\ &= -\frac{1}{2}\sum_t \sum_u \alpha^t \alpha^u r^t r^u (x^t)^T x^u + \sum_t \alpha^t. \end{aligned} \tag{5.12}$$

L_d is maximized with respect to α^t subject to $\sum_t \alpha^t r^t = 0$ and $\alpha^t \geq 0, \forall t$. Quadratic optimization methods are used to solve the dual problem whose solution complexity depends on the sample size N. There are N solutions for α^t, most of which vanish with $\alpha^t = 0$. Only a small fraction of the solutions has $\alpha^t \geq 0$ and their corresponding x^t are the support vectors. $\mathbf{w}$ is the weighted sum of these instances as shown in Equation 5.10. The support vectors are the x^t satisfying

$$r^t(\mathbf{w}^T x^t) = 1 \tag{5.13}$$

and lying on the margin. Any support vector can be used to compute w_0 as

$$w_0 = r^t - \mathbf{w}^T x^t. \tag{5.14}$$

In practice, it is customary to perform the calculation in Equation 5.14 for all support

vectors and take the average in order to attain numerical stability. The hyperplane that defines the discriminant is given by

$$\mathbf{H}(x) = \mathbf{w}^T x + w_0 \tag{5.15}$$

and the classifier decides the class to be $\mathbf{C}_1$ ($r^t = +1$) if $\mathbf{H}(x) > 0$ and $\mathbf{C}_2$ ($r^t = -1$) otherwise.

5.2. Nonseparable Case: Soft-Margin Hyperplane

When the input data are not linearly separable, a hyperplane that results in minimum error is sought. Slack variables ξ^t to store deviations from the margin are defined. Deviations may be such that an instance is located on the wrong side of the hyperplane or on the correct side of but not far enough from it. Figure 5.2 shows examples of training instances with no deviation (i.e. correctly classified with $\xi = 0$) and with the two types of deviations (i.e. misclassified with $\xi \geq 1$ and correctly classified but inside the margin with $0 \leq \xi < 1$). Using the slack variables, Equation

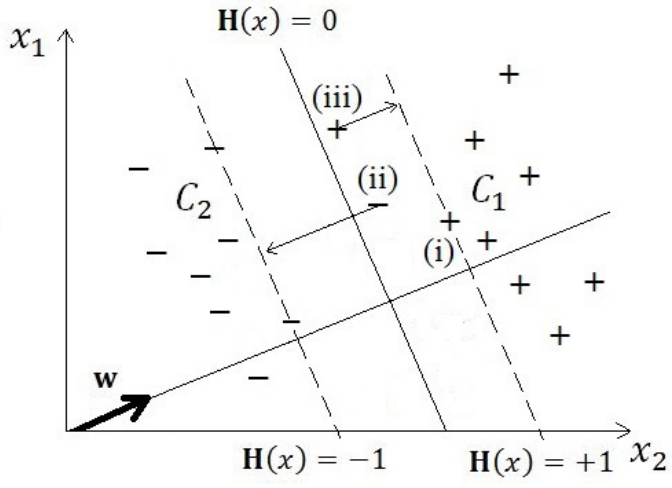

Figure 5.2. Linearly nonseparable inputs: (i) correct side, sufficiently away (ii) wrong side (iii) correct side, in the margin.

5.3 is relaxed as

$$r^t(\mathbf{w}^T x^t + w_0) \geq 1 - \xi^t. \tag{5.16}$$

and there is no problem as long as $\xi^t = 0$. If $0 < \xi^t < 1$, x^t is correctly classified but in the margin and if $\xi^t \geq 1$, x^t is misclassified. The *soft error* is defined as $\sum_t \xi^t$ and added to the primal equation L_p as a penalty term:

$$L_p(\mathbf{w}, w_0, \alpha) = \frac{1}{2}||\mathbf{w}||^2 + \mathrm{C}\sum_t \xi^t - \sum_t \alpha^t[r^t(\mathbf{w}^T x^t + w_0) - 1 + \xi^t] - \sum_t \beta^t \xi^t. \tag{5.17}$$

β^t are the new Langrange parameters to assure ξ^t are positive and C is the penalty factor. Equation 5.17 penalizes both misclassified instances and correctly classified ones that are inside the margin for better generalization. The dual problem L_d is defined such that the function to maximize with respect to α^t is identical to that of Equation 5.12 subject to the constraints $\sum_t \alpha^t r^t = 0$ and $0 \leq \alpha^t \leq \mathrm{C}, \forall t$. As in the separable case, $\mathbf{w}$ is defined by the few non-zero α^t and w_0 is solved for similarly.

5.3. Kernel Functions

An SVM is a linear classifier, but in most cases it is restrictive. SVMs can be extended to non-linear classifiers by mapping the input to a higher-dimensional space using suitably chosen basis functions $\bar{\Phi}(x)$. The linear model in the new k-dimensional z space achieved by a nonlinear transformation of the original input x corresponds to a nonlinear model in the original d-dimensional x space. New dimensions are calculated through the basis functions as

$$z = \bar{\Phi}(x) \tag{5.18}$$

where $z_j = \bar{\Phi}_j(x), j = 1, \ldots, k$. Assuming $z_1 = \bar{\Phi}_1(x) \equiv 1$ and not using a separate w_0 term, the discriminant in the z-space can be written as

$$\mathbf{H}(x) = \mathbf{w}^T z = \mathbf{w}^T \bar{\Phi}(x) = \sum_{j=1}^{k} w_j \bar{\Phi}_j(x). \tag{5.19}$$

Generally, k is much larger than d and also larger than N and hence, solving the dual problem is advantageous to solving the primal problem. Since there is no guarantee that the input is linearly separable in the new z-space, the more general case of the soft margin hyperplane is used to define the dual problem and obtain a solution. Selecting the value of the C parameter is critical since a large value would highly penalize nonseparable points and many support vectors would be stored causing overfitting, whereas a small C would lead to underfitting. The solution is

$$\mathbf{w} = \sum_t \alpha^t r^t z^t = \sum_t \alpha^t r^t \bar{\Phi}(x^t) \tag{5.20}$$

and the discriminant is given by

$$\mathbf{H}(x) = \mathbf{w}^T \bar{\Phi}(x) = \sum_t \alpha^t r^t \bar{\Phi}(x^t) \bar{\Phi}(x). \tag{5.21}$$

In *kernel machines*, the inner product $\bar{\Phi}(x^t)^T \bar{\Phi}(x)$ is replaced by a *kernel function*, $\mathrm{K}(x^t, x)$, between the support vectors and inputs in the original x-space to determine the discriminant as

$$\mathbf{H}(x) = \sum_t \alpha^t r^t \mathrm{K}(x^t, x). \tag{5.22}$$

Examples of kernel functions include linear functions ($\mathrm{K}(x^t, x) = x^T x^t$), polynomials of degree n ($\mathrm{K}(x^t, x) = (x^T x^t + 1)^n$), Gaussian radial-basis functions ($\mathrm{K}(x^t, x) = e^{-\frac{||x^t - x||^2}{2\sigma^2}}$) and sigmoidal functions ($\mathrm{K}(x^t, x) = tanh(2x^T x^t + 1)$).

6. SEGMENTATION

For a CAD system that uses image inputs to be of practical use, human intervention must be minimized. In an ideal scenario, an image is presented to the system and the system responds to this input. Unfortunately, the majority of input images are acquired in a raw format and robust operation of systems having image inputs (specifically the spina bifida detection system operating on fetal skull images) depends drastically on how those images are preprocessed and objects of interest (i.e. regions of interest (ROI)) are extracted.

Experience shows that as the degree of rawness increases, achieving a fully automatic segmentation with robust outputs becomes harder. Considering our case of fetal skull images acquired via US, the degrees of freedom include viewing planes (e.g. transcerebellar), scale, position, orientation, lighting conditions (settings), etc. When other conditions such as the lack of complete structures (e.g. some part of skull boundary is not visible – not due to occlusion but other factors that arise from US acquisition and that we can not clearly identify) and the presence of multiple similar structures (e.g. some portion of an image resembles a skull very much although it is not a skull or there is really more than one skull as in the cases of twin pregnancies) are taken into account, the difficulty of the segmentation task reveals itself solidly. In this section, the observations and remarks on the segmentation attempts that have shown unsuccessful are documented as well as those which can be deemed successful.

6.1. Automatic Segmentation Attempts

6.1.1. Model Fitting

Assuming that a fetal skull can coarsely be approximated with elliptical shapes, running the *Pseudo-Random Sample Consensus* (*PRANSAC*) [151] ellipse fitting algorithm and then extracting a more precise skull region using sorts of region growing approaches have been tried. PRANSAC ellipse fitting requires a three-point fitting

procedure where the ellipse center is estimated using the tangent lines to the ellipse at these points. The tangent lines are estimated using a least squares method with the feature points around the selected three ellipse points. At the very beginning, the set of image points from which the three ellipse points are selected must be determined. Various approaches have been tried including edge pixels (which often appear as discontinuous patches that harden the task), pixels with high intensities retrieved after thresholding and the *DBSCAN* (*Density Based Spatial Clustering of Applications with Noise*) [152] algorithm used to cluster neighboring pixels with similar intensities. No matter which individual method or which combination of the multiple methods are used, the problem with the selection of any three ellipse points is that there is a huge number of those triples to make it feasible for a robust fit. Leaving aside the complexity of fitting, it is difficult to decide whether some fit is good enough and if so whether another better fit exists. In short, a way (some criterion) for concluding if a particular fit is truly successful seems to be unavailable. PRANSAC ellipse fitting can be thought of as a "needle in a hay-stack" problem where no optimal (or even valid) solution is guaranteed. Moreover, even if an acceptable solution is assumed, the region growing methodology to follow can not be set with proper rules and hence the final segmentation is very likely to be poor.

6.1.2. Active Appearance Models

An *Active Appearance Model* (*AAM*) [122, 153] provides a statistical model of object shapes and appearances (i.e. texture) in training images (AAM model building). Similar objects in other images are later matched to this model and segmentation achieved (AAM fitting). AAM, in fact, is one of a group of methods that address the problem of variability commonly referred to as deformable template models. In the training phase of this sophisticated deformable template model, a set of images together with coordinates of landmark points that appear in all images is provided and statistical models of shape and appearance are produced. During fitting, the difference between a target image and the current estimate of shape and appearance are used to drive an optimization process to arrive at a matching. AAMs have been used to

model face images and in medical image interpretation. The experiments with fetal skull images show that AAM fitting fails for the majority of available samples for a number of reasons listed below:

- Modeling variances in shape and appearance is meaningful when coordinates of points that correspond and related quantities vary with small amounts. For instance; given two images and two corresponding landmark points in each image, if the point in one image is located to the top left corner of the image frame and the point in the other image is located to the lower right corner, there is no use to model shape variance since the variation spans almost the entire image frame. For another example, consider two corresponding points in two images. If the intensities at the pixels are, say zero at one image and 255 at the other, there is no use in modeling texture variance since the variation spans the entire intensity space. The scenarios of the two examples are not unrealistic with US images of fetal skulls. Figure 6.1 and Figure 6.2 show two sets of images that display the range of variation in shape and appearance, respectively.

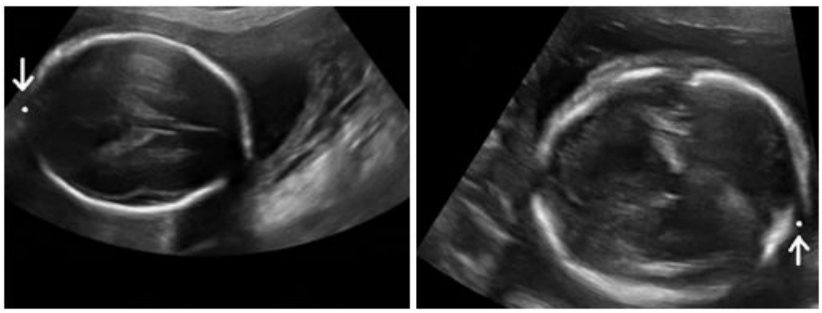

Figure 6.1. Two corresponding landmarks located very differently.

- Partially true for the case with skulls to an extent and being true in general, AAMs are not suitable for amorphous shapes such as clouds. In other words, it must be possible to obtain a training set of representative samples.
- Manual annotation of training images is cumbersome and prone to errors.
- The sizes of training sets may over-constrain variances in shape and texture.
- Initialization before AAM fitting is critical for good segmentation. If fitting starts with an inappropriate initialization, convergence to wrong outcomes is likely.

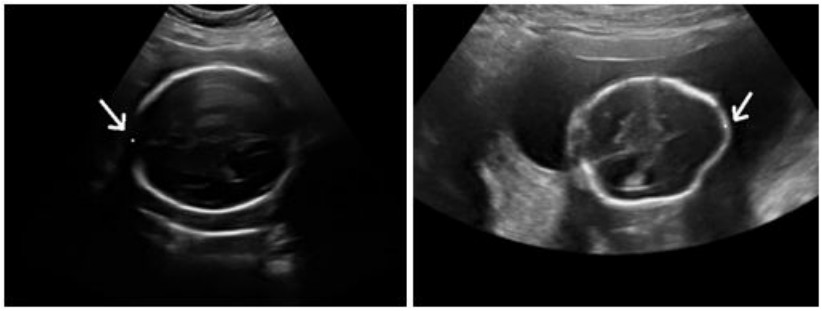

Figure 6.2. Two corresponding landmarks of considerably different intensities.

- Occlusions (e.g. missing data) result in optimization failure in AAM fitting.
- Noise may always cause problems in model building and fitting.

6.2. Full-Automaticity vs. Semi-Automaticity

Full automaticity in any machine application is a desired property. It would be very convenient if one was able to present any fetal US image to a machine-based decision-making system and get a sound response. However, this is practically impossible because available input images are acquired in very diverse conditions and automatic alignment of them is also as much impossible due to the unavailability of automatically retrieved landmarks. It would have been possible to detect landmark points (at least one) automatically if those landmarks were characterized with some property common to all such points and appropriate for machine processing, however, as far as we have experimented and to the best of our skills, such a property does not exist.

Once the chances of full-automatic segmentation are seen to have been eliminated, the objective is to achieve segmentation with minimal human interaction. This is acceptable as long as not much human effort is required and the entire CAD system serves useful purposes. Semi-automatic segmentation for the specific case of skull contour extraction is defined as follows: *Given an input image, the user marks four points on the skull contour which are the two ends of the occipitofrontal diameter (OFD) and the two ends of the biparietal diameter (BPD). After these four points are marked,*

the segmentation system runs to find a fixed number of skull boundary points evenly spaced on the boundary utilizing an average shape model obtained through marking points on a training set of images, considering the correspondences of points for all training set images and transforming all marked images to a normalized frame where normalized images are obtained by translating, rotating and scaling the training set images using appropriate parameters. The point-marking process need not know which point is which and the markings of the four points can be done in any order.

6.3. Semi-Automatic Segmentation

Segmenting fetal skulls consists of two main tasks:

(i) Constructing an average shape model using training images
(ii) Segmenting an image using the average shape model and four marked points

The first step is done once offline and thus does not contribute to running time. For a representative average model, we have preferred to do this construction using a set of training images, however, this action is in no way obligatory and other ways can be followed to obtain this average model. As long as the designer is confident, it may even be the case that an average shape model is imperatively dictated.

6.3.1. Average Shape Model Construction

For each training image, 240 (a design choice) points that are located on the object of interest (the skull boundary) are identified. Identifying the set of 240 points one by one manually is a tedious and cumbersome process. Instead of manually marking all these points, only 24 of them are marked starting from the frontal end (i.e. of the OFD) and proceeding in a clockwise orientation until the marking process arrives at the point that is only one step away from the frontal end point in a counterclockwise direction. During marking, care must be taken to ensure that each set of consecutive markings provides even spacing of points and that the definitive points of skulls (the two ends of the OFD and the two ends of the BPD) are marked at correct step indices

related to the traversal of contours. For example; the frontal end must be marked at the first step, the right end must be marked at the seventh step, the occipital end must be marked at the thirteenth step and the left end must be marked at the nineteenth step. Following the markings of the 24 points, cubic B-splines are used to locate nine more points between points in each pair of these 24 consecutive points. As a result, the contour is defined with 240 evenly spaced points (24 + 24x9 = 240 points in total). Clearly, the acquisition of the 240 boundary points is identical to the actions taken to obtain the shapes in the shapes database used in Zernike moments computation of Section 4.5. Figure 6.3 shows a sample US image that contains a fetal skull whose shape is defined with 240 points. Having annotated a training image, normalization actions

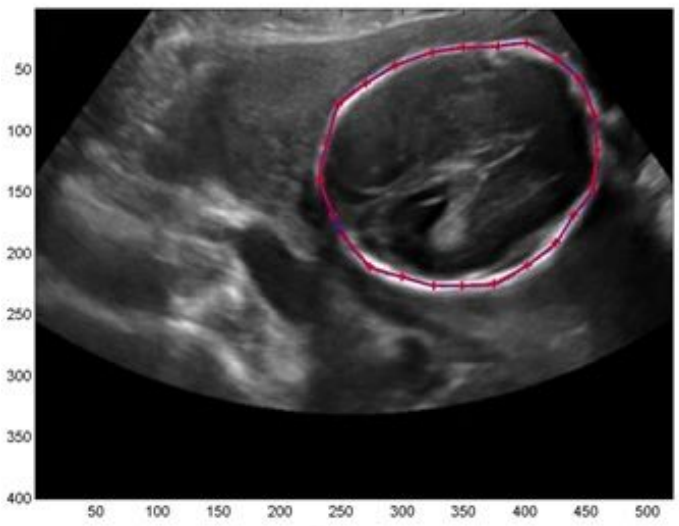

Figure 6.3. Skull contour defined with 240 points (only 24 manually marked).

to display and express the skull in a normalized frame are taken step by step. The following is an ordered list of actions that are carried out to perform the normalization for a single image:

(i) Rotate the annotated image I so that the line connecting the end points of the OFD is parallel to the horizontal axis of the image frame. The result image is I_r.

(ii) Crop I_r to get rid of unnecessary portions of the image. Denoting the length of the line connecting the end points of the OFD in I_r with M (i.e. length of major

axis) and the length of the line connecting the end points of the BPD with m (i.e. length of minor axis), cropping produces an image whose horizontal length is $12\frac{M}{10}$ and vertical length is $12\frac{m}{10}$. The result image I_{rc} contains the entire skull boundary, extends to left and right with a distance of 10% of M for the two directions, similarly extends to up and down with a distance of 10% of m for the two directions. Figure 6.4 shows how cropping is done for a hypothetical contour.

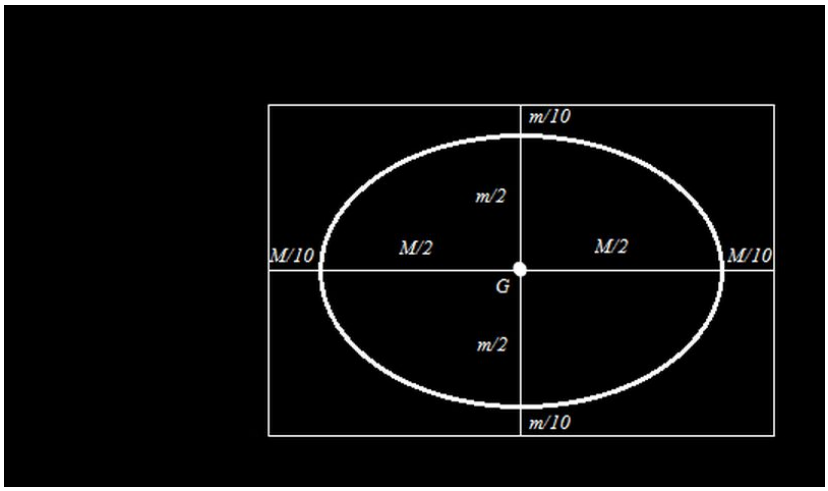

Figure 6.4. Cropping: the entire image I_r, the image in white rectangle I_{rc}.

(iii) An average model is to be computed using images of equal sizes. That is why, resize I_{rc} to an image of standard size. We have selected the horizontal size of such an image to be 228 pixels and the vertical size to be 200 pixels. These numbers approximate the ratio of major axis length M to the minor axis length m of skulls (M/m) well. The result image is I_{rcs}.

(iv) In a real scenario, the rotated version of I_{rcs} (with an angle of 180° either clockwise or counterclockwise) and the mirror images of I_{rcs} and its rotated version may be encountered. These variations are possible since "left" has no bias over "right" or "up" has no bias over "down" in biological skull formation and image acquisition. Obtain all these versions of I_{rcs} to be used in average shape model construction. The results of this step are $I^1_{rcs} = I_{rcs}$ (normalized image itself), I^2_{rcs} (rotation of I^1_{rcs} by 180°), I^3_{rcs} (mirror image of I^1_{rcs}) and I^4_{rcs} (mirror image of I^2_{rcs}). Figure 6.5 shows I^1_{rcs}, I^2_{rcs}, I^3_{rcs} and I^4_{rcs} for the image of Figure 6.3.

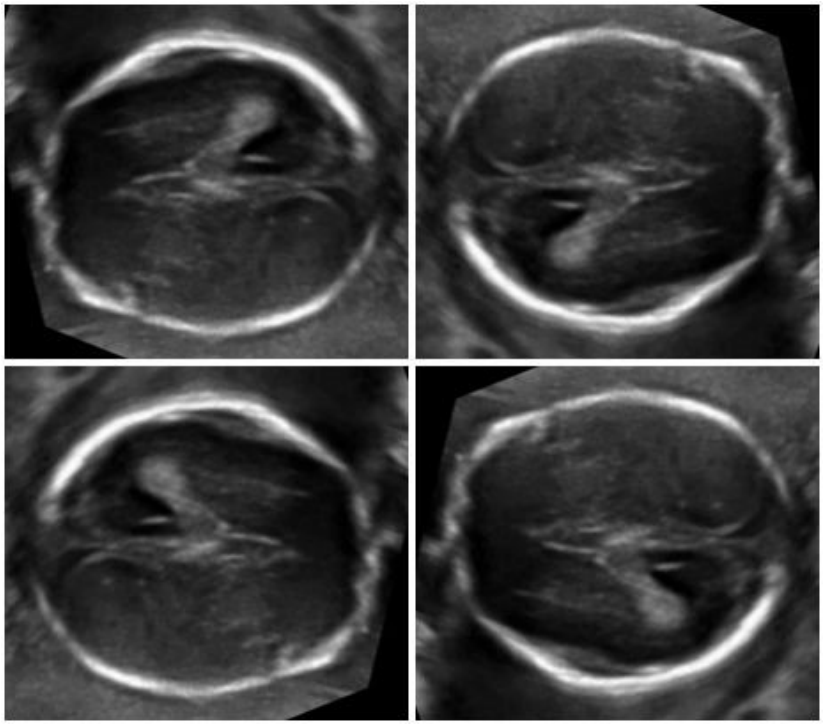

Figure 6.5. I^1_{rcs} (top left), I^2_{rcs} (top right), I^3_{rcs} (bottom left) and I^4_{rcs} (bottom right) of the image in Figure 6.3.

While running the normalization steps for a training image, the x and y coordinates of the annotated points in training images are adjusted so that they reflect the results of normalization actions. The adjustments can be in the form of updating the point coordinates or updating the point indices. In particular, the point with index one of I^1_{rcs} becomes the point with index 121 in I^2_{rcs}. Similarly, the point with index 61 of I^1_{rcs} becomes the point with index 181 in I^2_{rcs}. As another example, the point with index 10 of I^1_{rcs} becomes the point with index 112 in I^3_{rcs}.

Assuming that the normalization procedure has been performed for all N training images, the average shape model I_{avg} can be constructed using the $4N$ outputs of normalization. The construction consists of simply finding the averages of coordinates of 240 model points x_i and y_i over all $4N$ sets of points (x_{ij}, y_{ij}) where $1 \leq i \leq 240$

and $1 \leq j \leq 4N$, as in Equation 6.1:

$$x_i = \sum_{j=1}^{4N} x_{ij}/4N \text{ and } y_i = \sum_{j=1}^{4N} y_{ij}/4N \tag{6.1}$$

After average shape model construction and prior to its use for segmenting a particular image of the same size, the lines passing through the center of gravity of the model image and each model point are considered. For each of the 240 model points, the segmentation result (i.e. the appropriate pixel) is to be searched on the line segment of length 20 that is centered on the model point. The coordinates of points on line segments are taken to be pixel values which are pairs of integers (best approximations for these pairs are selected). The segmentation process described in Section 6.3.2 deals with pixel intensity distributions along line segments. Figure 6.6 displays the average shape model image I_{avg} that has been computed with $N = 358$ annotated training images (4x358 = 1432 images in total) and the line segments that pass through each model point.

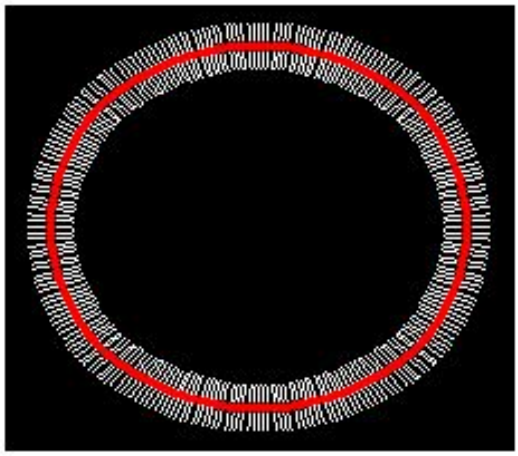

Figure 6.6. Average shape model (240 points) and corresponding line segments with pixel approximations.

6.3.2. Intensity-Based Averaging

Given an input image I to be segmented for a skull contour, the user is supposed to mark four points whose positions are approximately those of the two end points of the OFD and the two end points of the BPD of the fetal skull in I. The order that the user follows when marking these 4 points is of no importance because the preprocessing held prior to the segmentation algorithm takes necessary actions to handle assigning correct roles to the marked points. Figure 6.7 is an example to mark four points in the semi-automatic segmentation scheme. To arrive at a satisfactory segmentation result, the points *must* be marked with a minimal accuracy constraint; otherwise one can not expect successful segmentation. To make I fit to the frame of the average shape model

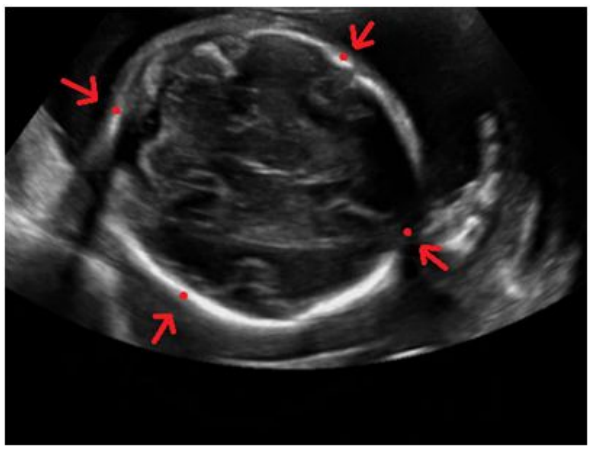

Figure 6.7. Marking four points in semi-automatic segmentation.

I_{avg} with identical sizes (200x228), a normalization procedure similar to that applied to every training image during model construction is carried out. Normalization consists of rotation, cropping and scaling. Determining the angle of rotation necessitates identifying which two of the marked four points stand as the end points of the major axis (OFD) of the coarse ellipse that corresponds to the skull region in I. The procedure for this identification and finding the true rotation angle is presented next:

(i) For all groupings of the marked four points (P_1, P_2, P_3, P_4) that puts two points in one group and the remaining two points in another group (e.g $\{(P_1, P_4)(P_2, P_3)\}$),

estimate the two lines $\mathbf{L}_1$ and $\mathbf{L}_2$ defined by the points (P_a, P_b) and (P_c, P_d) in the two groups and find the intersection point Q of $\mathbf{L}_1$ and $\mathbf{L}_2$. Let d_{aQ}, d_{bQ}, d_{cQ}, d_{dQ}, d_{ab} and d_{cd} be the straight-line distances between points denoted by subscript indices.

- If $\mathrm{d}_{aQ} + \mathrm{d}_{bQ} = \mathrm{d}_{ab}$ and $\mathrm{d}_{cQ} + \mathrm{d}_{dQ} = \mathrm{d}_{cd}$, decide Q to be the true center of gravity and proceed to (ii).
- Else repeat (i) with another grouping.

(ii) If $\mathrm{d}_{ab} > \mathrm{d}_{cd}$, decide $[P_a P_b]$ to be the major axis and $[P_c P_d]$ the minor axis.

(iii) Else decide $[P_c P_d]$ to be the major axis and $[P_a P_b]$ the minor axis.

(iv) Compute rotation angle θ using the orientation of the major axis.

Figure 6.8 shows the lines $\mathbf{L}_1$ and $\mathbf{L}_2$, the intersection point Q (which can also be conceived as the centroid of the skull) and the rotation angle θ. I is rotated using

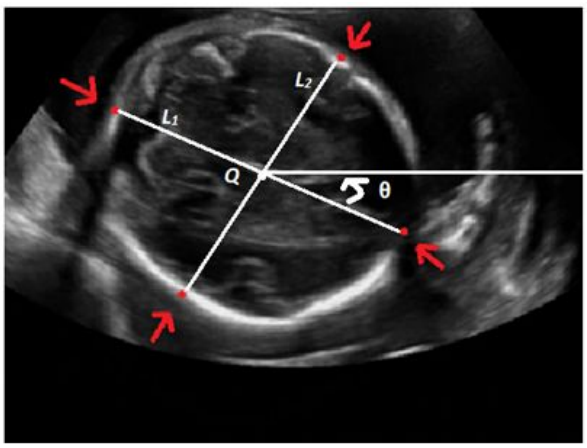

Figure 6.8. Determining the rotation angle θ for the image of Figure 6.7.

the outcome of the described procedure so that the major axis of the skull is aligned with the horizontal axis of the image frame. The rotation may result in the placement of the frontal end of the skull to the left and the occipital end to the right or vice versa. This is also not of any importance because the steps taken in average shape model construction handle both possibilities arising from rotation as well as mirror image cases. After the rotation is performed, cropping and scaling are applied similar to the ones used in the average shape model construction process. The result image

I_{inp} is of size 200x228. Figure 6.9 shows I_{inp} after the normalization process is applied on the image of Figure 6.7. The segmentation algorithm is to identify 240 different

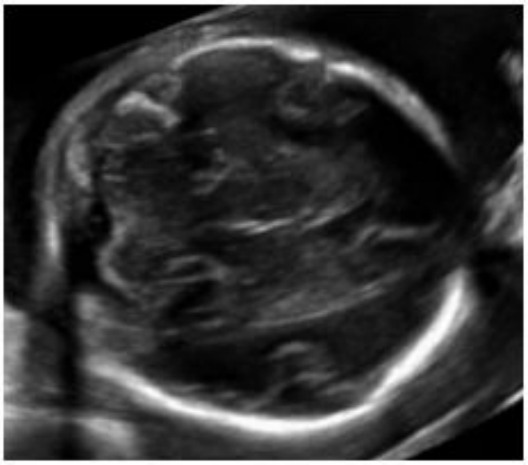

Figure 6.9. I_{inp} (the input to segmentation) for the image of Figure 6.7.

points, each of which lies on a separate line segment determined in the average shape model construction process. It is noted that 240 non-intersecting line segments, each corresponding to one of the 240 average shape model points, exist. Identifying the point on a line segment output by the segmentation heuristic that is used consists of considering the intensity distribution of pixels along the line segment and finding the point location with a weighting performed with respect to pixels intensities at all pixels of the considered line segment. The facts, decisions and assumptions that hold through the segmentation procedure, that are all visually verified by the outputs of the experiments, are presented as a list here:

(i) The pixels corresponding to the middle points of line segments are the actual model points (corresponding pixels).

(ii) The lengths of line segments corresponding to model points is 20. This number does not correspond to the number of pixels along the line segment but to the Euclidian distance between the end points of any line segment. The number of pixels on a line segment can be as many as 21 when the line segment is vertical or horizontal with respect to the image axes.

(iii) The underlying lines of the line segments pass through the centroid of the average

shape model and one of the model points.

(iv) Not all the intensities at the pixels of line segments are considered as decisive factors during segmentation. If any pixel intensity is lower than a threshold value τ, its value is set to zero and practically not used in computations that determine point locations that the segmentation procedure outputs.

(v) The threshold value τ is determined using Otsu's method [113]. Other methods can be applicable. During the computation of τ, the intensity values belonging to the input image I_{inp} at all the pixels on all the 240 line segments are used together. The threshold value τ is fixed for the image (i.e. for all line segments).

Figure 6.10 shows the binary image of the line segments of the average shape model accompanied with the line segment indices associated with frontal end point (of the OFD), right end point (of the BPD), occipital end point (of the OFD) and left end point (of the BPD). The numbering proceeds increasingly in a clockwise orientation. Figure 6.11 and Figure 6.12 show the original and simplified (some values eliminated

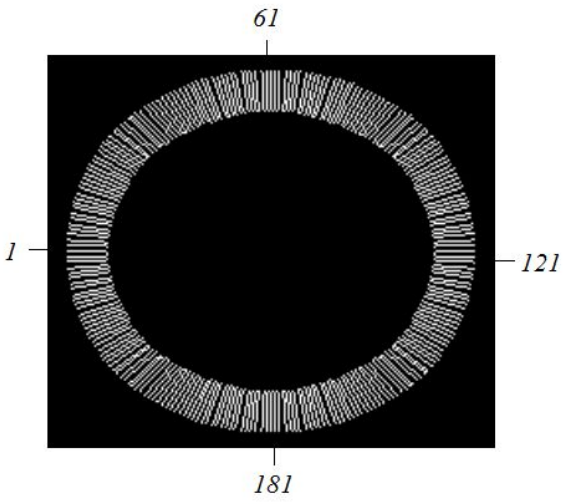

Figure 6.10. Binary image of line segments and the numbering scheme.

after thresholding) intensity distributions for the image of Figure 6.9 at pixel locations of two different line segments. Figure 6.13 shows the intensity distribution of pixels of the first line segment when no pixels can pass the threshold test. This is displayed for the input image at the top of the same figure.

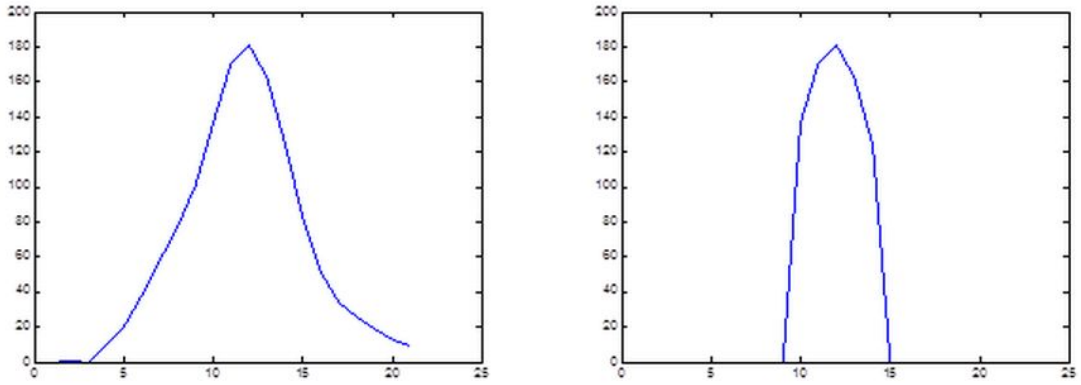

Figure 6.11. Intensity distributions ($\tau = 111$, left: original, right: simplified) of the image of Figure 6.9 for the first line segment.

The heuristic for determining a "correct" segmentation point along a line segment is computing a weighted average of the locations of the pixels of the line segment where the weights are the intensities of those pixels. Once more, we note that the intensities

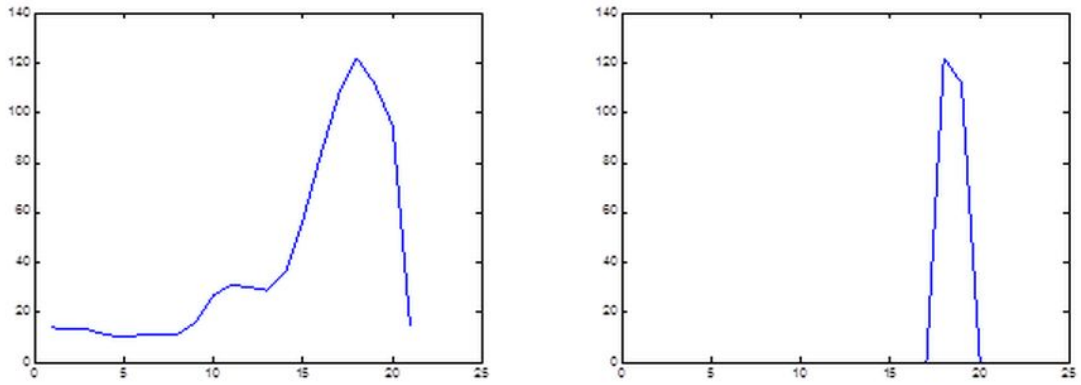

Figure 6.12. Intensity distributions ($\tau = 111$, left: original, right: simplified) of the image of Figure 6.9 for the 50th line segment.

below the threshold τ are treated zero. What refers to the "location" concept here is not the actual pixel coordinates but integers which stand for the index of pixels along the line segment of consideration. For example, if a line segment consists of 21 pixels, then the pixel indices satisfy $1 \leq i \leq 21$. All locations are paired with actual pixel

coordinates specific for each line segment. The implementation can keep track of which location corresponds to which pair of pixel coordinates (x, y).

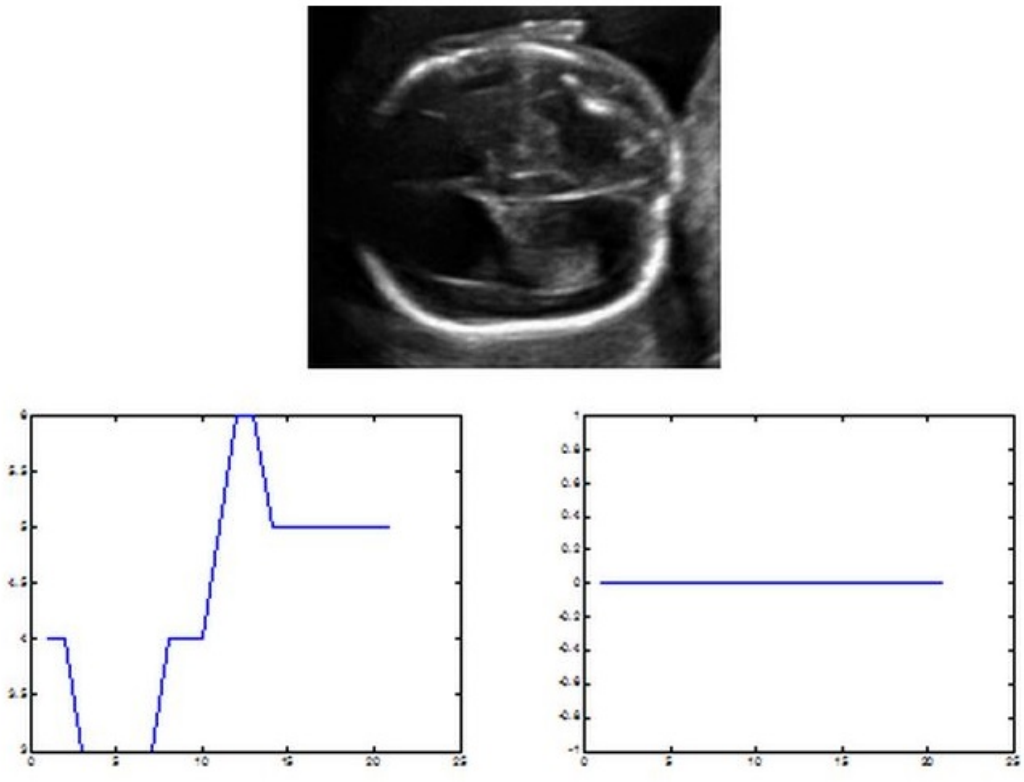

Figure 6.13. Input skull image (top) and intensity distributions ($\tau = 103$, left: original, right: simplified) for the first line segment.

The computation of the location of a "correct" point is given by

$$i_{seg} = \frac{\sum_{i=1}^{L} i.f(i)}{\sum_{i=1}^{L} f(i)} \tag{6.2}$$

where L is the total number of locations, i denotes location indices, i_{seg} is the location of the output point of segmentation and $f(i)$ is the function returning the *threshold-simplified* intensity of the pixel at location i. When i_{seg} does not show up as an integer, it is rounded ensuring a value in the correct range (i.e. $1 \leq i \leq L$). Furthermore, when all $f(i)$ are zero as in the example of Figure 6.13, the midpoint of the corresponding line segment (i.e. the corresponding point of the average shape model) is selected as the i_{seg} location. Finally, the pixel corresponding to the i_{seg} location is marked as

the segmentation point. The procedure is repeated for 240 points (line segments) to retrieve the 240 segmentation points.

6.3.2.1. Smoothing. Our heuristic seems to be working singularly for each individual point but there is one more action to be performed before connecting all adjacent point pairs with straight line segments and arriving at the final segmentation result of the entire image (i.e. the skull boundary). The extracted set of points does not contribute to a boundary that is smooth enough to be considered appropriate both visually and for machine applications. A *smoothing* process that adjusts the pixel coordinates of the 240 points is applied and the boundary is constructed with the smoothed set of points. For smoothing purposes, we have chosen to run a moving average window with size 11 on the x and y coordinates of all points and round the outcomes. Figure 6.14 shows the segmentation result for the image of Figure 6.9 before and after smoothing. Figure 6.15 shows the segmentation result where the extracted region is superimposed on the original input image of Figure 6.9.

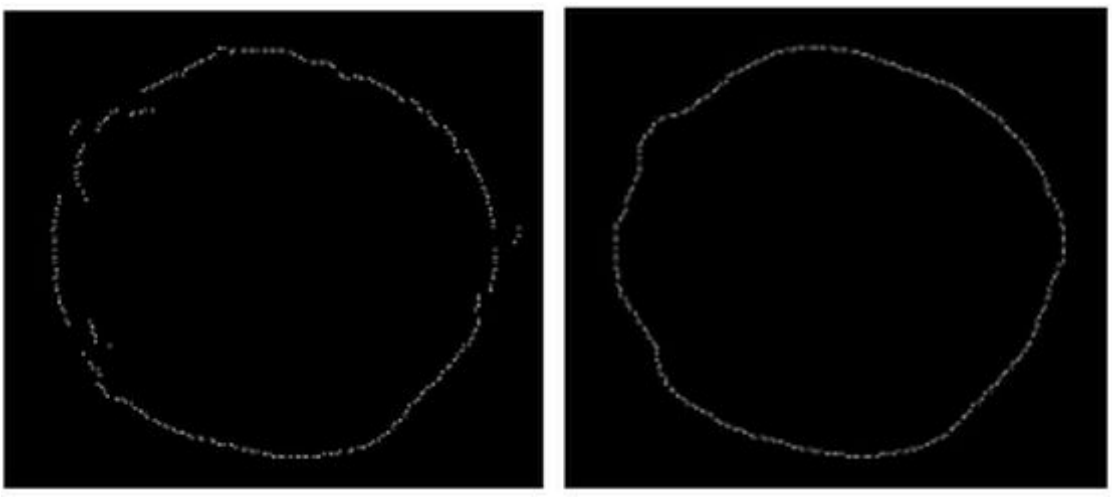

Figure 6.14. Segmentation results (left: before smoothing, right: after smoothing).

For reference and visually validating the segmentation procedure with intensity-based averaging, the inputs and outputs of four skull images from our collection are displayed in Figure 6.16.

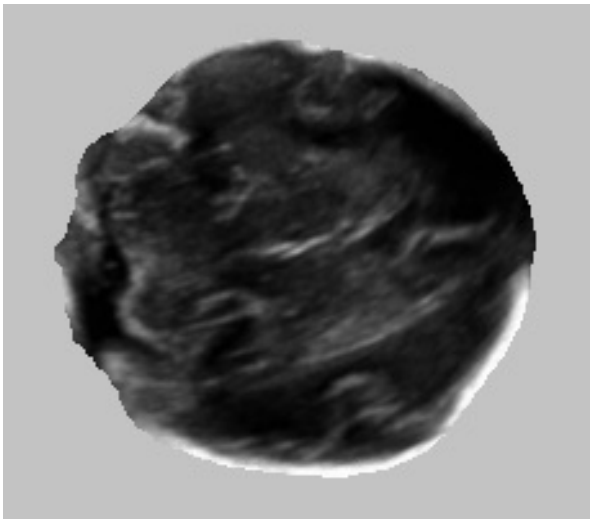

Figure 6.15. Segmented image of Figure 6.9.

6.3.3. Active Shape Models

Model-based techniques are useful and work well to locate rigid objects in images. When the appearance of objects of interest can vary, as in the case of skull boundaries, it is more problematic to apply model-based approaches. A number of deformable template models [119–122, 153] have been proposed to handle this variability. *Active Shape Models* (*ASM*) [121] is a technique that learns shape variability through the use of shapes in a training set, where all shapes are annotated with a number of points. For a reliable deformable model to be built, the annotations must be correct and the correspondence of all points in all images must be known. Active shape models do not sacrifice model specificity in order to accommodate variability and let the models deform only in ways characteristic of the objects they represent. The built models can be used to search images to locate objects of interest using proper initialization and iterative refinement. In our work, we use the ASM implementation of Kroon [154] to locate fetal skulls in US images as an alternative segmentation tool. Essential theory on ASM and the details specific to our application follow in the remainder of this chapter.

6.3.3.1. Statistical Shape Models. The shape of an object is a property that remains invariant under similarity transformations of translation, rotation and scaling; and is

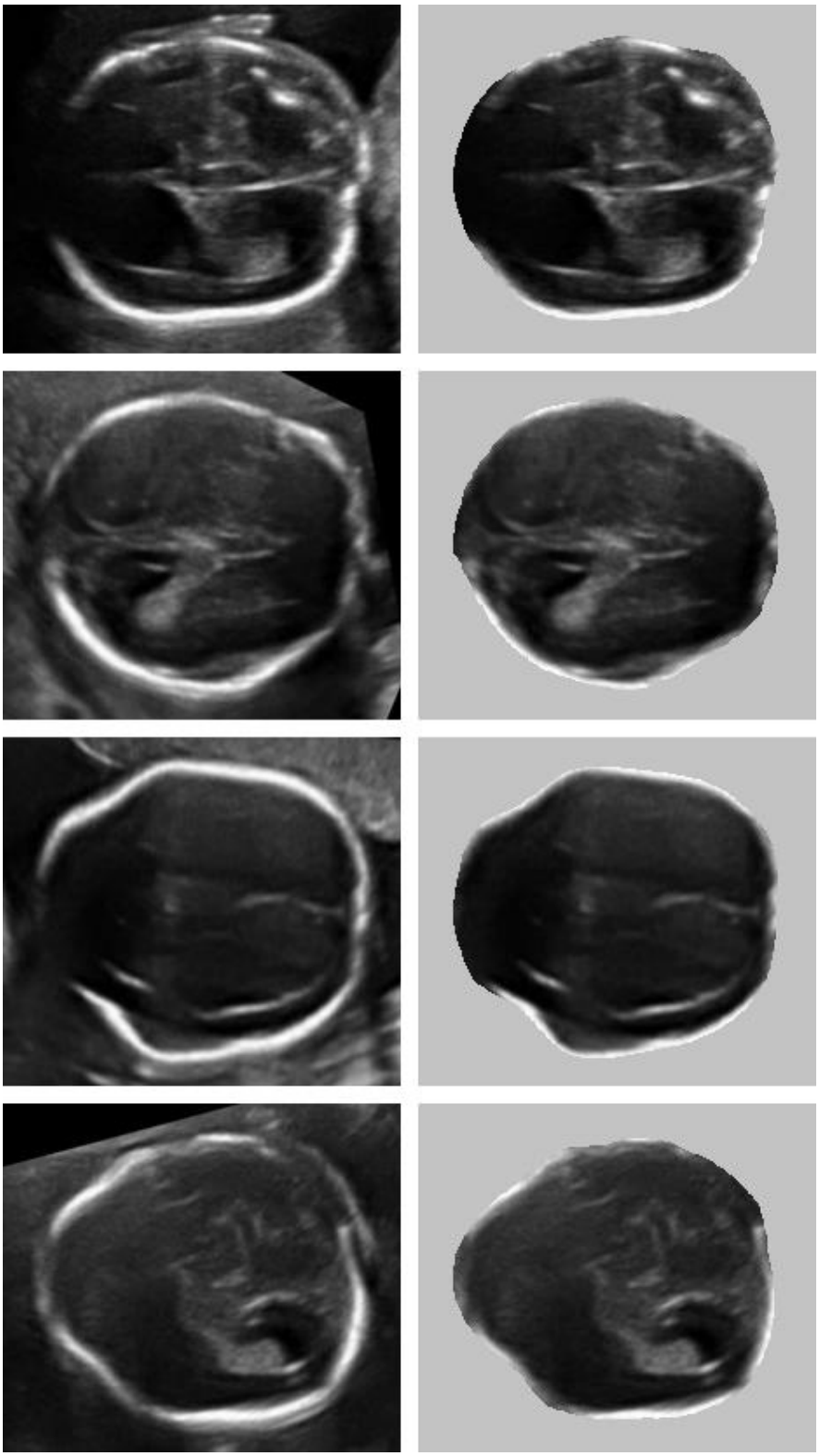

Figure 6.16. Segmentation with intensity-based averaging (left: inputs, right: outputs).

a signature of the object. It is described by the location of n points in any dimension (e.g. planar shapes are in 2D). Generalizing shapes such that the points describing them do not only include spatial coordinates but other dimensions such as time is also possible. Assuming that n points are used to annotate each shape in a training set of N shapes, statistical models of shapes can be derived both to analyze new shapes and to synthesize others similar to those in the training set. In a 2D image, the n points $\{(x_i, y_i)\}$ can be represented as a $2n$-element vector $\mathbf{x} = (x_1, \ldots, x_n, y_1, \ldots, y_n)^T$. For a training set with N shapes, N such vectors $\mathbf{x}_i$ can be generated.

Before performing statistical analysis on shapes, the training set must be aligned to a common coordinate frame such that the definitive (i.e. center of gravity) and annotated (i.e. those along the border) points of shapes correspond. In our application, we use the 4x358 = 1432 normalized (cropped, rotated and scaled) skull images for statistical analysis. The normalization procedure of Section 6.3.1 presents all images and their corresponding annotated points in a coordinate frame free of the variations attributable to similarity transformations. It should be noted here that the presented images are also used in the average shape model construction stage of segmentation with intensity-based averaging. Hence, it is easy to see the identicalness of the coordinate frame used in ASM segmentation to that of segmentation via intensity-based averaging. Although the aforementioned normalization is valuable in that variations of considerable magnitude are eliminated, it is rather coarse with respect to alignment required in ASM analysis. For the required fine alignment that precedes statistical shape analysis (shape model building), there exist methods among which the *Procrustes Analysis* [155] is the most popular. In Procrustes analysis, shapes are aligned so as to minimize the sum of distances of all shapes to the mean shape ($\sum |\mathbf{x}_i - \bar{\mathbf{x}}|^2$). Although analytic solutions for aligning a set exists, a simple iterative approach to follow is presented in the algorithm of Figure 6.17. The training set with N aligned examples forms a distribution in the nd-dimensional space. The aim is to obtain a parameterized model of the form $\mathbf{x} = \mathbf{M}(\mathbf{b})$, where $\mathbf{b}$ is a set of parameters describing this distribution. Such a model can be used to generate new vectors $\mathbf{x}$. If the distribution $\mathrm{pr}(\mathbf{b})$ can be modeled, one could limit $\mathbf{b}$ such that the generated $\mathbf{x}$ are similar to the examples in the training set. Estimating $\mathrm{pr}(\mathbf{x})$ from the model would then also

1: Translate all examples so that their centers of gravity are at the origin
2: Pick one example as the initial estimate of the mean shape and scale it ($|\bar{\mathbf{x}} = 1|$)
3: Record the estimate as $\bar{\mathbf{x}}_0$ to define the default reference frame
4: Align all shapes with the current estimate of the mean
5: Re-estimate mean from aligned shapes
6: Align mean shape with $\bar{\mathbf{x}}_0$ and scale it so that $|\bar{\mathbf{x}} = 1|$
7: If no convergence, go to 4 ▷ *Convergence is achieved if the current estimate of mean does not change significantly*

Figure 6.17. Procrustes analysis for aligning shapes.

be possible.

Modeling shape variation starts with applying *Principle Component Analysis* (*PCA*) to the cloud of data $\mathbf{x}_i$ in the nd-dimensional space to reduce the dimensionality and simplify the problem. This computes the main axes of the cloud and lets any shape be approximated with fewer than nd parameters. The mean of the data is

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{6.3}$$

and the covariance matrix is

$$\mathbf{S} = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T. \tag{6.4}$$

Denoting the eigenvectors of $\mathbf{S}$ with Φ_i and the corresponding eigenvalues with λ_i such that λ_i are sorted ($\lambda_i \geq \lambda_{i+1}$); $\mathbf{\Phi}$, shown in Equation 6.5 contains the t eigenvectors with the largest eigenvalues:

$$\mathbf{\Phi} = \{\Phi_1|\Phi_2|\ldots|\Phi_t\} \tag{6.5}$$

Any training set example $\mathbf{x}$ can be approximated by

$$\mathbf{x} \approx \bar{\mathbf{x}} + \mathbf{\Phi b} \tag{6.6}$$

where $\mathbf{b}$ is a t-dimensional vector of model parameters given by

$$\mathbf{b} = \mathbf{\Phi}^T(\mathbf{x} - \bar{\mathbf{x}}). \tag{6.7}$$

The vector $\mathbf{b}$ contains the parameters of a deformable template model. A shape $\mathbf{x}$ can be varied by varying the elements of $\mathbf{b}$ as suggested by Equation 6.6. λ_i corresponds to the variance of the i^{th} parameter b_i. Applying the limits $\mp 3\lambda_i$ to b_i guarantees the generation of shapes similar to the ones in the training set. t can be chosen in several ways. In the simplest case, it could be required that a particular fraction of the total variance is explained by t parameters. The selection of t can be based on the variation explained by t eigenvectors. The constraint to determine t is given by

$$\sum_{i=1}^{t} \lambda_i \geq \tilde{f}\sigma_T^2 \tag{6.8}$$

where $\tilde{f} \in [0, 1]$ is the fraction desired to be explained (e.g. 98%) and σ_T^2 is the total variance explained with all eigenvectors $\mathbf{\Phi}$.

6.3.3.2. Interpreting Images with Active Shape Models. Image interpretation involves finding the set of parameters that best match the model to the image. In ASM, the set of parameters defines the shape and position of the target object in the image. For a set of model parameters $\mathbf{b}$ that is characteristic of the shape and another set of pose parameters $\mathbf{p}$ that define how the object is translated, oriented and scaled; an instance of the model projected onto the image can be generated and the target object can be compared to this instance by a fit function $\mathbf{F}(\mathbf{b}, \mathbf{p})$. The best set of shape and pose parameters to interpret the target object in the image is that optimizing the fitness measure. For example, if $\mathbf{F}(\mathbf{b}, \mathbf{p})$ is an error measure, it is aimed to be minimized. Similarly, if it stands for the conditional probability $\Pr(\mathbf{b}, \mathbf{p}|I)$ that $(\mathbf{b}, \mathbf{p})$ is realized

when the interpreted image is I, the objective is to maximize the measure. Theoretically, a suitable fit function must be selected and a general-purpose optimizer used for optimization. Assuming that the shape model represents boundaries and strong edges, the fit function may measure the distance between a given model point and the nearest strong edge in the image. Figure 6.18 illustrates this scenario where the fit measure is defined as

$$\mathbf{F}(\mathbf{b}, \mathbf{p}) = \text{dist}(P, P_{se}). \tag{6.9}$$

The fit measure relies on the target points being the correct points. If some points are incorrect, the quality of fit is not indicated correctly.

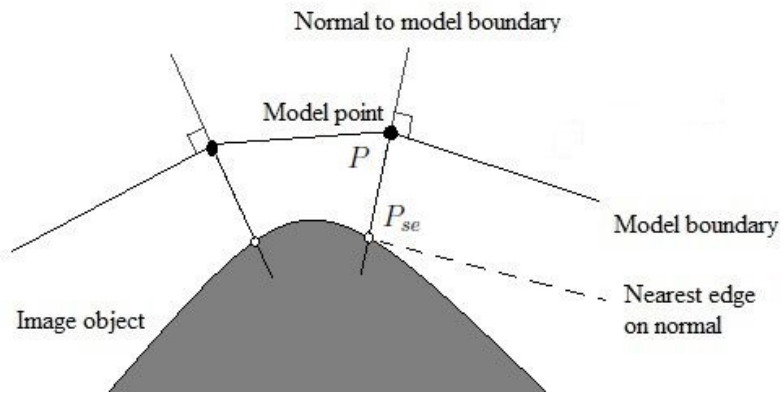

Figure 6.18. Example of fit function $\mathbf{F}(\mathbf{b}, \mathbf{p})$: distance between model point P and nearest strong edge P_{se}.

Looking for nearby strong edges is not the only choice for iteratively optimizing the fit. Local structure around model points may be learned from training images and the interpretation process may search for the best match around the current model point learned from training examples. Alternatively, an image can be sampled around the current model point and the match quality can be determined by how well the samples match the models derived from the training set. Interpretation is carried out for every model point, one by one, to arrive at a full interpretation of an image. It should be noted that, given poor initializations of model instances, the general

optimization problem is very hard. Obtaining good interpretations (i.e. segmentations) depends on how well model instances are initialized prior to running local optimization strategies. Starting from wrong positions leads to either divergence or convergence on wrong structures of locally similar shapes to those in the training set. With a good enough initialization Z_0 of an ASM projected on an image, an iterative approach to improve the fit Z_i works by examining the region around each model point to find best matches, updating Z_i according to the newly found model points and correspondingly updating the shape ($\mathbf{b}$) and pose ($\mathbf{p}$) parameters until convergence.

As stated before, the assumption that strong nearby edges are the true locations for which model points should converge in the search for correctly locating target objects does not always hold. This suggests learning what the neighborhood of model points is like from training examples through building models. It is customary to sample along profiles normal to the model boundary at each model point as shown in Figure 6.18. For a given model point in the i^{th} training image, sampling along the profile at k pixels either side of the point is performed and $2k+1$ samples are placed in a vector $\mathbf{g}_i$. The effects of global intensity changes can be eliminated by sampling the derivatives along the profile instead of the intensities. A normalization by dividing the sum of elements in $\mathbf{g}_i$ is then performed as in Equation 6.10:

$$\mathbf{g}_i \leftarrow \frac{1}{\sum_j |g_{ij}|}\mathbf{g}_i \tag{6.10}$$

The computations are repeated for all training images and the set of normalized samples $\{\mathbf{g}_i\}$ is obtained. Assuming $\{\mathbf{g}_i\}$ are distributed as multivariate Gaussian, the mean $\bar{\mathbf{g}}$ and variance $\mathbf{S}_g$ are computed giving a statistical model of gray-level profile about the point. This is repeated for every model point giving one gray-level model for each point. The quality of fit for a new sample $\mathbf{g}_{new}$ is measured by the Mahalanobis distance of the sample from the model mean given by

$$\mathbf{F}(\mathbf{g}_{new}) = (\mathbf{g}_{new} - \bar{\mathbf{g}})^T \mathbf{S}_g^{-1} (\mathbf{g}_{new} - \bar{\mathbf{g}}) \tag{6.11}$$

which is linearly related to the probability that $\mathbf{g}_{new}$ is drawn from the distribution. Minimizing $\mathbf{F}(\mathbf{g}_{new})$ is equivalent to maximizing this probability. During the search around a model point, sampling is performed for $2m + 1$ pixels (m pixels on either side) along the profile. Assuring that $m > k$, the quality of fit to the corresponding gray-level model is tested at $2(m-k)$ pixels along the sample and the location with the lowest $\mathbf{F}(\mathbf{g}_{new})$ value is selected as the new model point. The aforementioned procedure is repeated for all model points and suggested new locations are found. The shape and pose parameters are then updated to best match the model to the new points.

6.3.3.3. Multi-Resolution Active Shape Models. Statistical model building for gray-levels along normal profiles of points and the ASM search can be implemented in a multi-resolution framework, hence improving efficiency and robustness. For the multi-resolution implementation, a Gaussian pyramid representation of each image is required. The base of the pyramid is the original image and the successive levels are obtained by repeated smoothing and sub-sampling of images. Each subsequent level of the Gaussian pyramid is formed by smoothing and sub-sampling the image of the previous level to get an image with half the size in both spatial dimensions. Statistical models of gray-level values along normal profiles at each point are built for all levels of the pyramid. The same number of pixels, regardless of level, are generally used in profile models. This accounts to more of the image being represented at coarse levels. The search starting at coarse levels and refining the found points in finer resolutions leads to a faster search algorithm. Since the number of pixels used in search is the same for any level, coarse levels allow quite large movements and the model is expected to converge to a good solution as the levels move towards finer resolutions. Determining when the search process should continue at a finer resolution depends on the number of times the best found pixel is within the central 50% of the profile. For example, if for some pyramid level, at least 90% of the best found pixels of all current model points are within $ns/2$ pixels (ns being the length of normal profiles) of the corresponding model point, the algorithm decides the search to have converged at that pyramid level and moves to the next finer resolution.

6.3.3.4. Skull Segmentation with Active Shape Models. The ASM of skull images in our database is learned using the normalized images (i.e. the training set) obtained by the procedure of Section 6.3.1. The training set contains 4x358 = 1432 skull images with one skull in each. The skulls in the images are annotated with 240 landmarks. Kroon's implementation [154] aligns the vertices in the training set by centering the landmarks to remove translation effects, computing the angles defined by the vectors from each landmark to the mean of the landmarks (i.e. the center) and then rotating every landmark using the mean of the computed angles as the rotation angle, the same angle for all points. It is commented in the code that using Procrustes analysis for alignment is also possible, however the simpler alignment methodology is preferred. The aligned data of vertex coordinates are used to build the statistical shape model described in Section 6.3.3.1. The leading eigenvectors which explain 98% of the total variance are retained to achieve compactness and remove contour noise.

In constructing models for gray-level profiles of landmarks and in ASM search, a multi-resolution framework with two scales is used. The landmark profiles normal to the model boundary are acquired at 49 pixels ($k = 24$ pixels either side of the landmark and the same for both scales). Building gray-level profiles is performed in either of two ways: If the original search method using Mahalanobis distances to profile means $\bar{\mathbf{g}}$ is intended, the edge gradient information (derivatives) is used to compute means and covariances of landmark profiles over the training set. An alternative search method utilizes intensities of profile pixels on which PCA is applied. Similar to shape model construction, the eigenvectors explaining 98% of the intensity variation in gray-level profiles are kept. During search, the aim is to minimize the distance to the mean of the considered intensity profiles. We have observed that the second search method (using intensities of profile pixels) provides visually better segmentation with our data set.

The inputs to the ASM search are identical to those used in segmentation with intensity-based averaging. The user marks the two endpoints of the OFD and those of the BPD in the raw image of a transcerebellar skull. The normalization of Section 6.3.2 is performed and the outcome is fed as input to ASM search. The ASM search is preceded with the segmentation procedure of intensity-based averaging to get the

"non-smooth" contour points of the normalized skull image. We initiate the ASM search with the non-smooth model boundary instance obtained through intensity-based averaging and run the search until iteration limits are exhausted. At each scale, the search runs for 40 iterations. The search length to detect the optimal contour point position is one pixel in both normal directions. The methodology can be considered as smoothing applied to the contour points output by the segmentation procedure utilizing intensity-based averaging. The shape constraints, in effect, account for smoothing the initial contour discovered by intensity-based averaging. Figure 6.19 shows examples of ASM segmentation, where the input images are the same as those in Figure 6.16. It is evident that utilizing the intensities of profile pixels with PCA provides considerably better results than using the edge gradient information and measuring the quality with Mahalanobis distance to the mean of profile samples.

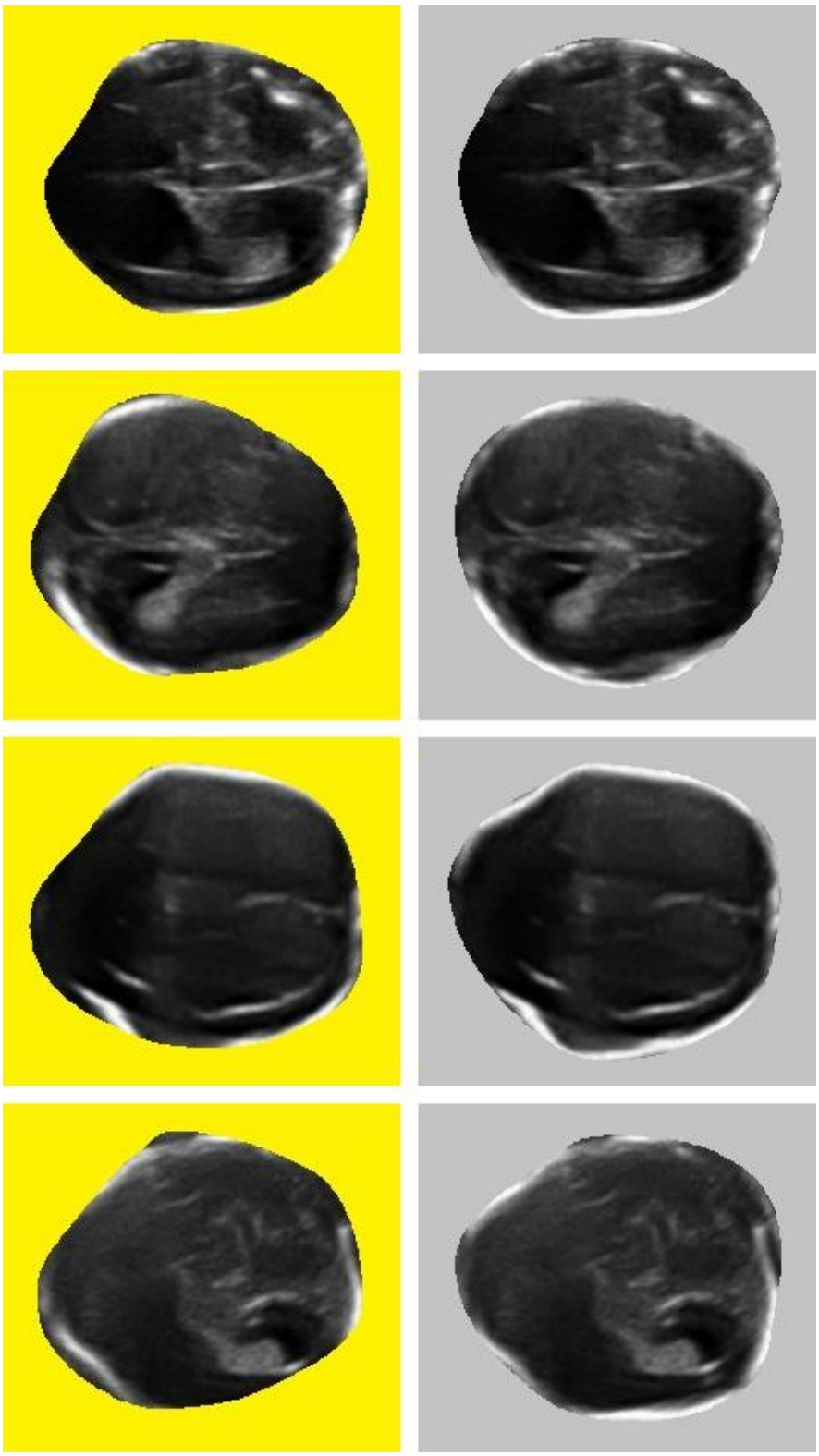

Figure 6.19. Segmentation with active shape models for the inputs of Figure 6.16 (left: edge gradient modeling, right: intensity modeling).

7. RARITY AND PERFORMANCE EVALUATION

The robustness of classifiers depends on the availability of learning data that evenly span the input space with members of every involved class of the problem. Unfortunately, available training data for some problems do not meet this requirement and hence designed classifiers can not be trusted to assign correct labels to new instances that were not used for training. The spina bifida detection is such a problem, where the obtainable learning data contain many healthy samples compared to few defective samples causing imbalance in data distribution and perhaps not covering some portions of the input space at all. Rare class mining requires tackling the problems arising with absolute and relative rarity. As far as we consider, modifying distributions of training data and properly assessing classification performance are the two branches that designers must cope with when attacking problems associated with rarity. In this chapter; we briefly elaborate on common performance evaluation metrics, the basics of ROC analysis [125] employed to evaluate classifier performance at a range of operating points and the SMOTE oversampling strategy [130] that has been used to resample data sets for SVM learning. Our experiments using kNN classifiers with CSS features utilize resampling of training instances using exact copies of the rare class instances with higher membership counts. Further details of actions taken to handle rarity and assess performance specific for the spina bifida detection problem are available in Chapter 8.

7.1. Performance Metrics

Once a two-class problem is assumed (i.e. positive and negative classes), accessing classification performance is achieved by means of performance indicators, the most general being the *accuracy* metric. Accuracy is defined as the ratio of the number of correct classifications to the number of all samples. Adopting the notation of the contingency table of Table 2.1, accuracy is defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{7.1}$$

The number of positive instances in a data set (either training or test) is $Np = TP + FN$, that of negative instances is $Nn = TN + FP$ and the number of all instances is $N = TP + FP + TN + FN$. Being a general metric, accuracy is not suitable for skewed (unbalanced) data distributions. In a rare-class problem where 99% of the samples are those of the prevalent class and the rest contains rare class samples; even if all samples are predicted to be of the majority class, the accuracy metric figure is 0.99 that signals very successful classification performance realized on the data set. However, it is clear that this is not what is intended and proper evaluation metrics that value the prediction of the rare class are required. *Recall* (also known as sensitivity) and *precision* are the two basic metrics which can be used on their own or in the derivations of other metrics to incorporate both of them in performance indicators. Since it is desired to evaluate classifier performance on the rare class rather than the majority class, recall and precision are usually defined with respect to the rare class as in Equation 7.2 and Equation 7.3:

$$Recall = \frac{TP}{Np} = \frac{TP}{TP + FN} \tag{7.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{7.3}$$

Recall is the fraction of the positive instances that are correctly classified as positive and precision is the fraction of correct positive decisions. Specificity refers to the fraction of negative samples that are correctly identified (i.e. similar to the concept of recall but accounting for negatives). Recall does not provide an insight into how many negative samples are misclassified and precision does not indicate how many positive instances are incorrectly classified as negative. Using recall and precision simultaneously can effectively evaluate classification performance in imbalanced learning problems.

F-measure weighs the relative importances of recall and precision. The generic

form of F-measure is

$$Fmeasure = \frac{1}{\lambda \frac{1}{Recall} + (1-\lambda)\frac{1}{Precision}}, 0 \leq \lambda \leq 1 \tag{7.4}$$

where λ can be selected based on how much weight one wants to assign to recall and precision. Increasing λ assigns more weight to recall (i.e. for reducing false negatives) and decreasing λ puts more weight on precision (i.e. for reducing false positives). When recall and precision get equal weight, $\lambda = 0.5$ and Equation 7.4 can be rewritten as

$$Fmeasure = \frac{2.Recall.Precision}{Recall + Precision}. \tag{7.5}$$

Geometric mean of Recall and Precision ($GMRP$) is another common performance metric defined as

$$GMRP = \sqrt{Recall.Precision}. \tag{7.6}$$

7.2. ROC Analysis

Section 7.1 describes the particular *point metrics* of recall, precision, F-measure and GMRP, which are all defined for classifiers operating at a single point. In contrast, performance evaluation of classifiers can be performed at a variety of operating points, each determined by how much true positive predictions is achieved at the expense of false positive predictions. ROC (receiver operating characteristics) analysis is a procedure for performance evaluation along a curve where each curve point corresponds to a classifier operating with the indicated TP and FP rates (tp and fp). In the ROC space shown by 2D graphs, tp is plotted on the Y axis and fp is plotted on the X axis. Figure 7.1 shows a ROC graph on which five classifiers (A, B, C, D and E) are marked. A perfect classifier is one that can predict all positive instances correctly without making any incorrect positive predictions. Classifier A is an ideal classifier with $tp = 1$ and $fp = 0$ (i.e. the ROC point (0, 1)). When comparing classifiers and

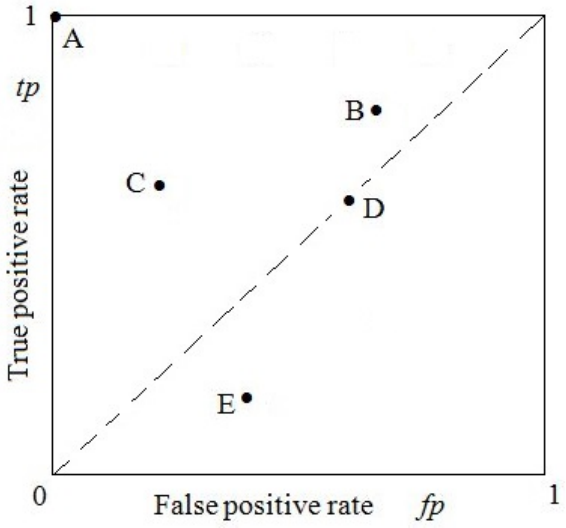

Figure 7.1. ROC graph of five classifiers A, B, C, D and E.

selecting better ones, it is desired to have larger tp as well as lower fp. This leads to the fact that a classifier is superior to another one if the corresponding ROC point of the former is located to the north-west of that of the latter. Referring to Figure 7.1, we can conclude that A is the best classifier among all and the performance of C is better than that of D. The point (0, 0) indicates that a classifier operating at that point makes no true positive decisions as well as causing no false positive errors. (0, 0) is in fact where a classifier does not make any decisions at all (i.e. a non-existent classifier). The other extreme point (1, 1) is where the classifier issues positive decisions to all samples, thus predicts all positive samples correctly and incorrectly decides all negatives to be positive. Making positive decisions easily (as in (1, 1)) and hardly (i.e. only with strong evidence) leads to discriminating classifiers as either "liberal" or "conservative" [125] respectively. A liberal classifier makes more true positive predictions accompanied with more false positive errors. Oppositely, conservative classifiers do not often predict instances as positive, hence false positive errors are few as well as true positive decisions. Classifier C of Figure 7.1 is more conservative than B (equivalently B is more liberal than C). The classifiers located on the diagonal line ($y = x$) are those that work by random guessing. A classifier that predicts instances as positive half of the time and as negative the other half is on (0.5, 0.5). Similarly, a classifier that makes positive

predictions 20% of the time is located on (0.2, 0.2). D is a classifier that operates by random guessing. If a classifier, such as E of Figure 7.1, is located below the diagonal (in the lower right triangle) of the ROC graph, it is not a realistic classifier since it performs worse than random guessing and simply reverting the decisions would result in classification above the diagonal line.

Most classifiers are designed to yield a single point in the ROC space. On the other hand, some classifiers produce a score of some sort such as an instance probability and base their decisions on this score. When the limits on this score are tuned to modify the classification rule, different points on the ROC graph are obtained each corresponding to a classifier operating with different values of the defined score, hence at different operating points. Connecting the points on the ROC graph showing different operating conditions of a single classifier produces a connected curve in the ROC space and such curves are indicative of classification performance along tp and fp. Obtaining a justifiable ROC curve depends on sampling the score at appropriate values, running the classifier for all sampled values (i.e. for all operating points) and properly connecting the points. Using linear interpolation, the ROC points (i.e. (fp, tp) pairs) are successively connected starting from the point with the smallest fp value towards that with the largest fp value. Covering the entire X and Y axes requires connecting the leftmost operating point to (0, 0) and the rightmost to (1, 1). Figure 7.2 shows a ROC curve of a classifier operating at four different points. Using the ROC curve of classifiers, the *whole-curve metric* of AUC (area under the ROC curve) [126] can be defined as the area between the curve and the X axis (fp). AUC evaluates the "overall" performance of a classifier along the whole curve. Greater AUC values indicate better classification performance and no realistic classifier has AUC < 0.5.

7.3. Synthetic Minority Oversampling Technique

Modifying class distributions in imbalanced sets of training samples is an essential action to be carried before the data is used to train classifiers. Without such an action, the trained classifier can not be supposed to perform well on minority class instances. Undersampling and oversampling are the options that designers of classifiers can take.

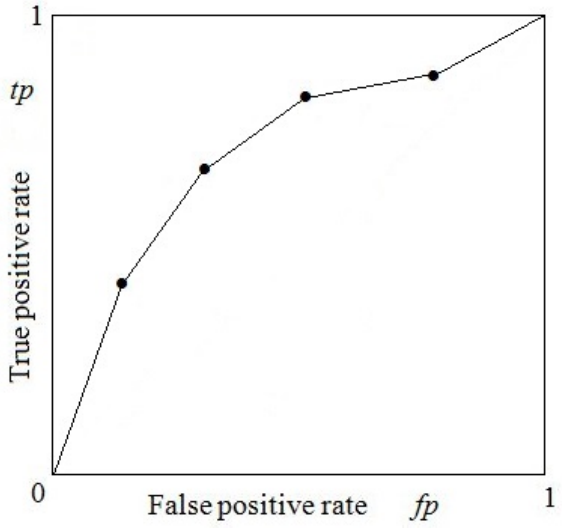

Figure 7.2. ROC curve of a classifier operating at four (fp, tp) points.

The trade-offs between the two selections are generally determined by empirical studies. When oversampling is considered, the most straightforward method is to resample data with replacement, that is to replicate exact copies of rare class samples in the training set. As an alternative, Chawla *et al.* [130] propose an oversampling approach that creates "synthetic" examples using instances of the minority class. The technique is called *Synthetic Minority Oversampling TEchnique* (SMOTE). Synthetic examples are generated in feature space rather than raw input space. As applicable to our database of skull images; instead of using two different skull images of defective fetuses to generate a new skull image, the Zernike features obtained for the two skulls are used to synthesize the Zernike features of a new skull.

Oversampling by SMOTE is performed by considering each minority class sample and introducing synthetic samples along the line segments connecting the sample to any of the k nearest minority class samples. The amount of oversampling determines how many of the k nearest neighbors are used in generating new samples. For instance, if the amount of oversampling required is 200%, then two of the k-nearest neighbors (assuming $k \geq 2$) of each rare class sample are used to create new samples. A synthetic sample along the line between two samples (the sample in consideration and its selected

nearest neighbor) is generated by picking a random number in $[0, 1]$, computing the difference between the two feature vectors, multiplying this difference vector by the random number and finally adding this vector to the feature vector of the considered sample. If the amount of sampling required implies that not all but some of the k nearest neighbors of each rare class sample are used to create new samples, the selection of which of the k nearest neighbors to use is also performed randomly. The overall expectation of SMOTE is to make the decision region of the rare class more general. Figure 7.3 presents the pseudo-code of SMOTE oversampling.

7.3.1. Borderline-SMOTE

Different from generating synthetic rare class samples considering all rare class samples, Han *et al.* [156] generate synthetic samples only for those rare class samples that are more apt to be misclassified. These samples lie on the borderline of minority and majority classes, hence are called *borderline* samples. The intuition is that the samples far from the borderline may contribute less to classification and SMOTE is used to oversample only borderline rare class samples to attain better classification performance. Whether a rare class sample is on the borderline is decided by counting its m nearest neighbors from both the rare and majority class samples. Given a specific rare class instance, let m' denote the number of majority class instances among its m nearest neighbors. If $m' = m$, all nearest neighbors are of the majority class, the sample is considered to be noise and not operated on any further. If $0 \leq m' < m/2$, the instance is *safe* and not a borderline sample. Only when $m/2 \leq m' < m$, the instance is in danger and can easily be misclassified. Denoting the set of borderline rare class samples with B and the set of all rare class samples with J ($B \subseteq$ J), the k nearest neighbors of each sample in B from J (excluding the sample itself) are found. Depending on the sampling rate, a subset of these k nearest neighbors and the considered rare class sample are used to synthesize a new sample by the algorithm of Figure 7.3. Repeating the oversampling procedure for all samples in B and as dictated by the desired oversampling rate synthesizes new rare class samples that are included in the training set. In our implementation, we use $k = 5$ and $m = 3$.

Algorithm $SMOTE(N_r, O, k)$

Input: Number of rare class samples N_r, Amount of oversampling $O\%$, Number of nearest neighbors k

Output: round$(O/100 * N_r)$ synthetic rare class samples as the array SYNTH[ ][ ]

1: Randomize the N_r rare class samples in the array ORIG[ ][ ] ▷ *Issue a random ordering and restore the samples in* ORIG[ ][ ]
2: $t_{all} \leftarrow \lfloor O/100 \rfloor$ ▷ *All N_r samples to be SMOTEd t_{all} times*
3: $N_{extra} \leftarrow$ round$((O/100 - t_{all}) * N_r)$ ▷ *N_{extra} samples SMOTEd once more*
4: $extra\_flag \leftarrow false$ ▷ *All samples to be SMOTEd t_{all} times*
5: **if** $N_{extra} > 0$ **then**
6: $extra\_flag \leftarrow true$ ▷ *Some samples to be SMOTEd $t_{all} + 1$ times*
7: SYNTH[ ][ ] $\leftarrow$ array of synthetic rare class samples, initially $\emptyset$
8: $no\_considered\_samples \leftarrow N_r$ ▷ *All samples are considered at least once*
9: **if** $t_{all} = 0$ **then**
10: $no\_considered\_samples \leftarrow N_{extra}$ ▷ *N_{extra} samples considered only once*
11: **for** $i \leftarrow 1$, $no\_considered\_samples$ **do**
12: $Attr_1 \leftarrow$ ORIG$[i][\,]$ ▷ *Features of the i^{th} sample*
13: Compute k nearest neighbors of $Attr_1$, randomize and store in $NN\_array[\,]$
14: $no\_synth \leftarrow t_{all}$ ▷ *The number of synthetic samples for* ORIG$[i][\,]$
15: **if** $extra\_flag = true$ AND $i \leq N_{extra}$ **then**
16: $no\_synth \leftarrow t_{all} + 1$
17: **for** $j \leftarrow 1$, no_synth **do**
18: $Attr_2 \leftarrow$ ORIG$[NN_array[j]][\,]$ ▷ *Features of the j^{th} nearest neighbor*
19: $diff \leftarrow Attr_2 - Attr_1$ ▷ *Difference*
20: $rand \leftarrow$ random number in $[0, 1]$
21: $Attr_s \leftarrow Attr_1 + rand * diff$ ▷ *Generated synthetic sample*
22: Append $Attr_s$ to SYNTH[ ][ ]
23: **return** SYNTH[ ][ ]

Figure 7.3. SMOTE oversampling.

8. EXPERIMENTS

Our data set of fetal skull images consists of 358 samples. 29 of these are labeled as *defective* and the remaining 329 are labeled as *non-defective* by medical experts. The data set is highly unbalanced and robust classification performance can be obtained using appropriate methods to handle this data set.

We run and collect results on three classifiers. The first two classifiers employ CSS images of skull contours and use them to measure the similarity between any two instances via either CSS matching of Abbasi *et al.* [74] or DMLM matching procedure of Kpalma *et al.* [138]. The distance (or similarity) scores are processed with a 20NN classifier. The results are reported over a range of operating conditions through ROC and precision-recall (PR) curves. The superiority of a classifier with a particular setting over another classifier with a different setting is made visible with the whole-curve metric of AUC [126]. Moreover, the point metrics of accuracy, recall, precision, specificity, F-measure and GMRP are provided for some operating points. Our third classifier is an SVM-based classifier utilizing magnitudes of Zernike moments of skull shapes. Like the CSS-based classifiers, the performance of the SVM classifiers trained with magnitudes of Zernike moments is reported using ROC, PR curves, the aforementioned point metrics and AUC. The SVM classification tool we employ is the *LIBSVM* library by Chang and Lin [157]. There are two types of kernels that we use to build SVM classifiers with scaled training data and for which we report results and elaborate. These are linear and RBF kernels. All other parameter values used by LIBSVM in SVM training and prediction are left as default.

Handling the class imbalance problem is achieved by a group of techniques. For the 20NN classifiers exploiting CSS images, the membership counts of defective samples are considered larger than those of healthy samples so as to reflect class ratios correctly. This can be treated as a simple oversampling strategy where the single occurrences of rare class samples are updated as multiple occurrences (i.e. oversampling with replacement). Alternatively, the same set of classifiers are run with random

undersampling (RU) to equalize the number of defective and healthy samples in training sets. Rare class mining for the SVM classifier is performed using combinations of borderline-SMOTE oversampling and random undersampling. Different parameter selections for oversampling and undersampling rates yield different operating conditions and hence different points on ROC curves. Our presentation of results based on SVM and Zernike moments is an analysis of how oversampling and undersampling rates affect classification performance for the spina bifida detection problem.

The skull images used in classification (either to train classifiers or assign a label to them with the trained classifier) are not usable in their raw states. For training, those skull contours and shapes extracted after manual segmentation (as accurately as possible) are used since ground truth instances are required for robust training. Evaluation of classification performance are then performed using ground truth instances and the outcomes of segmentation methods of both intensity-based averaging and active shape models. We concentrate on examples obtained with ground truth (i.e. manual) segmentations to evaluate the performance of our classifiers. The performances of the classifiers when the input data is obtained after applying the proposed semi-automatic segmentation methods to raw skull images are presented in Appendix A.

8.1. Operating Conditions and Configurations for CSS-Based Classifiers

Operating conditions corresponding to points on ROC curves are associated with particular values of a decision threshold $\tau \in [0, 1]$. Given the output score Ω_i of the 20NN classifier for any instance i, the decision strategy assigns the label *defective* if $\Omega_i \geq \tau$ and the label *non-defective* otherwise. The Ω_i value is the ratio of the sum of defective sample memberships to the sum of all sample memberships in the 20 nearest neighbors of instance i. Each instance in a group is normally to be accounted for a single membership (i.e. of value one), but for purposes of handling rarity in unbalanced data sets, rare-class instances may be assigned a membership value greater than one

to favor the presence of those instances. Ω_i is computed as

$$\Omega_i = \frac{m_d.N_d}{m_d.N_d + m_h.N_h} \tag{8.1}$$

where N_d is the number of defective examples and N_h is the number of healthy samples in the 20-neighborhood of instance i ($N_h + N_d = 20$), m_d and m_h are membership contributions of defective and healthy samples. Both of them can be taken one for balanced datasets where the number of defective and healthy samples in training sets are equal, however for data sets that contain considerably different numbers of defective and healthy samples, these coefficients can be adjusted to reflect class ratios.

A number of configurations were used to report the results of the experiments with CSS-based classifiers. The list of the elements of these configurations is as follows:

(i) *Feature space:* Representation of contours with CSS images
- using CSS images of only actual (real) contours
- using CSS images of both actual contours and their reflections

(ii) *Distance computation:* by means of CSS matching of pairs of contours
- employing u (position) and σ (height) of arcs – (i.e. C_1 of Equation 3.21)
- employing u, σ and w (width) of arcs – (i.e. C_2 of Equation 3.22)

(iii) *Similarity computation:*
- transforming the computed distance (cost) measure of CSS matching to a similarity measure (Equation 8.2)
- computing similarities of feature vectors in DMLM matching (Equation 3.26)

(iv) *Classification:* kNN algorithm with $k = 20$ and decision thresholds τ in [0.05, 1] with spacings of 0.05 (i.e. $\tau \in \{0.05, 0.1, 0.15, 0.2, \ldots, 0.8, 0.85, 0.9, 0.95, 1\}$).

(v) *Data set handling:*
- oversampling for unbalanced training sets by considering the membership contributions of the rare class examples larger than those of the majority class examples.
- undersampling the entire data set to equalize the number of defective and non-defective samples and working with balanced sets.

When only actual skull contours are used with CSS matching of Abbasi *et al.* [74], the distance between pairs is the cost C of CSS matching. In the case of using actual contours and their reflections at the same time, the matching costs C_{actual} (of two actual contours) and $C_{reflected}$ (of two reflected contours) of images are both considered. CSS matching costs can have values in the range $[0, \infty)$. A matching cost of ∞ signals that the matched CSS images can not have the same label, however, it may often be the case that the pair of actual contours has a finite matching cost C_{actual} and the pair of reflected contours has ∞ matching cost $C_{reflected}$ or vice verse. To be able to handle these cases as well as those when both costs are finite and incorporate both distances into the computation of a single matching cost C, a weighted average of similarities S_{actual} and $S_{reflected}$ for actual and reflected contours are first computed to obtain a single similarity S and S is later turned into a distance measure C. The relation between distance (cost) C and similarity S is defined in Equation 8.2:

$$S = 1/C \text{ or } C = 1/S \tag{8.2}$$

Given C_{actual} and $C_{reflected}$ as the CSS matching costs for a pair of actual contours and for their corresponding reflections, w_{actual} and $w_{reflected}$ as the weights of those costs, the computation of the single cost C_{all} is given in Equation 8.3, Equation 8.4 and Equation 8.5:

$$S_{actual} = \frac{1}{C_{actual}} \text{ and } S_{reflected} = \frac{1}{C_{reflected}} \tag{8.3}$$

$$S_{all} = w_{actual}.S_{actual} + w_{reflected}.S_{reflected} \tag{8.4}$$

$$C_{all} = \frac{1}{S_{all}} \tag{8.5}$$

The DMLM matching procedure of Kpalma *et al.* [138] does not cause a problem like the one mentioned for the CSS matching of Abbasi *et al.* [74] because similarities,

instead of costs, of actual and reflected contours are directly computed and they may possess only finite values in $[0, 100]$. As a result, Equation 8.4 can be used in the first place to compute the combined similarity score. In our experiments, when both contours are used, w_{actual} of Equation 8.4 is set to 0.7 and $w_{reflected}$ is set to 0.3.

8.2. Data Sets and Experimental Setup in CSS-Based Classification

The results of the experiments are obtained using 100 folds (different selections of training and test sets) for any data set. The unbalanced data set of 358 samples is divided into two sets, one training set and one test set having 298 and 60 samples, respectively. In the training set, 24 samples are defective and 274 are non-defective. In the test set, five defective and 55 non-defective samples exist. Distances (or similarities) of samples to be classified and all training samples are computed. During classification, when the sample to be classified is in the training set, its distance to itself is not taken into account in order to avoid bias. Since the training set is unbalanced, the membership contribution m_d for defective samples is taken as $274/24 = 11,416$ and m_h for non-defective samples is taken one. This simulates oversampling of the rare class. To perform experiments with balanced data sets, data sets consisting of all 29 defective samples and a subset of the non-defectives containing 29 samples are picked to form a balanced data set. This process is repeated 100 times to repeat the experiments with 100 different balanced data sets. Each data set is partitioned to a training set of size 48 (24 defectives and 24 non-defectives) and a test set of size 10 (five defectives and five non-defectives). The rule of 100 folds applies for each balanced data set. m_d and m_h are both one. Taking a small subset of non-defective samples in the formation of data sets is, in effect, undersampling.

All reported results are those averaged for all different data sets and all folds of each data set. There is one data set which is unbalanced and 100 different balanced data sets obtained by undersampling (discarding the majority of non-defective samples). In CSS-based classification, the following experimental settings are considered:

(i) Unbalanced data sets using only real (actual) contours, CSS matching, costs

computed with Equation 3.21 (first distance metric - C_1)

(ii) Unbalanced data sets using both real contours and their reflections, CSS matching, costs computed with Equation 3.21 (C_1)

(iii) Unbalanced data sets using only real contours, CSS matching, costs computed with Equation 3.22 (second distance metric - C_2)

(iv) Unbalanced data sets using both real contours and their reflections, CSS matching, costs computed with Equation 3.22 (C_2)

(v) Unbalanced data sets using only real contours, DMLM peak points matching

(vi) Unbalanced data sets using both real contours and their reflections, DMLM matching

(vii) Balanced data sets using only real contours, CSS matching, costs computed with Equation 3.21 (C_1)

(viii) Balanced data sets using both real contours and their reflections, CSS matching, costs computed with Equation 3.21 (C_1)

(ix) Balanced data sets using only real contours, CSS matching, costs computed with Equation 3.22 (C_2)

(x) Balanced data sets using both real contours and their reflections, CSS matching, costs computed with Equation 3.22 (C_2)

(xi) Balanced data sets using only real contours, DMLM matching

(xii) Balanced data sets using both real contours and their reflections, DMLM matching

8.3. CSS-Based Classification with Nearest Neighbors

ROC and PR curves of test sets obtained with the 20NN classifiers, the AUC values of both test and corresponding training sets, point metrics of accuracy, recall, precision, specificity, F-measure and GMRP associated with test sets at selected decision thresholds are all presented in Section 8.3.1 and Section 8.3.2. Training of all classifiers are performed with ground truth segmentations of fetal skull images. The test sets of this chapter are also the outcomes of ground truth segmentations.

8.3.1. Unbalanced Data and Ground Truth Segmentations

The CSS-based 20NN classifiers, defined for the possible experimental settings stated in Section 8.2 and running on unbalanced data sets, provides the ROC curves of Figure 8.1 and the PR curves of Figure 8.2 for test sets over 100 folds. Table 8.1 presents the F-measure and GMRP values of the same classifiers for test sets at selected operating points (the best among all considered decision thresholds τ). Precision and recall are assigned equal weights (i.e. $\lambda = 0.5$) in the computation of F-measure. Table 8.2 shows accuracy, recall, precision and specificity at the operating points for which the best F-measure and GMRP values in Table 8.1 are observed.

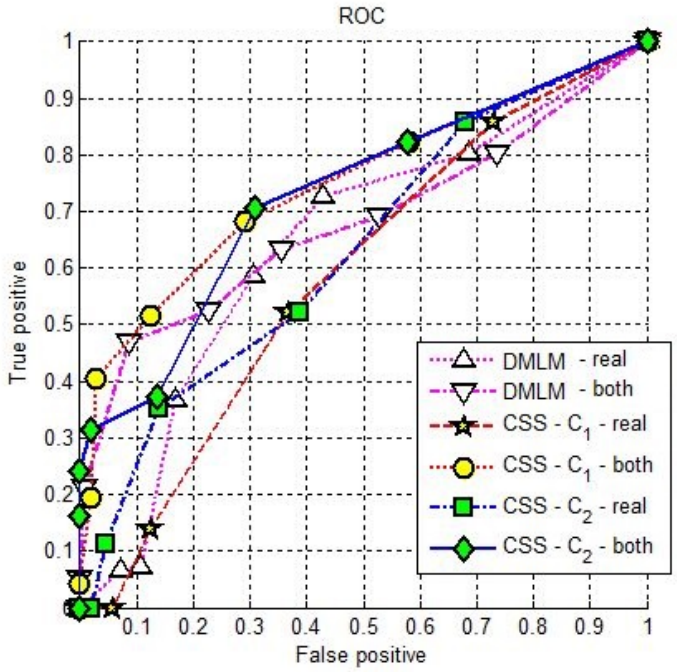

Figure 8.1. ROC curve of CSS-based classifiers with *unbalanced* test sets.

8.3.2. Balanced Data and Ground Truth Segmentations

The ROC, PR curves, F-measure and GMRP values; corresponding accuracy, recall, precision and specificity; all associated with balanced test sets are shown in Figure 8.3, Figure 8.4, Table 8.3 and Table 8.4, respectively.

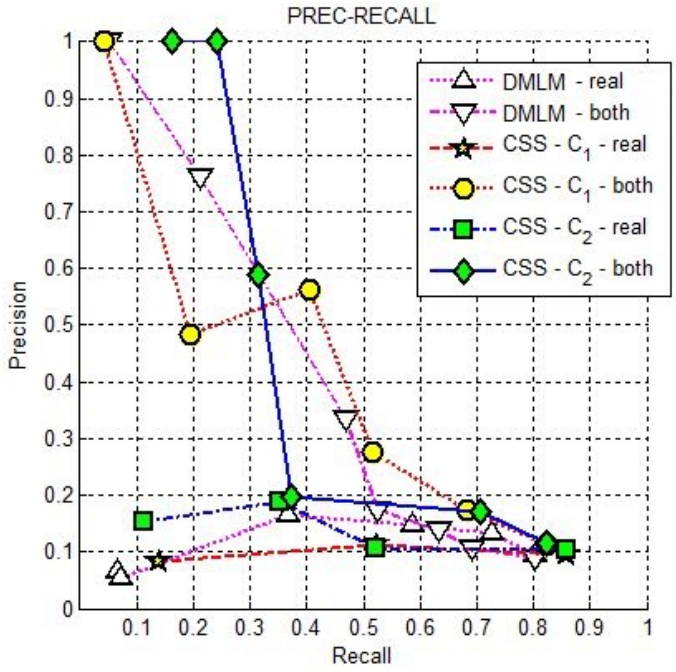

Figure 8.2. PR curve of CSS-based classifiers with *unbalanced* test sets.

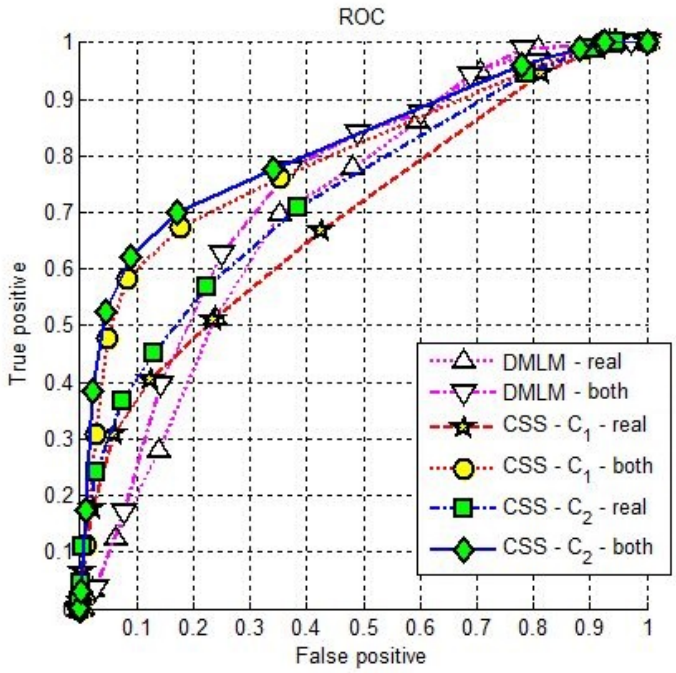

Figure 8.3. ROC curve of CSS-based classifiers with *balanced* test sets.

Table 8.1. F-measure, GMRP of CSS-based classifiers with *unbalanced* test sets.

UNBALANCED test	Threshold (τ)	F-measure	GMRP
CSS, C_1, real	0.35	0.1731	0.2874
CSS, C_1, both	0.70	**0.4706**	**0.4771**
CSS, C_2, real	0.60	0.2446	0.2569
CSS, C_2, both	0.70	0.4097	0.4301
DMLM, real	0.65	0.2360	0.2943
DMLM, both	0.75	0.3919	0.3974

Table 8.2. Metrics of CSS-based classifiers with *unbalanced* test sets.

UNBALANCED test	Threshold (τ)	Accuracy	Recall	Precision	Specificity
CSS, C_1, real	0.35	0.3207	**0.8580**	0.0963	0.2718
CSS, C_1, both	0.70	0.9250	0.4040	0.5635	0.9724
CSS, C_2, real	0.60	0.8188	0.3520	0.1874	0.8613
CSS, C_2, both	0.70	0.**9262**	0.3140	**0.5892**	**0.9818**
DMLM, real	0.65	0.6845	0.5860	0.1478	0.6935
DMLM, both	0.75	0.8773	0.4700	0.3360	0.9144

8.3.3. Evaluation of CSS-Based Classifiers

Table 8.5 shows the area under the ROC curve (AUC) scores in CSS-based classification for all the considered settings on the used training and test sets. The results obtained from the experiments using different settings and types of data sets reveal advantages of some choices over others. These advantages are most visible when the AUC outcomes in Table 8.5, with the best figures along each column highlighted, are examined. The following is a list of findings:

- Employing both actual contours and their reflections to obtain CSS features almost always provides better classification performance than using merely the actual contours.

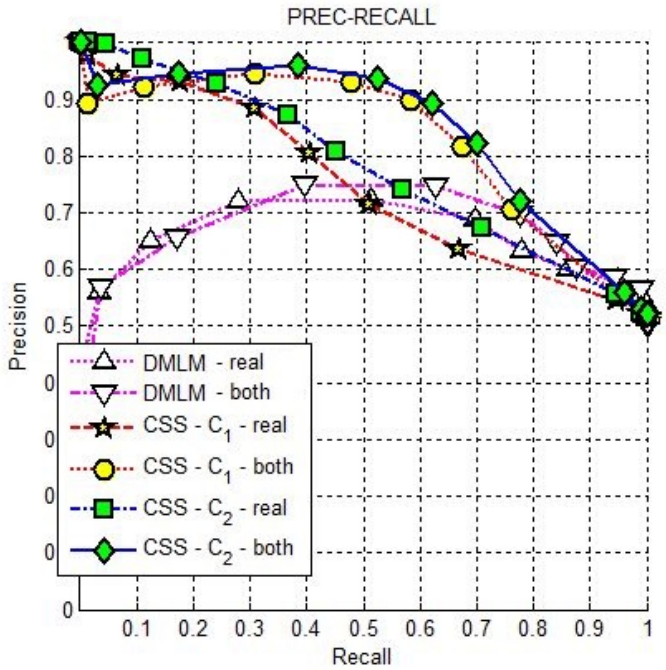

Figure 8.4. PR curve of CSS-based classifiers with *balanced* test sets.

- Evaluating how well contours match in CSS matching using the second distance metric is generally advantageous to that using the first distance metric.
- The performance of DMLM matching is comparable to that of CSS matching. It performs best for balanced training sets when both actual contours and their reflections are used.

We can experimentally conclude that enhanced feature sets and more proper distance measures improve classification performance. Encountering better results for test sets in some cases when the usual expectation is to observe more success with training sets may be judged due to several factors such as the overall unbalanced nature of the experimental data, the differences of sample populations in training and test data, etc. In fact, we have performed training using more samples to achieve better learning and formed test sets with fewer samples. Besides, when one takes into account the facts related to the rarity of one of two classes, partly-unexpected results can be overlooked. Observing worse figures for balanced data sets than those for unbalanced data sets is understandable since undersampling with the objective of getting balanced

Table 8.3. F-measure, GMRP of CSS-based classifiers with *balanced* test sets.

BALANCED test	Threshold (τ)	F-measure	GMRP
CSS, C_1, real	0.30	0.6931	0.7194
CSS, C_1, both	0.45	0.7401	0.7435
CSS, C_2, real	0.30	0.7009	0.7259
CSS, C_2, both	0.45	**0.7564**	**0.7590**
DMLM, real	0.45	0.7203	0.7421
DMLM, both	0.60	0.7354	0.7365

Table 8.4. Metrics of CSS-based classifiers with *balanced* test sets.

BALANCED test	Threshold (τ)	Accuracy	Recall	Precision	Specificity
CSS, C_1, real	0.30	0.5665	0.9468	0.5467	0.1863
CSS, C_1, both	0.45	0.7491	0.6753	0.8185	0.8229
CSS, C_2, real	0.30	0.5789	0.9472	0.5562	0.2106
CSS, C_2, both	0.45	**0.7633**	0.6991	**0.8240**	**0.8276**
DMLM, real	0.45	0.6206	**0.9487**	0.5805	0.2925
DMLM, both	0.60	0.7034	0.7768	0.6983	0.6300

data sets results in discarding valuable information content and forcing to perform experiments with even more unpopulated sets of samples. The F-measure and GMRP values in Table 8.1 and Table 8.3 do not contradict the conclusions drawn by observing the AUC values, however it should be noted that the magnitude orders are different for unbalanced and balanced data sets due to the ratio of sample populations of defectives and non-defectives.

8.4. Data Sets and Experimental Setup in SVM Classification

The data used for SVM training consist of sampled versions of the available data in the feature space of magnitudes of Zernike moments of skull shapes computed considering invariance under similarity transforms. Although training is performed

Table 8.5. AUC for all settings of CSS-based classification.

AUC	unbalanced training	unbalanced test	balanced training	balanced test
CSS, C_1, real	0,6626	0,5866	0,5748	0,6937
CSS, C_1, both	0,7472	**0,7504**	0,6450	0,7970
CSS, C_2, real	0,6839	0,6315	0,6236	0,7337
CSS, C_2, both	**0,7538**	0,7297	0,6743	**0,8177**
DMLM, real	0,6730	0,6442	0,6435	0,7066
DMLM, both	0,6451	0,6802	**0,6837**	0,7427

with sampled data, the results are reported using non-sampled (i.e. original) data sets.

Sampling in SVM classification refers to different combinations of SMOTE oversampling of the defectives and random undersampling of the non-defectives. The five nearest minority instances of each borderline minority instance are considered while generating new (synthetic) instances with borderline-SMOTE. $N\%$ oversampling of the minority class corresponds to $N/100$ new samples generated from the borderline defectives in the training set. If the number of borderline defectives in the training set is M, each of those M defectives and $\lfloor N/100 \rfloor$ nearest neighbors among its five nearest defective neighbors are used to generate new samples along the connecting lines. When $N/100 \notin Z^+$, some of the borderline instances may be sampled once more to achieve the desired rate $N\%$. Our assumption is that at most all five nearest neighbors of all borderline minority samples are used to generate synthetic samples, thus the maximum allowed oversampling rate is 500%. An $N\%$ random undersampling (RU) of the majority class requires randomly discarding some common class samples so that the number of minority samples is $N\%$ of that of the majority samples at the end of sampling. Larger N corresponds to discarding more majority samples and thus to more undersampling. In the following presentation, 0% oversampling refers to no oversampling performed and 0% undersampling means that none of the majority samples are discarded (conceptually different from what has been described for $N\%$ undersampling). Table 8.6 shows some scenarios related to the number of minority and

majority instances in training sets before and after various combinations of SMOTE oversampling followed by random undersampling. All training sets are composed of

Table 8.6. Examples of numbers of samples for various sampling scenarios.

Before sampling			**SMOTE rate**	**RU rate**	**After sampling**	
#minority	#majority	#borderline			#minority	#majority
24	274	irrelevant	0%	0%	24	274
24	274	7	100%	100%	31	31
24	274	12	500%	50%	84	168
24	274	8	300%	125%	48	38
24	274	9	200%	100%	42	42

274 non-defective samples and 24 defectives. The test sets contain 55 non-defectives and five defectives. 100 folds are used to perform experiments and the average results are reported. The rule applied in forming training and test data is identical to that used in CSS-based classification (unbalanced configuration). Training for any data set formed from normalized Zernike moments magnitudes of shapes attained from manual segmentation (i.e. ground truth) of skulls is carried with the sampled data obtained after applying borderline-SMOTE and RU, each with particular rates. SMOTE oversampling is performed at either of 0%, 100%, 200%, 300%, 400% or 500% followed by RU at either of 0%, 50%, 60%, 75%, 100%, 125%, 150%, 175%, 200%, 300%, 400%, 500%, 600%, 700%, 800%, 1000% or 2000%. The operating points of classifiers are defined by a pair of values for SMOTE rate and RU rate (102 in total).

8.5. SVM Classification

The performance of SVM classifiers for ground truth data is presented in Section 8.5.1 with the aid of ROC and PR curves, AUC and point metric realizations of accuracy, recall, precision, specificity, F-measure and GMRP. Comparative treatments, observations and findings are given in Section 8.5.2.

8.5.1. Ground Truth Segmentations

Figures 8.5 and Figure 8.6 illustrate the ROC and PR curves of linear-SVM and RBF-SVM classifiers on test sets when the SMOTE oversampling rate is 500%. Points on the curves correspond to different RU rates mentioned in Section 8.4.

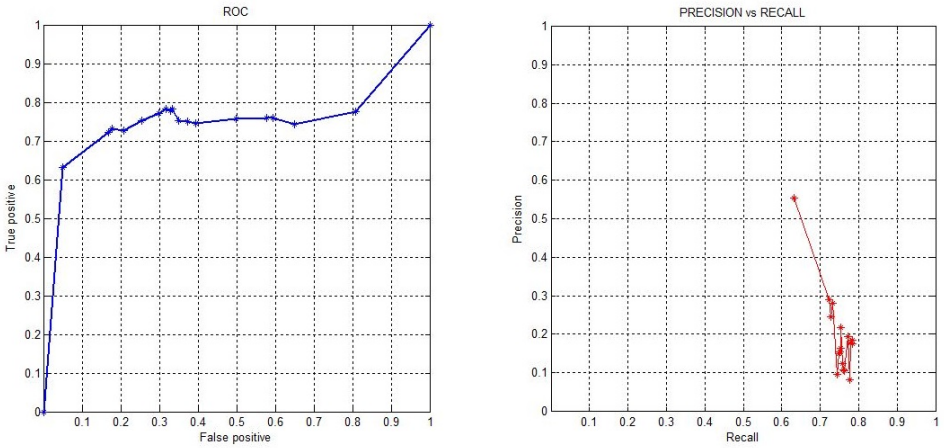

Figure 8.5. ROC and PR curves of test sets (linear-SVM, 500% SMOTE).

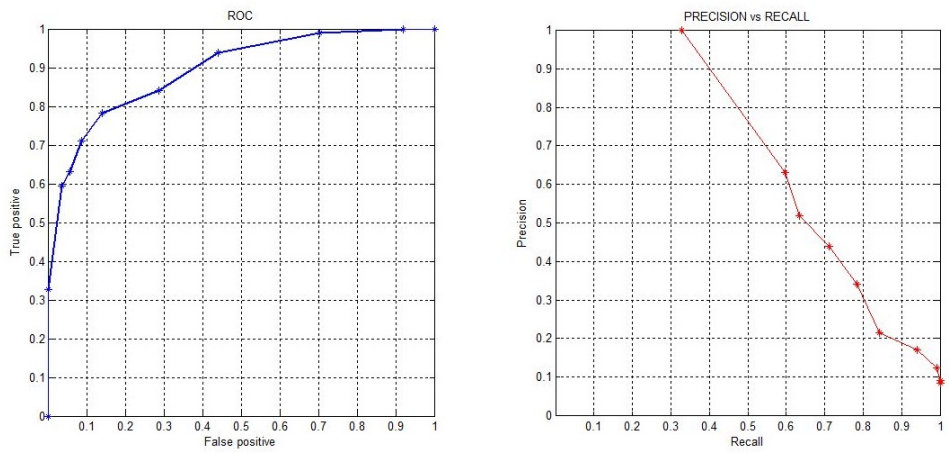

Figure 8.6. ROC and PR curves of test sets (RBF-SVM, 500% SMOTE).

Table 8.7 shows the AUC scores of training and test sets with the linear-SVM

and RBF-SVM classifiers when the corresponding ROC curves are obtained by fixing the SMOTE rate and letting the RU rate vary in our experimental configurations.

Table 8.7. AUC for SVM classifiers.

AUC	**linear-SVM**		**RBF-SVM**	
	Training	Test	Training	Test
0% SMOTE	0.9642	**0.7583**	0.6186	0.7886
100% SMOTE	0.9726	**0.7583**	0.7536	0.8367
200% SMOTE	0.9785	0.7510	0.7927	0.8816
300% SMOTE	0.9812	0.7497	0.8152	0.8848
400% SMOTE	**0.9862**	0.7464	0.8231	0.8956
500% SMOTE	0.9851	0.7515	**0.8348**	**0.8961**

In Table 8.8 and Table 8.9, the best F-measure and GMRP values observed for training and test sets with the two SVM types at each SMOTE rate are listed. The RU rates involved in sampling and which produced the *best* values are also displayed. Table 8.10, 8.11, Table 8.12 and Table 8.13 show the accuracy, recall, precision and specificity values of training and test sets with respect to the performance achieved with either linear-SVM or RBF-SVM classifiers, at the points where the best F-measure and GMRP are observed.

Table 8.8. F-measure, GMRP of SVM classifiers with training sets.

F-measure & GMRP Training sets	**linear-SVM**			**RBF-SVM**		
	RU rate	F-measure	GMRP	RU rate	F-measure	GMRP
0% SMOTE	0%	0.8530	0.8624	100%	0.3493	0.3756
100% SMOTE	0%	0.8604	0.8687	100%	0.4423	0.4423
200% SMOTE	0%	0.8648	0.8719	75%	0.4571	0.4723
300% SMOTE	0%	0.8738	0.8797	75%	0.5051	0.5175
400% SMOTE	0%	0.8820	0.8860	75%	**0.5212**	**0.5251**
500% SMOTE	0%	**0.8846**	**0.8873**	60%	0.5153	0.5227

Table 8.9. F-measure, GMRP of SVM classifiers with test sets.

F-measure & GMRP	**linear-SVM**			**RBF-SVM**		
Test sets	RU rate	F-measure	GMRP	RU rate	F-measure	GMRP
0% SMOTE	0%	**0.7071**	**0.7088**	100%	**0.6314**	**0.6320**
100% SMOTE	0%	0.6829	0.6830	75%	0.6091	0.6091
200% SMOTE	0%	0.6322	0.6324	75%	0.5782	0.5794
300% SMOTE	0%	0.6416	0.6421	60%	0.6265	0.6304
400% SMOTE	0%	0.5953	0.5969	50%	0.6276	0.6306
500% SMOTE	0%	0.5899	0.5912	50%	0.6125	0.6128

Table 8.10. Metrics of linear-SVM classifiers with training sets.

METRICS	**linear-SVM**				
Training sets	RU rate	Accuracy	Recall	Precision	Specificity
0% SMOTE	0%	0.9794	0.7438	**1.0000**	**1.0000**
100% SMOTE	0%	0.9802	0.7558	0.9984	**1.0000**
200% SMOTE	0%	0.9807	0.7671	0.9910	**1.0000**
300% SMOTE	0%	0.9817	0.7829	0.9885	0.9964
400% SMOTE	0%	**0.9826**	0.8050	0.9752	0.9964
500% SMOTE	0%	**0.9826**	**0.8213**	0.9586	0.9927

8.5.2. Evaluation of SVM Classification

SVM classifiers have proven to be well-performing and robust in discriminating members of different classes. As is the case with any classification technique, rarity and the associated imbalance issues cause problems such as incorrect generalization, overlearning, etc. with the model-based SVM approach. When proper methods to handle rarity/imbalance are employed, the achieved performance can be improved as shown by the experimental results we have attained.

The area under the ROC curve (AUC) values reported in Table 8.7 indicate that resampling training sets by creating synthetically-generated minority class samples

Table 8.11. Metrics of RBF-SVM classifiers with training sets.

METRICS	**RBF-SVM**				
Training sets	RU rate	Accuracy	Recall	Precision	Specificity
0% SMOTE	100%	0.9225	0.2554	0.5524	**0.9745**
100% SMOTE	100%	0.9037	0.4454	0.4391	0.9015
200% SMOTE	75%	0.9282	0.3650	0.6112	0.9453
300% SMOTE	75%	**0.9326**	0.4150	**0.6453**	0.9526
400% SMOTE	75%	0.9284	**0.4650**	0.5930	0.9489
500% SMOTE	60%	0.9315	0.4413	0.6192	0.9599

Table 8.12. Metrics of linear-SVM classifiers with test sets.

METRICS	**linear-SVM**				
Test sets	RU rate	Accuracy	Recall	Precision	Specificity
0% SMOTE	0%	**0.9540**	0.6620	**0.7589**	**0.9636**
100% SMOTE	0%	0.9470	**0.6700**	0.6962	**0.9636**
200% SMOTE	0%	0.9358	0.6460	0.6190	**0.9636**
300% SMOTE	0%	0.9362	0.6660	0.6190	0.9455
400% SMOTE	0%	0.9265	0.6420	0.5550	0.9091
500% SMOTE	0%	0.9260	0.6320	0.5531	0.9273

has the potential of performance improvement. Specifically, 400% borderline-SMOTE oversampling achieves the best performance on the ROC space for training sets with a linear-SVM classifier. 500% borderline-SMOTE performs best on both training and test sets when RBF-SVM classifiers are employed. Increasing the random undersampling rate with any fixed SMOTE rate tends to move the corresponding ROC point in the northeast direction, that is increasing both *tp* and *fp*.

As pointed out in the literature and experimentally verified, best performance need not be obtained for training sets where the numbers of minority and majority samples are equal. Other ratios such as 4:3, 2:1 may produce better results. This observation leads to combining SMOTE oversampling and RU. Point metrics of F-

Table 8.13. Metrics of RBF-SVM classifiers with test sets.

METRICS	**RBF-SVM**				
Test sets	RU rate	Accuracy	Recall	Precision	Specificity
0% SMOTE	100%	0.9397	0.6040	0.6613	0.9455
100% SMOTE	75%	0.9317	0.6060	0.6122	0.9091
200% SMOTE	75%	0.9207	**0.6180**	0.5432	0.8909
300% SMOTE	60%	0.9403	0.5640	**0.7047**	0.9273
400% SMOTE	50%	**0.9413**	0.5720	0.6953	**0.9818**
500% SMOTE	50%	0.9348	0.5960	0.6300	0.9455

measure and GMRP reveal the RU rates where optimal performance for the spina bifida detection problem with particular borderline-SMOTE rates can be obtained. With linear-SVM classifiers, 500% SMOTE and 0%RU (no undersampling) work best on training sets, whereas 400% SMOTE and 75% RU with RBF-SVM classifiers provides best performance on the same training sets. When test sets are considered, it is observed that the best linear-SVM classifier is one trained with no oversampling and no undersampling. The RBF-SVM classifier trained with no oversampling and 100% RU turns out to be the best running on test sets. The *best* classifiers reported correspond to single points on the ROC space (i.e. single classifiers) and can not be used to evaluate classification performance along a whole ROC curve. Performance indicators on training, test data and their comparison may occasionally look unexpected. The confusion can be resolved by considering what is meant by a training set on which performance is reported and the data with which actual training is performed. While training of classifiers is done using resampled versions of the original data, the performance is computed on the original data itself. The number of samples and distributions in the two may considerably differ and hence original training sets should be considered as sorts of other test sets. Noting that original training sets have 298 samples and test sets have 60 samples further convince in the possibility of the observed results.

9. CONCLUSION

Spina bifida is among the most common birth defects that may seriously affect the quality of life of individuals. There is no cure for nerve damage caused by the defect. If no action is taken before birth, what can be done to prevent further damage and infections is neurosurgical operation to close the opening on the back by putting the spinal cord and nerve roots back inside the spine and covering with nervous membrane (i.e. meninges). A shunt may also be surgically installed to drain excessive cerebrospinal fluid back into the abdomen or chest wall. The clinical importance of prenatal detection arises as reducing the occurrence of new cases due to terminating pregnancies with the fear that newborns might have a poor quality of life in the future or letting surgeons perform open or minimally-invasive fetal surgery.

Fetal US screening can usually detect the presence of the defect and CAD to help specialists in this task is a challenge. The easiest and most tractable CAD solution manifests itself as detecting the existence of lemon sign around transcerebellar fetal skulls. In this dissertation, we have realized two schemes that could be used in automatized detection of the defect. Both methods employ transcerebellar fetal skull images acquired via US and attack a two-class classification problem using the features extracted from skull shapes (or boundaries).

The first solution relies on measuring the similarity of skull boundaries and running a lazy nearest neighbor classifier for label assignment. Estimating the similarity of skull contours is achieved by the CSS representation of curves and the features acquired from this representation. Since features from two different curves may possess different numbers of entities, parametric (i.e. model-based) classification has been found inappropriate. CSS matching and DMLM matching are the two utilized methods for matching any two CSS images to output a matching score that represents either the distance between curves (cost of matching them) or their similarity. CSS images consist of curvature zero-crossings (γ points) that curves possess at a continuum of scales (from fine to coarse). The desirable properties of the CSS representation are its translation,

rotation and scale invariance, which are deemed critical for robust recognition and retrieval. Moreover, it provides compact description of contours with sufficient level of detail and abstracts the descriptions at multiple scales. In our implementations, an extended CSS representation, using both actual contours and their reflections has also been used. The core of the CSS-based system depends on computing CSS images from which features are derived and the time cost can be reported based on how long it takes to obtain a CSS image from a given contour. With our initial settings and the way followed in CSS images computation, it takes 0.399 seconds on average to arrive at the CSS image of a contour with an Intel Core i7 CPU running at 1.73 GHz speed and using 6 GB RAM capacity on 64-bit Windows 7 operating system. A standard deviation of 0.236 seconds of this measure is observed. The variation of processing time is associated with the variation of contour characteristics in that γ points for some contours disappear at earlier scales whereas those for others survive for more scales. Retrieving features of a specific CSS image and executing the matching algorithms for that image with all the images in training sets is quite fast. The kNN classifier requires sorting matching scores where the complexity depends log-linearly on the number of training samples.

In the second solution method, magnitudes of Zernike moments computed from normalized skull shapes are considered and the parametric SVM classifier is utilized. For any skull shape, 504 Zernike features are involved because so many moments are sufficient to reconstruct a shape within certain accuracy limits. Solving dual problems in SVM model construction does not depend on the input dimensionality but on the number of training instances N. Upper bounds of time and space complexity are $O(N^3)$ and $O(N^2)$, respectively.

Major drawbacks of experimenting with medical datasets include absolute and relative rarities. The samples that a designer is able to collect are few. In addition, members of different classes have very unbalanced distributions. The population of the common class is very likely to dominate that of the rare class. These factors prevent effective classifier design and healthy judgment of classifier performance. Handling the imbalance problem in CSS-based classification has been performed via oversampling

rare class samples with replacement or undersampling randomly selected majority class samples to balance class distributions. Different combinations of borderline-SMOTE oversampling of the rare class and random undersampling of the prevalent class have been used to resample training sets prior to SVM training. One of the reasons of preferring a nearest neighbor classifier in CSS-based classification can be stated as avoiding errors arising from misestimating parameter values due to lack of learning data (i.e. overlearning), which is supposed to prevent wrong generalization. Building optimal classifiers using rare and unbalanced data with a preferred classification technique, such as for spina bifida detection using either CSS-based kNN classifier or SVM, is a matter of considering the operation objectives, identifying them through the use of performance metrics and points on ROC and precision-recall curves, and finally selecting the classifier that best satisfies the objectives.

A general observation is that discriminating samples using simple structural and global shape descriptors is hard especially when all or most samples share close (almost identical) values for these descriptors, thus insufficient feature representations are likely to fail. This is why CSS representation describing contours (curves) at multiple scales has been preferred with a non-parametric classification approach. Zernike moments and their magnitudes are non-simple descriptors to describe shapes, that perform well in classification and appropriate for use with parametric techniques. The deductions on which representation and classification methodology to use with shape data are general and each specific problem may break the statements with alternative solutions working quite well. Rich data sets with balanced distributions are supposed to perform much better, since the probability of missing rare cases would lower by better spanning the input space with instances of correct labels.

Classification with shapes, in reality, is accompanied with the preceding problem of segmentation. Although experimental results in this work are mainly reported with ground truth (manually-segmented) shapes, two semi-automatic segmentation techniques in the context of fetal skull localization have been proposed. These are the intensity-based averaging technique and ASM segmentation, which both depend on constructing average shape models and manually marking few (four) points on the in-

put images to be segmented. The ultimate goal for segmentation is full-automaticity, that is fetal skulls are supposed to be segmented with no user intervention. Automatic segmentation naturally starts with attempts to detect discontinuities; however, edge structures in US images are not clearly identifiable, they contain considerable amount of noise and they do not appear as single and connected structures but with broken boundaries that must artificially be completed. Model fitting approaches and edge detection strategies do not perform well. Absolutely precise fetal skull segmentation seems to be performable by only the eye of a human (i.e. a medical specialist). Our foresight for full-automaticity is that a learning-based method such as the matching pursuit [158–160] has the chance of being utilized for representing structures of interest, decomposing them using a set of primitives (i.e. atoms) and in turn recognizimg similar structures in images that were not used in the learning process. The quality (correctness) of segmentation for a robust real-time system would certainly play a consequential part in CAD with skull shapes (and in any application working on shape data that is available through segmentation).

APPENDIX A: EXPERIMENTS WITH SEMI-AUTOMATICALLY SEGMENTED IMAGE DATA

The properties of the test sets related to how the instances in Appendix A are obtained (i.e. which semi-automatic segmentation technique is used) are indicated in the headings of Section A.1, Section A.2, Section A.4 and Section A.5. The training set instances employed in actual training after sampling via oversampling with replacement, random undersampling, borderline-SMOTE or combinations of borderline-SMOTE and RU; are those of ground truth segmentations. The scenarios and rules of Chapter 8 are valid for the presentations in the appendix.

A.1. CSS-Based Classification and Segmentations with Intensity-Based Averaging

Table A.1 shows the *best* F-measure and GMRP for unbalanced data sets (test sets) whose samples are obtained by segmentation through intensity-based averaging. Table A.2 lists the accuracy, precision, recall and specificity values at the corresponding operating points.

Table A.3 shows the best F-measure and GMRP for balanced test sets whose samples are obtained by segmentation through intensity-based averaging. The corresponding values of accuracy, precision, recall and specificity are shown in Table A.4.

A.2. CSS-Based Classification and ASM Segmentations

Table A.5 shows the best F-measure and GMRP for unbalanced test sets whose samples are obtained by ASM segmentation. The accuracy, recall, precision and specificity values at the corresponding ROC points are given in Table A.6.

Table A.7 shows the best F-measure and GMRP values for balanced test sets

Table A.1. F-measure, GMRP of CSS-based classifiers with *unbalanced* test sets: *segmentations with intensity-based averaging.*

UNBALANCED test	Threshold (τ)	F-measure	GMRP
CSS, C_1, real	0.60	0.2522	0.2743
CSS, C_1, both	0.35	0.2195	0.3289
CSS, C_2, real	0.70	0.2707	0.2750
CSS, C_2, both	0.35	0.2210	**0.3302**
DMLM, real	0.70	0.2713	0.2824
DMLM, both	0.65	**0.2727**	0.2934

Table A.2. Metrics of CSS-based classifiers with *unbalanced* test sets: *segmentations with intensity-based averaging.*

UNBALANCED test	Threshold (τ)	Accuracy	Recall	Precision	Specificity
CSS, C_1, real	0.60	0.7940	0.4160	0.1809	0.8284
CSS, C_1, both	0.35	0.4910	**0.8600**	0.1258	0.4575
CSS, C_2, real	0.70	**0.8577**	0.3280	**0.2305**	**0.9058**
CSS, C_2, both	0.35	0.4962	**0.8600**	0.1268	0.4631
DMLM, real	0.70	0.8343	0.3760	0.2122	0.8760
DMLM, both	0.65	0.8098	0.4320	0.1993	0.8442

whose samples are obtained by ASM segmentation. The corresponding accuracy, recall, precision and specificity are shown in Table A.8.

A.3. CSS-Based Classification and Area under the ROC Curves

AUC scores with CSS-based classifiers on unbalanced and balanced test sets for different settings using the two proposed semi-automatic segmentation schemes are displayed in Table A.9.

Table A.3. F-measure, GMRP of CSS-based classifiers with *balanced* test sets: *segmentations with intensity-based averaging.*

BALANCED test	Threshold (τ)	F-measure	GMRP
CSS, C_1, real	0.25	0.6936	0.7284
CSS, C_1, both	0.30	**0.7248**	**0.7457**
CSS, C_2, real	0.25	0.6934	0.7283
CSS, C_2, both	0.30	0.7245	0.7454
DMLM, real	0.45	0.7043	0.7184
DMLM, both	0.50	0.7098	0.7181

Table A.4. Metrics of CSS-based classifiers with *balanced* test sets: *segmentations with intensity-based averaging.*

BALANCED test	Threshold (τ)	Accuracy	Recall	Precision	Specificity
CSS, C_1, real	0.25	0.5536	**0.9989**	0.5312	0.1082
CSS, C_1, both	0.30	0.6277	0.9475	0.5868	0.3078
CSS, C_2, real	0.25	0.5532	**0.9989**	0.5310	0.1075
CSS, C_2, both	0.30	0.6273	0.9474	0.5865	0.3071
DMLM, real	0.45	0.6134	0.8777	0.5881	0.3490
DMLM, both	0.50	**0.6391**	0.8359	**0.6168**	**0.4423**

A.4. SVM Classification and Segmentations with Intensity-Based Averaging

The best F-measure and GMRP values arising for test sets with linear-SVM and RBF-SVM classifiers when samples of test sets are obtained through intensity-based averaging are shown in Table A.10. All results stand for each of the particular SMOTE rates and corresponding RU rates where best performances are observed.

Table A.11 and Table A.12 show the values of accuracy, recall, precision and specificity belonging to test sets for the operating points in Table A.10 for linear-SVM

and RBF-SVM classifiers, respectively.

Table A.5. F-measure, GMRP of CSS-based classifiers with *unbalanced* test sets: *ASM segmentations.*

UNBALANCED test	Threshold (τ)	F-measure	GMRP
CSS, C_1, real	0.60	0.2651	0.2768
CSS, C_1, both	0.55	0.2425	0.2909
CSS, C_2, real	0.60	**0.2874**	0.2976
CSS, C_2, both	0.60	0.2747	0.2831
DMLM, real	0.70	0.2642	0.2717
DMLM, both	0.35	0.2068	**0.3202**

Table A.6. Metrics of CSS-based classifiers with *unbalanced* test sets: *ASM segmentations.*

UNBALANCED test	Threshold (τ)	Accuracy	Recall	Precision	Specificity
CSS, C_1, real	0.60	0.8200	0.3720	0.2059	0.8607
CSS, C_1, both	0.55	0.7192	0.5420	0.1562	0.7353
CSS, C_2, real	0.60	0.8353	0.3880	**0.2283**	0.8760
CSS, C_2, both	0.60	0.8415	0.3620	0.2213	0.8851
DMLM, real	0.70	**0.9012**	0.2140	0.3450	**0.9636**
DMLM, both	0.35	0.4418	**0.8740**	0.1173	0.4025

A.5. SVM Classification and ASM Segmentations

The best F-measure and GMRP values for test sets with SVM classifiers when the samples are obtained with ASM segmentation are shown in Tanle A.13.

Table A.14 and Table A.15 show the accuracy, recall, precision and specificity values of test sets for the operating points in Table A.13 for linear-SVM and RBF-

SVM classifiers, respectively.

Table A.7. F-measure, GMRP of CSS-based classifiers with *balanced* test sets: *ASM segmentations.*

BALANCED test	Threshold (τ)	F-measure	GMRP
CSS, C_1, real	0.45	0.5204	0.5208
CSS, C_1, both	0.30	0.5575	0.5575
CSS, C_2, real	0.45	0.5210	0.5215
CSS, C_2, both	0.45	0.5576	0.5577
DMLM, real	0.45	0.5181	0.5185
DMLM, both	0.30	**0.6667**	**0.6866**

Table A.8. Metrics of CSS-based classifiers with *balanced* test sets: *ASM segmentations.*

BALANCED test	Threshold (τ)	Accuracy	Recall	Precision	Specificity
CSS, C_1, real	0.45	0.4797	0.5410	0.5013	0.4183
CSS, C_1, both	0.30	0.5464	0.5520	0.5631	0.5407
CSS, C_2, real	0.45	0.4804	0.5440	0.4999	0.4167
CSS, C_2, both	0.45	0.5466	0.5515	**0.5639**	**0.5418**
DMLM, real	0.45	0.4764	0.5409	0.4970	0.4119
DMLM, both	0.30	**0.5537**	**0.8760**	0.5382	0.2314

Table A.9. AUC of CSS-based classifiers with semi-automatic segmentations.

AUC	**Intensity-based averaging**		**Active shape models**	
	unbalanced test	balanced test	unbalanced test	balanced test
CSS, C_1, real	0.6739	0.6626	0.6162	0.4673
CSS, C_1, both	0.6589	0.6891	0.6261	0.5293
CSS, C_2, real	0.7048	0.6631	0.5829	0.4679
CSS, C_2, both	0.6197	**0.7018**	0.6293	0.5331
DMLM, real	0.6936	0.6934	0.6065	0.4739
DMLM, both	**0.7107**	0.6958	**0.6554**	**0.5604**

Table A.10. F-measure, GMRP of SVM classifiers: *segmentations with intensity-based averaging.*

F-measure & GMRP	**linear-SVM**			**RBF-SVM**		
Test sets	RU rate	F-measure	GMRP	RU rate	F-measure	GMRP
0% SMOTE	0%	**0.7223**	**0.7391**	100%	0.4463	0.4472
100% SMOTE	0%	0.6903	0.6948	100%	0.5542	0.5687
200% SMOTE	0%	0.6190	0.6198	100%	0.4988	0.5373
300% SMOTE	0%	0.6404	0.6410	100%	0.5001	0.5383
400% SMOTE	0%	0.6147	0.6147	75%	0.5511	0.5585
500% SMOTE	0%	0.6045	0.6045	60%	**0.5681**	**0.5793**

Table A.11. Metrics of linear-SVM: *segmentations with intensity-based averaging.*

METRICS	**linear-SVM**				
Test sets	RU rate	Accuracy	Recall	Precision	Specificity
0% SMOTE	0%	**0.9612**	0.5960	0.9165	**1.0000**
100% SMOTE	0%	0.9538	**0.6200**	**0.7785**	0.9818
200% SMOTE	0%	0.9410	0.5900	0.6511	0.9818
300% SMOTE	0%	0.9433	0.6140	0.6692	0.9818
400% SMOTE	0%	0.9367	0.6180	0.6115	0.9818
500% SMOTE	0%	0.9353	0.6080	0.6010	0.9636

Table A.12. Metrics of RBF-SVM: *segmentations with intensity-based averaging.*

METRICS	**RBF-SVM**				
Test sets	RU rate	Accuracy	Recall	Precision	Specificity
0% SMOTE	100%	0.9172	0.4200	0.4762	**0.9818**
100% SMOTE	100%	0.9032	0.7140	0.4529	0.8727
200% SMOTE	100%	0.8640	**0.7940**	0.3636	0.8364
300% SMOTE	100%	0.8637	**0.7940**	0.3650	0.8182
400% SMOTE	75%	0.9138	0.6580	0.4741	0.9273
500% SMOTE	60%	**0.9173**	0.6060	**0.4890**	0.9273

Table A.13. F-measure, GMRP of SVM classifiers: *ASM segmentations.*

F-measure & GMRP	**linear-SVM**			**RBF-SVM**		
Test sets	RU rate	F-measure	GMRP	RU rate	F-measure	GMRP
0% SMOTE	0%	**0.7071**	**0.7088**	100%	**0.6314**	**0.6320**
100% SMOTE	0%	0.6829	0.6830	100%	0.6091	0.6091
200% SMOTE	0%	0.6322	0.6324	75%	0.5782	0.5794
300% SMOTE	0%	0.6416	0.6421	60%	0.6265	0.6304
400% SMOTE	0%	0.5953	0.5969	50%	0.6276	0.6306
500% SMOTE	0%	0.5899	0.5912	50%	0.6125	0.6128

Table A.14. Metrics of linear-SVM: *ASM segmentations.*

METRICS	**linear-SVM**				
Test sets	RU rate	Accuracy	Recall	Precision	Specificity
0% SMOTE	0%	**0.9540**	0.6620	**0.7589**	**0.9636**
100% SMOTE	0%	0.9470	**0.6700**	0.6962	**0.9636**
200% SMOTE	0%	0.9358	0.6460	0.6190	**0.9636**
300% SMOTE	0%	0.9362	0.6660	0.6190	0.9455
400% SMOTE	0%	0.9265	0.6420	0.5550	0.9091
500% SMOTE	0%	0.9260	0.6320	0.5531	0.9273

Table A.15. Metrics of RBF-SVM: *ASM segmentations*.

METRICS	**RBF-SVM**				
Test sets	RU rate	Accuracy	Recall	Precision	Specificity
0% SMOTE	100%	0.9397	0.6040	0.6613	0.9455
100% SMOTE	100%	0.8677	**0.6700**	0.3536	0.8545
200% SMOTE	75%	0.9207	0.6180	0.5432	0.8909
300% SMOTE	60%	0.9403	0.5640	**0.7047**	0.9273
400% SMOTE	50%	**0.9413**	0.5720	0.6953	**0.9818**
500% SMOTE	50%	0.9348	0.5960	0.6300	0.9455

REFERENCES

1. Wingate, E.E., J.A. Hancock, K.E. Churchwell, and M.J. Pipkins, "Spina Bifida: A Review of the Importance of Sonography's Role in Its Detection", *Journal of Diagnostic Medical Sonography*, Vol. 20, No. 4, pp. 227-237, 2004.

2. Mariam, T., "The Lemon Sign", *Radiology*, Vol. 228, No. 1, pp. 206-207, 2003.

3. Konur, U. and F.S. Gürgen, "Computer Aided Diagnosis for Spina Bifida", *International Symposium on Health Informatics and Bioinformatics*, 2010.

4. Konur, U., F. Gürgen, and F. Varol, "A Two-View Ultrasound CAD System for Spina Bifida Detection using Zernike Features", *SPIE Medical Imaging Conference*, 2011.

5. Konur, U. and F.S. Gürgen, "Curvature-Based Multi-Scale Classification of Fetal Skulls for Spina Bifida Detection", *International Conference on Applied Informatics for Health and Life Sciences*, 2013.

6. Konur, U., F.S. Gürgen, F. Varol, and L. Akarun, "Computer Aided Detection of Spina Bifida using Nearest Neighbor Classification with Curvature Scale Space Features of Fetal Skulls Extracted from Ultrasound Images", *Knowledge Based Systems*, 2015, `http://dx.doi.org/10.1016/j.knosys.2015.04.021`, [Accessed June 2015].

7. Konur, U., F. Gürgen, and F. Varol, "Segmentation of Fetal Skulls using Ellipse Fitting and Active Appearance Models", *IEEE Signal Processing and Communication Applications Conference (SIU)*, 2012.

8. Lodwick, G.S., C.L. Haun, W.E. Smith, R.F. Keller, and E.D. Robertson, "Computer Diagnosis of Primary Bone Tumor", *Radiology*, Vol. 80, No. 2, pp. 273-275, 1963.

9. Myers, P.H., C.M. Nice, H.C. Becker, W.J. Nettleton, J.W. Sweeney, and G.R. Meckstroth, "Automated Computer Analysis of Radiographic Images", *Radiology*, Vol. 83, No. 6, pp. 1029-1033, 1964.

10. Winsberg, F., M. Elkin, J. Macy, V. Bordaz, and W. Weymouth, "Detection of Radiographic Abnormalities in Mammograms by Means of Optical Scanning and Computer Analysis", *Radiology*, Vol. 89, No. 2, pp. 211-215, 1967.

11. Kruger, R.P., J.R. Townes, D.L. Hall, and S.J. Dwyer, "Automated Radiographic Diagnosis via Feature Extraction and Classification of Cardiac Size and Shape Descriptors", *IEEE Trans. Biomedical Engineering*, Vol. 19, No. 3, pp. 174-186, 1972.

12. Toriwaki, J.I., Y. Suenaga, T. Negoro, and T. Fukumura, "Pattern Recognition of Chest X-ray Images", *Computer Graphics and Image Processing*, Vol. 2, No. 3-4, pp. 252-271, 1973.

13. Kruger, R.P., W.B. Thompson, and A.F. Turner, "Computer Diagnosis of Pneumoconiosis", *IEEE Trans. Systems, Man and Cybernetics*, Vol. 4, No. 1, pp. 40-49, 1974.

14. Huang, H.K., *PACS and Imaging Informatics: Basic Principles and Applications*, John Wiley & Sons, Inc., Hoboken, 2010.

15. Nakamura, K., H. Yoshida, R. Engelmann, H. MacMahon, S. Katsuragawa, T. Ishida, K. Ashizawa, and K. Doi, "Computerized Analysis of the Likelihood of Malignancy in Solitary Pulmonary Nodules with Use of Artificial Neural Networks", *Radiology*, Vol. 214, No. 3, pp. 823-830, 2000.

16. Aoyama., M., Q. Li, S. Katsuragawa, H. MacMahon, and K. Doi, "Automated Computerized Scheme for Distinction between Benign and Malignant Solitary Pulmonary Nodules on Chest Images", *Medical Physics*, Vol. 29, No. 5, pp. 701-708, 2002.

17. Shiraishi, J., H. Abe, R. Engelmann, M. Aoyama, H. MacMahon, and K. Doi, "Computer-Aided Diagnosis to Distinguish Benign from Malignant Solitary Pulmonary Nodules on Radiographs: ROC Analysis of Radiologists' Performance - Initial Experience", *Radiology*, Vol. 227, No. 2, pp. 469-474, 2003.

18. Katsuragawa, S., K. Doi, and H. MacMahon, "Image Feature Analysis and Computer-Aided Diagnosis in Digital Radiography: Detection and Characterization of Interstitial Lung Disease in Digital Chest Radiographs", *Medical Physics*, Vol. 15, No. 3, pp. 311-319, 1988.

19. Katsuragawa, S., K. Doi, and H. MacMahon, "Image Feature Analysis and Computer-Aided Diagnosis in Digital Radiography: Classification of Normal and Abnormal Lungs with Interstitial Disease in Chest Images", *Medical Physics*, Vol. 16, No. 1, pp. 38-44, 1989.

20. Ishida, T., S. Katsuragawa, K. Nakamura, K. Ashizawa, H. MacMahon, and K. Doi, "Computerized Analysis of Interstitial Lung Diseases on Chest Radiographs Based on Lung Texture, Geometric-Pattern Features and Artificial Neural Networks", *SPIE Medical Imaging Conference*, 2002.

21. Ashizawa, K., T. Ishida, H. MacMahon, C.J. Vyborny, S. Katsuragawa, and K.Doi, "Artificial Neural Networks in Chest Radiographs: Application to Differential Diagnosis of Interstitial Lung Disease", *Academic Radiology*, Vol. 6, No. 1, pp. 2-9, 1999.

22. Ashizawa, K., H. MacMahon, T. Ishida, K. Nakamura, C.J. Vyborny, S. Katsuragawa, and K. Doi, "Effect of Artificial Neural Network on Radiologists' Performance for Differential Diagnosis of Interstitial Lung Disease on Chest Radiographs", *American Journal of Roentgenology*, Vol. 172, No. 5, pp. 1311-1314, 1999.

23. Nakamori, N., K. Doi, V. Sabeti, and H. MacMahon, "Image Feature Analysis and Computer-Aided Diagnosis in Digital Radiography: Automated Analysis of

Sizes of Heart and Lung in Digital Chest Images", *Medical Physics*, Vol. 17, No. 3, pp. 342-350, 1990.

24. Nakamori, N., K. Doi, H. MacMahon, Y. Sasaki, and S. Montner, "Effect of Heart Size Parameters Computed from Digital Chest Radiographs on Detection of Cardiomegaly: Potential Usefulness for Computer-Aided Diagnosis", *Investigative Radiology*, Vol. 26, No. 6, pp. 546-550, 1991.

25. Sanada, S., K. Doi, and H. MacMahon, "Image Feature Analysis and Computer-Aided Diagnosis in Digital Radiography: Automated Detection of Pneumothorax in Chest Images", *Medical Physics*, Vol. 19, No. 5, pp. 1153-1160, 1992.

26. Kano, A., K. Doi, H. MacMahon, D.D. Hassell, and M.L. Giger, "Digital Image Subtraction of Temporally Sequential Chest Images for Detection of Interval Change", *Medical Physics*, Vol. 21, No. 3, pp. 453-461, 1994.

27. Difazio, M.C., H. MacMahon, X.W. Xu, P. Tsai, J. Shiraishi, S.G. Armato, and K. Doi, "Effect of Time Interval Difference Images on Detection Accuracy in Digital Chest Radiography", *Radiology*, Vol. 202, No. 2, pp. 447-452, 1997.

28. Ishida, T., S. Katsuragawa, K. Nakamura, H. MacMahon, and K. Doi, "Iterative Image Warping Technique for Temporal Subtraction of Sequential Chest Radiographs to Detect Interval Change", *Medical Physics*, Vol. 26, No. 7, pp. 1320-1329, 1999.

29. Li, Q., S. Katsuragawa, and K. Doi, "Improved Contralateral Subtraction Images by Use of Elastic Matching Technique", *Medical Physics*, Vol. 27, No. 8, pp. 1934-1942, 2000.

30. Kakeda, S., K. Nakamura, K. Kamada, H. Watanabe, H. Nakata, S. Katsuragawa, and K. Doi, "Improved Detection of Lung Nodules by Using a Temporal Subtraction Technique", *Radiology*, Vol. 224, No. 1, pp. 145-151, 2002.

31. Kakeda, S., K. Kamada, Y. Hatakeyama, T. Aoki, T. Ohguri, Y. Korogi, and K. Doi, "Effect of Temporal Subtraction of Technique on Reading Time and Diagnostic Accuracy in the Interpretation of Chest Radiographs", *American Journal of Roentgenology*, Vol. 187, No. 5, pp. 1253-1259, 2006.

32. Shiraishi, J., F. Li, and K. Doi, "Computer-Aided Diagnosis for Improved Detection of Lung Nodules by Use of PA and Lateral Chest Radiographs", *Academic Radiology*, Vol. 14, No. 1, pp. 28-37, 2007.

33. Fraioli, F., L. Bertoletti, A. Napoli, F. Pediconi, F.A. Calabrese, R. Masciangelo, C. Catalano, and R. Passariello, "Computer-Aided Detection (CAD) in Lung Cancer Screening at Chest MDCT: ROC Analysis of CAD versus Radiologist Performance", *Journal of Thoracic Imaging*, Vol. 22, No. 3, pp. 241-246, 2007.

34. Freer, T.W. and M.J. Ulissey, "Screening Mammography with Computer-Aided Detection: Prospective Study of 12,860 Patients in a Community Breast Center", *Radiology*, Vol. 220, No. 3, pp. 781-786, 2001.

35. Gur, D., J.H. Sumkin, H.E. Rockette, M. Ganott, C. Hakim, L. Hardesty, W.R. Poller, R. Shah, and L. Wallace, "Changes in Breast Cancer Detection and Mammography Recall Rate after the Introduction of a Computer-Aided Detection System", *Journal of the National Cancer Institute*, Vol. 96, No. 3, pp. 185-190, 2004.

36. Birdwell, R.L., P. Bandodkar, and D.M. Ikeda, "Computer-Aided Detection with Screening Mammography in a University Hospital Setting", *Radiology*, Vol. 236, No. 2, pp. 451-457, 2005.

37. Cupples, T.E., J.E. Cunningham, and J.C. Reynolds, "Impact of Computer Aided Detection in a Regional Screening Mammography Program", *American Journal of Roentgenology*, Vol. 185, No. 4, pp. 944-950, 2005.

38. Morton, M.J., D.R. Whaley, K.R. Brandt, and K.K. Amrami, "Screening Mam-

mograms: Interpretation with Computer Aided Detection - Prospective Evaluation", *Radiology*, Vol. 239, No. 2, pp. 375-383, 2006.

39. Dean, J.C. and C.C. Ilvento, "Improved Cancer Detection using Computer-Aided Detection with Diagnostic and Screening Mammography: Prospective Study of 104 Cancers", *American Journal of Roentgenology*, Vol. 187, No. 1, pp. 20-28, 2006.

40. Rangayyan, R.M., F.J. Ayres, and J.E.L. Desautels, "A Review of Computer-Aided Diagnosis of Breast Cancer: Toward the Detection of Subtle Signs", *Journal of the Franklin Institute - Special Issue: Medical Applications of Signal Processing Part I*, Vol. 344, No. 3-4, pp. 312-348, 2007.

41. Linguraru, M.G., S. Zhao, R.L. Van Uitert, L. Jiamin, J.G. Fletcher, A. Manduca, and R.M. Summers, "CAD of Colon Cancer on CT Colonography Cases without Cathartic Bowel Preparation", *IEEE Annual International Conference of the Engineering in Medicine and Biology*, 2008.

42. Arnoldi, E., M. Gebregziabher, U.J. Schoepf, R. Goldenberg, L. Ramos-Duran, P.L. Zwerner, K. Nikolaou, M.F. Reiser, P. Costello, and C. Thilo, "Automated Computer-Aided Stenosis Detection at Coronary CT Angiography: Initial Experience", *European Radiology*, Vol. 20, No. 5, pp. 1160-1167, 2010.

43. Halpern, E.J. and D. J. Halpern, "Diagnosis of Coronary Stenosis with CT Angiography: Comparison of Automated Computer Diagnosis with Expert Readings", *Academic Radiology*, Vol. 18, No. 3, pp. 324-333, 2011.

44. Kang, K.W., H.J. Chang, H. Shim, Y.J. Kim, B.W. Choi, W.I. Yang, J.Y. Shim, J. Ha, and N. Chung, "Feasibility of an Automatic Computer-Assisted Algorithm for the Detection of Significant Coronary Artery Disease in Patients Presenting with Acute Chest Pain", *European Journal of Radiology*, Vol. 81, No. 4, pp. 640-646, 2012.

45. Dominguez, A.R. and A.K. Nandi, "Detection of Masses in Mammograms via Statistically Based Enhancement, Multilevel Thresholding, Segmentation and Region Selection", *Computerized Medical Imaging and Graphics*, Vol. 32, No. 4, pp. 304-315, 2008.

46. Chen, C.Y., H.K. Chang, Y.T. Kuo, and H.J. Chiou, "P2D-10 Computer-Aided Diagnosis of Peripheral Soft Tissue Tumors using Geometric and Texture Features", *IEEE Ultrasonics Symposium*, 2006.

47. Kasai, S., F. Li, J. Shiraishi, Q.Li, Y. Nie, and K. Doi, "Development of Computerized Method for Detection of Vertebral Fractures on Lateral Chest Radiographs", *SPIE Medical Imaging Conference*, 2006.

48. Kasai, S., F. Li, J. Shiraishi, Q. Li, and K. Doi, "Computerized Detection of Vertebral Compression Fractures on Lateral Chest Radiographs: Preliminary Results of a Tool for Early Detection of Osteoporosis", *Medical Physics*, Vol. 33, No. 12, pp. 4664-4674, 2006.

49. Arimura, H., Q. Li, Y. Korogi, T. Hirai, H. Abe, Y. Yamashita, S. Katsuragawa, R. Ikeda, and K. Doi, "Automated Computerized Scheme for Detection of Unruptured Intracranial Aneurysms in Three-Dimensional MRA", *Academic Radiology*, Vol. 11, No. 10, pp. 1093-1104, 2004.

50. Arimura, H., Q. Li, Y. Korogi, T. Hirai, S. Katsuragawa, Y. Yamashita, K. Tsuchiya, and K. Doi, "Computerized Detection of Intracranial Aneurysms for 3D MR Angiography: Feature Extraction of Small Protrusions Based on a Shape-Based Difference Image Technique", *Medical Physics*, Vol. 33, No. 2, pp. 394-401, 2006.

51. Shiraishi, J., Q. Li, D. Appelbaum, Y. Pu, and K. Doi, "Development of a Computer Aided Diagnostic Scheme for Detection of Interval Changes in Successive Whole-Body Scans", *Medical Physics*, Vol. 34, No. 1, pp. 25-36, 2007.

52. Doi, K., "Computer-Aided Diagnosis in Medical Imaging: Historical Review, Current Status and Future Potential", *Computerized Medical Imaging and Graphics*, Vol. 31, No. 4-5, pp. 198-211, 2007.

53. Zhang, D. and G. Lu, "Review of Shape Representation and Description Techniques", *Pattern Recognition*, Vol. 37, No. 1, pp. 1-19, 2004.

54. Huttenlocher, D.P., G.A. Klanderman, and W.J. Rucklidge, *Comparing Images using the Hausdorff Distance*, Technical Report: CUCS-TR-91-1211, Department of Computer Science, Cornell University, 1991.

55. Belongie, S., J. Malik, and J. Puzicha, "Matching Shapes", *IEEE International Conference on Computer Vision*, 2001.

56. Gonzalez, R.C. and R.E. Woods, *Digital Image Processing*, Addison-Wesley Longman Publishing Co., Inc., Boston, 3rd edition, 1992.

57. Bimbo, A.D. and P. Pala, "Visual Image Retrieval by Elastic Matching of User Sketches", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 19, No. 2, pp. 121-132, 1997.

58. Chellappa, R. and R. Bagdazian, "Fourier Coding of Image Boundaries", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 6, No. 1, pp. 102-105, 1984.

59. Lin, C.C. and R. Chellappa, "Classification of Partial 2D Shapes using Fourier Descriptors", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 9, No. 5, pp. 686-690, 1987.

60. Arbter, K., "Affine-Invariant Fourier Descriptors", in J.C. Simon (ed.), *From Pixels to Features*, pp. 153-165, Elsevier Science Publishers, Amsterdam, 1989.

61. Arbter, K., W.E. Snyder, H. Burkhardt, and G. Hirzinger, "Application of Affine-

Invariant Fourier Descriptors to Recognition of 3-D Objects", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 12, No. 7, pp. 640-647, 1990.

62. Granlund, G., "Fourier Preprocessing for Hand Print Character Recognition", *IEEE Trans. Computers*, Vol. 21, No. 2, pp. 195-201, 1972.

63. Rauber, T.W., *Two-Dimensional Shape Description*, Technical Report: GR UNINOVA-RT-10-94, University Nova de Lisboa, 1994.

64. Richard, C.W. and H. Hemami, "Identification of Three-Dimensional Objects using Fourier Descriptors of the Boundary Curve", *IEEE Trans. Systems, Man and Cybernetics*, Vol. 4, No. 4, pp. 371-378, 1974.

65. Ohm, J.R., F.B. Bunjamin, W. Liebsch, B. Makai, K. Muller, A. Somlic, and D. Zier, "A Set of Visual Feature Descriptors and Their Combination in a Low-Level Description Scheme", *Signal Processing: Image Communication*, Vol. 16, No. 1-2, pp. 157-179, 2000.

66. Tieng, Q.M. and W.W. Boles, "Recognition of 2D Object Contours using the Wavelet Transform Zero-Crossing Representation", *IEEE Trans. Patttern Analysis and Machine Intelligence*, Vol. 19, No. 8, pp. 910-916, 1997.

67. Yang, H.S., S.U. Lee, and K.M. Lee, "Recognition of 2D Object Contours using Starting-Point-Independent Wavelet Coefficient Matching", *Visual Communication and Image Representation*, Vol. 9, No. 2, pp. 171-181, 1998.

68. Das, M., M.J. Paulik, and N.K. Loh, "A Bivariate Autoregressive Modeling Technique for Analysis and Classification of Planar Shapes", *IEEE Trans. Patttern Analysis and Machine Intelligence*, Vol. 12, No. 1, pp. 97-103, 1990.

69. Asada, H. and M. Brady, *The Curvature Primal Sketch*, MIT AI Memo 758, 1984.

70. Asada, H. and M. Brandy, "The Curvature Primal Sketch", *IEEE Trans. Patttern*

Analysis and Machine Intelligence, Vol. 8, No. 1, pp. 2-14, 1986.

71. Mokhtarian, F. and A. Mackworth, "Scale Based Description and Recognition of Planar Curves and Two Dimensional Shapes", *IEEE Trans. Patttern Analysis and Machine Intelligence*, Vol. 8, No. 1, pp. 34-43, 1986.

72. Mokhtarian, F., S. Abbasi, and J. Kittler, "Robust and Efficient Shape Indexing through Curvature Scale Space", *British Machine Vision Conference*, 1996.

73. Mokhtarian, F., S. Abbasi, and J. Kittler, "Efficient and Robust Retrieval by Shape Content through Curvature Scale Space", *International Workshop on Image Databases and Multimedia Search*, 1996.

74. Abbasi, S., D. Mokhtarian, and J. Kittler, "Curvature Scale Space Image in Shape Similarity Retrieval", *Multimedia Systems*, Vol. 7, No. 6, pp. 467-476, 1999.

75. Freeman, H., "On the Encoding of Arbitrary Geometric Configurations", *IRE Trans. Electronic Computers*, Vol. 10, No. 2, pp. 260-268, 1961.

76. Freeman, H. and A. Saghri, "Generalized Chain Codes for Planar Curves", *International Joint Conference on Pattern Recognition*, 1978.

77. Grosky, W.I. and R. Mehrotra, "Index-Based Object Recognition in Pictorial Data Management", *Computer Vision, Graphics and Image Processing*, Vol. 52, No. 3, pp. 416-436, 1990.

78. Grosky, W.I., P. Neo, and R. Mehrotra, "A Pictorial Index Mechanism for Model-Based Matching", *Data and Knowledge Engineering*, Vol. 8, No.4, pp. 309-327, 1992.

79. Mehrotra, R. and J.E. Gary, "Similar-Shape Retrieval in Shape Data Management", *IEEE Computer*, Vol. 28, No. 9, pp. 57-62, 1995.

80. Berretti, S., A.D. Bimbo, and P. Pala, "Retrieval by Shape Similarity with Per-

ceptual Distance and Effective Indexing", *IEEE Trans. Multimedia*, Vol. 2, No. 4, pp. 225-239, 2000.

81. Dudek, G. and J.K. Tsotsos, "Shape Representation and Recognition from Multiscale Curvature", *Computer Vision and Image Understanding*, Vol. 68, No. 2, pp. 170-189, 1997.

82. Fu, K.S., *Syntactic Methods in Pattern Recognition*, Academic Press, New York, 1974.

83. Sonka, M., V. Hlavac, and R. Boyle, *Image Processing, Analysis and Machine Vision*, Chapman & Hall, London, 1993.

84. Li, S.Z., "Shape Matching Based on Invariants", in O.M. Omidvar (ed.), *Shape Analysis*, Vol. 6, pp. 203-228, Ablex, 1999.

85. Huang, C.L. and D.H. Huang, "A Content-Based Image Retrieval System", *Image and Vision Computing*, Vol. 16, No. 3, pp. 149-163, 1998.

86. Squire, D.M. and T.M. Caelli, "Invariance Signature: Characterizing Contours by Their Departures from Invariance", *Computer Vision and Image Understanding*, Vol. 77, No. 3, pp. 284-316, 2000.

87. Hu, M.K., "Visual Pattern Recognition by Moment Invariants", *IRE Trans. Information Theory*, Vol. 8, No. 2, pp. 179-187, 1962.

88. Taubin, G. and D.B. Cooper, "Recognition and Positioning of Rigid Objects using Algebraic Moment Invariants", *SPIE Geometric Methods in Computer Vision Conference*, 1991.

89. Taubin, G. and D.B. Cooper, "Object Recognition Based on Moment (or Algebraic) Invariants", in J. Mundy and A. Zisserman (eds.), *Geometric Invariance in Computer Vision*, pp. 375-397, MIT Press, Cambridge, 1992.

90. Teague, M.R, "Image Analysis via the General Theory of Moments", *Journal of Optical Society of America*, Vol. 70, No. 8, pp. 920-930, 1980.

91. Teh, C.H. and R.T. Chin, "On Image Analysis by the Method of Moments", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 10, No. 4, pp. 496-513, 1988.

92. Liao, S.X. and M. Pawlak, "On Image Analysis by Moments", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 18, No. 3, pp. 254-266, 1996.

93. Zhang, D.S. and G. Lu, "Generic Fourier Descriptor for Shape-Based Image Retrieval", *IEEE International Conference on Multimedia and Expo*, 2002.

94. Lu, G.J. and A. Sajjanhar, "Region-Based Shape Representation and Similarity Measure Suitable for Content-Based Image Retrieval", *Multimedia Systems*, Vol. 7, No. 2, pp. 165-174, 1999.

95. Goshtasby, A., "Description and Discrimination of Planar Shapes using Shape Matrices", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 7, No. 6, pp. 738-743, 1985.

96. Blum, H., "A Transformation for Extracting New Descriptors of Shape", in W. Whaten-Dunn (ed.), *Models for the Perception of Speech and Visual Forms*, pp. 362-380, MIT Press, Cambridge, 1967.

97. Alpaydın, E., *Introduction to Machine Learning*, MIT Press, 2nd edition, 2010.

98. Quinlan, J.R., "Induction of Decision Trees", *Machine Learning*, Vol. 1, No. 1, pp. 81-106, 1986.

99. Quinlan, J.R., *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

100. Breiman, L., J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth International Group, 1984.

101. Liu, B., W. Hsu, and Y.Ma, "Integrating Classification and Association Rule Mining". *International Conference on Knowledge Discovery and Data Mining*, 1998.

102. Jensen, F., *An Introduction to Bayesian Networks*, Springer, New York, 1996.

103. Hertz, J., A. Krogh, and R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, 1991.

104. Boser, B., I. Guyon, and V.N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers", *Annual Workshop on Computational Learning Theory*, 1992.

105. Vapnik, V.N. and A.Y. Chervonenkis, "On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities", *Theory of Probability and Its Applications*, Vol. 16, No. 2, pp. 264–280, 1971.

106. Shapiro, L. and G. Stockman, *Computer Vision*, 2000.

107. Forsyth, D.A. and J. Ponce, *Computer Vision: A Modern Approach*, Pearson, 2nd edition, 2011.

108. Haralick, R.M. and L.G. Shapiro, "Image Segmentation Techniques", *Computer Vision, Graphics and Image Processing*, Vol. 29, No. 1, pp. 100-132, 1985.

109. Sobel, I., *An Isotropic 3x3 Image Gradient Operator*, Stanford Artificial Intelligence Project, 1968.

110. Prewitt, J.M.S., "Object Enhancement and Extraction", in B. Lipkin and A. Rosenfeld (eds.), *Picture Processing and Psychopictorics*, pp. 75-149, Academic Press, New York, 1970.

111. Roberts, G.L., *Machine Perception Of Three-Dimensional Solids*, Ph.D. Thesis, Massachusetts Institute of Technology, 1963.

112. Canny, J., "A Computational Approach to Edge Detection", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, pp. 679-714, 1986.

113. Otsu N., "A Threshold Selection Method from Gray-Level Histograms", *IEEE Trans. Systems, Man and Cybernetics*, Vol. 9, No. 1, pp. 62-66, 1979.

114. Ohlander, R., K. Price, and D.R. Reddy, "Picture Segmentation using a Recursive Region Splitting Method", *Computer Graphics and Image Processing*, Vol. 8, No. 3, pp. 313-333, 1978.

115. Shi J. and J. Malik, "Normalised Cuts and Image Segmentation", *IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

116. Dempster, A.P., N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", *Journal of Royal Statistical Society*, Vol. 39, No. 1, pp. 1-38, 1977.

117. Redner, R.A. and H.F. Walker, *Mixture Densities, Maximum Likelihood and the EM Algorithm*, SIAM Review 26, pp. 195-239, 1984.

118. Belongie, S., C. Carson, H. Greenspan, and J. Malik, "Color and Texture-Based Image Segmentation using EM and Its Applications to Content Based Image Retrieval", *International Conference on Computer Vision*, 1998.

119. Kass, M., A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models", *International Journal of Computer Vision*, Vol. 8, No. 2, pp. 321-331, 1988.

120. Yuille, A.L., P.W. Hallinan, and D.S. Cohen, "Feature Extraction from Faces using Deformable Templates", *International Journal of Computer Vision*, Vol. 8, No. 2, pp. 99-111, 1992.

121. Cootes, T.F., C.J. Taylor, D.H. Cooper, and J. Graham, "Active Shape Models - Their Training and Application", *Computer Vision and Image Understanding*,

Vol. 61, No. 1, pp. 38-59, 1995.

122. Cootes, T.F., G.J. Edwards, and C.J. Taylor, "Active Appearance Models", *European Conference on Computer Vision*, 1998.

123. Weiss, G.M., "Mining with Rarity: A Unifying Framework", *ACM SIGKDD Explorations Newsletter - Special Issue on Learning from Imbalanced Datasets*, Vol. 6, No. 1, pp. 7-19, 2004.

124. Swets, J.A., "Measuring the Accuracy of Diagnostic Systems", *Science*, Vol. 240, No. 4857, pp. 1285-1293, 1988.

125. Fawcett, T., "An Introduction to ROC Analysis", *Pattern Recognition Letters - Special Issue: ROC Analysis in Pattern Recognition*, Vol. 27, No. 8, pp. 861-874, 2006.

126. Bradley, A., "The Use of the Area under the ROC Curve in the Evaluation of Machine Learning Algorithms", *Pattern Recognition*, Vol. 30, No. 7, pp. 1145-1159, 1997.

127. Van Rijsbergen, C.J., *Information Retrieval*, Butterworths, London, 1979.

128. Joshi, M.V., "On Evaluating Performance of Classifiers for Rare Classes", *IEEE International Conference on Data Mining*, 2002.

129. Kubat, M. and S. Matwin, "Addressing the Curse of Imbalanced Training Sets: One-Sided Selection", *International Conference on Machine Learning*, 1997.

130. Chawla, N.V., K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer, "SMOTE: Synthetic Minority Over-Sampling Technique", *Journal of Artificial Intelligence Research*, Vol. 16, No. 1, pp. 321-357, 2002.

131. Schapire, R.E., "A Brief Introduction to Boosting", *International Joint Conference on Artificial Intelligence*, 1999.

132. Joshi, M.V., V. Kumar, and R.C. Agarwal, "Evaluating Boosting Algorithms to Classify Rare Cases: Comparison and Improvements", *IEEE International Conference on Data Mining*, 2001.

133. Chawla, N.V., A. Lazarevic, L.O. Hall, and K. Bowyer, "SMOTEBoost: Improving Prediction of the Minority Class in Boosting", *European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2003.

134. Han, S., B. Yuan, and W. Liu, "Rare Class Mining: Progress and Prospect", *Chinese Conference on Pattern Recognition*, 2009.

135. Mokhtarian, F. and A.K. Mackworth, "A Theory of Multiscale, Curvature-Based Shape Representation for Planar Curves", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 14, No. 18, pp. 789-805, 1992.

136. Yang, M., K. Kpalma, and J. Ronsin, "Affine Invariance Contour Descriptor Based on Iso-Area Normalization", *Electronic Letters*, Vol. 43, No. 7, pp. 1-2, 2007.

137. Kpalma, K. and J. Ronsin, "Turning Angle Based Representation for Planar Objects", *Electronic Letters*, Vol. 43, No. 10, pp. 1-2, 2007.

138. Kpalma, K., M. Yang, and J. Ronsin, "Planar Shapes Descriptors Based on the Turning Angle Scalogram", *International Conference on Image Analysis and Recognition*, 2008.

139. Kpalma, K. and J. Ronsin, "A Novel Multi-Scale Representation for 2-D Shapes", *International Conference on Image Analysis and Recognition*, 2007.

140. Avrithis, S.Y., Y. Xirouhakis, and S.D. Kollias, "Affine-Invariant Curve Normalization for Object Shape Representation, Classification and Retrieval", *Machine Vision and Applications*, Vol. 13, No. 2, pp. 80-94, 2001.

141. Kpalma, K. and J. Ronsin, "Multiscale Contour Description for Pattern Recognition", *Pattern Recognition Letters*, Vol. 27, No. 13, pp. 1545-1559, 2006.

142. Kopf, S., T. Haenselmann, and W. Effelsberg, "Enhancing Curvature Scale Space Features for Robust Shape Classification", *International Conference on Multimedia and Expo*, 2005.

143. Richter, S., G. Kühne, and O. Schuster, "Contour-Based Classification of Video Objects", *SPIE Storage and Retrieval for Media Databases Conference*, 2001.

144. Zernike, F., "Beugungstheorie des Schneidenver-Fahrens und Seiner Verbesserten Form, der Phasenkontrastmethode", *Physica*, Vol. 1, No. 7-12, pp. 689-704, 1934.

145. Hosny, K.M., "A Systematic Method for Efficient Computation of Full and Subsets Zernike Moments", *Information Sciences*, Vol. 180, No. 11, pp. 2299-2313, 2010.

146. Khotanzad, A. and Y.H. Hong, "Invariant Image Recognition by Zernike Moments", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 12, No. 5, pp. 489-497, 1990.

147. Hosny, K.M., "Exact and Fast Computation of Geometric Moments for Gray Level Images", *Applied Mathematics and Computation*, Vol. 189, No. 2, pp. 1214-1222, 2007.

148. Vapnik, V., *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.

149. Vapnik, V., *Statistical Learning Theory*, Wiley, New York, 1998.

150. Cortes, C. and V. Vapnik, "Support Vector Networks", *Machine Learning*, Vol. 20, No. 3, pp. 373-297, 1995.

151. Song, G. and H. Wang, "A Fast and Robust Ellipse Detection Algorithm Based

on Pseudo-Random Sample Consensus", *International Conference on Computer Analysis of Images and Patterns*, 2007.

152. Ester, M., H.P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", *International Conference on Knowledge Discovery and Data Mining*, 1996.

153. Stegmann, M. B., *Active Appearance Models: Theory, Extensions and Cases*, M.S. Thesis, Technical University of Denmark, 2000.

154. Kroon D.J., *Active Shape Model and Active Appearance Model*, 2010, `http://www.mathworks.com/matlabcentral/fileexchange`. [Accessed November 2014].

155. Goodall, C., "Procrustes Methods in the Statistical Analysis of Shape", *Journal of the Royal Statistical Society B*, Vol. 53, No. 2, pp. 285-339, 1991.

156. Han, H., W.Y. Wang, and B.H. Mao, "Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning", *International Conference on Advances in Intelligent Computing*, 2005.

157. Chang, C.C. and C.J. Lin, "LIBSVM : A Library for Support Vector Machines", *ACM Trans. Intelligent Systems and Technology*, Vol. 2, No. 3, pp. 1-27, 2011.

158. Bergeaud, F. and S.G. Mallat, "Matching Pursuit of Images", *IEEE International Conference on Image Processing*, 1995.

159. Mallat, S.G., "Matching Pursuits with Time-Frequency Dictionaries", *IEEE Trans. Signal Processing*, Vol. 41, No. 12, pp. 3397-3415, 1993.

160. Mendels, F., P. Vandergheynst, and J.P. Thiran, "Matching Pursuit-Based Shape Representation and Recognition Using Scale-Space", *Wiley Periodicals*, Vol. 16, No. 5, pp. 162-180, 2007.