ABSTRACT

SHETTY, SANKETH V. A Biologically Plausible Architecture for Shape Recognition. (Under the direction of Professor Wesley E. Snyder).

This thesis develops an algorithm for shape representation and matching. The algorithm is an object centered, boundary-based method for shape recognition. Global features of the shape are utilized to define a frame of reference relative to which local shape features are characterized. The curvature of the boundary at a point is the local feature used. Curvature is computed by the Digital Straight Segments algorithm. Matching is done using the process of evidence accumulation similar in approach to the Generalized Hough Transform. The algorithm is tested for invariance to similarity transforms. Its robustness to noise and blurring is also tested. A multi-layer, feed-forward neural network architecture that implements the algorithm is proposed.

A Biologically Plausible Architecture for Shape Recognition

by

Sanketh V. Shetty

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Electrical Engineering

Raleigh

2006

Approved By:

Dr. Griff Bilbro	Dr. David S Lalush
Dr. Wesley E. Snyder Chair of Advisory Committee	Dr. H Joel Trussell

To Mom and Dad . . .

Biography

Sanketh Shetty was born in the south Indian port city of Mangalore on the 24th of August 1982. He did his primary schooling at Bishop Cottons Boys School, Bangalore and St.Dominics Savio College, Lucknow. He attended middle school at La Martiniere Boys Junior College, Lucknow. He completed his higher secondary education at DAV Boys Senior Secondary School in Chennai.

In 2000, he enrolled into the Bachelor of Engineering program at Birla Institute of Technology and Science (BITS), Pilani, India. There he majored in Electrical and Electronics Engineering with an emphasis in Robotics. In his senior year he completed a six month internship at Hewlett Packard Laboratories in Bangalore, working in Speech Recognition. In Spring 2004 he graduated with distinction and the Fall of the same year he joined the Master's program in Electrical and Computer Engineering at North Carolina State University, Raleigh. Since the summer of 2005 he has been a member of the Image Analysis Laboratory and worked under the guidance of Dr.Wesley Snyder. He has been accepted into the PhD program at the Department of Electrical and Computer Engineering, University of Illinois, Urbana-Champaign. He plans to enroll there in the fall semester of 2006.

Sanketh's interests are in Computer Vision, Visual Neuroscience, Cognitive Science and Artificial Intelligence.

Acknowledgements

As a novice to research in the field of Image Analysis it was a matter of tremendous honor and privilege for me to be associated with Dr. Wesley Snyder. As my advisor he helped me formulate and tackle the problem addressed in this thesis. At every juncture he was readily available with critiques of the ideas that my research associate and I threw at him. I can not thank him enough for the belief he showed in me and for educating me in the process of research. I express my sincerest gratitude to Dr. Joel Trussell, Dr. David Lalush and Dr. Griff Bilbro for agreeing to serve on my thesis committee and for providing valuable feedback on my work.

Over the last year I have worked closely with Karthik Krish at the Image Analysis Laboratory in developing ideas and testing them through elaborate experiments. In him I found a valuable research associate with whom I could discuss ideas for the algorithm and engage in passionate debate over the merits of various approaches. I have gained immensely from his programming practices and I thank him for the experience. To Geoff Chang, our fellow research associate, I extended gratitude for all the help and support over the year. I wish to acknowledge Dr.Mark White, Dr. Ronald Endicott and Dr. James Lester for helping me develop my interest in Artificial Intelligence and Cognitive Science. They have, in their own ways, through teaching and discussions, helped me understand and define my research. I thank them for that.

My parents have always been the strongest source of support and inspiration to me. I wish to acknowledge and thank them for never letting anything affect my education. To my friends Nilesh, Gopal, Ravi, Coma and Ranjith I express my thanks for all the good times.

Contents

List of Figures **vii**

List of Tables **ix**

1 Introduction **1**

2 Background **5**
- 2.1 2D Shape Representation Paradigms 6
- 2.2 Computational Models of Object Recognition 7
- 2.3 The Generalized Hough Transform 9

3 The Computational Algorithm **12**
- 3.1 Model of Shape Representation 13
  - 3.1.1 Description of the Algorithm 14
- 3.2 Shape Matching 19
  - 3.2.1 Preprocessing of Shape Contour 20
  - 3.2.2 Description of Matching Algorithm 21
- 3.3 Examples of Model Building and Matching 27
- 3.4 Strategies to Reduce Computational Cost 32

4 Experiments and Results **34**
- 4.1 The Data Set 35
- 4.2 Robustness to Similarity Transforms 38
  - 4.2.1 Experiment on Rotational Invariance 38
  - 4.2.2 Experiment on Scale Invariance 38
- 4.3 Effect of Noise and Blur on Performance 41
- 4.4 Effect of the Increment Function on Performance 45
  - 4.4.1 Effect on Rotational Invariance 45
  - 4.4.2 Effect on Scale Invariance 45
  - 4.4.3 Effect of Contour Noise and Blur on Performance 47

4.5 Similarity Detection 48

5 Neural Network Architecture for Shape Recognition 55

5.1 The Visual Pathway 56

5.1.1 V1 58

5.1.2 Extra-striate Processing of Shape Information 59

5.2 VisNet and the Standard Model 60

5.2.1 VisNet 61

5.2.2 The Standard Model 61

5.3 The SKS Neural Network Architecture 63

5.3.1 Description of the Architecture 63

5.3.2 Equivalence to the SKS algorithm 67

6 Conclusion and Future Directions 71

6.1 Future Work 73

Bibliography 75

A Estimation of Curvature using Digital Straight Segments 81

A.1 Discrete Curvature Estimation 81

A.2 Digital Straight Segments 82

A.2.1 Algorithm for Recognition of *Standard* Lines 83

A.3 Estimation of Curvature and Tangents Using DSS 84

A.4 Experiments and Results 84

List of Figures

3.1 Geometric Center and the principal axis of the rotated image. 14
3.2 Plot of the function that remaps the curvature calculated by the DSS algorithm to values between 0 and 1. 18
3.3 As the principal axis by nature is not directed we have to select one of the two possible directions. This results in ambiguity in angle descriptions. The two possibilities are shown in (a) and (b). It is clear that the angles calculated for the same point is dependent on how the rhs system is set up. 24
3.4 Contours of the tanks Chaff (top) and Chief (bottom) 28
3.5 The two tank contours after transformation to their new coordinate systems defined by their principal axes. 29
3.6 Model of the tank Chaff 30
3.7 Model of the tank Chief 30
3.8 Target:Chaff, Model:Chaff(left),Chief(right) 31
3.9 Target:Chief, Model:Chaff(left),Chief(right) 31

4.1 The data set of tanks. 36
4.2 The 31 contours sample from the SQUID Data Set. 37
4.3 Variation of Accumulator Peaks for tanks contours, matched against their own models, with orientation. 39
4.4 Variation of Accumulator Peaks for tanks contours, matched against their own models, with size. 40
4.5 A tank contour with jitter added to the boundaries. 42
4.6 A fish contour (right) shown on the left after 2000 iterations of blurring. 42
4.7 Variation of Accumulator Peaks for tanks contours, matched against their own models, with noise. 43
4.8 Variation of Accumulator Peaks for tanks contours, matched against their own models, with increasing blurring. 43
4.9 Variation of classification accuracy with increasing blur (tested on the SQUID data set). 44

4.10 Variation of Accumulator Peak for Chaff with standard deviation of increment function for different angles of rotation. 46
4.11 Variation of Accumulator Peak for Chaff with standard deviation of increment function for different scales. 46
4.12 Variation of Accumulator Peak for Chaff with standard deviation of increment function for different levels of contour noise. 48
4.13 Variation of Accumulator Peak for Chaff with standard deviation of increment function for different levels of contour blur. 49
4.14 Top 5 matches (black contours) from the SQUID data base for the model (white contour) of contour kk87. 52
4.15 Top 5 matches (black contours) from the SQUID data base for the model (white contour) of contour kk732. 53
4.16 Top 5 matches (black contours) from the SQUID data base for the model (white contour) of contour kk942. 53
4.17 Top 3 matches (black contours) from the SQUID data base for the model (white contour) of contour kk779. Other matches were below the threshold of 60. 54

5.1 A block diagram of the Visual Pathway. 57
5.2 The SKS Neural Network Architecture. Alternating layers of Simple and Complex cells produce cells that are sensitive to low order combinations of features and at the same time a degree of translational and scale invariance. Additional layers can be added after C2 in order to arrive at more complicated features and larger receptive field sizes. . . 64
5.3 Sigma Pi connections gate the connections between the C2 layer and the M-Layer. Each cell in the M-layer is connected to equivalent cells in the C2 layers. When an afferent cell fires the input is communicated to the cell in the M-Layer that is consistent with that assignment of principal axis. 65
5.4 The architecture of the F-Layer for determination of the principal axis. The red connections are in a mutually inhibitory connections. The neurons in the hidden layer pool inputs from a similarly tuned neurons in the C1 layer. They are connected to the output layer to produce an estimate of the principal axis. The output layer is what produces the gated sigma-pi connections between the C2 layer and the M-layer. . . 68

A.1 Average Curvature Estimation Error 85

List of Tables

3.1 Max accumulator values for matching of the two contours in figure 3.4 against their own models and that of the other contour. 28

4.1 Percentage match for rotation through 135,225 and 315 degrees. . . . 39
4.2 Table shows number of matches above the threshold (columns) for different models (rows) when the 1100 contours of the SQUID data base were matched against the models. 51

Chapter 1

Introduction

"Nothing can be believed but what one sees, or has from an eye witness." This quote by Thomas Jefferson underlines the significance of our ability to process visual information. Yet for something that humans rely on so explicitly to navigate and function in their environment, research is still far from a comprehensive (and consistent) explanation of how people see. An activity considered as trivial by people, has been notoriously difficult to implement on digital computers. The field of Computer Vision has been constantly inspired by human (and animal) vision. From the work of David Hubel and Torsten Wiesel [23] to more recent advances in the understanding of the functioning of higher visual areas, human vision has contributed to the design of better algorithms for analysis of digital images. Object recognition is still by far an unsolved problem, and shape is only one of the features of an object. It however is a feature that contributes significantly to our perception of the object. An algorithm that performs well in recognizing shapes is expected to perform equally well in recognizing and categorizing objects [15].

The objective of this thesis is to develop an algorithm for shape representation and matching. This algorithm is inspired from the working of the higher visual areas of primates. The algorithm in its original form was proposed by Dr.Welsey **S**nyder and later developed by Karthik **K**rish and the author of this thesis (**S**). The shape modeling and matching process developed in this thesis is consequently referred to as the **SKS** algorithm. Significant insights into human visual processing were gained through a survey of the latest neuroscience literature. This understanding of the functioning of the visual cortex was applied in the development of a neural network architecture to implement the SKS algorithm.

This thesis is organized in five chapters.

In the second chapter, a background for this thesis is established. This enables a presentation of this work in the context of previous efforts in shape recognition. Research in shape recognition in general is first reviewed. A discussion of the structural and view based approaches to shape representation is then presented. This is followed by a discussion of the Hough Transform [20] and its application to encoding shape [1]. The Generalized Hough Transform is similar in philosophy to the SKS algorithm and hence merits discussion.

In chapter 3, the SKS algorithm is discussed. The SKS algorithm is a boundary based, object centered shape representation method. It utilizes global characteristics of a shape to describe its local characteristics in an invariant fashion. In particular two global characteristics, the geometric center of a shape and the principal axis are utilized. These two shape characteristics are functions of the boundary points of the shape. The principal axis defines the abscissa of the reference frame attached to the object and the geometric center defines the origin of this frame. Once these have been defined a feature vector is computed for each point on the object boundary. The feature vector, $\mathbf{v}$, contains the distance of the point from the origin, the angular position relative to the reference frame and the curvature of the local object boundary. Curvature is a good feature to characterize local shape because it is invariant to translation and rotation (at least in the absence of noise). There are several techniques for estimating curvature of a digital contour [41]. Most methods in literature are derivative based. However, in this algorithm, Digital Straight Segments (DSS) are used to determine curvature[8]. The computed feature vectors populate the model space ($\kappa(curvature) \times d(distance) \times \theta(angular\ position)$). The representation of different shapes in model space is expected to be unique. As the shapes become more different their representations vary. However, the model still captures similarities between shapes.

For the process of matching a Hough Transform [1] like technique, of evidence accumulation, is used. The feature vectors for a target shape are computed and compared with that of the models. An accumulator is utilized to determine the similarity between two shapes. A match is determined by presence or absence of a peak in the accumulator. It was observed that a shape that is nearly identical to another shape produces a sharp peak in the accumulator (at or close to the geometric center). The value of the peak is used as a measure of similarity. The more dissimilar the shapes, the more diffused the accumulator looks.

Claims of invariance to similarity transforms and robustness to noise made in Chapter 3 are verified in a series of experiments in Chapter 4. Two data sets are utilized for the experiments. The first is a data set of twelve tank contours and the second is a data set of contours of marine animals (**SQUID**). Robustness to similarity transforms is verified on the **tank** data set. The sensitivity of the algorithm to boundary jitter and smoothing is determined in the subsequent section. Both the **tank** and **SQUID** data sets are used in these experiments. Some of the parameters that affect the performance of the algorithm (invariance and robustness) are validated with experiment. These parameters affect the ability of the algorithm to generalize. This ability to generalize between shapes is tested in the final section. The algorithm is used to identify similar shapes in the 1100 shape **SQUID** database.

One of the primary objectives of the larger research project undertaken at the Image Analysis Laboratories is to explain how humans recognize shapes. In Chapter 5, a neural network architecture that implements the SKS algorithm is proposed. The network draws from the work of Poggio et al. [32][33][36][7][35]and Rolls and Deco[34]. This neural network is designed to implement a hierarchical feature matching system as suggested by Connor in [10]. The network maps a contour onto a model space. The model space is used to index a shape by its parts in a framework similar in principle to the Generalized Hough Transform [1]. Scale and translational invariance are properties of the architecture proposed in [32][33][36][7][35] and [34]. The SKS neural network is designed to incorporate invariance to rotation in the plane similar to the SKS algorithm. For this purpose a layer that computes the principal axis of a shape is introduced. This layer gates the outputs of the model space (layer) to bring about a representation invariant to the angular position of a feature. This chapter addresses only the design of the neural network. Additionally, an overview of shape processing in the visual cortex and the summary of the *standard model* [32][33][36][7][35] and *VisNet* [34] is presented.

The thesis concludes with a summary of the algorithm and a discussion in the context of the criteria for shape representation as recommended by Marr [28]. Additionally, some suggested directions of future research are discussed.

Chapter 2

Background

The crux of any computational algorithm for shape recognition is the representation it adopts. This is where various computational theories of shape recognition differ. Representation may be comprehensive where the entire object is utilized to represent shape or landmark based where only certain prominent features of the shape are utilized. A shape may be represented as a vector of numbers computed from its characteristics. Alternatively, features of the shape can be estimated and their spatial relationships can be encoded [42]. What is important though is that the representation be compact and unique. Representation also brings with it a trade off between specificity and invariance [35]. A representation of shape must be unique enough to distinguish between dissimilar shapes, while simultaneously retaining the ability to identify similar ones. The ability to generalize between shapes aids in categorization. Marr in [28] identifies the following essential criteria of a shape representation paradigm:

- **accessibility**: This criterion reflects on how much computation is required to compute the description, and if any added computation is required, to interpret it. That is, it signifies how readily available shape information is from its representation.

- **scope**: A good paradigm should be able to represent a large variety of shapes.
- **uniqueness**: The representation for a shape should be unique enough for it to be able to distinguished from other shapes. Two shapes with identical characteristics should map to the same shape model.
- **stability**: The representation should not change drastically with small changes to shape. In other words the change in representation should be related to the degree of shape deformation.
- **sensitivity**: The representation should be able to distinguish between similar shapes. It should be sensitive to small variations in shape.

These criteria will be utilized later in the thesis to evaluate the SKS algorithm.

2.1 2D Shape Representation Paradigms

Various paradigms of shape recognition are described and compared in a recent paper by Zhang and Lu in [42]. In this paper the authors identify two distinct approaches to shape representation. The first is by the shape contour and the second by the region described by the shape. Both cases are further divided into structural and global approaches. In global methods of shape representation, the shape as a whole is considered to generate a vector or feature that is then utilized to distinguish between shapes. Structural methods utilize parts of the shape to compute features and index these features to arrive at a shape representation. There is a significant difference between the boundary-based and region-based paradigms. In boundary-based methods boundary primitives like curvature or tangent angle are utilized. Other properties, such as the direction of the principal axis, or ratio of the major and minor axes, can also be computed. Some of the boundary-based shape recognition methods include: correspondence-based methods (like Hausdorff distance[5]), boundary signatures, boundary moments, spectral transforms, chain code, curve decomposition and syntactic analysis [42]. Region-based methods include: geometric/algebraic/orthogonal moments [21], generic Fourier descriptors, medial axis and shape matrix methods [42]. In theirpaper, Zhang and Lu recommend region-based methods over boundary-based methods since they appear to be more robust to noise. Additionally, they also recommend moments- based descriptors and general Fourier descriptors as good shape representations. However, no quantitative tests that measure performance of all these methods on a standard database are done.

The SKS algorithm is a boundary-based shape representation algorithm. In the chapters ahead the robustness to contour jitter and blurring will be established. Low computational complexity and compactness were cited in [42] as important characteristics of a shape representation scheme. Though no quantitative measures of the two are given, the SKS algorithm, in general, after the speed-ups (section 3.4) performed equivalently to the implementation of the Hu moments [21] classifier and faster than the Hausdorff distance [5] classifier.

2.2 Computational Models of Object Recognition

There are two major approaches to explaining object recognition in humans. The first approach draws from the work of Marr [28] and was further developed by Biederman in his paper [6]. This approach is the *structural description* [6][15][24] approach to shape representation. The structural description theory proposes a parts-based representation of an object. An object is described in terms of a set of shape primitives (called geons) which are derived from the 2D projection of the object onto the retina. These geons coupled with information about their spatial relationships form the crux of the representation. These geons are identified by their non-accidental properties (NAP) [6] and these properties do not vary with translation, scale, left-right inversion and depth rotation to a certain degree [24]. Kayaert, Biederman and Vogel in [25] study the importance of non-accidental properties in representing shapes. They show that cells in the inferotemporal cortex of monkeys are more sensitive (on an average) to changes in NAP than to changes in metric properties (e.g. size, degree of curvature). They argue that this neurological evidence supports Biederman's Geon Structural Description theory [6]. The invariance in representation is largely brought about by the fact that the description is simplified by the limited alphabet and that spatial relationships do not vary with the above mentioned transforms. However, structural descriptions as such have been shown to be sensitive to how image segmentation is done[24]. Using the same alphabet, it is possible to describe an object in different ways. Additionally, generating structural descriptions from 2D images has been found to be difficult [24].

An alternative approach arose in terms of the *view-based description* of objects [39][32][15][24]. The key idea in this approach is that "object representations encode visual information as it appears to the observer from a specific vantage point."[39] The representation of shape arises from the features that are apparent to the observer from a particular viewpoint. The features adopted are simple when compared to the geons in the structural description approach. Since it is computationally prohibitive to store each and every view of an object, matching is done by interpolating between stored viewpoints [39][24]. The idea is to represent the spatial arrangement of shape features relative to a reference (relative to the view) coordinate frame. View-based theories offer a strong explanation of how object recognition is done. They however have their shortcomings. Going from view based representations of single objects to representation of categories is difficult [39]. Additionally, view based representations speak very little about how two objects are similar or how they differ. The representation is holistic, therefore it fails to identify spatial relationships between individual features [24].

Hummel in [24] summarizes the differences between these approaches as being that of the complexity of representation vs. the complexity of matching. Representation is simpler in the view-based approach and the matching process compensates for this simplicity through interpolations and operations like alignment to match two shapes. In structural descriptions the representation in terms of geons is complex. Inferring three-dimensional volumetric primitives out of two-dimensional retinal-projections involves complicated computation. However, the process of matching is simplified by the fact that once the geons are presented in a relational structure. The comparison between two structures is straightforward and can be implemented computationally [24].

The neural network architecture for the SKS algorithm is developed in chapter 5 keeping the context of this debate in mind. The SKS algorithm leans towards being a view-based theory. Though it is object-centered, it does not qualify as a structural description method. The SKS algorithm incorporates a simple representation in terms of the boundary position (angular and radial) and the feature computed at the boundary. The relative spatial arrangement between features is of prime importance in identifying similar shapes. It also employs techniques to realign a shape to obtain consistent descriptions. The neural architecture developed is also influenced by architectures developed to support view-based theories, like *the standard model*[32][33][36][7][35].

2.3 The Generalized Hough Transform

The approach to shape representation and matching adopted in the SKS algorithm is similar to that of the Generalized Hough Transform (GHT) [1] proposed by Ballard. The Hough Transform (HT)[20][38] was first introduced for the purpose of identifying straight lines. It was extended to match circles and other parametric shapes like ellipses. Ballard identified the possibility of extending evidence accumulation, as proposed by the HT, to representing and matching arbitrary shapes. For each point on the object boundary its vector position relative to the center of gravity is computed. Furthermore, the angle (θ_t) between this vector and the tangent to that point is computed. The angle is quantized into a set of bins. For each angle the position vectors of all the points at which that particular angle was calculated, are stored. This forms the shape representation. In the matching process an accumulator is used. The angle θ_t is computed for all points on the object boundary. For each θ_t

the corresponding position vectors from the model are retrieved. Points in the accumulator, corresponding to these position vectors relative to the point on the object boundary, are incremented. In case the object being matched is similar to the model shape a peak is expected in the accumulator.

Translational invariance is achieved by referencing all points relative to the geometric center of the shape. Rotational and scale invariance are achieved by utilizing a accumulator that compensates for similarity transforms. In other words computation is done across a fixed number of scales and rotations by assigning accumulators for each of these. It is a straightforward observation that the computational complexity of this algorithm is high. The accumulation of evidence across different orientations and scale further compound this complexity. The SKS algorithm is philosophically similar. However, scale and orientation invariance are achieved in a different manner. The possibility of graded tuning to features enables greater invariance and robustness to noise and blurring. However, the principle of evidence accumulation still holds. The computational complexity of the SKS algorithm is also significantly lower than the GHT.

Basak and Das in [3] discuss a neural network implementation of the Hough Transform to detect parametric shapes in images. In an earlier paper, Basak and Pal [4] describe a neural network architecture to detect arbitrary shapes utilizing a Hough like approach. They incorporate two streams of information processing for feature detection and pose estimation. Their implementation was however, restricted to using a limited set of features (corner detectors) and polygonal shapes.

In this chapter a summary of the literature related to shape and object recognition was presented. The different approaches to 2D shape representation and matching were discussed. The SKS algorithm was compared against some of these algorithms. The results obtained show promise in this approach. A review of the theories in human object recognition was also presented. The neural architecture described in chapter 5 is developed in the context of this debate. Next, a description of the SKS computational algorithm is presented.

Chapter 3

The Computational Algorithm

The shape recognition algorithm discussed in this thesis adopts an object centered, boundary-based representation of the object contour. The algorithm involves describing the shape in terms of object primitives encoded in terms of their angular position and distance, relative to a reference frame. Global and local features are used to characterize the shape. In this approach, global features are utilized to overcome the variation in shape description due to translation, rotation in the plane and scale. Local features describe the geometric properties of the shape contour in the neighborhood of the point being characterized. In this implementation of the algorithm, the local feature used is curvature. Feldman and Singh in [16] discuss the significance of curvature in conveying information regarding shape. They show that a significant amount of shape information is stored at points of high curvature (convex and concave). This algorithm, however, utilizes the entire range of curvatures to encode shape. Curvature is not necessarily the only feature that can be used with the algorithm. For all practical purposes, any other suitable local feature (for example the tangent angle as described in [1]) can also be utilized. The proposed algorithm is for 2D contours, which are Jordan curves, i.e. they are simple (do not cross themselves) and closed. With minor modifications, it can also be extended to open contours. One of the primary assumptions made in this algorithm is that segmentation of the object contour has already been completed. No assumptions are made regarding the quality of segmentation. The input to the algorithm is a set of 8-connected points that describe the object of interest. The algorithm proposed has two distinct modules. One for shape representation (model generation) and the second for shape matching.

3.1 Model of Shape Representation

Before we arrive at a model for representing the object contour, some concepts essential to the implementation of the algorithm are discussed. As mentioned above, global characteristics of the shape are utilized to attain invariance to translation, rotation and scale. *Translation invariance* is attained by defining all object primitives relative to the geometric center of the contour. The geometric center is the mean of the (x,y) coordinates of the shape. From here on the geometric center will be denoted by ($\bar{x}$,$\bar{y}$). This choice of the geometric center makes our representation *object-centric* as opposed to the *view-centric* approach (object primitives are described relative to a fixed frame).

The object center does not in itself define a frame of reference. It determines only the origin of this frame. A frame of reference is essential in order to represent angular positions of features on the object boundary. It is also desirable that the angular ordering of features does not change with the rotation of the object contour (in the plane).

Having a fixed frame (e.g. parallel to the scene coordinates) results in very little rotational invariance. Therefore another characteristic of the object contour that would make the representation independent of rotation in plane is sought. The *principal axis* of the object is one such feature. Using *principal components analysis*(PCA) [38][14] the major axis of the object is determined. Using the major axis, a right handed coordinate system is set up (as shown in the figure 3.1). Setting up a right handed coordinate system is important as this makes the shape representation consistent. This description of the object contour relative to the right handed system derived from the principal axis makes the representation *invariant to rotation* in the

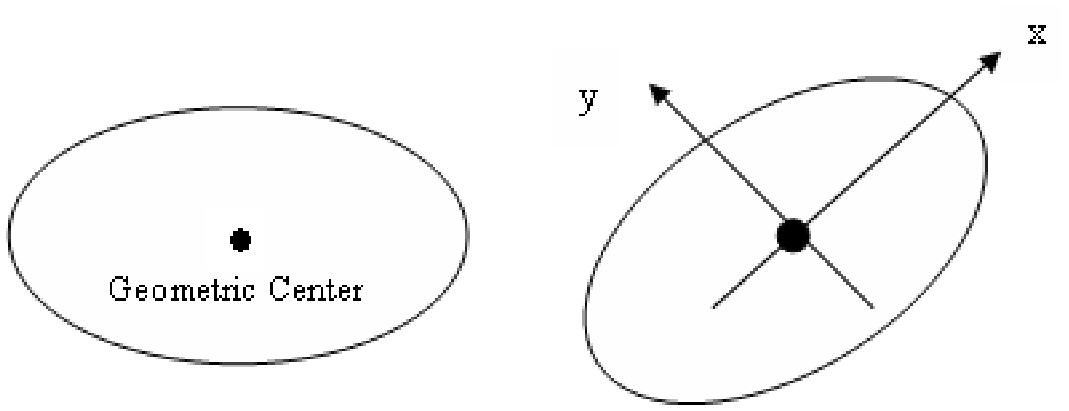

Figure 3.1: Geometric Center and the principal axis of the rotated image.

plane (up to a symmetry).

Shape representation is also sensitive to object size. To make this representation *invariant to scale*, the object is resized to a predetermined size. In the model building and matching process all contours are scaled so that the distance of the most distant point from the geometric center of the object is uniform across all objects. The convex hull of the object can also be used in re-scaling. However, here the maximum distance is used.

These global transforms (and features) ensure our representation is invariant to any geometric transform in the plane. The shape model is built on the object contour, once all these transforms have been accounted for.

3.1.1 Description of the Algorithm

The input to the algorithm is a set of 8-connected contour points. The first step in the algorithm is to make the contour size invariant and fix a reference coordinate system. These operations involve the global features of the object and are significant because they ensure consistent object descriptions. The operations can be listed as:

1. Calculate the geometric center ($\bar{x}$,$\bar{y}$), the mean of the x and y coordinates of the points on the contour.
2. Shift the contour such that the geometric center is at (0,0).
3. Determine the principal axis by PCA. (Since the principal axis is not directed, one direction is randomly chosen.)
4. Setup a right handed coordinate system using the principal axis as the abscissa.
5. Transform contour points to this frame of reference.
6. Determine the most distant point on the contour and re-scale all contour points so that their new distance from the origin is $(\frac{distance_{original}}{distance_{max}}) * scale_{model}$

Here $scale_{model}$ is a global parameter that determines the uniform size (in pixels) to which all objects are scaled.

With the global characteristics ensuring invariance to similarity transforms, local features are encoded with respect to their position vector and their feature values.

Shape Characterization

The approach adopted to shape characterization is comprehensive. Shape information at all contour points is utilized to determine the shape characteristics. Fundamentally, the algorithm characterizes a contour point by its distance from the origin (d_{point}), the angle (θ_{point}) with respect to the abscissa of the reference frame and curvature at that point(κ_{point}). In essence, the direction vector and the curvature at a point encode shape. This can be viewed as encoding “what” is the feature (the curvature value) and “where” it is. Such processing is analogous to the functioning of the visual cortex in which there are two distinct pathways for “what” (ventral) and "where" (dorsal) information [22][34]. It is also noteworthy that research [29][30][31] in visual neuroscience shows this approach to shape representation is adopted by primate cortices (in the higher visual areas, V4-IT). Connor in [10] discusses a parts based representation of boundary fragments and documents that response of neurons in the visual cortex of a macaque. These neurons in the V4 show sensitivity to the curvature, the radial position and the context (connectivity to other parts).

Model Space and Quantization

The model of a contour can be viewed as a scatter of points (κ, d, θ) in a 3-dimensional space defined by the curvature, distance, and angular position. A continuous representation of all possible values of (κ, d, θ) is computationally prohibitive. This is overcome by quantizing each "feature" in $N_{feature}$ bins. This not only ensures computation is tractable, it also ensures the model is not sensitive to small variations in measurement. In experiments conducted, quantizing the model space into 100 discrete values for distance, 180 values for angular position and 100 values for curvature, were found to generalize well. The Euclidian distance of a contour point is rescaled to account for object size and then quantized. The angular position is invariant to size and is estimated after the points have been transformed to the new coordinate system.

Connor and Pasupathy in [30][31] have shown that there are cells in the V4 area of the visual cortex that are sensitive to the curvature of a contour. Curvature is a good feature to characterize shapes because it is invariant to translation and rotation. It is sensitive to object size but this can be accounted for by scaling it by $distance_{max}$. A formal definition of continuous curvature is presented. Curvature a of curve at a point is defined as the directional change of the tangent with arc length at that point. Worring and Smeulders in [41] and Coeurjolly in [8] give some ways to think of curvature as shown below:

Definition 1 Consider a curve $(x(s))$ parameterized by arc length s. One way of defining curvature is as a second derivative:

$$k(s) = x''(s) \tag{3.1}$$

where $x''(s)$ is the second derivative of $x(s)$.

Definition 2 Curvature can also be defined as the rate of change of tangent angle:

$$k(s) = \theta'(s) \tag{3.2}$$

where $\theta(s)$ is the angle made by the tangent at $x(s)$ with a given axis.

Definition 3 And finally, the curvature at a point can be estimated by fitting a (osculating) circle to its neighborhood and utilizing the inverse of the radius as an estimate:

$$k(s) = \frac{1}{r(s)} \tag{3.3}$$

where $r(s)$ is the radius of the osculating circle at $x(s)$.

These definitions of curvature in equations 3.1-3.3 are equivalent.

In [8], a method is proposed where the curvature estimation is done in a purely discrete way using osculating circles ,which avoids any parameters based on the nature of the data. In this implementation, the Digital Straight Segments (DSS) algorithm [8][27][12][13] was used to estimate curvature. The DSS algorithm was tested with other curvature estimation algorithms by a colleague in the Image Analysis Laboratory. DSS is independent of the data set and is very robust to noise (as opposed to derivative based methods). It is also highly parallel. Analysis of this algorithm is present in the Appendix.

The output of the curvature estimation algorithm varies from $-\infty$ to ∞. These values do not have biological analogues. Additionally, research [30] has shown that it is reasonable to map the outputs of the algorithm to values between -1 and 1. The

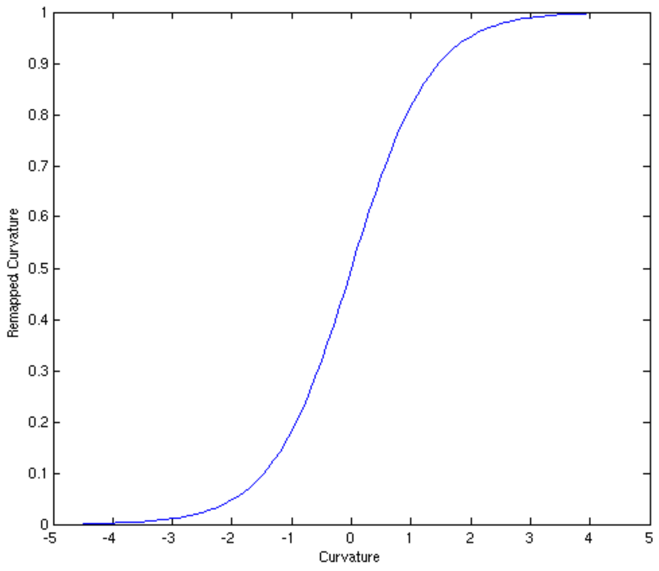

Figure 3.2: Plot of the function that remaps the curvature calculated by the DSS algorithm to values between 0 and 1.

assumption made here is that beyond a particular absolute value (in our experiments 4) all high curvatures are the same to the system. However, in the range -4 to 4 the curvature is mapped between 0 and 1 (since the actual value does not matter it is not necessary to map it to -1 to 1). A sigmoid is used to implement the (non-linear) remapping as suggested by [30]. The idea is to emphasize finer curvature differences as the absolute values get smaller and to treat larger absolute (concave and convex) curvatures less rigorously.

The function for rescaling is given by:

$$\kappa_r = \frac{1}{1 + e^{-\alpha\kappa}} \tag{3.4}$$

After mapping the curvatures to values between 0 and 1, the curvatures are quantized it into $N_{curvature}$ bins. The curvature values are rescaled to with object size. Quantization is done after rescaling.

Model Generation

The model is generated by populating the feature space by estimating the (κ, d, θ) tuple at each point. The three features are scaled and quantized. The bin represented by the quantized values is then incremented. This is repeated for all the points in the shape. Owing to the fact that the model determined by each and every point, we normalize the accumulated model by the number of points to obtain a percentage representation. The model can now be viewed as a density function. It represents the fraction of points that satisfy a particular combination of features.

The density function thus can be represented as:

$$M(\kappa, d, \theta) = \frac{1}{P}\int_S \delta(|\kappa - \kappa(s)|, |d - d(s)|, |\theta - \theta(s)|)dS \tag{3.5}$$

Here, $\delta(x, y, z)$ is the Dirac (delta) function and the integral is over the contour S. P is the length of the contour. However, our representation is a discrete version. Owing to this there is a possibility of multiple points on the contour incrementing the same bin in the model space. This is a disadvantage of quantization. Some of the shape information maybe lost and the representation may not be truly invertible. Quantization is a parameter in the algorithm and the resolution of the model space can be set to make the representation as unique as possible. The resolution of the model space determines the ability of the algorithm to generalize and also discriminate between similar shapes. At small resolutions the generalization will be good but this comes at the cost of discrimination. On the other hand, at high resolutions the generalization gets progressively poorer, but discrimination improves drastically. Higher resolutions also imply larger computational complexity (both space and time). The resolution (100 x 100 x 180) was arrived at after experimenting.

3.2 Shape Matching

The matching philosophy is similar to that of the Generalized Hough Transform[1]. An accumulator is used to calculate increments in the matching process. The model discussed in section 3.1 can be viewed as a look up table (similar to the r-table in [1]) . The basic principle is the same. However, some of the strategies adopted in the matching process reduce the computational complexity when compared with the Generalized Hough Transform. Additionally, speed-ups for the matching process are also discussed in Section 3.4. In this section, the matching algorithm is discussed. First the basic algorithm is described in Section 3.2.2. There some of the parameters key to the implementation of the matching algorithm are also analyzed. One of the major problems associated with the Generalized Hough Transform is the computational complexity [1]. In Section 3.4, methods to speed-up processing and the trade-offs involved are discussed.

As in the case of model generation no assumptions are made regarding the quality of segmentation. In Chapter 4, noise analysis is done to simulate poor segmentation. The methodology developed involves building models of the shape "classes" first. The matching process matches the input against the models of each of the shape classes. The input to the matching process, as in the case of model generation, is a set of 8-connected points. The DSS algorithm, discussed in section 3.1.1, generates curvatures of all the points in the shape.

3.2.1 Preprocessing of Shape Contour

As in the case of model generation, the first steps in the matching process involve:

- Determination of the center of gravity.
- Determination of the principal axis and assignment of a right handed coordinate system.
- Transformation of the shape points to the new coordinate system.
- Normalization of object size (rescaling distances and curvatures)

As mentioned in Section 3.1, the principal axis is not directed. In the process of model generation, one of the directions is chosen and a right handed coordinate system is set up. This is important because all the angular positions of the contour points are calculated relative to this coordinate system.

3.2.2 Description of Matching Algorithm

The preprocessing step outputs a set of points that have been normalized for object size, translation and rotation in the plane (using principal axis description). The curvatures of the shape are also rescaled to adjust for object size. The principal approach of the matching algorithm is to calculate the probability that a point in the accumulator is the geometric center of the object. For this an accumulator A is set up. The accumulator (A) has $2 \times \sqrt{2} \times scale_{model}$ rows and columns (rounded off to the nearest integer). This is to ensure that the rescaled object is centered in the accumulator and fits completely in it. The algorithm calculates the total increment for each position in the accumulator. The logic is that since the shape model is built using the geometric center as the reference, if a position in the accumulator is such that the shape description (of the target shape) at that position is consistent with the description (of the model shape) relative to its geometric center, a large number of increments would occur at that position. A clear peak would be expected at this position. This peak indicates a match.

The input to the algorithm is a set of ordered pairs of contour points $C(\mathbf{x})$. Now for each point in on the contour a feature vector $\mathbf{v}$ is calculated. This feature vector is the (κ, d, θ) tuple estimated at this point. For a given model $\mathbf{M}$ the accumulator increment is calculated as $f(\mathbf{v}, \mathbf{M})$, where f is some increment function (it can be uniform or graded). The function f measures the consistency of the tuple $\mathbf{v}$ with the model $\mathbf{M}$. It is noted that $\mathbf{v}$ is estimated relative to (x_i, y_i) which corresponds to a row and column of the accumulator. Therefore, the net increment at a position (x_i, y_i) in the accumulator is calculated by summing f over all points of the contour:

$$A(x_i, y_i, \mathbf{M}) = \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{v}_n(x_i, y_i), \mathbf{M}) \tag{3.6}$$

Here N is the number of points in the contour being matched. The normalization ensures all values in the accumulator are between 0 and 1. In addition to this, it brings about an invariance to perimeter. Since rescaling the object size does not imply fewer points there is a need to ensure that a contour with a larger number of points does not imply a better match. This is achieved by the normalization.

In equation 3.6, the clear dependency of the feature vector **v** on the accumulator position is also shown. Distance and angular position calculations are sensitive to the reference point whereas curvature is not. Overall, the feature vector, **v**, is a function of the accumulator position. Equation 3.6 is a discrete implementation of the accumulator. As the resolution of the sampling increases, these accumulator increments can be expressed as integrals. The accumulator increment, in the basic implementation of the algorithm, is calculated for the entire accumulator. This is an inefficient practice and alternatives are suggested in section 3.4. The result of the matching process is the largest value in the accumulator. If the shape being matched is "similar enough", there is usually a sharp peak in the accumulator. In case there is very little similarity, (for example a square shape matched against a circle) the accumulator is blurred and there is no peak that clearly stands out.

Given a set of models:

$$\mathcal{M} = \{M_1, M_2, ..., M_K\} \tag{3.7}$$

and a set of matching scores:

$$S_k = \max\{(A(M_k))\} \tag{3.8}$$

where $k = \{1, 2, .., K\}$.

The best matching class and the best score for the corresponding match are defined as:

$$\begin{aligned} S_{best} &= \max\{S_k\} && (3.9) \\ M_{best} &= \arg\max\{S_k\} && (3.10) \end{aligned}$$

Disambiguation of Object Description

One of the problems that arises from utilizing the *principal axis* to determine angular positions of features is that the *principal axis* is not directed. As mentioned in the algorithm description, one of the two possible directions is chosen, at random, as the positive half of the abscissa. Figure 3.3 illustrates the ambiguities that this method brings. For the given shape, there are two possible right handed coordinate systems that can be set up (relative to the principal axis). They are shown in 3.3(a) and (b). It can be clearly seen that for a point the angular position is dependent on the decision of which direction of the principal axis to choose. Some experiments were carried out to incorporate a bias so that the algorithm always selects a particular direction. The choice of a fool-proof characteristic was complicated by the fact that the decision was being made relative to the center of gravity and the principal axis. It is possible to use higher order characteristics like third order moments to determine a particular direction [21]. However, these methods are not known to be robust to noise. Furthermore, the drastic change in shape descriptions that would arise on swapping the directions would imply the algorithm does not fit Marr's [28] requirement for stability (Chapter 2). A simpler (less efficient) solution is proposed. Two accumulators A_1 and A_2 are used for the two possible shape descriptions (figures 3.3 (a) and (b)). The algorithm requires less than twice the amount of computation previously needed. This is because most of the calculations are the same except one additional look up in the model. Since two accumulator values are present the final output of the matching process is decided as:

$$A_{out} = \max(\max(A_1), \max(A_2))$$

The accumulator with the larger peak is chosen as one being consistent with the object description in the model. This modifies equation 3.8 to:

$$S_k = \max\{\max(A_1(M_k)), \max(A_2(M_k))\} \tag{3.11}$$

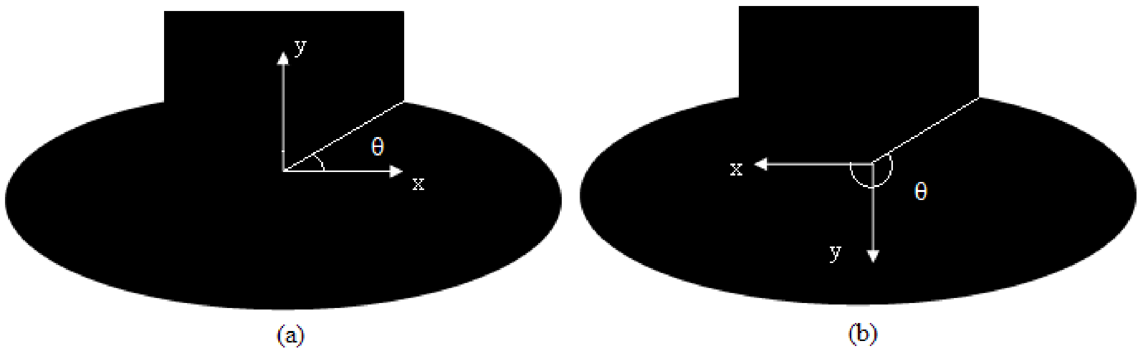

Figure 3.3: As the principal axis by nature is not directed we have to select one of the two possible directions. This results in ambiguity in angle descriptions. The two possibilities are shown in (a) and (b). It is clear that the angles calculated for the same point is dependent on how the rhs system is set up.

Increment Functions

The increment function f needs to be defined. One of the simplest formulations of this function is as a look up table:

$$f(\mathbf{v}, \mathbf{M}) = \begin{cases} 1 & \mathbf{M}(\mathbf{v}) > 0 \\ 0 & Otherwise \end{cases} \tag{3.12}$$

The function described here is similar to a delta function. If a vector $\mathbf{v}$ is present in the model $\mathbf{M}$ (indicated by a non-zero value) then the function returns 1. Experiments were performed using this function. This function is expected to generalize very poorly and it does. The reasoning is straightforward. The function matches only a shape description that is exactly the same as the model. However, in most cases, there are some small variations in the estimation of $\mathbf{v}$, especially, the distance d and curvature κ. Most contours in the experiments are made up of 1000-10000 points. The model space has $100 \times 100 \times 180$ (2×10^6) bins. Therefore the model is sparse. Using this increment, even a small offset would imply a very low match score. Shape "similarity" is not reflected in the scores obtained.

To account for the spare model, the following function is proposed:

$$f(\mathbf{v}, \mathbf{M}) = \begin{cases} 1 & \mathbf{M}(\mathbf{v} \pm \delta) > 0 \\ 0 & \quad Otherwise \end{cases} \tag{3.13}$$

Here δ defines a neighborhood around $\mathbf{v}$. This function (equation 3.13) does a better job of generalization. It allows (pre-determined) variations in estimation of the feature vector. Generalization comes at the cost of discriminability. The ability of a model to generalize is desirable however the ability to discriminate is also important. If there are small changes in the shape resulting from digitization noise, it is important that this be apparent in the matching score. This formulation of the function does not make these differences apparent. Hence, an increment function is sought that not only emphasizes the shape information present in the model but also allows for some variability in matching.

For this purpose an increment function with a graded response is proposed:

$$f(\mathbf{v}, \mathbf{M}) = G_{\sigma_\kappa, \sigma_d, \sigma_\theta}(\mathbf{v} - g(\mathbf{v})) \tag{3.14}$$

where $G_{\sigma_\kappa,\sigma_d,\sigma_\theta}$ is a zero mean, three-dimensional Gaussian function (without the normalization constant). The function $g(\mathbf{v})$ returns the nearest neighbor (measured using the Mahanalobis distance [14]) of the feature vector $\mathbf{v}$ in the model. That is it returns the nearest (again in the Mahanalobis definition) non-zero point in the model. The Gaussian function in turn rewards (with outputs closer to 1) target shapes that are are "more similar" to the model. This formulation of the increment function allows generalization while not completely compromising on discrimination. σ_κ, σ_d and σ_θ still need to be determined and these parameters can be set depending on desired generalizability of the model. The increment function of equation 3.14 is what is used in experiments and the results documented in chapter 4.

Model Saturation

A problem that arises owing to the quantization of the model space is the mapping from the contour, $C(x, y)$ to the model M is many-to-one. In the process of matching this may result in some shapes (especially ones with a greater number of points) exhibiting a better match than they should. This situation is exacerbated by the graded increment function used for matching. The problem was more severe in an earlier approach to shape characterization where only distance and curvature were used to characterize shapes. A solution to this problem was engineered by keeping track of the number of times a particular "cell" (a κ, d, θ tuple) in the model is looked up. The argument is that only a specific fraction (ideally $\frac{1}{N}$, where N is the number of points) of the target shape should correspond to any "cell" in the model. For example, suppose the shape from which the model is constructed is made up 1000 points of which 10 points map (or 1%) to a cell of values $(\kappa_1, d_1, \theta_1)$. Take a target shape that is made up of 10,000 points. This shape may be the same shape at a larger scale or a totally different shape. In either case, it is desirable that the same fraction (1%) of the contour points map to the cell with the values $(\kappa_1, d_1, \theta_1)$ and hence increment their respective accumulators. It is easy to see how in some cases even though the shapes are different some contour points may enable false increments of the accumulators. To solve this problem a saturation process is implemented. A copy of the model is maintained and the value at a particular "cell" in the model is decremented every time that cell is looked up in the matching process. The decrement is shown below:

$$\mathbf{M}(\mathbf{v}) = \mathbf{M}(\mathbf{v}) - \frac{1}{N_{target}} \tag{3.15}$$

where N_{target} is the number of points in the target.

Thus, the accumulator increment function as defined in equation 3.14 will return zero as soon as an expected fraction of points have looked up the "cell". This makes the matching process thorough, as only the proportion of points consistent with the model can actually contribute to the match score. However, the computational cost added is severe and it does not work with some of the speed-ups recommended in section 3.4.

3.3 Examples of Model Building and Matching

In this section, the theory developed in the sections 3.1 and 3.2 is applied to build models for the contours in figure 3.4 and then match them against each other. The aim is to run through step-wise the process of model building and matching. The data set from which these contours are drawn is shown in Chapter 4.

For the model building process first the preprocessing is done on the tanks. After being scaled and transformed to a new coordinate system, the contours appear as shown in figure 3.5.

Next the model of each of these tanks is built. The models are shown in figures 3.6 and 3.7. The difference between the tanks can be seen clearly in their shape signatures in model space.

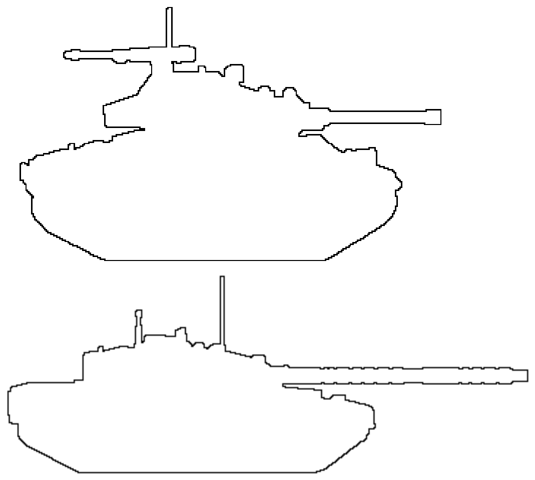

Figure 3.4: Contours of the tanks Chaff (top) and Chief (bottom)

In the matching process, each of the tanks is matched against a model of itself and the other tank. First the accumulators of the various matches are shown in figures 3.8 - 3.9. In figures 3.8(left) (target: Chaff, model: Chaff) and 3.9 (right) (target: Chief, model: Chief) the accumulators show clear peaks around the geometric center where as in figures 3.9 (left) (target: Chief, model: Chaff) and 3.8 (right) (target: Chaff, model: Chief) a large number of points are incremented and there is no clear winner. It should be noted that these accumulators are the “max” accumulators. The other accumulators corresponding to the “wrong” object descriptions are not shown here. For the sake of completeness all the accumulator outputs are documented in table 3.1.

Table 3.1: Max accumulator values for matching of the two contours in figure 3.4 against their own models and that of the other contour.

Model	Target	A_1	A_2
Chaff	Chaff	100	30
Chaff	Chief	50	60
Chief	Chaff	56	45
Chief	Chief	30	100

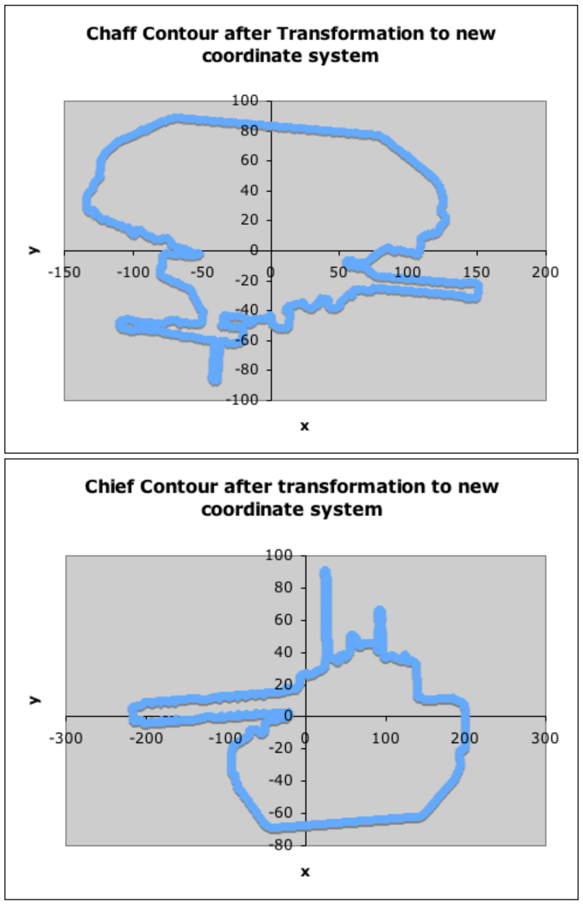

Figure 3.5: The two tank contours after transformation to their new coordinate systems defined by their principal axes.

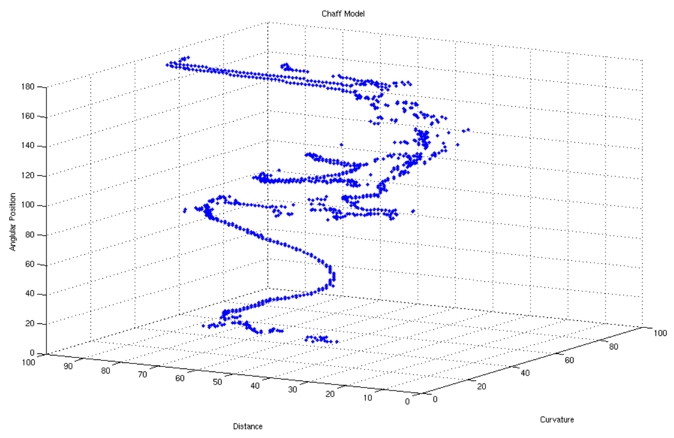

Figure 3.6: Model of the tank Chaff

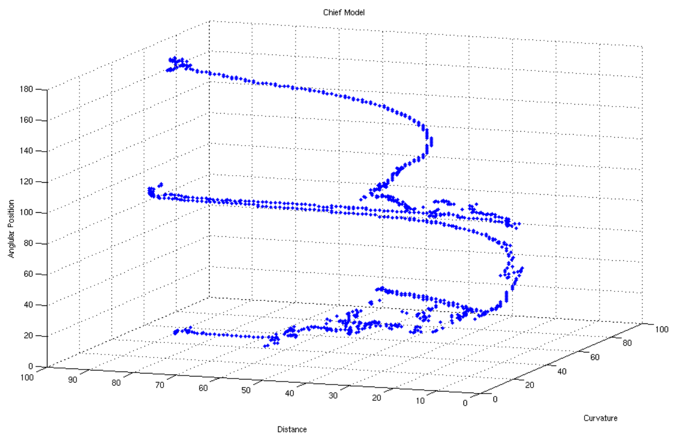

Figure 3.7: Model of the tank Chief

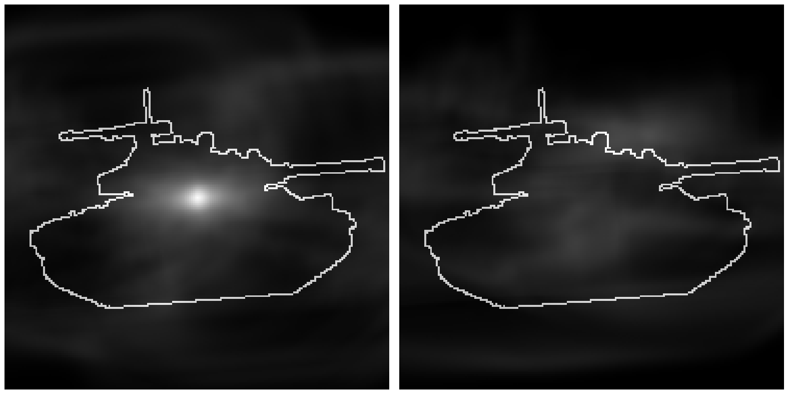

Figure 3.8: Target:Chaff, Model:Chaff(left),Chief(right)

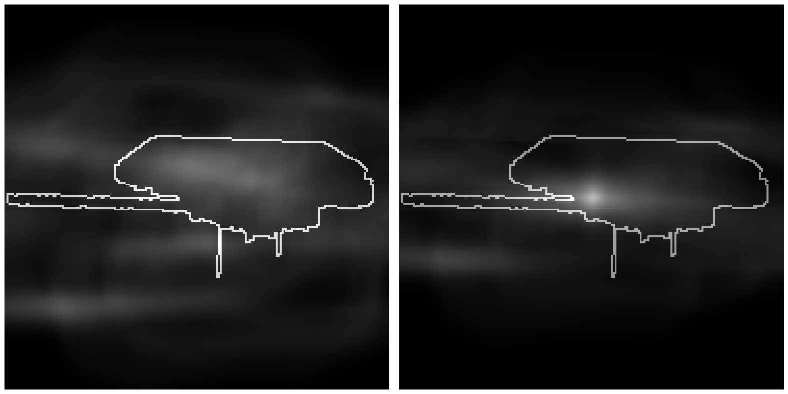

Figure 3.9: Target:Chief, Model:Chaff(left),Chief(right)

3.4 Strategies to Reduce Computational Cost

One of the problems associated with the Generalized Hough Transform and any look up based techniques is the large computational complexity both in terms of space and time. In addition, to bring about properties of invariance to scale and rotation the methods suggested in [1] make the complexity even worse. The algorithm discussed in this chapter tackles the problem of space and time complexity in the following ways:

Principal Axis Approach: In the original formulation of the Generalized Hough Transform the accumulator was expanded to add a θ axis and a s (scale) axis to exhaustively compute all possible configurations of the shape and for all possible sizes. In this algorithm, the use of PCA narrows down the possible angles to two. Additionally, the rescaling of the contour by maximum distance forgoes the use of the scale axis in the accumulator. It is easy to see how this reduces both space and time complexity of the algorithm.

Reducing Accumulator Size: One way to further speed-up the algorithm is to reduce the accumulator size. Since shape description is done using the geometric center, it would be logical to expect the peak of the accumulator to also be present in the vicinity of the geometric center of the target contour (if not at exactly the geometric center). Therefore, as a speed-up we reduced the accumulator size to just a $r \times r$ neighborhood of the geometric center of the target shape. In experiments, r=5 was sufficient.

Precomputation of accumulator increments: During the initial testing of the algorithm, it was observed that model building overall required much lesser time than matching. The matching process was loaded with heavy computation that is largely redundant. Additionally, model building is done once whereas matching is a more frequent operation. Therefore to make the implementation efficient and to improve speed most of the computational load of the matching process was shifted over to the model generation phase. The values for the increment function for different values of (κ, d, θ) for a given model M are computed and stored in advance. This increases the space requirements for the entire algorithm. However, the time speed-ups achieved are worth the space requirements. It should be noted that the dynamic space requirements (during run time) are minimal. The speed-up in matching is largely because the whole matching process is reduced to look ups: order O(n).

Saturation Computation: In section 3.2 "model saturation" is discussed. It was stated that this is an expensive operation both in terms of space and time. For some sacrifice in correctness speed-ups can be achieved by abandoning this operation. It was observed in experiments that the lack of saturation only brought about small changes in the outputs. This operation can be safely abandoned without any loss in accuracy.

These methods resulted in major speedups when it came to run time of the algorithm. The space requirement is larger than the initial formulation but still significantly lesser than that of the Generalized Hough Transform and its variants.

In this chapter an analysis of the SKS algorithm developed in this thesis was presented. The SKS algorithm is, in theory, robust to rotation, translation and zoom. It is also expected to be robust to noise and blur. These assertions are verified in the following chapter.

Chapter 4

Experiments and Results

An algorithm that characterizes shapes and matches shapes against models is described in the previous chapter. It is claimed that the algorithm is invariant to similarity transforms (translation, rotation in the plane and zoom)[1] and robust to noise and blur. In this chapter these claims are experimentally verified. Additionally, some factors were introduced in Chapter 3 that have an effect on the matching performance of the algorithm. The most prominent of these factors is the degree of tolerance of the model (defined in terms of the increment function f). It was postulated that a more tolerant model (larger variance in the matching process) would be able to generalize better. It was also postulated that this generalizability would come at the cost of discrimination. These postulations are verified in this chapter in a set of experiments where the noise is progressively added to the shape contour. The output of the shape matching algorithm is a number that represents how similar the given contour is to the model of a particular shape and in turn to that shape itself. How this number translates as a "shape metric" merits further investigation. In this chapter, a summary of some initial experiments in this investigation is also presented.

[1]A similarity transform corresponds to a rigid body motion in 3-space, a motion which does not change the length of vectors. That is, translation and rotation, but not zoom. However, the definition used here includes zoom. This definition of similarity is related to the concept of similar triangles. Therefore, invariance to zoom is also considered.

4.1 The Data Set

Two data sets were used in the experiments documented in this chapter. The first is a data set of 12 **tank** contours (figure 4.1). The second data set (SQUID database from VSSP Surrey)[2] is a collection of 1100 contours of **marine animals** (fish, eels, sea horse, sting ray, etc.). For the experiments on invariance to similarity transforms the **tank** data set was utilized. Though the initial data set was of 12 contours, additional contours were generated from these contours. As the objective was to test the robustness to similarity transforms the following sets of contours were generated for each tank:

- 9 contours of the tank rotated by $15^o, 30^o, 45^o, 60^o, 75^o, 90^o, 135^o, 225^o$ and 315^o.
- 5 object sizes of the tank scaled to 0.25,0.5,0.75,1.25 and 1.5 times its original size.

A total of 15 contours of each tank at different sizes and rotations constitute the data set. In all there are 180 contours. All the results presented in sections 4.2-4.4 are for this data set. All tank contours were generated using the public domain NIH Image program (developed at the U.S. National Institutes of Health and available on the Internet at http://rsb.info.nih.gov/nih-image/).

The second data set was utilized primarily for the experiments described in section 4.5. It is also used for determining the robustness of the algorithm to contour blur. The data base contains several shapes of marine animals of the same species (with some having similar shapes). Of the data set 31 contours (figure 4.2) were sampled and models were generated. Experiments were conducted to find the contours most similar to these models. The objective was to get a qualitative measure of how well the algorithm is able to identify similar shapes.

[2]http://www.ee.surrey.ac.uk/Research/VSSP/imagedb/demo.html, used with permission.

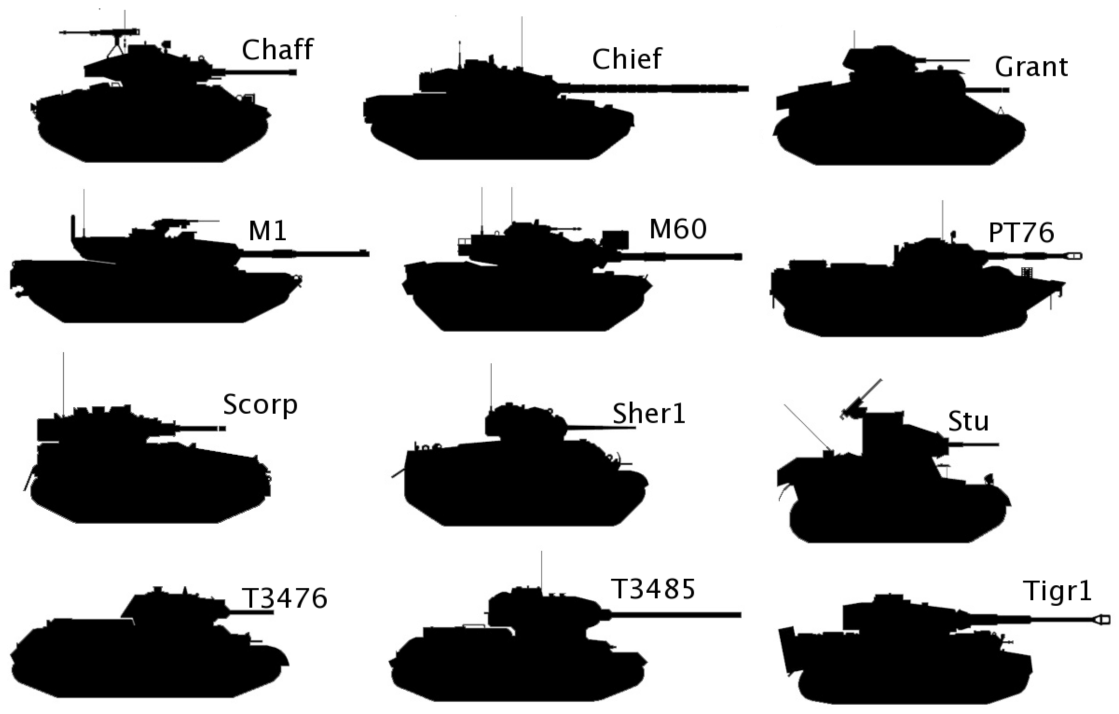

Figure 4.1: The data set of tanks.

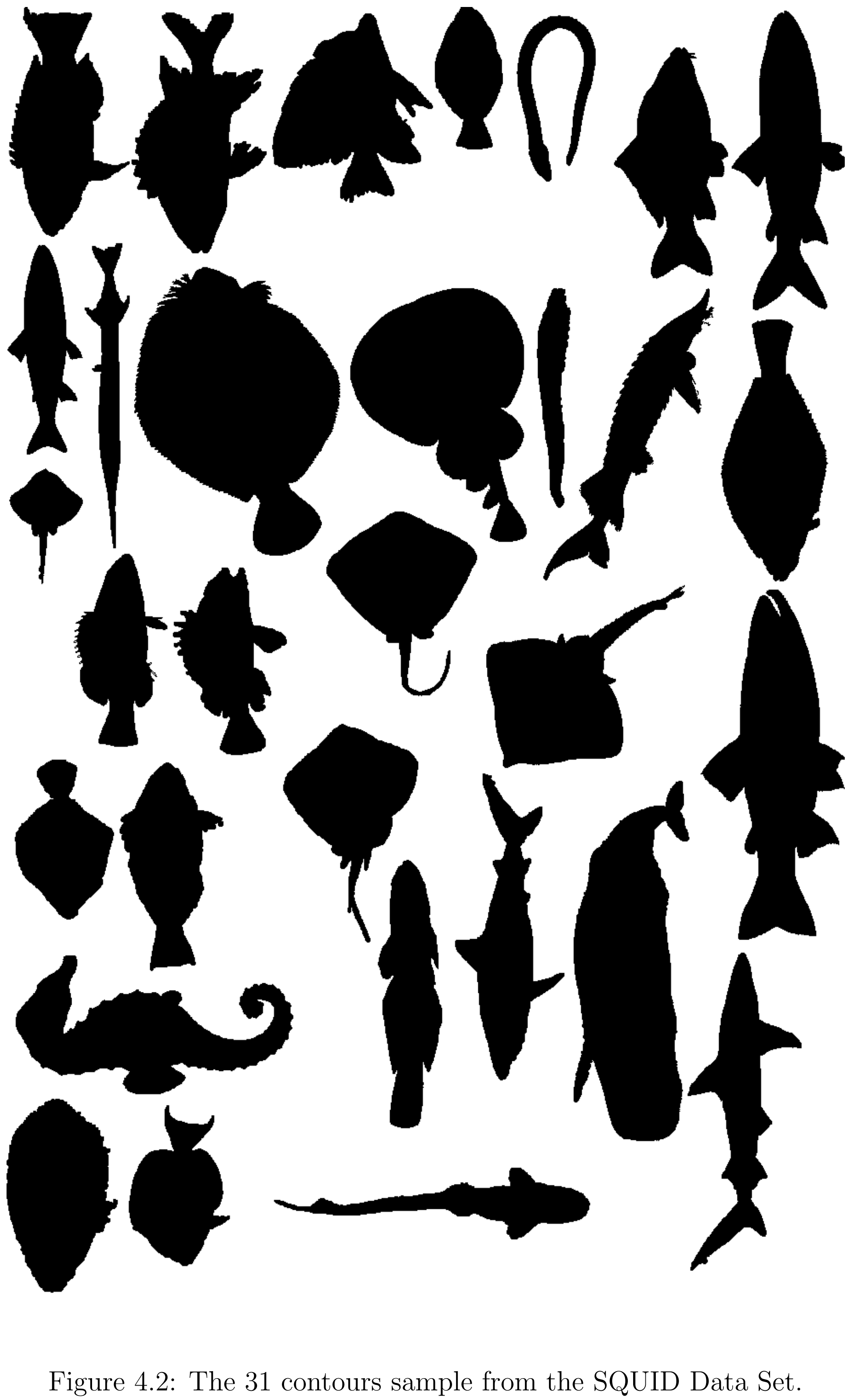

Figure 4.2: The 31 contours sample from the SQUID Data Set.

4.2 Robustness to Similarity Transforms

4.2.1 Experiment on Rotational Invariance

In this experiment, first, the models for each of the 12 tanks were built. Next, contours oriented at 9 different angles were matched against all 12 tank models. No mismatches were observed in the matching process. A plot of the variation of the output of the (max) accumulator with change in orientation (when matched against a model of the unrotated contour of the same tank) is shown in figure 4.3. This plot is done for angles 0-90 at 15 degree intervals. The three remaining orientations ($135^o, 225^o$ and 315^o) are not shown in the figure because intermediate angles were not evaluated. The data for these angles is presented in table 4.1. This was done primarily to test if the PCA results in consistent assignment of right handed coordinate systems.

Figure 4.3 shows the model is robust to rotation in the plane. It is observed that the peak values of the accumulator are not 100%. This however can be attributed to digitization noise that is associated with rotation. When a contour is rotated, owing to the digital nature of the data some artifacts are introduced (as the sampling is in cartesian coordinates and not along arc length). This results in (κ, d, θ) tuples that have little or no correlation with the model. Consequently, the peak values of the accumulator are lower than expected. It is interesting to note that the rotation through 90^o produces the highest accumulator values. It was observed that contours rotated through this angle have, in general, the least number of artifacts because in essence rotation through 90^o involves swapping x and y values of the contour.

4.2.2 Experiment on Scale Invariance

In this experiment the 12 tanks (original silhouettes) were scaled to different shape sizes. The new silhouettes varied in size from a quarter of the original contour

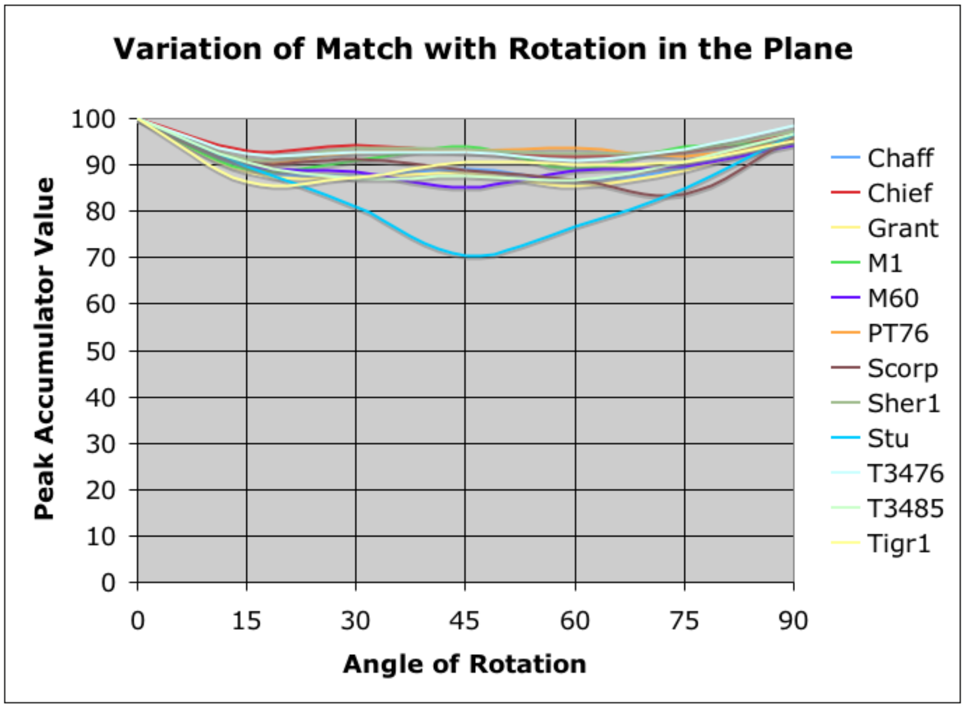

Figure 4.3: Variation of Accumulator Peaks for tanks contours, matched against their own models, with orientation.

Table 4.1: Percentage match for rotation through 135,225 and 315 degrees.

Target	135	225	315
Chaff	91.47	92.60	91.47
Chief	93.51	95.05	92.59
Grant	94.79	95.18	94.85
M1	94.26	94.17	92.84
M60	85.00	86.27	85.87
PT76	91.66	92.57	91.69
Scorp	88.75	88.22	87.95
Sher1	93.69	93.49	93.60
Stu	69.55	69.84	70.35
T3476	91.90	92.89	92.96
T3485	87.09	88.38	88.53
Tigr1	90.47	90.00	90.26

to one and half times that of the original contour. Samples were taken at sizes 0.25,0.5,0.75,1.25 and 1.5 (in addition to the original size). The models of the tanks built in the previous section were used to evaluate "scale invariance". A plot of the peak values of the accumulators of the tanks contours (of different sizes) when matched against their own model is shown in figure 4.4. The different lines represent the 12 different tanks in the data set.

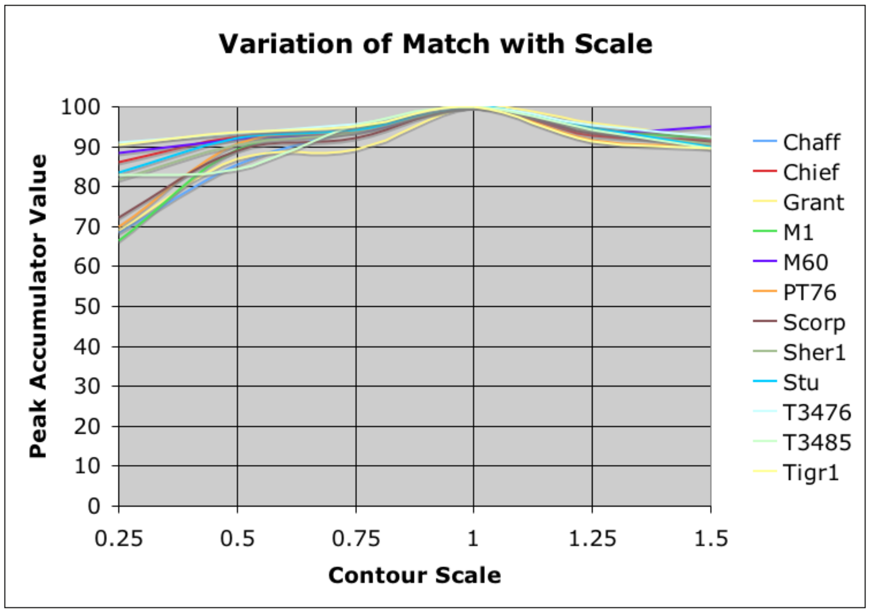

Figure 4.4: Variation of Accumulator Peaks for tanks contours, matched against their own models, with size.

At small object sizes (especially 0.25) many of the features of the tanks were lost. In particular, the antennae. Additionally, many of the smooth shape features become sharp and noisy at smaller sizes. This explains the lower match at small object sizes. At larger sizes the peak values are still not 100%. The introduction of additional points in larger contours and the fact that the "increment function" f produces a "graded" response accounts for this reduction in peak values. Overall, figure 4.4 shows that the algorithm is robust to scale. One of the weaknesses of the algorithm is the dependence on maximum distance for rescaling (of both curvatures and distances). However, this experiment shows this does not have a significant impact on performance.

4.3 Effect of Noise and Blur on Performance

In chapter 3 it was stated that no assumptions are made regarding the quality of segmentation. In terms of the algorithm there are no parameters that need to be set relative to the noise and blur in an image. Since the algorithm involves characterizing the shape that is described as a set of 8 connected points, the noise is also added to these contours. This can be viewed as boundary jitter. A real world process that adds jitter to contours as shown in figure 4.5 is difficult to conceive. Instead we propose to demonstrate robustness to bad segmentation. The jitter to simulate this is Gaussian random in nature and is added to the cartesian coordinates of the shape. The experiments here study the variation of peak values of the accumulator with increasing Gaussian random noise. The standard deviation of this additive was increased in steps of 0.1 between 0.1 and 1. There on it was increased in steps of 0.5 to 3. The peak value of the accumulator is expected to degrade gracefully with increasing noise. The reason for this degradation is the error introduced in the measurement of curvature. The curvature estimation algorithm also degrades in performance with increasing noise. Addition of noise does not significantly affect the calculation of θ and d values. However, the error introduced (both in terms of additional points) and curvature is significant enough to overcome the use of the graded increment function. Figure 4.7 shows a plot of the performance of the algorithm with increase in noise in the contour. The curves for all 12 tanks are plotted to show that they follow similar trends with increasing noise.

A phenomenon that does occur is contour blurring. The sharp features of the contour get smoothed over and the contour begins to lose its original shape (figure 4.6). An experiment was set up where the given contours were blurred continuously and the resulting degradation in performance was recorded. The variation of peak values of the accumulator with blur is shown in the figure 4.8. As expected there

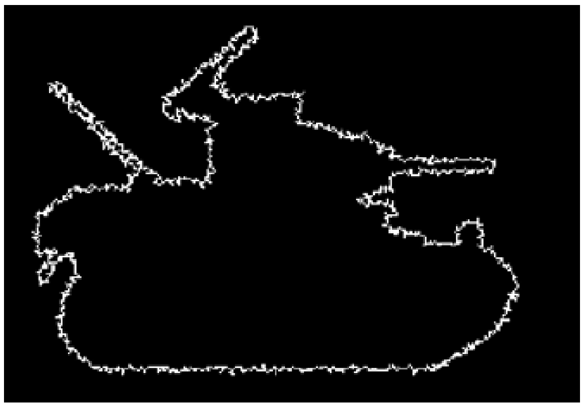

Figure 4.5: A tank contour with jitter added to the boundaries.

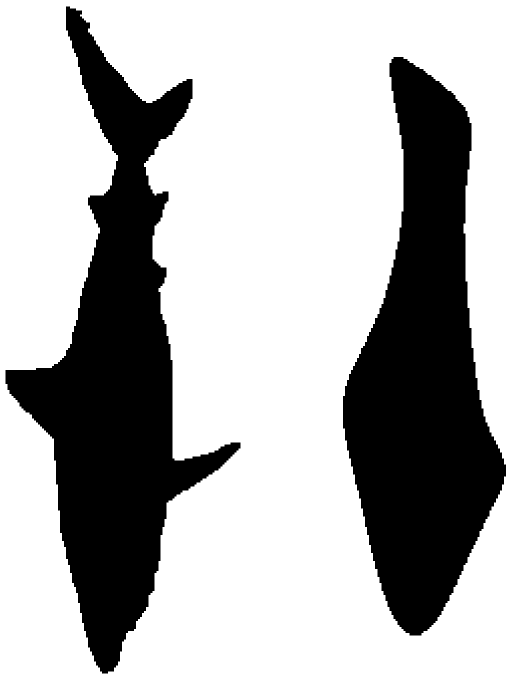

Figure 4.6: A fish contour (right) shown on the left after 2000 iterations of blurring.

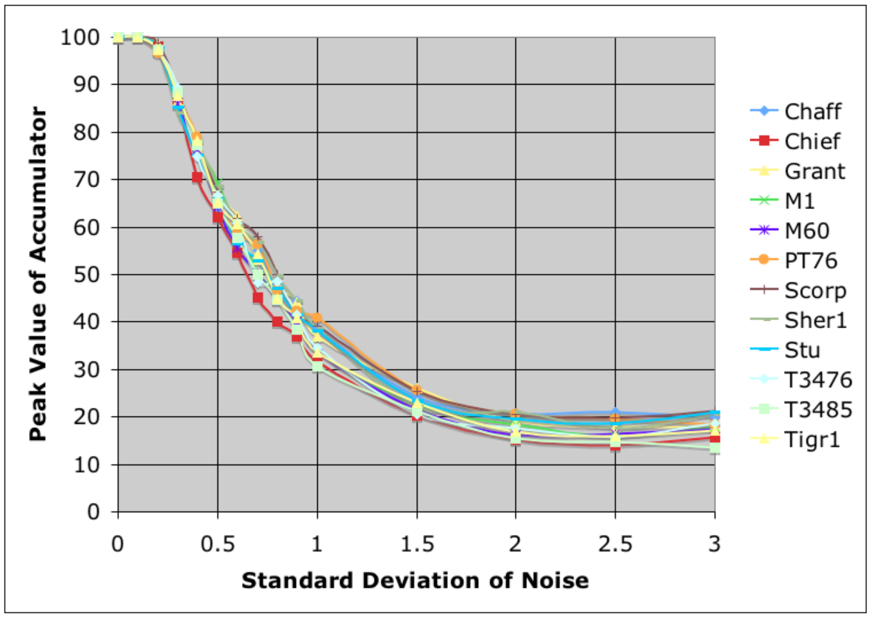

Figure 4.7: Variation of Accumulator Peaks for tanks contours, matched against their own models, with noise.

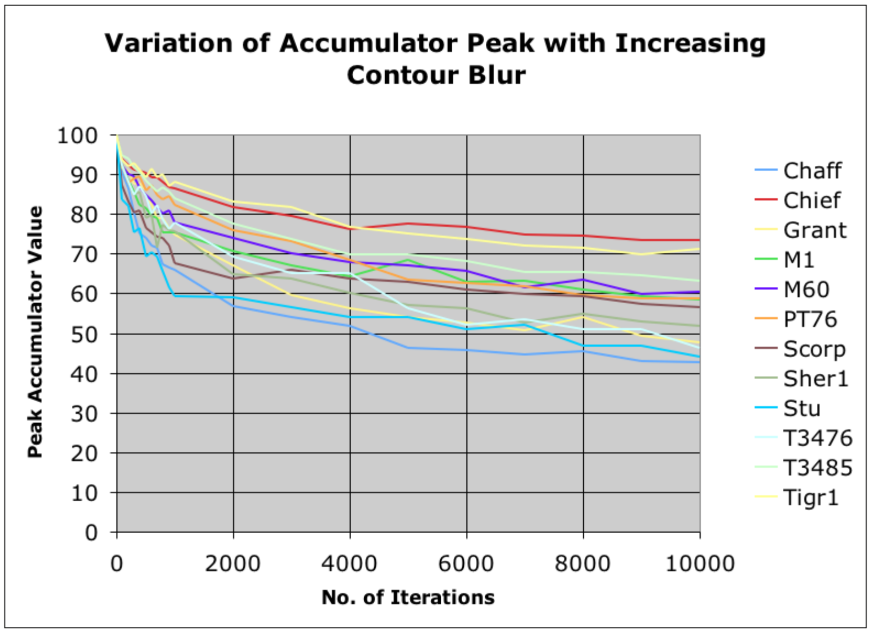

Figure 4.8: Variation of Accumulator Peaks for tanks contours, matched against their own models, with increasing blurring.

is a definite degradation in performance with increased blur. This however did not translate into any significant inaccuracy in classification as far as the tank data set is concerned. In order to get a more reliable estimate on the effect of blur on classification accuracy an experiment was conducted in which the SQUID data set was tested. In the experiment models were generated for the 31 contours sampled from the data set (figure 4.2). Next all the sampled contours were classified after the application of a predetermined number of blur steps. The accuracy of the matching algorithm for this data set is shown in figure 4.9.

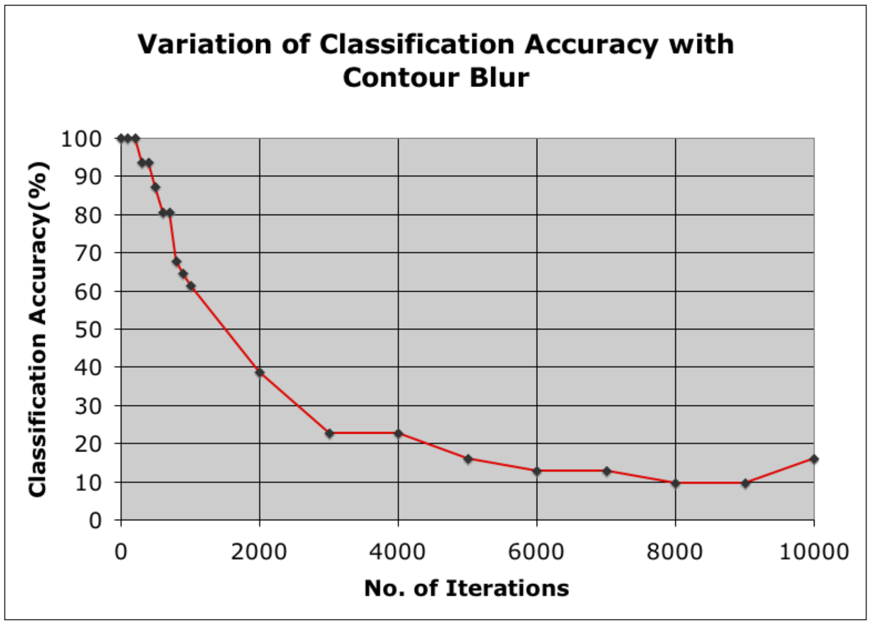

Figure 4.9: Variation of classification accuracy with increasing blur (tested on the SQUID data set).

4.4 Effect of the Increment Function on Performance

In chapter 3 a graded increment function was proposed for the matching process. The nature of the increment function has a significant effect on the ability of the algorithm to generalize across similar shapes while retaining the ability to discriminate between them. The increment function (equation 3.14) is Gaussian. The standard deviation of this function specifies the extent to which the measured feature vector can deviate from the "true" value, as derived from the representative shape, and still contribute a reasonable increment to the accumulator. For the purpose of this experiment the standard deviation of the Gaussian is specified in terms of the cells in the model ($\kappa \times d \times \theta$) space. Accumulator increments were calculated for standard deviations of 1,2,3,4,5 and 6. The goal is to observe the effect of this variation on the robustness of the algorithm to similarity transforms and to noise.

4.4.1 Effect on Rotational Invariance

Statistics for rotational invariance were computed on the entire data set in section 4.2.1. This however was at a fixed standard deviation. To study the effect of the nature of the increment function on rotational invariance, matching was performed with increment functions of increasing standard deviation. The results for one shape (tank: Chaff) are shown in figure 4.10. Other tanks follow similar trends.

4.4.2 Effect on Scale Invariance

The statistics for scale invariance were computed on the entire data set in section 4.2.2. The plot in figure 4.11 shows the variation in peak values of the accumulator with the standard deviation of the increment function. This plot was computed for one shape (tank: Chaff). Other tanks show similar variation.

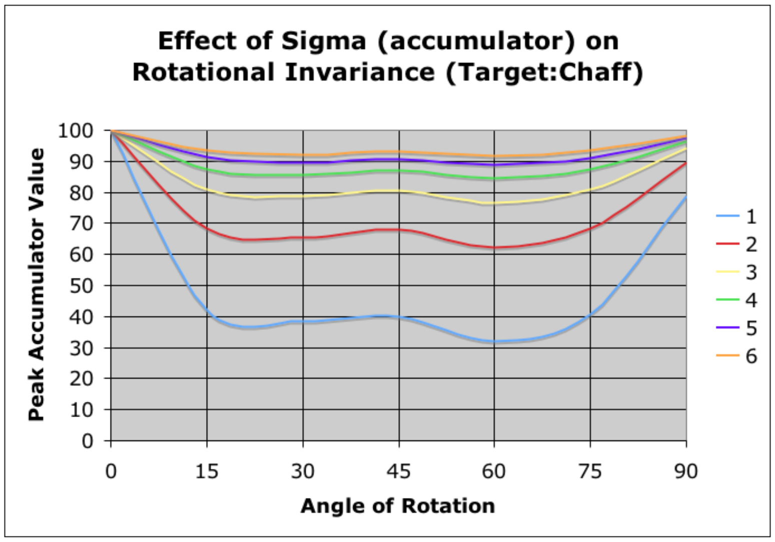

Figure 4.10: Variation of Accumulator Peak for Chaff with standard deviation of increment function for different angles of rotation.

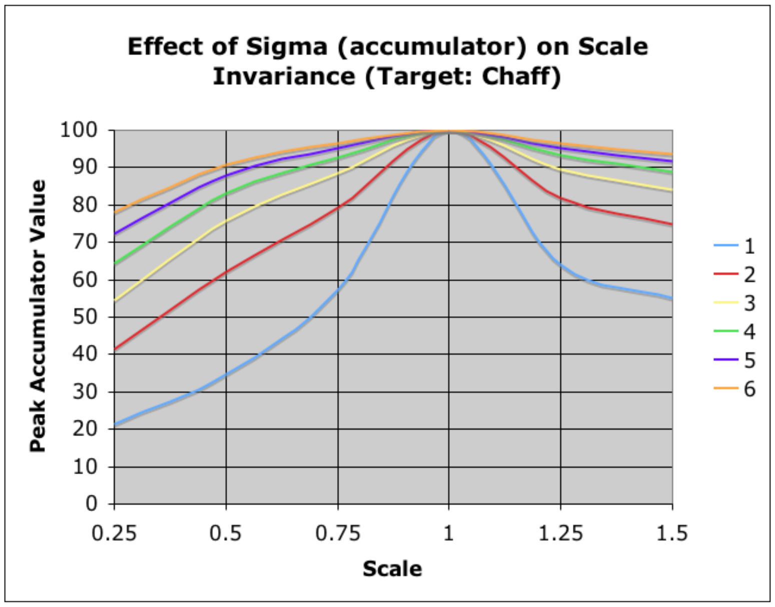

Figure 4.11: Variation of Accumulator Peak for Chaff with standard deviation of increment function for different scales.

Figures 4.10 and 4.11 are easy to interpret. As the standard deviation of the increment function increases, the ability of the algorithm to generalize also increases. There is a reasonable addition of digitization noise with rotation and scaling of a digital contour. This results in erroneous estimation of feature vectors at specific points in the contour and hence smaller accumulator peaks. The graded nature of the increment function offsets some of this variation in estimation of feature vectors. The extent of this offset is determined by the standard deviation of the increment function. For a small standard deviation (1) the offset is significantly smaller and the performance of the algorithm is poor. The increment function at a standard deviation of 1 implies if any of the features computed at a point is off by more than 1 "cell" the increment drops to below 0.367. This is indicative of a low tolerance to variation in estimation of the feature vector. This explains the nature of the plot for rotational invariance at a standard deviation of 1. As the standard deviation increases this tolerance (and hence ability to generalize) also increases. The plots show that for increasing standard deviations the algorithm performs better in terms of rotational and scale invariance. The plot also documents the diminishing returns with this increase. The level of improvement in generalization is relatively less between standard deviations 5 and 6. It is here that the maxim of discriminability becomes significant. Increasing the standard deviation brings with it a loss in discriminability between shapes. Therefore, it is prudent not to increase the standard deviation significantly. In our experiments a standard deviation between 4 and 5 showed good robustness to similarity transforms and noise.

4.4.3 Effect of Contour Noise and Blur on Performance

To examine the effect of noise and blur on the performance of the algorithm an experiment similar to those described in sections 4.4.1 and 4.4.2 was set up. In one experiment the tank contour of Chaff was taken and jitter was added to the cartesian coordinates. The standard deviation of noise was varied between 0.1 and 3. In the second, the Chaff contour was blurred continuously and the degradation of the accumulator peak was studied. Upto ten thousand iterations of blurring was applied to the contours. These experiments were repeated at different standard deviations of the increment function(1-6, shown in figures). A plot of the noise performance is shown in figure 4.12 and that of blur is shown in figure 4.13.

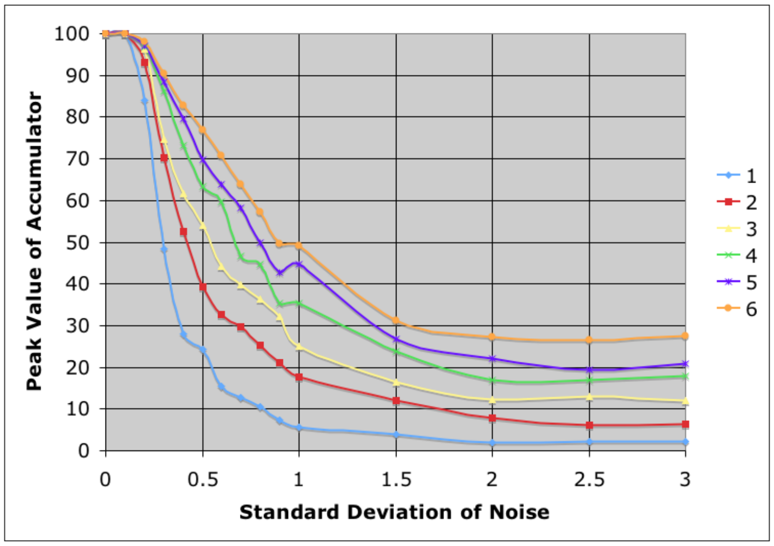

Figure 4.12: Variation of Accumulator Peak for Chaff with standard deviation of increment function for different levels of contour noise.

As expected the performance of the algorithm degrades with noise. However, the degradation is more gradual in the case of increment functions with larger standard deviations. This is expected owing to the increasing generalization that comes with more tolerant increment functions. A similar effect is seen in the case of the blurred contours.

4.5 Similarity Detection

One of the desirable properties of the algorithm is its ability to detect similarities between shapes. Since the algorithm involves encoding the structural information (curvature) of a shape relative to a set of global features (principal axis and geometric center), it is expected to produce similar models for shapes with similar structure. In the previous sections the ability of the algorithm to discriminate between some-

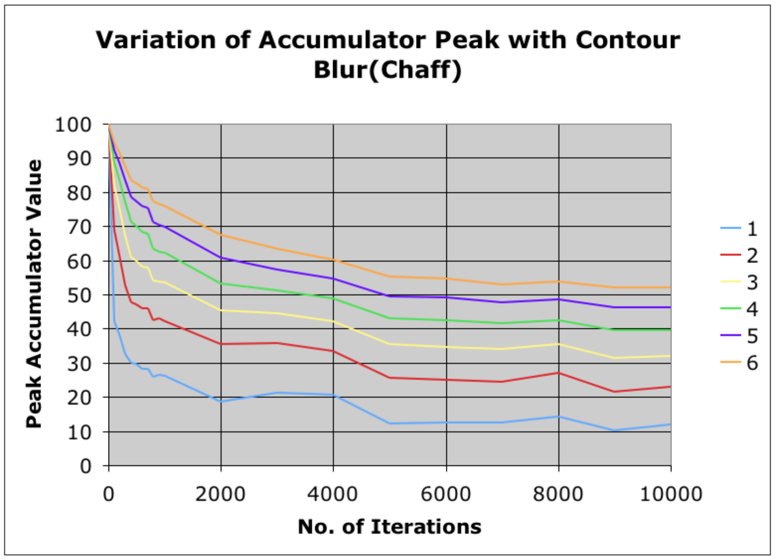

Figure 4.13: Variation of Accumulator Peak for Chaff with standard deviation of increment function for different levels of contour blur.

what "similar" shapes (all tanks) has been documented. In this section the ability of the algorithm to detect similar shapes is investigated. A formal evaluation of the output of the algorithm as a shape metric is out of the scope of this thesis. However, preliminary evaluation of the algorithm in this regard is presented here.

The SQUID data set was sampled at random and 31 contours were selected randomly. Models were generated for each of the contours. Next, the entire data set of 1100 contours was matched against the models. The top five matches for each model were isolated. An artificial threshold of t was set and the number of contours with peak accumulator value above t for each model was calculated. Some of the matches are shown in figures 4.14-4.17. Table 4.2 shows the number of contours (of the total 1100) with matches above the threshold t for different values of t.

Table 4.2 shows the number of contours with accumulator peaks higher than a particular threshold (t) when matched against each of the 31 models. Some of the models are more "similar" to other shapes in the data base, as can be clearly seen from the table. For example contour kk492 over 100 contours with an accumulator peak of greater than 60. The same is not true for other contours. The contour kk127 had all peaks below 50 (excluding itself). In order to attain a qualitative feel for how similar some of the matches are the the top matches for some of the models are shown in figures 4.14-4.17. In all the figures the shapes that show a high degree of match (in terms of accumulator peaks) are visually very "similar" to the model. For example, all the shapes in figure 4.17 are sea-horses. Similarly, all the shapes in figure 4.15 look like "sting rays". This ascertains the ability of the algorithm to detect similarity between shapes.

Table 4.2: Table shows number of matches above the threshold (columns) for different models (rows) when the 1100 contours of the SQUID data base were matched against the models.

Model	$t = 90$	$t = 80$	$t = 70$	$t = 60$	$t = 50$
kk5	1	1	5	66	289
kk19	1	1	3	28	135
kk30	1	1	2	31	80
kk42	1	1	1	12	22
kk72	1	1	9	48	183
kk83	1	1	5	78	384
kk87	1	5	16	31	48
kk100	1	1	1	12	162
kk127	1	1	1	1	1
kk184	1	1	1	1	1
kk209	1	1	1	3	58
kk250	1	1	1	4	17
kk263	1	1	3	7	34
kk273	1	2	9	38	105
kk310	1	1	1	12	86
kk378	1	1	2	5	19
kk430	1	3	8	36	136
kk464	1	1	1	3	14
kk473	1	1	1	1	5
kk480	1	1	1	7	183
kk495	1	2	8	138	445
kk502	1	3	25	122	359
kk558	1	3	6	50	231
kk676	1	1	1	1	5
kk689	1	1	7	50	210
kk717	1	1	3	22	169
kk732	1	8	11	16	29
kk766	1	4	31	146	370
kk779	1	1	2	3	19
kk810	1	1	5	33	234
kk942	3	5	33	136	366

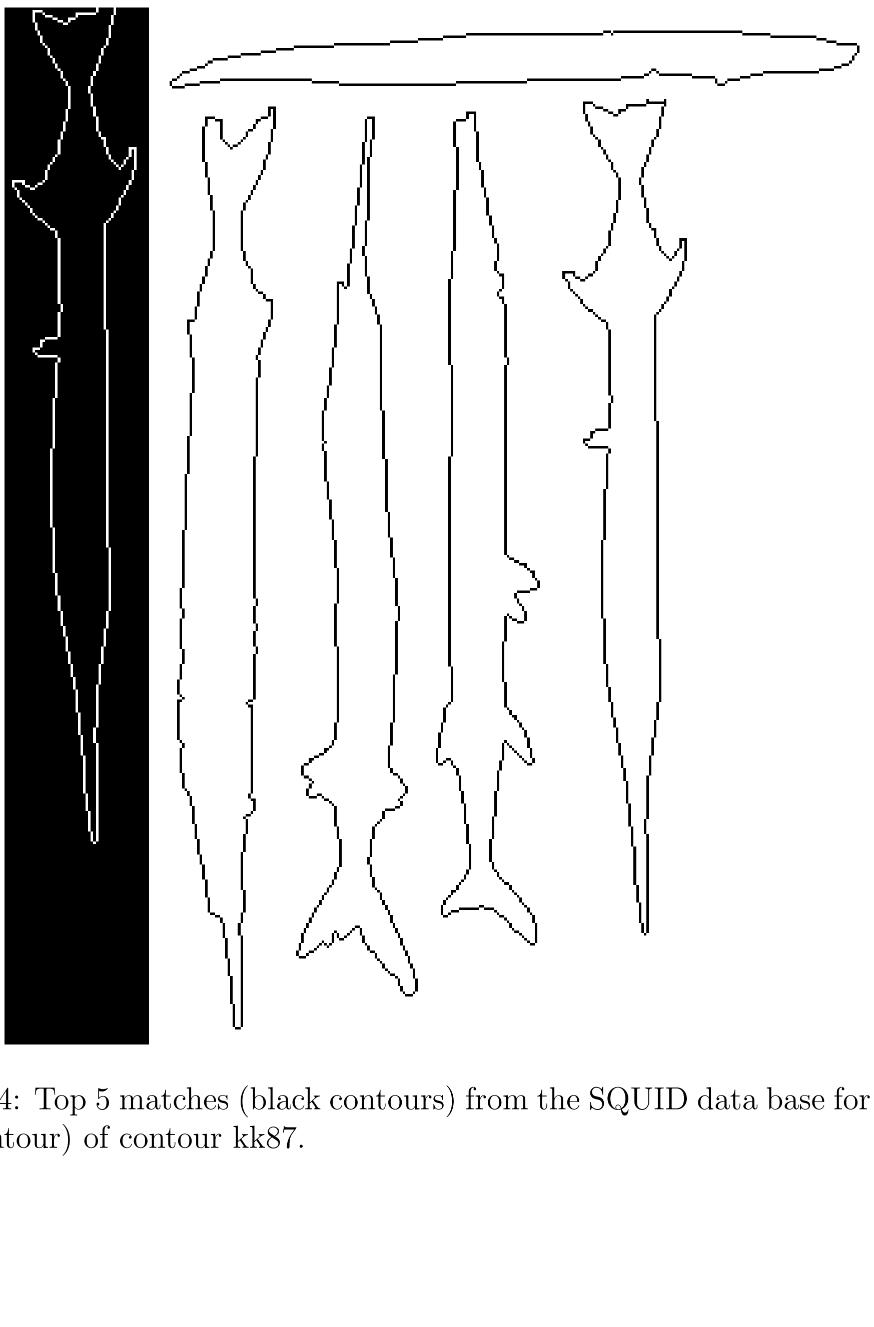

Figure 4.14: Top 5 matches (black contours) from the SQUID data base for the model (white contour) of contour kk87.

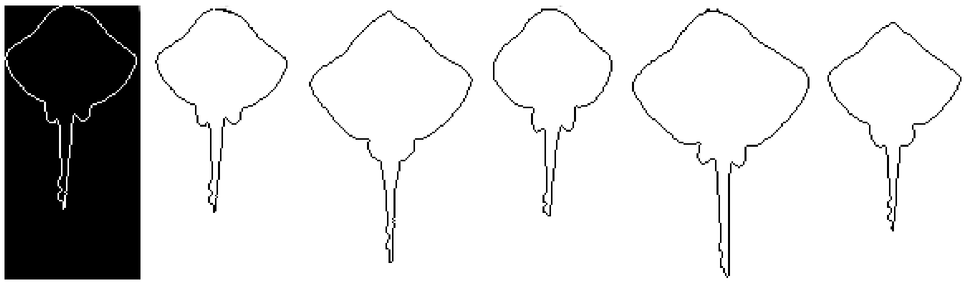

Figure 4.15: Top 5 matches (black contours) from the SQUID data base for the model (white contour) of contour kk732.

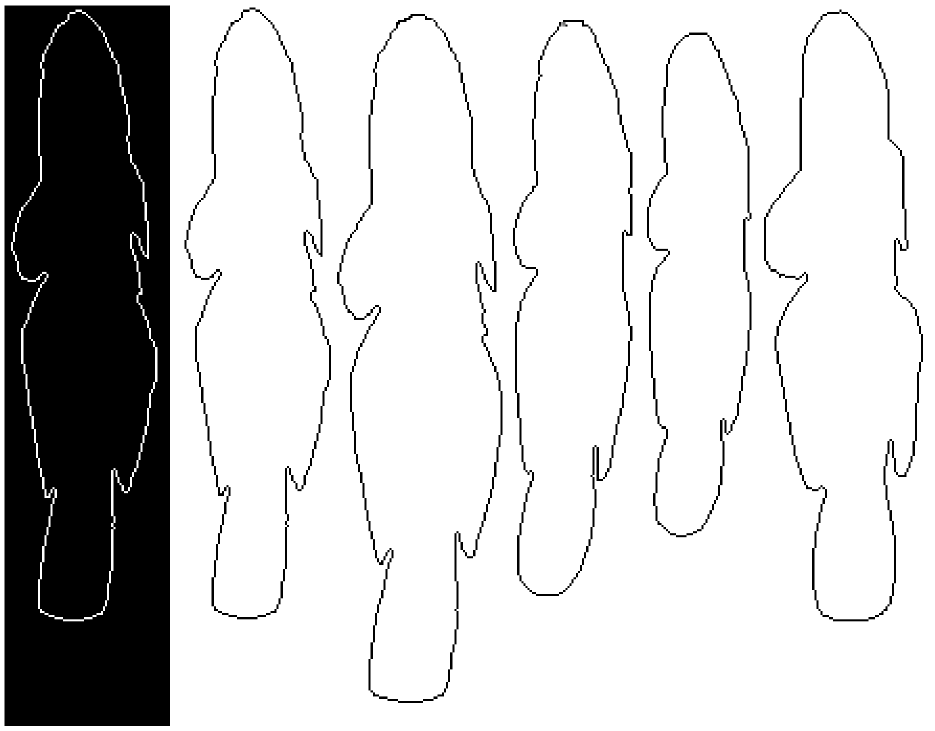

Figure 4.16: Top 5 matches (black contours) from the SQUID data base for the model (white contour) of contour kk942.

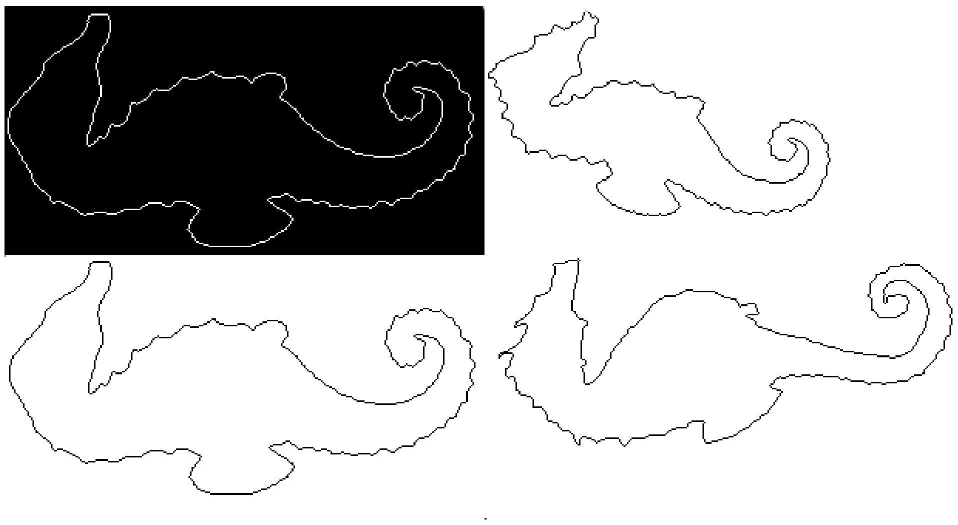

Figure 4.17: Top 3 matches (black contours) from the SQUID data base for the model (white contour) of contour kk779. Other matches were below the threshold of 60.

In this chapter experiments have established that the algorithm presented in chapter 3 is invariant to similarity transforms and robust to noise and blur. The algorithm can also detect similarity between shapes and a combination of the outputs (accumulator peak and peak-offset from the center of the accumulator) may be useful as a "shape metric". The algorithm has not been tested for robustness to occlusions though this is something that needs to be done to establish its effectiveness as a shape characterization and matching technique. Some experiments with variants of the algorithm have shown it to be invariant to occlusion also. However, this came at the cost of scale invariance.

Chapter 5

Neural Network Architecture for Shape Recognition

The philosophy behind the SKS algorithm, developed in the chapter 3, is largely influenced by the working of the human visual system. In addition to developing a robust and invariant representation for shapes, this thesis is a first step towards the larger goal at the Image Analysis Group, developing an understanding of how humans recognize shapes. One of the significant aspects of the SKS algorithm is that it is highly parallel. Ballard, Hinton and Sejnowski identified the importance of parallel architectures to visual processing in [2]. They hypothesized an object centered frame work that describes shape features relative to a frame attached to the object. A parallel architecture implementing feature matching like the Generalized Hough Transform is recommended by them. All the elements of the SKS algorithm can be implemented on a parallel architecture, like a neural network. The curvature computation algorithm is local to the object boundary. The feature vector computed for each point is also independent of the process of calculation of feature vectors at other locations. In this chapter, a neural network architecture that implements the algorithm, developed and tested in the previous chapters, is discussed. First, a summary of the visual pathway and the processing of shapes in the human visual cortex is presented. This is followed by a discussion of some neural network architectures that are present in literature. *The Standard Model*[32][33][36][7][35], an architecture for shape processing developed at the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology and the VisNet proposed by Rolls and Deco in [34] are discussed here. Both are hierarchical neural network implementations of the visual pathway. They achieve scale and translational invariance and closely mimic the behavior of actual cortical recordings of cells in the higher visual areas (V4 and IT). Finally, an augmented model that ensures invariance to rotation and brings about an object centered representation of shapes is described.

5.1 The Visual Pathway

The processing of shape information starts very early in the visual pathway (figure 5.1). The receptors (cones) on the retina are packed densely in a region called the fovea. This region is generally where the focus of attention is. The receptors are connected to the bipolar cells in the retina either directly (in an excitatory role) or through horizontal cells (in an inhibitory role)[34]. These generate center surround ON and OFF center cells. A detailed description of this architecture and the processing is presented in [22][34]. The bipolar cells feed into the retinal ganglion cells and feed through the optic nerve onto the *lateral geniculate nuclei*(LGN). The LGN retains the retinotopic mapping that is maintained right through to higher visual areas. The LGN contains two kinds of cells - Magno-cells (M-LGN) and Parvo-cells (P-LGN). Of these cells, the P-LGN are primarily responsible for communicating shape information. Both these layers of the LGN feed into different regions of the primary visual cortex (V1). The focus of this description is on shape processing and so only the section of the visual cortex relevant to shape is discussed from here.

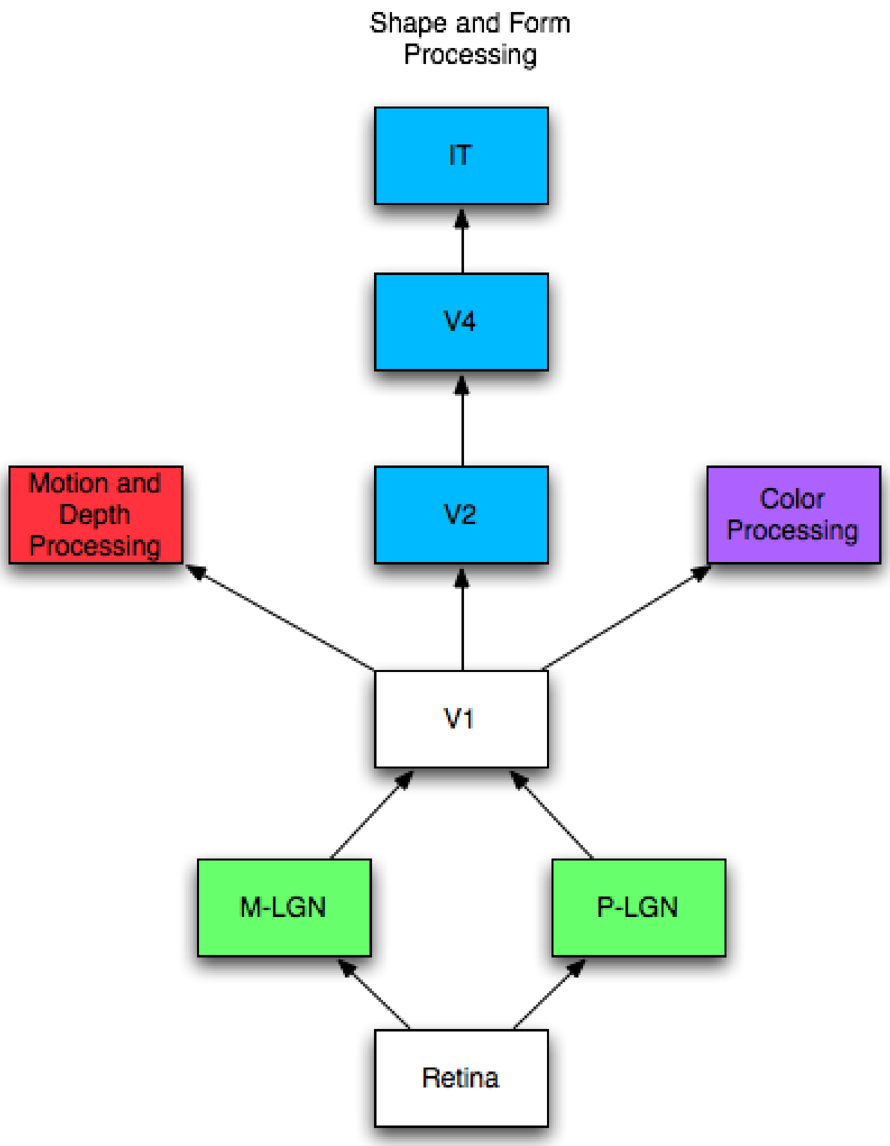

Figure 5.1: A block diagram of the Visual Pathway.

5.1.1 V1

The primary visual cortex constitutes several layers (hence the name *striate cortex*). There are rich vertical connections between these layers with very little horizontal or diagonal spreading between layers. This would suggest that most of the processing in the V1 area is local and it clearly is not the *seat of perception* [22]. There are three types of cells in this visual area. *Simple Cells* respond best to optimally oriented bars or edges. They are tuned to four different orientations with a tuning window of 45^o [34]. They are said to arise out of the spatial summation of ON and OFF center cells[22][34]. Research by De Valois and De Valois also show that these cells are sensitive to the spatial frequency of the visual stimuli[34]. These properties of the Simple Cells are essential to understanding their implementation using Gabor spatial filters [11] in later sections. *Complex Cells* are also sensitive to oriented bars but they show some translational invariance. The bars can be present anywhere in their receptive field. This response of complex cells was explained to be due to the integration of the responses of simple cells with similar orientation tuning but different (adjacent) positions in the visual field[22]. *End Stopped Cells* are the third type of cells present in the V1 area. They are sensitive to curves of appropriate curvatures and properly oriented lines of fixed lengths.

The cells in V1 are arranged in "hypercolums" [34]. These "hypercolumns" are responsible for the processing of visual information from one specific area in the visual field. They contain columns of neurons tuned to a particular orientation with multiple columns representing the four different tunings. This is important because the retinotopic mapping is still maintained and the different hypercolumns can be viewed as tiny processors of local shape information. In [34] it is suggested that V1 layer neurons "may function collectively to incorporate contextual information from outside their classical receptive fields and in turn serve pre-attentive visual segmentation". This is important to the implementation of the neural network architecture for the SKS algorithm. This is because global shape information like the principal axis may require such mechanisms to enable its incorporation to the processing of contours.

5.1.2 Extra-striate Processing of Shape Information

There is very little understood about the behavior of extra-cortical circuits. However, some studies [35][29][30] have shown that these neurons are sensitive to combinations of features from their afferent layers. The extra-striate areas primarily involved in handling shape information are regions of the V2, V4 and IT. The cells in the V2 are known to be sensitive to low order combinations of the afferent[3] cells from the striate cortex (the primary visual cortex) [34][35]. These combinations represent the growing complexity of features along the visual hierarchy. They also represent the growing invariance of this architecture to small translations. This has been replicated by architectures like the VisNet [34] and *the Standard Model*[32][33][36][7][35].

The V4 is known to be sensitive to low order feature combinations [30][10]. There are cells in the V4 that are tuned to simple features like oriented line segments and others that are tuned to more complex curvatures. In [10], Connor summarizes his research into neurons in the V4. He shows that these neurons are sensitive to arrangement of shape features relative to the object center. The V4 neurons are shown to be sensitive to angular position of a feature, the normal at the feature, the curvature of the feature and the context (the curvatures of the segments adjacent to features). These cells also show some invariance to object size and object location (translation). They also have larger receptive fields when compared to the cells lower in the hierarchy. These properties are of significance to the SKS algorithm and also the neural network architecture. Connor [10] hypothesizes an object recognition framework that is a hierarchical feed forward network with neurons sensitive to feature combinations of their afferent layers. This network not only encodes what features are present but also their relative spatial arrangements. This hypothesis along with the research of Poggio et al. [32][33][36][7][35] and Rolls' VisNet[34] were essential to the develop-

[3]conducting or conducted inward or toward something (for nerves or central nervous system).

ment of the neural network architecture to implement the SKS algorithm.

The final higher visual area that is significant to shape processing is the Inferior Temporal cortex (IT). The neurons in this area of the visual pathway are sensitive to moderately complex features [34]. Neurons that are sensitive to similar features are clustered together and are involved in competitive inhibition to ensure discrimination. These neurons also exhibit significant translational invariance [34] in addition to scale invariance of up to two octaves [35]. Poggio et al., [35], argue that this layer contains several view-tuned neurons whose responses are pooled to produce view-independent neurons that identify specific objects. Additionally, Rolls [34] has documented that this representation of objects in the IT is distributed as this enables the encoding of a significantly larger number of shapes. This also ensures a graceful degradation of performance in the presence of noise. A significant finding about the view-independent cells in the IT [34] is that a set of neurons was shown to fire even when the object presented was completely inverted. This suggests an object based framework for recognition in these areas.

5.2 VisNet and the Standard Model

Several architectures for invariant object recognition are present in literature [35][34] [17][6]. Of these neural network implementations, those of Poggio et al. [35] and Rolls [34] present a biologically plausible framework for processing shapes (2D and 3D). In this section, these two models of human vision are presented and discussed. Both are feature hierarchies and exhibit strong invariance to translation and scale changes. Both are intent on explaining the processing of shapes in the visual pathway using biologically plausible circuits. They both are feed forward networks with absolutely no top-down feedback.

5.2.1 VisNet

VisNet is a neural network architecture for shape recognition developed by Rolls and Deco and described in their book [34]. The general philosophy involves a feature hierarchy going from simple (oriented lines) to complex (curves and corners) features. The network has 4 layers of neurons that learn and classify using mutual inhibition (over short ranges) and competition. The units from one layer converge onto neurons in the higher layers. In other words, several neurons in the lower layers are afferent on neurons in the higher layers. This implies an increasing size of receptive fields as we go higher up in the network. The input to the network is from a set of 2D spatial filters implemented as Difference of Gaussians. These are intended to mimic the orientation and spatial frequency sensitivities of *simple* cells in V1. Rolls and Deco do not assume any preexisting affinity for any combination of features. The layers through self-organization learn to represent the entire feature space. Feature combinations are not replicated at all positions. Instead a representative sample of images with all possible features is learnt by the lower layers of the network. These images are presented at all possible positions in the input layer. The higher layers learn feature associations specific to objects and do not bother with translational invariance. Rolls and Deco also suggest a trace rule for learning. This is similar to Hebbian learning except there is a temporal aspect to it. This temporal aspect to learning brings about some of the invariance to scale and translation. This form of Hebbian-like learning also makes the network biologically plausible. Additionally, the competitive nature of learning also brings about the distributed representation like the neurons in the IT exhibit. Overall, this hierarchical representation exhibits the ability to differentiate between different spatial arrangements of features and is similar to the behavior of cells between the V1 and IT layers of the ventral pathway.

5.2.2 The Standard Model

The Standard Model [32][33][36][7][35] is a hierarchical model of shape recognition developed by Poggio, et al. It is a strictly feed forward architecture and seeks to explain the first 150 ms of vision. It is similar to the hierarchical model proposed by Fukushima [17] in that there are alternating layers of *simple* and *complex* cells that learn feature associations. In the Standard Model, these two types of cells are responsible for two distinct tasks. Unlike the VisNet their roles are predefined and do not emerge out of self-organization. The simple cells (S) are responsible for template matching. Each cell is active when the features in the object at a particular spatial location have high correlation with the feature that the S cell is tuned for. At the very base, the S cells mimic the behavior of the *simple* cells in V1. They are sensitive to oriented line segments. The entire space of orientations is covered by cells with a tuning peaks at 45^o intervals. In the Standard Model 2D Gabor filters serve as inputs to this layer of S-cells. These cells are sensitive to location of the oriented bars and can be assumed to be part of the "hypercolumns" that are found in V1. The *complex* cells (C) are responsible for the invariance characteristics of the network. They achieve this by pooling in inputs from S-cells that are tuned to the same feature but located at slightly different positions in the visual field. This operation is achieved by a softmax of inputs from the S-cells layer.

These layers of C and S cells are alternated to produce features of increasing complexity while ensuring a small degree of invariance to scale and translation. A complete discussion of this model is presented in [35]. In the same paper the authors discuss the biological plausibility of these operations of shape tuning and softmax. They argue that circuits for these operations are biologically plausible and also offer evidence for the same. This hierarchy of alternating layers agrees with neurophysiological data in literature. Most significantly, the layers at the top of the hierarchy (C2) show considerably similar sensitivities as cells in area V4 as reported in [29][30]. The authors state [7] that this hierarchy results in an object centered representation of features as hypothesized by Connor [10]. The results presented confirm this claim.

5.3 The SKS Neural Network Architecture

The neural network architecture proposed in this section is influenced by VisNet and the Standard Model. The philosophy of the SKS algorithm is to utilize global characteristics of a shape to overcome rigid similarity transforms and use local features to arrive at a shape description. The SKS algorithm brings about significant invariance to rotation, scale and translation. In addition, it also exhibits robustness to noise and blur. Translating the serial computations of the algorithm discussed in chapter 3 to a parallel architecture like a neural network is simplified by the fact that the modules of the algorithm are inherently parallel. The DSS algorithm for calculating curvature is also remarkably parallel. In this section the specifics of the architecture to implement the SKS algorithm in a neural network are discussed. No experiments were carried out with the proposed architecture.

5.3.1 Description of the Architecture

The SKS architecture inherits some prominent features from the VisNet and the Standard Model. In addition to the "what" information that is processed, it also seeks to incorporate an object based frame of reference to arrive at a consistent and invariant representation of shape. The Standard Model achieved scale and translation invariance by utilizing tuning functions and the pooling outputs over spatial location and scale. Rotational invariance is not discussed. Different view-tuned neurons are present for different views of the object and intermediate views are determined by graded response of these neurons. In this hypothesis however, it is proposed that view-tuned neurons exist but they are for depth-rotated analogues. The presence of different neurons for objects rotated in the plane is inefficient because the same set of features are "visible" to the shape processors. It is possible to propose a "biologically plausible" architecture that could implement this kind of processing. The hypothesis is influenced by the fact that vision involves the perception of "orientation" of an object (usually in terms of the principal axis or the axis of elongation) in addition to its spatial location (in terms of its geometric center).

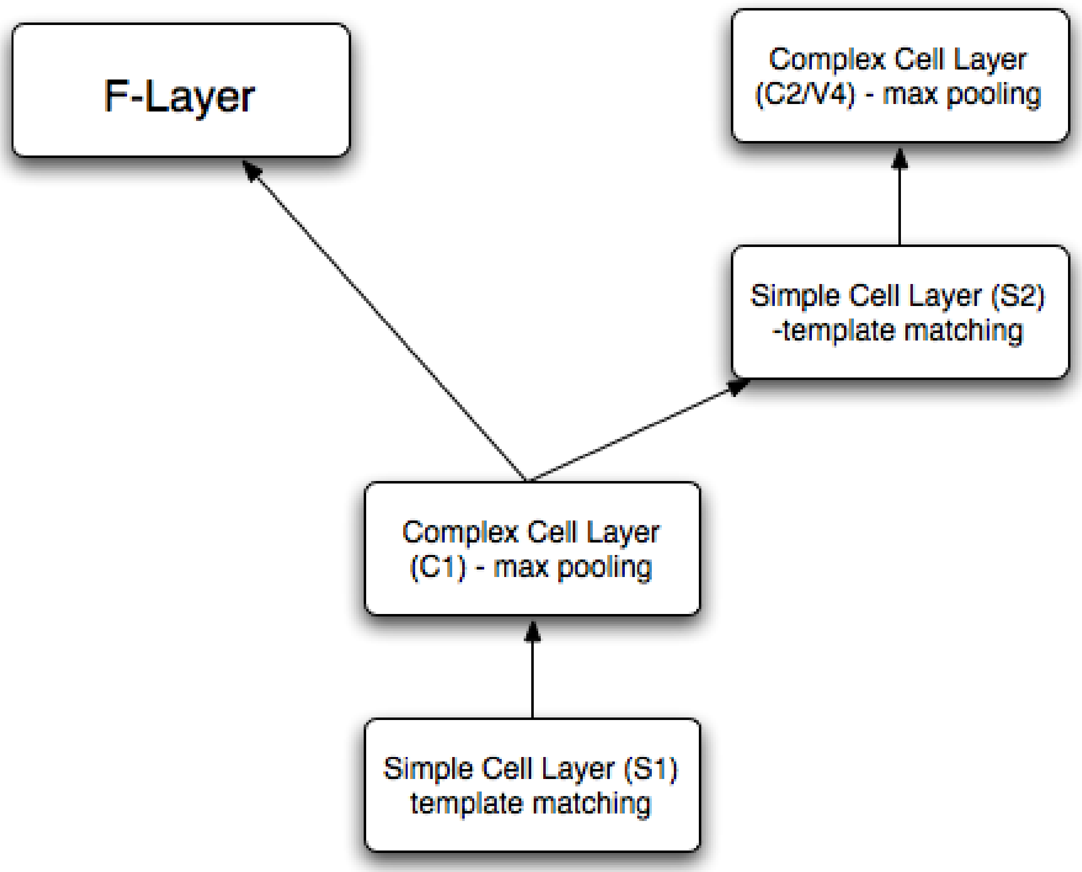
F-Layer
Complex Cell Layer (C2/V4) - max pooling
Simple Cell Layer (S2) -template matching
Complex Cell Layer (C1) - max pooling
Simple Cell Layer (S1) template matching

Figure 5.2: The SKS Neural Network Architecture. Alternating layers of Simple and Complex cells produce cells that are sensitive to low order combinations of features and at the same time a degree of translational and scale invariance. Additional layers can be added after C2 in order to arrive at more complicated features and larger receptive field sizes.

The neural network architecture hypothesized has the following features (figure 5.2):

- The architecture is a hierarchical, feed forward, multilayer, competitive neural network with mutual inhibition within a layer. The alternating simple(S) and complex(C) layers design of the Standard Model is retained. In essence it is a

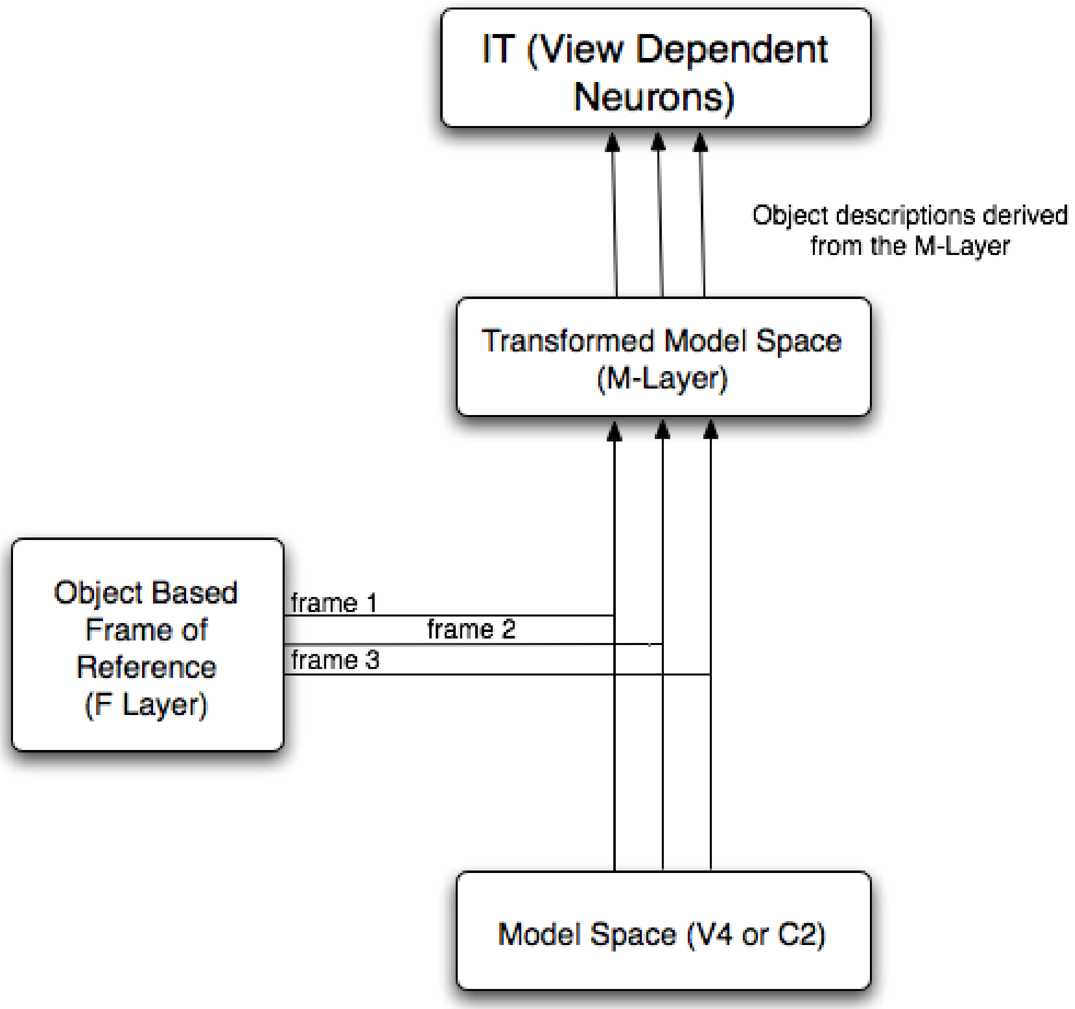
IT (View Dependent Neurons)
Object descriptions derived from the M-Layer
Transformed Model Space (M-Layer)
Object Based Frame of Reference (F Layer)
frame 1
frame 2
frame 3
Model Space (V4 or C2)

Figure 5.3: Sigma Pi connections gate the connections between the C2 layer and the M-Layer. Each cell in the M-layer is connected to equivalent cells in the C2 layers. When an afferent cell fires the input is communicated to the cell in the M-Layer that is consistent with that assignment of principal axis.

feature hierarchy going from units sensitive to oriented edges to complex stimuli.

- The input to the network is a 2D Gabor filtered image. The filtering of images using 2D Gabor spatial filters is discussed in [11].
- The architecture presented hypothesizes that the “feature-specific” information and the “global” information with regard to a shape are processed in parallel. After one layer of alternating S and C cells, the output of the C layer branches out into two distinct branches. One branch is for processing the orientation of the shape(F layer). The second branch continues the process of combining the outputs of layer C1 to produce low order feature combinations in layer S2.
- The C2 layer is equivalent to cells in the V4. This in turn is equivalent to the “model” space discussed in section 3.1. The cells are sensitive to angular position and curvature (added to distance) relative to the object center. However, the angular position here is relative to the viewer coordinates.
- The M-Layer, in figure 5.3, is a transform space of the C2 layer. Considering the C2 layer to be parameterized by distance, angular position and curvature of a segment [29][30][10], the M-Layer is also parameterized similarly. Each cell in C2 tuned to a particular feature at particular distance from the object center is connected to each and every cell tuned to the same feature at the same distance from the object center (but at different angular positions) in the M-Layer. These connections are gated by the outputs of the F-layer. For a particular orientation of an object a combination of the firing of cells in the F-layer and the C2 layer will produce (related) activity in the M-Layer.
- The M-Layer then communicates by simple feature combination a view (rotation in depth) dependent representation to the IT.

The F-Layer and the M-Layer

The critical additions to the architecture are the M-layer and the F-layer. Hinton in [18][19] proposes an architecture to enable invariance to similarity transforms by modulating shape descriptions by object based frames of reference. The F-layer computes an object based frame of reference relative to which the object description is communicated to the IT (through the M-layer). The significance of the F layer is shown in figure 5.3. The F-layer outputs gate the neural connections between the V4 (C2) layer and the M-layer. The gating enables only consistent object descriptions to pass through to the cells in the IT. Object descriptions in which there is correlation between the principal axis and the arrangement of features (segments of different curvatures) relative to this framework are considered to be consistent. This gating is accomplished by "biologically plausible" sigma-pi connections between the two sets of neural circuits [34]. The M-layer is a transform space where all features are transformed relative to a reference frame defined by the principal axis. Each view-dependent neuron is activated by an object description corresponding to combinations of features from the M-layer. The outputs of the view dependent neurons are pooled for the activation of view independent neurons in higher areas of the IT.

5.3.2 Equivalence to the SKS algorithm

There is a considerable level of similarity between the SKS algorithm and the neural network described in the previous section. There are parallels that can been drawn between the modules of the SKS algorithm and this neural network. The two are philosophically equivalent. In model building using the SKS algorithm, the first steps of processing include determining the geometric center of the contour and the principal axis. This is accomplished in part by the alternating S and C layers of the hierarchy, as they achieve a degree of translational and scale invariance.

The determination of the principal axis is done by the F-layer. The inputs to this layer are the outputs of the C1 layer. So the inputs are the response to orientation filters that are obtained from the 2D Gabor filtering of the image (with added translational and scale invariance). The F-layer is composed of two sub layers. They first layer contains cells tuned to the four directions similar to the S1 layer. They are not distributed over space however. Instead these cells pool inputs from similarly tuned

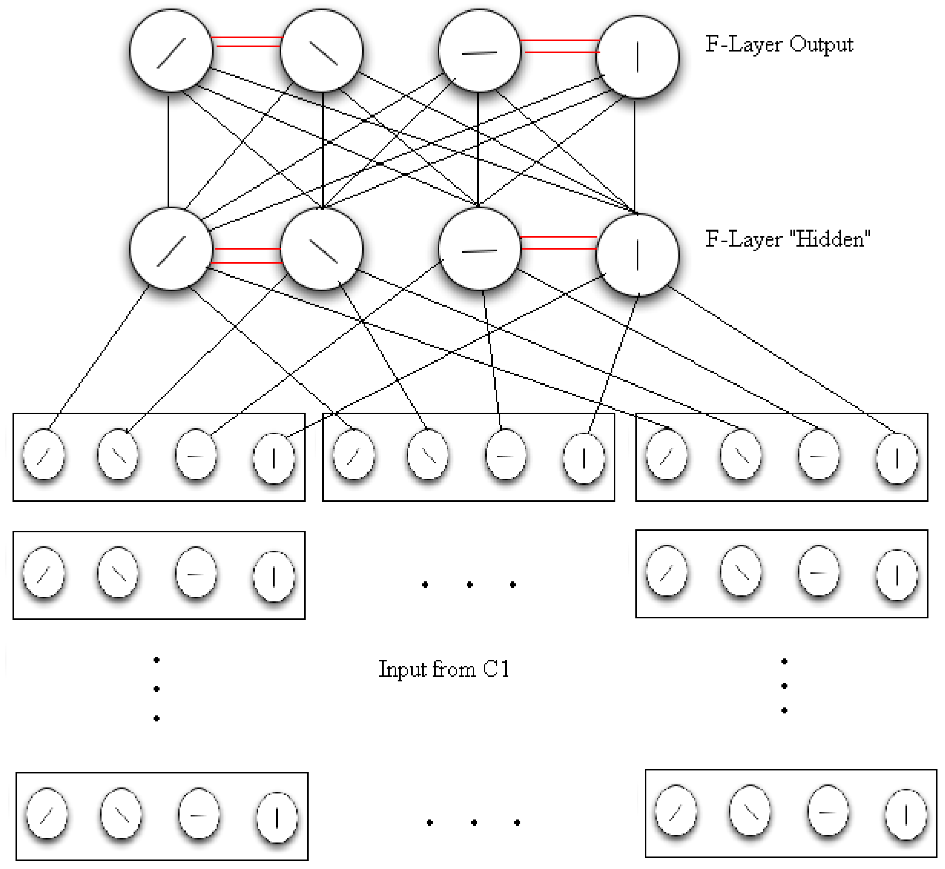

Figure 5.4: The architecture of the F-Layer for determination of the principal axis. The red connections are in a mutually inhibitory connections. The neurons in the hidden layer pool inputs from a similarly tuned neurons in the C1 layer. They are connected to the output layer to produce an estimate of the principal axis. The output layer is what produces the gated sigma-pi connections between the C2 layer and the M-layer.

cells (over all spatial locations) in the C1 layer (see figure 5.4)(This architecture can be adapted to pool inputs over smaller image regions and introduce a hierarchy to determine the principal axis. However, these two are still similar in philosophy.) This pooling for each orientation is expected to give a measure of the number of segments in C1 with the same orientation. Intermediate orientations are expected to contribute differently to the two orientations that they are closest to. Once this pooling is done by mutual inhibition, two directions are expected to stand out distinctly (with un-normalized values). Their values are expected to be small as a proportionally small percentage of their afferent cells are expected to be active for most images. The values of the responses of these orientations are fed to the output F-layer which determines a normalized value so that a proper orientation can be inferred.The F-layer connections can be assumed to be hard wired as of now. The outputs of these F-layer are used to gate the connections between the C2 layer and the M-Layer. In the SKS algorithm, the contour is transformed to the new frame of reference defined by the principal axis. In the network, the model space equivalent of the contour is transformed instead. These two are equivalent operations. An assumption made here is that the gating circuits and the F-layer circuits are hard-wired (either genetically or during development) and are not dependent on visual experience.

In the SKS algorithm, curvature is determined utilizing the digital straight segments (DSS) algorithm. This algorithm can be implemented in a parallel architecture. The functioning of the feature hierarchy can be viewed as curvature filtering [36][35]. Along the hierarchy low order combinations of afferent features produces sensitivity to more complex stimuli (like curves of varying curvature). In the SKS algorithm each and every point is treated uniquely and technically an infinity of “features” (different curvatures) is present. In the network however, these combinations are dictated by visual experience and the statistics of real images. Additionally, graded response of different features can help represent a virtual infinity of feature space. The model space in the SKS algorithm is similar to the C2 layer which is parameterized by curvature of the feature, angular position and distance from the object center. The significant difference is in the matching process. There is no accumulator present in the network discussed above. However, the accumulator can be viewed as 2D spatial correlation between the models of two contours. This correlation is what is implemented in the connections between the neurons of the M-Layer and the view-tuned neurons of the IT. If no depth rotation is assumed, these view tuned cells should be sufficient to represent and classify the database of shapes presented in the previous chapter.

In this chapter a neural network architecture for shape recognition is proposed. It is largely derivative from models developed by Poggio and Riesenhuber , and Rolls and Deco. The models have been augmented with additional layers (F-layer and M-layers) that enables a rotationally invariant representation of shapes. There are distinct representations for depth rotated views as the feature sets that are visible are not only different but they also have different spatial arrangements. The final neural network developed is influenced by the SKS algorithm and is philosophically similar to it. Implementation of this network is beyond the scope of this thesis.

Chapter 6

Conclusion and Future Directions

In this thesis an algorithm for shape modeling and matching is developed. The algorithm was described and discussed in detail in chapter 3. The SKS algorithm is shown to be invariant to translation, rotation and zoom. Additionally, robustness to noise and blur is also established. A reasonable implementation of the algorithm as a multi-layered, hierarchical neural network is also proposed. However, the invariance to occlusions was not tested as part of the thesis. Human object recognition exhibits tremendous robustness to occlusion. A reasonable description of human object recognition needs to identify mechanisms to achieve this. One point that is often pointed out in models of human object recognition is that we are not completely invariant to rotation in the plane. Humans do exhibit a degree of invariance to rotation in the plane but not to complete image inversion. The SKS neural network architecture is for a completely rotationally invariant representation. Shepard and Metzler in their classic paper [37] show that humans may utilize mental rotation of images to identify objects in unfamiliar orientations. It is possible to think of the F-layer and M-layer (chapter 5)as providing the computation required for these alignment operations.

The approach described in this thesis is however is only one variant of the algorithm. The first attempt at the algorithm was done using just the curvature and the distance at a point as features. This formulation of shape representation however does not satisfy the uniqueness criterion of shape representation as discussed by Marr [28]. Another version of the algorithm was developed using curvature, distance and the tangent angle (angle between the position vector and tangent vector) as features. This showed promise and the performance on data sets was better than the previous case. However, it is difficult to prove the uniqueness of models in this scheme. It is possible to conceive of shapes which have the same models. Additionally, the mapping from the contour space to the model space is many to one. Several points in the point space can be mapped to the same point in model space. A circle is a case in point. The tangent angle, curvature and distance are equal (in theory) for all points on the circle. Describing the position vector (in terms of angular and radial position) of a boundary point is one way of ensuring uniqueness. No two points on the boundary map to the same point in model space (especially in the continuous case). The SKS modeling of shape satisfies all of Marr's criterion for shape representation:

- **accessibility**: Once the shape representation is computed, matching is a simple look up process.
- **scope**: There is no limitation on the variety of shapes that this algorithm can represent.
- **uniqueness**: The utilization of radial and angular position of boundary points ensures uniqueness of models. Additionally, the models are invertible, i.e. the original shape can be retrieved from the model.
- **stability**: The biggest factor that affects stability in the algorithm is the definition of the principal axis. The matching process is designed to ensure descriptions are stable. However, explicit tests need to be done to test the effect of noise and blurring on the determination of the principal axis.
- **sensitivity**: Experiments in chapter 4 established this aspect of the representation.

6.1 Future Work

- The SKS algorithm was tested for invariance to similarity transforms and robustness to noise and blur. However, the performance under occlusion was not evaluated. This is something that needs to be done.

- Furthermore, the algorithm needs to be compared in quantitative terms with some popular shape representation algorithms. Initial tests with Hu Moments [21] and Hausdorff distance [42] have been completed. However, thorough comparison against robust implementations of state of the art shape recognition algorithms is necessary.

- A neural network architecture for the SKS algorithm is discussed in chapter 5. An implementation of this architecture is essential. In particular the working of the F-layer and the Sigma-Pi connections in the M-layer needs to be verified.

- The computational complexity of the SKS algorithm can be further reduced if the curve is broken down into constant curvature segments. The curvature of the segments and their spatial position will suffice reduce the model size significantly. In initial experiments with this approach there was a ten fold reduction in model size utilizing this approach.

- One of the features that the cells in V4 [30] are shown to be sensitive to is shape context. Cells were found to be sensitive not only to the curvature of to which they were tuned but also to the curvature of the neighboring segments. This can be easily appended to the shape models. It will be worthy to see if the added complexity in representation improves the performance of the algorithm.

- A suggestion made by a member of the thesis committee was to utilize the product of curvature and distance as a feature. This would forgo the necessity to resize the object. The feature suggested is theoretically sound. However, its utility needs to be verified experimentally.

- Investigation into extending this work to arriving at shape representations for three-dimensional shapes should also be considered.

Bibliography

[1] D.H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[2] D.H. Ballard, Geoffrey E. Hinton, and Terrence J. Sejnowski. Parallel visual computation. *Nature*, 306(3):21–26, 1983.

[3] Jayanta Basak and Anirban Das. Hough transform network: Learning conoidal structures in a connectionist framework. *IEEE Transactions on Neural Networks*, 13(2):381–392, March 2002.

[4] Jayanta Basak and Sankar K. Pal. Psycop - a psychologically motivated connectionist system for object perception. *IEEE Transactions on Neural Networks*, 6(6):1337–1354, November 1995.

[5] E. Belogay, C. Cabrelli, U. Molter, and R. Shonkwiler. Calculating the hausdorff distance between curves. *Information Processing Letters*, 1997.

[6] Irving Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.

[7] C. Cadieu, M. Kouh, M. Riesenhuber, and T. Poggio. Shape representation in v4: Investigating position-specific tuning for boundary conformation with the standard model of object recognition. Technical Report CBCL Paper 241/AI Memo 2004-024, Massachusetts Institute of Technology, 2004.

[8] David Coeurjolly, Serge Miguet, and Laure Tougne. Discrete curvature based on osculating circle estimation. *Lecture Notes in Computer Science*, 2059:303–, 2001.

[9] David Coeurjolly and Stina Svensson. Estimation of curvature along curves with application to fibres in 3d images of paper. *Lecture Notes in Computer Science*, 2749:247–254, 2003.

[10] Charles E. Connor. *The Visual Neurosciences*, volume 2, chapter Shape Dimensions and Object Primitives, pages 1080–1089. The MIT Press, 2003.

[11] John G. Daugman. Complete discrete 2-d gabor transforms by neural networks for image analysis and compression. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36(7), July 1988.

[12] Francois de Vieilleville, Jacques-Olivier Lachaud, and Fabien Feschet. Maximal digital straight segments and convergence of discrete geometric estimators. *Lecture Notes in Computer Science*, 3540:988–997, 2005.

[13] J.-P. Debled-Rennesson, I. Reveilles. A linear algorithm for segmentation of digital curves. *International Journal of Pattern Recognition and Artificial Intelligence*, 9(4):635–662, 1995.

[14] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* Wiley Interscience, 2nd edition, 2000.

[15] Shimon Edelman. *Representation and Recognition in Vision.* The MIT Press, Cambridge, Massachusetts, 1999.

[16] Jacob Feldman and Manish Singh. Information along contours and object boundaries. *Psychological Review*, 112(1):243–252, 2005.

[17] K. Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119–130, 1988.

[18] G. E. Hinton. A parallel computation that assigns canonical object-based frames of reference. In *Proceedings of Seventh International Joint Conference on Artificial Intelligence*, volume 2, pages 683–685, Vancouver, BC, Canada, 1981.

[19] G. E. Hinton. Shape representation in parallel systems. In *Proceedings of Seventh International Joint Conference on Artificial Intelligence*, volume 2, pages 1088–1096, Vancouver, BC, Canada, 1981.

[20] P. V. C. Hough. Method and means for recognizing complex patterns. *U.S. Patent 3069654*, 1962.

[21] M K Hu. Visual pattern recognition by moment invariants. *IRE Trans. Information Theory*, 1962.

[22] David H. Hubel. *Eye, Brain and Vision.* W.H. Freeman and Company, 1995.

[23] David H. Hubel and Torsten Wiesel. *Brain and Visual Perception: The Story of a 25-Year Collaboration.* Oxford University Press, 2004.

[24] John E. Hummel. *Cognitive Dynamics: Conceptual Change in Humans and Machines*, chapter Where view-based theories break down: The role of structure in shape perception and object recognition, pages 157–185. Erlbaum, Hillsdale, NJ, 2000.

[25] Greet Kayaert, Irving Biederman, and Rufin Vogels. Shape tuning in macaque inferior temporal cortex. *The Journal of Neuroscience*, 23(7):3016–3027, April 2003.

[26] Reinhard Klette and Azriel Rosenfeld. *Digital Geometry: Geometric Methods for Digital Picture Analysis.* Morgan Kaufmann, 2004.

[27] V.A. Kovalevsky. New definition and fast recognition of digital straight segments and arcs. *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, 2:31–34, 16-21 Jun 1990.

[28] David Marr. *Vision.* W.H. Freeman and Co., San Francisco, 1982.

[29] Anita Pasupathy and Charles E. Connor. Responses to contour features in macaque area v4. *Journal of Neurophysiology*, (82):2490–2502, 1999.

[30] Anita Pasupathy and Charles E. Connor. Shape representation in area v4: Position-specific tuning for boundary conformation. *Journal of Neurophysiology*, (86):2505–2519, 2001.

[31] Anita Pasupathy and Charles E. Connor. Population coding of shape in area v4. *Nature Neuroscience*, 5:1332–1338, 2002.

[32] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, pages 1019–1025, 1999.

[33] M. Riesenhuber and T. Poggio. *The Visual Neurosciences*, volume 2, chapter How visual cortex recognizes objects: The tale of the standard model, pages 1640–1653. The MIT Press, 2003.

[34] Edmund Rolls and Gustavo Deco. *Computational Neuroscience of Vision*. Oxford University Press, 2001.

[35] T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, and T. Poggio. A theory of object recognition: computations and circuits in the feedforward path of the ventral stream in primate visual cortex. Technical Report CBCL Paper 259/AI Memo 2005-036, Massachusetts Institute of Technology, 2005.

[36] T. Serre, L. Wolf, and T. Poggio. A new biologically motivated framework for robust object recognition. *MIT AIM*, 2004.

[37] Roger N. Shepard and Jacqueline Metzler. Mental rotation of three-dimensional objects. *Science*, 171(3972):701–703, Feb. 1971.

[38] Wesley E. Snyder and Hairong Qi. *Machine Vision*. Cambridge University Press, 2003.

[39] Michael J. Tarr and Heinrich H. Bülthoff. Image-based object recognition in man, monkey, and machine. *Cognition*, 1998.

[40] Anne Vialard. Geometrical parameters extraction from discrete paths. In *DCGA '96: Proceedings of the 6th International Workshop on Discrete Geometry for Computer Imagery*, pages 24–35, London, UK, 1996. Springer-Verlag.

[41] Marcel Worring and Arnold W. M. Smeulders. Digital curvature estimation. *CVGIP: Image Underst.*, 58(3):366–382, 1993.

[42] Dengsheng Zhang and Goujun Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37(1):1–19, January 2004.

Appendix

Appendix A

Estimation of Curvature using Digital Straight Segments

A.1 Discrete Curvature Estimation

A complete analysis of the various discrete curvature estimation techniques is available in the paper by Worring and Smeulders [41], which concludes by recommending that the best method to find discrete curvature is by differential filtering of tangent angle using a Gaussian kernel.

$$k(i) = \frac{\theta(i) * {G'}_{\sigma}}{1.107} \tag{A.1}$$

$$\theta(i) = \tan^{-1}\left[\frac{y_{res}(i+1) - y_{res}(i)}{x_{res}(i+1) - x_{res}(i)}\right] \tag{A.2}$$

x_{res} and y_{res} are the resampled versions of the points on the curve (x, y). 1.107 is the average distance between two points on a discrete contour and is a multiplicative bias correction. The resampling is done on a straight line joining two adjacent points on the curve. As [41] states such a procedure will reduce the errors due to non-uniform sampling.

Vialard [40] suggests an improvement to the above method where the tangent is estimated in a purely discrete way using Digital Straight Segments. However, the biggest problem with these methods is the estimation of the Gaussian parameter σ which is the standard deviation. σ has to be determined based on the nature of the data.

Coeurjolly, Miguet, and Tougne [8] proposed a method where the curvature estimation is done using a purely discrete way using osculating circles which avoids any parameters based on the nature of the data. This method is explained in detail in the next few sections.

A.2 Digital Straight Segments

A Digital Straight Segment(DSS) is defined using two support lines as follows:

$$D_{a,b,\mu,\omega} = \{(i,j) \in Z^2 : \mu \leq ai + bj < \mu + \omega\} \tag{A.3}$$

where a/b is the slope. a and b are relatively prime integers. μ is the approximate intercept and ω is the arithmetic width.

If $\omega = max(|a|, |b|)$, then the lines are called *naive* lines. These lines are 8-connected lines in Z^2. If $\omega = |a| + |b|$, then the lines are called *standard* lines. These lines are 4-connected lines. The term connected here implies adjacency. The connectivity is not absolutely necessary. A set of points are part of a straight segment as along as they lie within the support lines. Therefore, if two points are not connected and lie within the support lines then it is possible to find 4 or 8 connected points joining them.

It is interesting to note that *standard* lines are a bit more resistant to noisy points than *naive* lines because of greater separation between the support lines. *Standard* lines are used for curvature estimation in our algorithm.

There are two important algorithms found in the literature to find Digital Straight Segments. For *standard* lines, the algorithm is defined in [27]. Recognition of *naive* lines is discussed in [13]. The next sub section describes the algorithm to find *standard* lines as described in [27].

A.2.1 Algorithm for Recognition of *Standard* Lines

The algorithm was initially proposed in [27]. The description given below is based on [26]. The algorithm is based on the principle that a set of points (x_i, y_i) are part of a DSS if:

$$0 \leq bx - ay + c \leq |a| + |b| - 1 \tag{A.4}$$

where a, b and c are integers with a and b being relatively prime. The algorithm assumes that the points are 4-connected.

1. $p_N = q_P = (x_1, y_1), q_N = p_P = (x_2, y_2), a = x_2 - x_1, b = y_2 - y_1$
2. $n = 3$
3. Repeat steps below until (x_n, y_n) are not DSS.
4. $c = b * x_{n-1} - a * y_{n-1}$
5. $r_n = (x_n, y_n)$
6. $h = b * x_n - a * y_n - c$
7. if $0 \leq h \leq |a| + |b| - 1 :$ (x_n, y_n) is a part of the DSS.
8. if $h = -1$ then $q_N = r_n, p_P = q_P, (a, b) = r_n - p_N$. (x_n, y_n) are part of the DSS.
9. if $h = |a| + |b|$ then $q_P = r_n, p_N = q_N, (a, b) = r_n - p_P$. (x_n, y_n) are part of the DSS.
10. Otherwise, (x_n, y_n) are not DSS. Stop at previous vertex (x_{n-1}, y_{n-1}).

A.3 Estimation of Curvature and Tangents Using DSS

Digital Straight Segments can be used to estimate discrete curvature. One of the easiest and simplest way to do this is defined in [9]. Let $v_1 = (x_1, y_1)$ and $v_2 = (x_2, y_2)$ be the end points of the maximum length DSS around point v_0 on the curve. Note that v_1 and v_2 are also points on the curve. Let a be the Euclidean distance between v_0 and v_1. Similarly, let b be the distance between v_0 and v_2 and c be the distance between v_1 and v_2.

Then, the radius of the osculating circle at v_0 can be approximated by finding the radius of circumcircle(R_c) of the triangle with vertices v_0, v_1 and v_2.

$$R_c = \frac{abc}{4A} \tag{A.5}$$

$$A = \frac{\sqrt{(b+c)^2 - a^2}.\sqrt{a^2 - (b-c)^2}}{4} \tag{A.6}$$

A is the area of the triangle. Curvature at v_0 is, therefore, approximated by:

$$\kappa = \frac{1}{R_c} \tag{A.7}$$

The line joining v_1 and v_2 is the approximation of the tangent at v_0.

A.4 Experiments and Results

In this section, we test the accuracy of the DSS curvature estimator. 11 circles of radii 5,10,20,30,40,50,60,70,80,90 and 100 were taken and their curvature was estimated using the algorithm. A plot of the average estimation error versus radius is shown in A.1. The plot clearly shows the convergence of the average curvature to the true continuous curvature with increase in resolution. However, the convergence is true only for average curvature. The curvature at a point does not have asymptotic

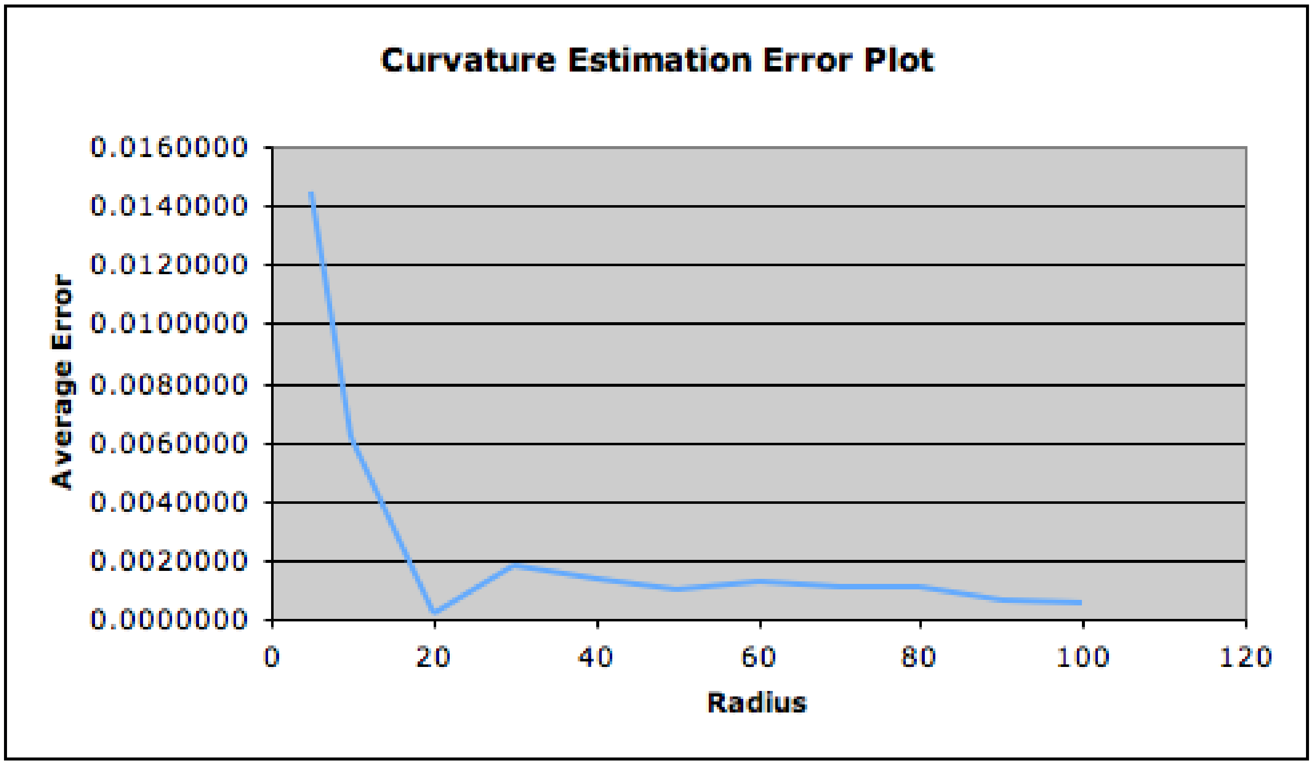

Figure A.1: Average Curvature Estimation Error

convergence. According to [12], the asymptotic convergence of a DSS curvature estimator is still an open problem.

However, the advantages of DSS curvature estimators clearly outweigh this disadvantage. The biggest advantage of DSS based curvature estimators is that they are independent of the nature of the data when compared to classical techniques. The algorithm is highly parallel and requires no parameters to set, giving very good accuracy at the same time.