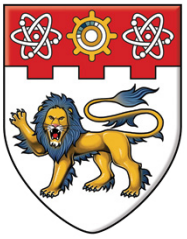
Exploiting Shape Properties for Improved Retrieval, Discrimination and Recognition



NANYANG TECHNOLOGICAL UNIVERSITY

by

Vittal Premachandran

Thesis submitted in partial fulfillment of the requirements for the award of

Doctor of Philosophy

in the

School of Computer Engineering

February 2014

To my parents.

"If we knew what it was we were doing, it would not be called research, would it?"

- Albert Einstein

Acknowledgements

It is a great pleasure for me to thank the people who have helped me during my PhD. First and foremost, I am indebted to my supervisor, Dr. Ramakrishna Kakarala. I am thankful to him for the amount of independence he gave me during my PhD, which allowed me to grow as a researcher. I would also like to thank him for the confidence he has showed in me and the patience that he had during some of the difficult times in my research. I am also thankful to him for always keeping his door open for me to discuss my research problems, many of which have gone on for long periods of time. Finally, I am grateful for the support and motivation that he has provided during the course of my PhD.

Secondly, I would like to thank my parents, V. R. Premachandran and Priya Premachandran, for all the help, motivation, love and support that they have provided. I am thankful to them for supporting me in my decision to embark on doctoral studies. It would have been impossible to complete this thesis without the confidence that I had the support, and help, of my parents during the entire period of my PhD.

Finally, I would like to thank my friends, and colleagues, especially Ishtapran Sahoo, Karthik Muthuswamy, Nikhil Narayan, Prabhu Kaliamoorthi, Prashant Goswami and Pravin Kakar for the various technical and non-technical discussions that we have had.

Contents

List of Figures

List of Tables

Abbreviations

AP	**A**verage **P**recision
CBIR	**C**ontent-**B**ased **I**mage **R**etrieval
DN	**D**ominant **N**eighbors
DR/FPPI	**D**etection **R**ate v. **F**alse **P**ositive **P**er **I**mage
HOG	**H**istogram of **O**riented **G**radients
IDSC	**I**nner **D**istance **S**hape **C**ontext
k-NN	**k**-**N**earest **N**eighbors
LCDP	**L**ocally **C**onstrained **D**iffusion **P**rocess
MRF	**M**arkov **R**andom **F**ield
P/R	**P**recision v. **R**ecall
SC	**S**hape **C**ontext
SSC	**S**olid **S**hape **C**ontext
SVM	**S**upport **V**ector **M**achine
TPG	**T**ensor **P**roduct **G**raph
TRW-S	**S**equential **T**ree **R**e-**W**eighted Message Passing

Abstract

Recognition of categories of objects is one of the central problems of computer vision. The human visual system has an unmatched ability to recognize objects across multiple modalities and appearances. Recognition of objects via their shapes is one of the primary reasons why humans are able to perform well in vision-related tasks. However, automatic algorithms that try to recognize objects are far from such ability. Therefore, this thesis concentrates on developing better representations of shapes to aid in object retrieval and object detection.

We begin by questioning the implicit assumption that all of the shape information lies in the contours, and show that making use of the interior properties of the shapes will produce better results while matching shapes. To this end, we introduce a novel descriptor, namely, the Solid Shape Context.

We then hypothesize that not all parts of the shape are equally important while extracting the shape properties, and try to identify the most discriminative parts. We extract the most discriminative parts of a shape by comparing each shape to its closest rivals and propose a means to improve the discriminative capability of standard shape descriptors.

We then propose a simple and intuitive way to obtain robust neighborhoods, which help in computing the "true" distances between shapes. This is achieved by mining additional information that was, until now, unidentified. In addition, we provide soft probabilistic measures for the inclusion or removal of a node from a local neighborhood. The ability to measure confidence of nodes being a part of the neighborhood was not explored till now. This work opens many avenues for future research in the rapidly growing field of retrieval.

Finally, as an unifying work, we propose a framework for performing shape-based object detection in real-world images, which also allows for the identification of object parts. This work bridges the gap between two independent, but actively researched, threads in the field of object detection. We propose a structured prediction approach for predicting object part labels, where the label of each part gets influenced by its neighboring parts.

Chapter 1

Introduction

The foundation of many image analysis functionalities usually boils down to object recognition and detection. The importance of object recognition and object detection in automated computer vision techniques cannot be underestimated. Be it in a face recognition feature of a camera, or a smile detection algorithm, object recognition plays an important role. There are many applications that benefit from object detection. A few examples include image retrieval (important in image-based web search), robot navigation (localizing objects is key in avoiding them), medical image analysis (matching shapes to detect abnormalities in organs), and surveillance (tracking objects in videos).

The human brain has an uncanny knack for matching shapes, visualizing object instances, ignoring background noise, and localizing objects in the wild. The motivation to achieve human-like performance has fueled researchers to propose various ways (rigid templates, color models, texture matching, etc.) to solve the problem of object recognition; many of which have resulted in significant advancements in the field. Though recent advances in computer vision have enabled the field to grow tremendously, we are nowhere close to achieving human-like performance.

One area that has recently gained attention is that of shape-based object recognition. Shapes are robust properties of objects that are sufficiently invariant to variations in lighting conditions, while still retaining the ability to enable generic object recognition. However, shape matching and shape-based object recognition is a hard and unsolved problem with lots of challenges to overcome. This thesis explores some of the issues with shape matching and proposes ways to improve upon the state-of-the-art algorithms. Before delineating the list of problems and

potential solutions, it would be best to understand the motivation behind the use of shapes in vision problems.

1.1 Motivation: Why use Shapes?

The shape of an object located in some space is a geometrical description of the part of that space occupied by the object. Kendall writes in his paper [45], "We here define 'shape' informally as 'all the geometrical information that remains when location, scale and rotational effects are filtered out from an object'". In many recognition instances, the object's shape plays a very important role. Often times, shape alone is sufficient to recognize an object. Color and texture are just attributes of the object that add-on to an object's characteristic. Consider the example shown in Figure 1.1. Even though there is no color and texture on the body, the two objects can easily be recognized as cars.

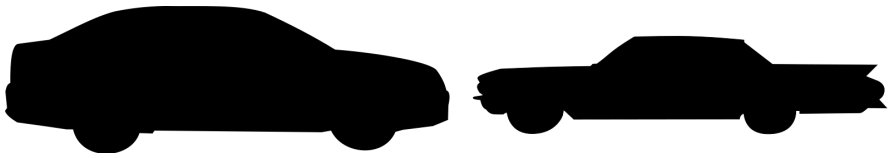


FIGURE 1.1: **Car Silhouettes** - The objects shown above can be recognized as cars even though it is just their silhouette that is visible (Source: Google Images).

Moreover, even among images that are defocussed, or the ones with lots of background clutter, the shape of an object stands out as the most prominent object characteristic. See examples in Figure 1.2.

In addition, shapes are so powerful that even partial shape information is sufficient to identify the objects. Look at the examples in Figure 1.3. It is quite apparent that the contours belong to that of a human face, a horse and an airplane (from left to right), even though significant portions of it are missing. This observation that humans can recognize objects using just fragments of contours is also corroborated by psychophysical studies such as the ones by Biederman and Ju [15].

Apart from recognition, other areas of computer vision rely on shape properties as well. Image segmentation algorithms rely heavily on object boundaries for crisp segmentation; edge maps are used to allow label transition while performing segmentation [16]. Contours play an important role in object localization. Areas

(A) Defocus (B) Clutter

FIGURE 1.2: The characteristic shape of the Eiffel tower can be recognized even though the image is defocussed and the shape of the starfish makes it stand out from background clutter (Source: Google Images).

FIGURE 1.3: Partial external contours are sufficient to recognize objects as human ace, horse and airplane (left to right). This shows the importance of shape information over color and texture information (Source: [88]).

such as tracking and pose estimation make use of shape priors as constraints during optimization [18]. Finally, in order to perform generic object recognition, shape is the most consistent object property that can be found. Objects such as cars, bottles, mugs, etc., can vary in their body color and texture, but possess a uniquely identifiable shape.

1.2 Issues to tackle

While shape matching may seem pretty straight forward for humans, it is an extremely hard problem to automate. What constitutes a shape? How to encode shape properties into features? What are the best ways to perform shape matching? Is a shape uniformly discriminative all across the body? How can we go beyond pairwise matching of shapes and leverage information from other similar

shapes to get better dissimilarities? How can an object's shape be used to recognize it in the wild? Questions such as these are open questions in the community and need answering.

This thesis aims to provide insights into four of these fundamental questions, which are listed below.

1. How can we take motivation from human perception and come up with better descriptors for matching visually similar shapes?
2. Do all parts of a shape contribute equally to its identification? Are there some parts that are more important and discriminative than others? If so, how do we identify such parts?
3. Can we leverage the abundance of data that is now available and learn "true" dissimilarities between shapes? If so, how can we do this in a robust fashion with as little supervision as possible?
4. Object shapes are not some random entities; they are highly structured in their appearance. So, is there a way to learn this structure given training data and how can the learned statistics be used for the identification of new object instances?

1.3 Applications

One might be tempted to ask what potential applications benefit from answers to the above questions. With 1.5 billion camera-based devices (phones with cameras, point and shoot cameras, DSLRs, surveillance cameras, etc.) sold in 2011 alone, and with 300 million photos being uploaded to Facebook each day, it is becoming increasingly important to develop robust vision-based applications (apart from other miscellaneous applications) that make sense of visual data. Two major vision applications that can benefit from the study of shape analysis are given below.

Content-Based Image Retrieval (CBIR): Many image retrieval systems make use of the surrounding text, or image tags, to retrieve photographs. Text-based retrieval systems have their drawbacks as it is extremely time consuming to manually tag photographs and/or describe them succinctly. Visual data, on the other

hand, provides rich information about the scene. Therefore, content-based image retrieval has recently gained much importance in the community and is being actively researched upon. Google Image Search is a product that relies on CBIR. As was motivated in Section 1.1, object shapes provide a rich source of visual information, and it is still an open question as to how shapes can be represented and used in shape-based retrieval systems. In chapters 3 and 5, we talk about retrieval in much more detail and show experiments on a standard shape-based retrieval database.

Object Detection: Apart from retrieval, it is also important to "understand" the visual content in a photograph. Google Goggles, for instance, tries to recognize objects in a scene and provide more information about them. Figure 1.4 shows many examples of recognizing objects. Performing such tasks will allow us to extract higher-level semantics from images. In Chapter 6, we will explain the detection process in detail and show how shape can be used in a structured fashion to recognize objects in the wild.

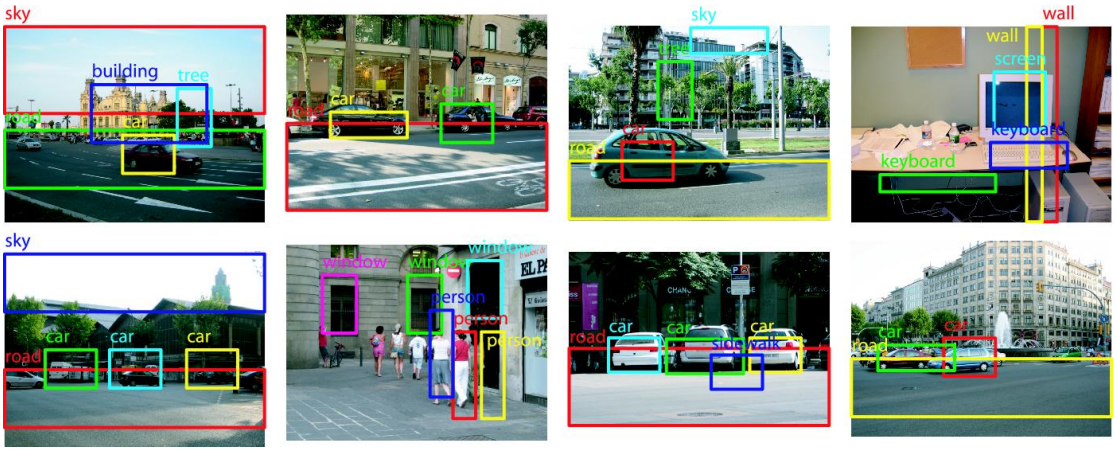


FIGURE 1.4: Example of recognizing objects from images (Source: [82]).

Miscellaneous Applications: Apart from the above mentioned major applications that we look at in this thesis, there are many other applications that benefit from shape analysis and shape matching. Many industrial testing algorithms look at the shapes of machinery to ascertain the presence/absence of defects. Ex: A lack of smoothness in the object contours, for instance, can help identify cracks in delicate parts.

Object contours are also used in medical imaging applications to check for abnormalities in organs. Parameters related to the contours of organ segments such as perimeter, area, curvature, etc. are used to check for tumors in body parts.

1.4 Methods and Approaches

Having looked at the exciting potential applications, and having identified the questions that need answering, it would be worthwhile to get a glimpse of the approaches that currently exist in the literature. The important areas that must be addressed while using shape are, shape extraction, representation and matching. In addition, for object detection, localization based on object contours also play an important role.

1.4.1 Shape Extraction

The standard way to extract an object's shape is by the using edge detectors. Edges constitute the extremities of the object and contains most of the shape information. The most common edge detection algorithm that is used is the Canny edge detector [22]. More recently, state-of-the-art edge detectors such as the Pb Edge Detector [63] (also known as the Berkeley edge detector) have gained importance in the community. Though such edge detectors are much slower than the Canny edge detector, their superior performance in terms of extracting semantically meaningful contour fragments, far outweighs the issues such as execution speed. Linking the extracted edges by using an edge linking algorithm also helps in developing better representations of shapes. For the experiments in this thesis, the Pb edge detector is used for extracting edges from real-world images.

1.4.2 Shape Representation

One of the most important parts of shape analysis is in coming up with a good representation for shapes. Significant amount of work has been done in this area and the literature is filled with proposals for extracting features from shapes. The fact that there is still no consensus on which shape feature to use shows the difficulty in coming up with good features for representing shapes. Shape features vary between global features and local features, with many intermediate versions in between. Purely global features such as Fourier descriptors cannot capture deformations in shapes, while purely local features such as corners or key-point based features cannot capture the important non-locality information of shapes.

Ever since the innovation of SIFT [59], histogram-based approaches have found a fair amount of success even in shape representation. Features such as Shape Context [13] has shown promise in capturing the shape properties, and many variants of it have since been proposed. However, they still have some issues that need to be solved such as robustness to indentations, etc. Due to the importance that features play in further steps, such as shape matching, we study some of the issues in coming up with a good shape descriptor in this thesis.

1.4.3 Shape Matching

After extracting statistics about shapes from the objects (i.e. features), in order to find similarities or dissimilarities between pairs of objects, some sort of a matching of the respective features have to be performed. Matching of shapes can depend on how the features were extracted and the nature of the features. Chamfer matching is used for direct matching of shape templates. Bipartite graph matching is utilized in some cases for matching respective features. If order preserving constraints can be incorporated, more efficient techniques such as dynamic programming can be used to match the respective features.

1.4.4 Model Representation

For a more robust representation of shapes, objects are modeled to account for deformations due to appearance or articulations. Early techniques included representing objects in terms of high-level representations using 2D primitives (lines, rectangles, etc.) for 2D object silhouettes, or 3D primitives (cylinders, conic sections, etc) for 3D objects (ex: humans). These models are now considered to be crude and the more recent trend in the literature is to make use of statistical machine learning techniques to learn structured probabilistic graphical models of object templates. One of the recent and highly-acclaimed template models is the deformable parts model [31], which makes use of abstract object parts connected using spring models, which allows for deformations in objects.

1.5 Outline and Contribution

The above discussion of methods related to shape extraction and matching shows that there are many different areas that need attention. This thesis investigates the questions outlined in the Section 1.2 and provides potential solutions to the solve the listed problems. Before delving into the proposed solutions, in Chapter 2, we give a background of shape matching techniques in the areas of shape-based retrieval and shape-based object detection. We list some of the milestones in the literature and point out their drawbacks.

In Chapter 3 of this thesis, we question how shape matching can be taken beyond the matching of respective contours. Does the object's contour contain all of the shape information, or are there other aspects to a shape that have been ignored? If so, how can we develop better descriptors for matching visually similar shapes? The investigation of these questions led to the development of a new perceptually-motivated shape descriptor, which has proved to overcome the limitations of the current-day shape matching techniques. The contributions in this part of the work involves the development of a novel approach to matching shapes that goes beyond the traditional approach of matching respective contours. We identified that contours, alone, are not sufficient, and sometimes, even, misleading, while matching shapes. This realization led to the development of an intuitively simple shape descriptor, namely, Solid Shape Context, which takes into consideration, not only the layout of the contour, but also the interior properties of the shape. This descriptor is immune to indentations in the object's contour, thus providing an acceptable match between visually similar objects having vastly different contour properties. The use of Solid Shape Context proved to significantly improve retrieval results on standard shape matching databases.

The question of whether all parts of a shape contribute equally to its identification is an extremely intriguing question. Are there some parts of a shape that are more discriminative than others? Most current-day shape descriptors have an implicit assumption that all parts of an object are equally important in identifying the object. However, we hypothesize that this is not the case. We surmise that different parts of the shape have different levels of importance and set out to learn a probabilistic importance map of the object whose domain is its contour. We learn the distribution of importance by making use of nearest neighbor algorithms on object parts. Surprisingly, we found that our method of trying to identify

important parts was able to identify semantically meaningful and discriminative parts of the object. An importance sampling routine on these heat maps led to the development of discriminative shape descriptors, which had better discriminative capability among confusing classes. This work is explained in detail in Chapter 4 of the thesis.

Performance of shape-based object retrieval systems can be significantly improved by going beyond the pairwise analysis of shapes. In today's world, where there is abundance of data, it makes sense to leverage information from other similar shapes while performing object retrieval. Similar shapes can be made to influence object retrieval by making use of label propagation techniques. However, propagating such similarity information requires careful selection of a local neighborhood, which has proven to be a difficult task in the unsupervised setting. Therefore, we propose a simple technique to select a robust local neighborhood by extracting more information from the standard k-Nearest Neighbors (k-NN) selection criterion. We propose to do this by utilizing the consensus information that can be easily collected from multiple rounds of k-NNs. In Chapter 5 of this thesis, we provide the intuition behind the approach and explain the process in detail.

With the above improvements provided to the area of shape-based object retrieval, we move on to the area of shape-based object detection. Recognizing objects in the wild is an extremely difficult problem and we propose a structured prediction approach to facilitate shape-based object recognition. We leverage the fact that objects are highly-structured entities, and therefore make use of structured prediction techniques for object part labeling. The object is modeled as a probabilistic graphical model with each node in the model representing a part of the object. We make use of variational techniques for approximate inference on the fully-connected graphical model. The unary potentials for nodes and higher-order potentials for interactions are designed based on the insights that were collected from the previously described contributions. We describe this work in detail in Chapter 6 of the thesis.

Chapter 7 summarizes the works in this thesis and provides a conclusion to the same. It also provides directions for future extensions of the works in this thesis and a high-level direction of where the field should head in order to solve some of the most interesting and important problems that exist in the field of computer vision.

Some of the material in this thesis has appeared elsewhere, including [77], [76], [78], [79] and [43].

Chapter 2

Background

Matching of shapes involves finding out how similar the given shapes are to each other. Such matching can be done either by using distance-based techniques or by using feature-similarity matching. Distance-based shape matching primarily involves matching two shapes using some sort of a distance metric. If the "distance" between the two given shapes is less, then the shapes are said to be similar to each other. Section 2.1 explains this technique in detail. Feature-similarity or descriptor-based shape matching involves the matching of the extracted shape features. Moreover, feature-based matching techniques also have to solve the correspondence problem. Section 2.2 gives a detailed discussion of feature-based matching techniques.

2.1 Distance-Based Shape Matching

Given two shapes, their similarity could be computed using certain distance-based techniques such as the Hausdorff distance [40] or the Chamfer distance [17, 88]. These distance-based techniques have been previously used for template matching. The smaller the distance value, the more similar the shapes are to each other. The Hausdorff distance, $h(.,.)$, between two shapes A and B, is computed as,

$$h(A,B) = \max_{a \in A}\{\min_{b \in B}\{d(a,b)\}\}, \tag{2.1}$$

where a is a point in the set A, b is a point in the set B, and $d(a,b)$ is the distance between the point a and point b. The distance metric can be Euclidean distance

or any other distance metric, based on the application. For shape matching, the most common distance metric that is used is the Euclidean distance. The Hausdorff distance, $h(A, B)$, is usually not symmetric i.e., $h(A, B) \neq h(B, A)$. Also, $h(A, B) = 0$ does not imply that $A = B$. It only implies that $A \subseteq B$. Therefore, the more common form of the Hausdorff distance that is used is given as,

$$H(A, B) = \max\{h(A, B), h(B, A)\}. \tag{2.2}$$

Early implementations of these techniques for matching shapes, were not scale and rotation invariant [4]. Moreover, the Hausdorff distance is not invariant to translation either.

Chamfer distance is another distance metric, which was first introduced by Barrow et al. [7]. It is defined as the average of the distances from a nearest point in a set B to all the points in a template set A. The Chamfer distance, d_{cham}, between two sets A and B is given as,

$$d_{cham}^{A,B} = \frac{1}{|A|} \sum_{\mathbf{x}_a \in A} \min_{\mathbf{x}_b \in B} ||\mathbf{x}_a - \mathbf{x}_b||_2 \tag{2.3}$$

where $|A|$ is the number of points in the set A, $\mathbf{x}_a$ is a point in set A and $\mathbf{x}_b$ is a point in set B. The above equation is one of the most basic versions of the Chamfer distance. As in the case for Hausdorff distance, it can be seen from Equation 2.3 that the distance metric does not account for similarity transformations.

2.2 Feature-Based Shape Matching

The distance-based matching that we just saw in the previous section is not invariant to basic transformations and is susceptible to output false distances in the presence of noise. Therefore, recent research has moved towards feature-based matching of shapes. The question that was being asked about a decade ago is what kind of features to use. Primitive-based features were the most favorable direction at that point in time and was actively researched upon since primitives provided a somewhat semantic description of object parts.

Corner detectors gathered some momentum as they were easy to compute and since corners were considered to be the "important" locations in the object's boundary. Some notable corner detectors include the Harris corner detector [38], SUSAN

corner detector [90] and the FAST corner detector [95]. The appeal of higher-order primitives have also resulted in other higher-order primitive detectors such as the Line Segment Detector [99] and rectangle detectors [42]. However, such detectors always make us ask whether the specific primitive under consideration is the best primitive to use in order to describe object shapes.

The fact that the above question has, to date, not be answered, and the fact that we find it difficult to describe a shape in words, shows the abstract nature of shapes. Bridging the gap between higher-level object semantics and lower-level image pixels has been an extremely difficult task and is a largely unsolved problem. The difficulty in developing semantically meaningful features is more-or-less accepted in the community and there is now a wide-spread interest in the use of statistical techniques. The following sections give a sense of how the field has progressed in the past decade. The evolution of shape features and shape matching techniques are laid out in the rest of the chapter.

2.2.1 Shape Descriptors

Early work used shape silhouettes as the input data and tried to extract descriptors out of it. Fourier descriptors [74, 110] are an example of such shape descriptors. Fourier descriptors consider the shape boundary as a periodic function and represent it using the frequency coefficients. A lack of invariance of the Fourier descriptors to shape deformations has somewhat diminished the interest in this area. Following the success story of the SIFT feature descriptor [59, 60], lots of attention has been diverted towards the development of histogram-based shape features.

Shape Context: A major milestone in the advancement of shape descriptors came from the work of Belongie et al. [11]. They proposed a histogram-based shape descriptor namely, Shape Context (SC), which effectively captured the shape properties of an object. The SC descriptor is computed at a finite set of points, which are uniformly distributed across the shape's boundary. The first major contribution came from the claim (verified experimentally) that these set of finitely sampled points need not be corner points or "special" in any way. The experimental results validated this claim and showed that there was nothing "special" about an object's corner points. Two notable findings have paved way for the development of statistical shape descriptors. Firstly, the subjectivity in the definition

of corner points made it hard to develop corner detectors that produced semantically meaningful points on the contours, thus diminishing interest in this area. Secondly, the claim that the corner points did not contain any "special" value also diverted the attention from developing more sophisticated corner detectors onto the development of more robust histogram-based shape descriptors.

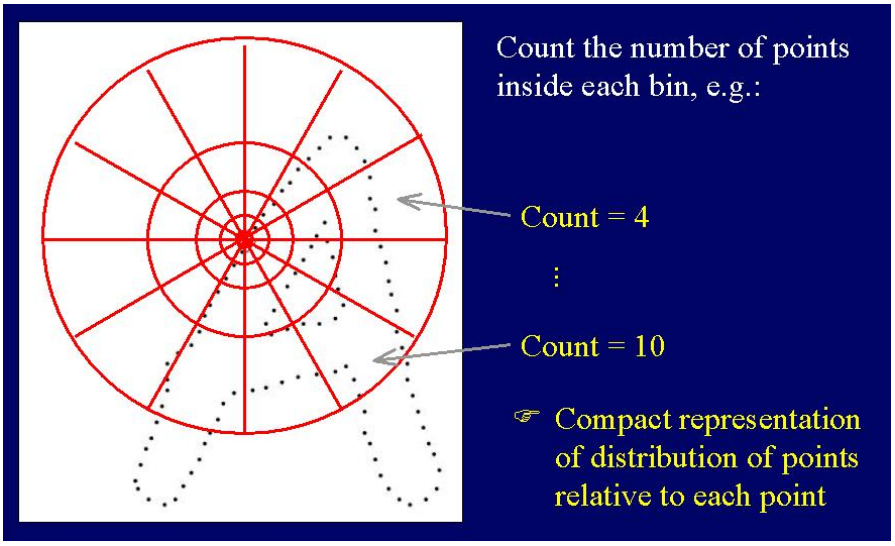
The Shape Context is basically a 2-D histogram of distances and angles. The distances are obtained by calculating the Euclidean distance between the point under consideration and every other sampled point. Similarly, the angular information between any two points is obtained by finding the angle between the tangent drawn at the point under consideration and the displacement vector between the two points. Armed with the distance and the angular information, a histogram is then created at each sampled point, which is the shape context histogram of that point. Similar such shape contexts are generated for every other sampled point on the object's boundary. Furthermore, the binning is done in such a way that points that are close to each other are given more importance than the points that are far away from each other. To specify such importance, instead of uniform binning, the authors of [11] adopt a log-polar style of binning.

Figure 2.1a gives an example of the shape context at a particular sampled point on the character 'A'. The red circular disc is a log-polar histogram with twelve angular orientations and five distance bins. Therefore, the SC descriptor is a 60-D feature vector at each sampled point. The value in bin (i, j) is the number of points that fall in the intersection of the i-th distance bin and the j-th angular bin. Figure 2.1b shows how points that are at similar locations on different instances, belonging to the same class, have similar shape contexts, as compared to the shape contexts of the points at different locations.

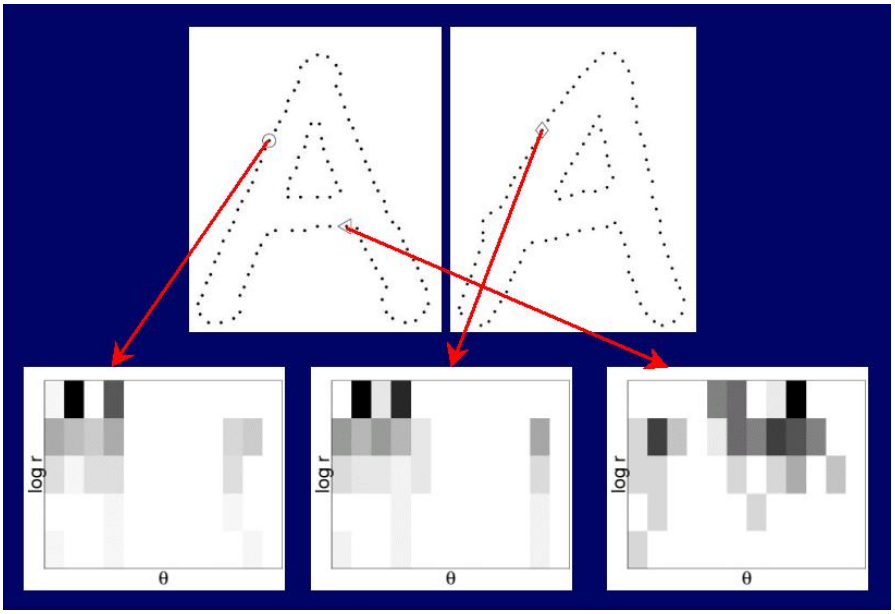
More formally, we could define the shape context as follows. Suppose we are given a shape, S, we first sample the shape into a set of n equally spaced points, $S = \{p_1, p_2, ...p_n\}$. Each point p_t, is a two dimensional vector, with the x-coordinate and the y-coordinate as its dimensions. The histogram at point p_t is defined as

$$\mathcal{H}_t^S(k) = \#\{p_s : p_s \in bin(k)\}, \tag{2.4}$$

where the value in the k-th bin is the number of points, p_s, falling into $bin(k), \forall s \in \{1, ..., n\}$. The shape context and the matching stages have since been improved by the original authors leading to a series of publications [12–14, 67]. The Shape



(A)



(B)

FIGURE 2.1: (a) The red circular disc is a log-polar histogram with twelve angular bins and five distance bins. (b) The shape context at points in similar locations in different characters belonging to the same class have similar looking shape contexts, as compared to the shape contexts of the points at different locations (Source: [10]).

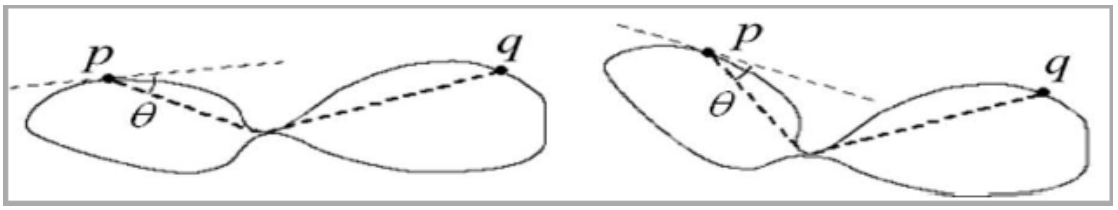


FIGURE 2.2: The inner distance between any two points, p and q, is the shortest distance along the path joining the two points, such that the path lies completely within the boundary of the object. The inner angle is the angle between the tangent at the first point and the first part of the path leading from the first point, p, to the second point, q (Source: [55]).

Context does have a lot of advantages. Since the histogram is composed of relative distances, it is inherently invariant to translations. The angles are computed relative to the tangent at the point under consideration, thus making it invariant to rotations as well. In order to make the descriptor invariant to scale, the distances are all normalized by the mean of the distances, before binning.

With all its advantages, the Euclidean distance that the SC uses does not allow it to be invariant to articulations in shapes. This problem led to the next major milestone in the form of Inner Distance Shape Context (IDSC).

Inner Distance Shape Context: Ling and Jacobs [54, 55] identified the lack of invariance of the Shape Context to articulations and proposed a variant of the SC in the form of Inner Distance Shape Context. The IDSC at a point is also made of the same two ingredients; distances and angles. However, it differs in the manner in which the distances and angles are calculated. For any two given points, p and q, the inner distance between the two points is calculated as the shortest distance along the path joining the two points, such that the path lies completely within the boundary of the object. The angle between the points is the angle between the tangent drawn at point p, and the first part of the path leading to point q. The authors name the angle as the inner angle. Ling and Jacobs found that their descriptor worked well with 8 distance bins, as opposed to the 5 distance bins that was used in the original SC. Therefore, the IDSC is a 96-D descriptor at each sampled point on the object's boundary. Figure 2.2 gives an illustration of how the inner distance and the inner angle is calculated.

In addition to the new descriptor, Ling and Jacobs also followed the idea suggested by Thayanathan et al. [94] to make use of the continuity constraints of the sampled points. The continuity constraints basically states that, on an object contour, a point p_i neighbors just two other points p_{i-1} and p_{i+1} and that they follow a

sequential ordering. This observations allows for the use of efficient matching techniques such as dynamic programming, thus allowing for a quick matching of shapes.

Non-Planar Shapes: A final variant of the SC that is worth mentioning is that of Gopalan et al. [37]. IDSC assumes planarity of shapes and thus, it does not perform well when the silhouettes under consideration are non-planar projections of 3-D objects. This issue was identified by Gopalan et al. and they propose to first affine-normalize the object parts before calculating the IDSC. They assume that the parts are near-convex regions of the shape boundary and propose an algorithm to extract all near-convex regions. They then affine normalize all the parts and then use IDSC to compute the shape features.

3-D Shape Descriptors: While this thesis concentrates predominantly on 2-D shape retrieval and 2-D shape descriptors, for completeness, we would like to mention some works in 3-D shape descriptors as well. Kokkinos et al. [47] have extended the 2-D shape descriptor, Shape Context, to help describe 3-D shapes. Their descriptor is still made distances and angles. However, their distances are intrinsic to the shape. Therefore, they name their 3-D shape descriptor as Intrinsic Shape Descriptor. Kalogerakis et al. [44] try to learn an automatic segmentation algorithm on 3-D shapes and use a list of features (curvature, principal components, shape diameter, average geodesic distance, orientation features, etc.) to train a discriminative labeling algorithm using Conditional Random Fields (CRF). The area of 3-D shape descriptors and retrieval is quite new and has lots of potential for future work.

2.2.2 Shape Matching

Once the shape features have been computed, matching respective shapes boils down to matching their respective features. The cost of matching a point p_i in shape, S_1, to a point, q_j, in shape, S_2, is the cost of matching the shape contexts at the two points. Shape contexts, being histogram-based, allows for the use of standard histogram matching functions such as the χ^2 test statistic,

$$C_{ij} \equiv C(p_i, q_j) = \frac{1}{2} \sum_{k=1}^{K} \frac{[\mathcal{H}_i^{S_1}(k) - \mathcal{H}_j^{S_2}(k)]^2}{\mathcal{H}_i^{S_1}(k) + \mathcal{H}_j^{S_2}(k)}, \tag{2.5}$$

where C_{ij} is the cost of matching point p_i to point q_j. Given all pairwise matching costs, the objective is to minimize the total cost of matching,

$$\Psi(\pi) = \sum_i C(p_i, q_{\pi(i)}), \tag{2.6}$$

where $\pi(.)$ is a one-to-one mapping function that maps points in the first shape to the points in the second shape. Standard bipartite graph matching algorithms such as the Hungarian algorithm [69] were previously used to find the optimal matches. However, as stated in the previous section, adding figural connectivity constraints allows us to make use of more efficient algorithms such as dynamic programming.

2.3 Partial Shape Matching

The previous section described the well-known Shape Context shape descriptor and its variants. The SC, being a 2-D histogram of distances and angles, where the distances and angles are computed between every point and every other point on the shape, makes it a purely global shape descriptor. Such descriptors can be used when the object is well-segmented from the image. However, segmentation is still an unsolved problem. Also, most state-of-the-art edge detectors are nowhere close to producing smooth closed object boundaries. Therefore, ability to match shapes partially is extremely important while recognizing objects in real-world images.

Bronstein et al. [20, 21] tackle the problem of partial similarity and show how objects that have large similar parts (but not completely similar) can be matched. They define their own intrinsic shape features and find correspondences between shapes. They also present a novel approach, which shows how partiality can be quantified using the notion of Pareto optimality. Matching shapes at different levels of partiality gives rise to the Pareto boundary. Any point on the Pareto-optimal boundary has the same cost of matching. Different points on the boundary account for a trade-off between partiality and dissimilarities. Figure 2.3 gives a good illustration of the trade-off between partiality and dissimilarity.

The interesting notion of Pareto optimality has since been applied by Donoser et al. [29] for matching partial shapes. They also develop a new angle-based shape

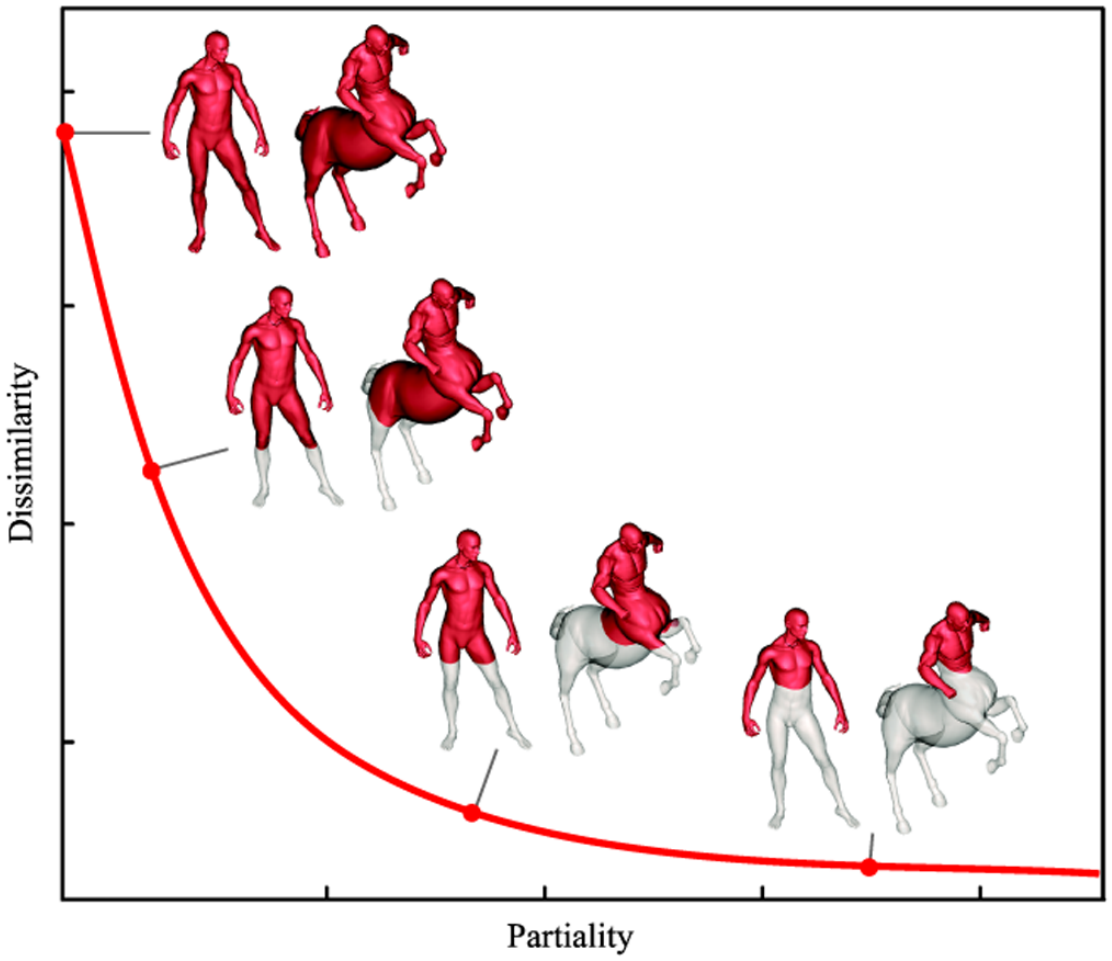


FIGURE 2.3: The figure shows a good illustration between partiality and dissimilarity. The red curve is the Pareto boundary. All points on the Pareto boundary have equal matching costs. Different points on the Pareto boundary trades-off between different levels of partiality and dissimilarity. (Source: [21])

descriptor that allows for efficient matching of partial shapes. They identified that combining the features into histograms make the descriptor purely global and, therefore, retained the true angular values (without binning) in the form of a matrix. The descriptor of a contour segment with n landmark points is a matrix of size $n \times n$, with the (u, v)-th entry given as,

$$\alpha_{uv} = \angle(L_{u,v}, L_{v,v-\Delta}), \tag{2.7}$$

where u and v are two points on the contour, $L_{u,v}$ is the line segment joining the two points, and Δ is an indicator of the number of points before point v with respect to which the angle α_{uv} is calculated. The authors also propose an efficient way of partially matching shapes by making use of the integral images data structure, which was first introduced by Crow [24], and was later made popular by Viola and

Jones [98] in their famous face detector.

Ma and Latecki [61] took motivations from the work by Donoser et al. [29] and utilized the partial shape matching technique for matching contour fragments extracted from real-world images. They not only used an angular matrix, similar to the one defined in [29], but also define a similar distance matrix between all pairs of points and use a combination of the distance and angle matrix to describe contour fragments.

We will use the notion of partial shape matching while performing object recognition from real world images. However, before moving on to recognizing objects in the wild, we will now explore the interesting area of going beyond pairwise matching of shapes to improve image retrieval scores.

2.4 Beyond Pairwise Shape Comparison

What we have looked at until now, is a scheme for comparing a shape template to a given test example, and calculating how "far off" the test object is from the template that we are comparing to. Such comparison can be considered as pairwise comparisons of shapes since there are only two objects that are involved in the matching process. However, consider the following scenario. Given three shapes, S_1, S_2 and S_3, if we have a sense of how "far" S_2 is from S_1, and how "far" S_3 is from S_1 as well, then we can make an intelligent guess of the "distance" between S_2 and S_3. This example can be extrapolated to more than three shapes. The ability to uncover distances between shapes, without actually comparing the shapes explicitly, motivates us to go beyond pairwise comparison of shapes. This idea has caught a lot of attention in the recent past and has seen contributions from multiple avenues such as dimensionality reduction, manifold learning and label propagation.

We now explain the basic idea of going beyond pairwise shape comparison. Consider the set, $\mathcal{S}$, that contains all that shapes that can exist. It must be apparent that there is no possible way to embed all these shapes in low dimensional spaces, ex: 1-D, 2-D, or 3-D. Therefore, these shapes are represented in extremely high dimensional spaces. If we consider 128×128 image patches containing objects, and if the shapes features were pixel intensities, we would have a 16384-D feature space. Such values for dimensionalities are common in computer vision. No

one really knows the true dimensionality for representing shapes, and therefore, they are usually represented by such high-dimensional, but extremely rich, features. Moreover, it is not even known whether shapes lie in an Euclidean space. Recent works which assume that the set of shapes lie on a low-dimensional non-linear manifold, but embedded in a high-dimensional Euclidean space have shown promising results. The study of shape manifolds obtained by normalizing point clouds for translation, rotation, and scaling was pioneered by Kendall [45].

With this background, we can now formulate the problem of finding the "true" dissimilarities between pairs of shapes as finding the geodesic distance between them, along the non-linear low-dimensional manifold. Therefore, it makes sense to utilize dimensionality reduction techniques to reduce the dimensionality of the feature space and then compute the Euclidean distance in the reduced feature space. Some of the most famous non-linear dimensionality reduction algorithms include Isomap [93], Locally Linear Embedding [81], Local Tangent Space Alignment (LTSA) [114], Laplacian Eigen Maps [9] and Diffusion Maps, to name a few.

All these algorithms have to tackle two parts of the larger problem of identifying the geodesic distance. How does one approximate the non-linear manifold? And how does one discover the geodesic distance on the approximated manifold? The approaches that exist in the literature is to approximate the non-linear manifold as a combination of linear sub-spaces and then compute the geodesic distance along the new path. Most algorithms approximate the manifold by building graphs and then sparsify them preserving local linearities. Isomap, for example, uses this extremely simple idea to approximate the non-linear dimensionality of the shape space. The geodesic distance is then obtained by running a simple shortest-path algorithm on the sparsified graph. The description can be visualized in Figure 2.4.

One of the biggest challenges is in obtaining the sparse graph from the fully-connected data graph. How does one retain a subset of edges emanating from a particular node in the graph? Current techniques make use of neighborhood principles in performing the sparsification. The most common methods of sparsification include k-NN sparsification and ϵ-neighborhood graph sparsification. As the names suggest, in k-NN graph sparsification technique, all edges apart from those that join a node to its k-nearest neighbors are pruned off; in the case of ϵ-neighborhood graph, only those edges are retained, which connects to nodes that are at a distance of less than ϵ from the node under consideration. Graph

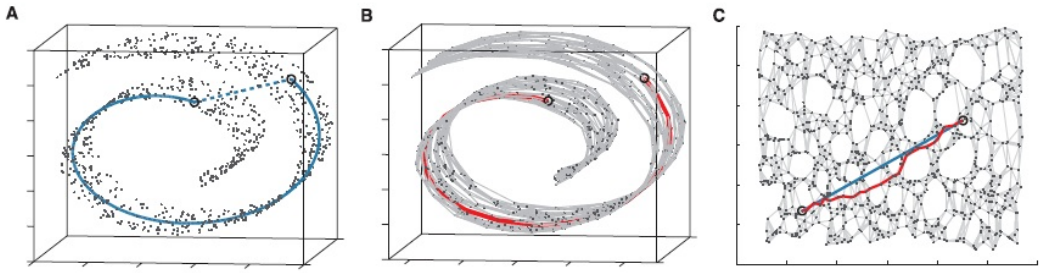


FIGURE 2.4: The figure shows an example of a 2-D manifold lying in a 3-D space. The goal is to find the geodesic distance between two points shown in subfigure (a). The data points are first connected into a graph with edges between nearby points (subfigure (b)). The geodesic distance is approximated by calculating the shortest path distance on the graph (subfigure (c)) (Source: [93]).

sparsification is based on the assumption that the true geodesic distances between neighborhood nodes (usually a small locality) can be well-approximated by the original Euclidean distance.

Other tangent direction-based adaptive graph sparsification techniques have been recently proposed ([113]) and they do show promise. However, techniques that make use of explicit feature values cannot be used in most cases. This is because, in many cases, the features are not known (but only the distances in the high-dimensional space is known), or the feature dimensionality is extremely large making the number of shapes populating the feature space extremely sparse leading to unstable algorithms. In moving beyond pairwise shape retrieval, we are usually provided with the pairwise distance matrix and we are left with the task of finding the true geodesic distances directly in the data space.

One of the ways to learn the geodesic distance in the data space is to use label propagation, or diffusion-based techniques, which we briefly explain, below. Diffusion maps leverages the relationship between heat diffusion and Markovian random walk. Consider the set of data points (i.e., shapes) given as $\mathcal{X} = \{x_1, x_2, ...x_n\}$. We find the similarity between pairs of points x_i and x_j, using some kernel similarity, $\mathcal{K}(x_i, x_j)$. A common kernel that is used is the Gaussian kernel,

$$\mathcal{K}(x_i, x_j) = \begin{cases} \exp(-d(x_i, x_j)/\sigma^2), & \text{if } x_i \in Nbhd(x_j) \\ 0, & \text{otherwise} \end{cases} \tag{2.8}$$

where $d(.,.)$ is the distance between two points, which is usually obtained after matching the respective shape features, $Nbhd(x_j)$ denotes the neighborhood of x_j,

which could be one of the many definitions of neighborhood, and σ is the kernel's proximity parameter, which can also be considered as a scaling factor. The term 'affinity' is interchangeably used with the term 'kernel'.

Now, to generate a random walk on the neighborhood affinities, we have to define a transition probability matrix, P, which is basically a normalized version of the kernel matrix, $\mathcal{K}$,

$$P = D^{-1}\mathcal{K}. \tag{2.9}$$

In the above equation, D, is the degree matrix, which is used to normalize the kernel matrix. The entries of the matrix P, i.e., $P(x_i, x_j)$ gives the transition probabilities of transitioning from point x_i to point x_j. The probability after t time steps is written as $P^t(x_i, x_j)$, which is obtained by multiplying P with itself for t times. The probability between two points after t iterations, reflects the true affinity between the two points, which is therefore, inversely proportional to the true geodesic distance between the two points. Therefore, after running the Markovian chain for t iterations, the true geodesic distance between points gets uncovered.

Recently, there have been many variants of the Markov chain-based diffusion processes that have been proposed in the literature. Two diffusion processes that are worth mentioning are the Locally Constrained Diffusion Process (LCDP) [107], and Tensor Product Graph diffusion (TPG) [108].

Locally Constrained Diffusion Process (LCDP): Yang et al. [107] identified that performing diffusion along a locally constrained neighborhood made the diffusion process robust to noise. Therefore, they define the diffusion process as,

$$P^{t+1}(x_i, x_j) = \sum_{x_k \in k-NN(x_i), x_l \in k-NN(x_j)} P_{FC}(x_i, x_k) P^t(x_k, x_l) P_{FC}(x_l, x_j), \tag{2.10}$$

where P_{FC} is the transition probability matrix from the fully-connected graph, and k-NN(x_i) is a function that returns the set of all shapes that lie in the k-Nearest Neighborhood of the shape x_i. Equation 2.10 can be succinctly written as,

$$P^{t+1} = P_{FC} P^t P_{FC}^{\top}. \tag{2.11}$$

Here, $\top$ is the transpose of the matrix. Because of its constrained nature, the authors call the diffusion process as a Locally Constrained Diffusion Process.

Tensor Product Graph (TPG): More recently, Yang et al. [108] proposed a more robust version of the diffusion process. They make use of higher-order information by performing the diffusion on the tensor product graph, instead of the original graph. Moreover, their method is robust to the number of iterations as well.

They start by generating the affinity matrix, A (this is the same as the kernel matrix defined above). They then normalize it and prune out certain edges so that the sum of all rows of the matrix $A < 1$. The fact that the maximum eigen value of any matrix is upper bounded by the maximum of the sum of the rows of that matrix, guarantees that diffusion performed on the current matrix, A, will always converge. To extract higher-order information from the matrix A, they compute the tensor product graph as $\mathbb{A} = A \otimes A$, where $\otimes$ denotes the Kronecker product. The t-th iteration in their definition of diffusion is defined as,

$$\mathbb{A}^t = \sum_{i=0}^{t} \mathbb{A}^i. \tag{2.12}$$

In addition, they show that the TPG diffusion can be performed in the same amount of complexity as on the original graph. The t-th stage of the diffusion process is given as,

$$Q^{t+1} = AQ^tA^\top + I, \tag{2.13}$$

where, $Q^1 = A$, and I is the identity matrix. The proof of the above equation is not relevant to us. It suffices to know that there exists a way to perform diffusion on the tensor product graph, at the same complexity as on the original graph, and the way to do it is given by Equation 2.13. We suggest that the reader look through the original publication ([108]) for the proof.

The success of the diffusion process is highly dependent on the graph sparsification step. Accurately sparsifying the graph is a difficult problem as there is no information about the local geodesics in the data space. Therefore, most graph sparsification techniques use domain knowledge to set the parameter k, in the k-NN graph sparsification method, or the parameter ϵ, in the ϵ-neighborhood graph. The robustness of the diffusion process is highly dependent on the value of these parameters. In Chapter 5, we propose a novel method for attaining a robust neighborhood by mining the information from multiple rounds of k-NNs.

Comparison of shapes in conjunction with other shapes is an exciting area which has gained significant interest in the recent past. The ability to mine information from surrounding shapes to find the "true" distances between pairs of shapes can be applied to various other areas as well. As a final section in this survey, we will now look into the techniques in shape-based object detection. Some parts of the following section is also applicable to object recognition, in general.

2.5 Object Detection

Detecting objects in real-world images is a challenging task and is an actively pursued area in the field of computer vision. The goal of an object detection algorithm is to identify where in the image a particular object instance exists. While the literature consists of many bag-of-words-based recognition and detection approaches, in this thesis, we will concentrate on the techniques involved in shape-based object detection.

The seminal work of Viola and Jones [98] is a success story of detecting faces in images. The intelligent usage of data strucutres such as integral images, a learning algorithm that is a variant of AdaBoost, and a cascade architecture to quickly reject false sub-windows, meant that their method not only detected faces in images, but did so in real time. The goal of detection in images has now moved on from detecting faces to detecting generic objects. Ever since the development of the brilliantly-performing Histograms of Oriented Gradients (HOGs) by Dalal and Triggs [25], the field of object detection has seen major improvements. HOGs capture the gradient information from local patches and bins them into histograms. HOGs are definitely a step up from the non-spatial bag-of-words approaches for object detection. The general approach to object detection using HOGs is as follows. Given a training set of positive and negative images, HOG features are extracted from both instances. A classifier is then trained using these positive and negative features (one of the most commonly used classifier is the Support Vector Machine). Given a new test image, all locations in the image are scanned using a sliding window-based approach. At each location, the HOG features are computed and tested for the presence, or absence, of objects using the previously trained SVM. The SVM score gives the object confidence. The test image is resized to multiple resolutions to account for scale variances and the detection procedure is repeated over each scale.

Parallel to recognizing objects using HOGs, object detection based on their shape properties has gained much importance in the recent past. This is due to the strong localization capability of the contours and their invariance to lighting conditions. Shape-based object recognition techniques extract the shape information from the edge map of an image and then perform shape matching to ascertain the similarity of the contour fragments. The edge map can be extracted using any standard edge detector. A commonly used edge detector is the Berkeley edge detector [63] (also called as the Pb edge detector since it produces probabilistic edge maps) as it is one of the best performing edge detectors and produces semantically meaningful edge maps. Dealing with contours for shape-based detection poses its own challenges. Even the state-of-the-art edge detectors [63, 102] cannot produce clean and smooth object contours. Therefore, problems such as broken contours, missing contours and noisy background contours have to be dealt with appropriately while making use of edge maps.

Shotton et al. [88] learn a codebook of edge fragments and use chamfer matching for matching the learnt fragments. They introduce a variant of the chamfer matching technique, namely, Oriented Chamfer Matching (OCM), which takes into account the orientation of the contour points, in addition to the distance between them. Ferrari et al. published a series of works [32–34] advancing the area of contour-based shape recognition. In one of their first works [32], they partitioned the image edges into contour segments and encoded them in an image representation that encoded their interconnections. This representation was named the Contour Segment Network. The object detection was formulated as finding paths in the network by reassembling the model outlines. Subsequently, in [33], they presented a family of scale-invariant local shape features that were formed by k connected, roughly straight contour segments, called kAS. They claimed that the kAS was able to encode fragments from the object boundary without including fragments from background clutter. Finally, in [34], they presented an object detection procedure which integrated the complementary strengths offered by shape matchers. To localize objects, they integrated a Hough-style voting technique, along with a non-rigid point matching algorithm.

A simple approach for contour-based object detection was proposed by Felzenszwalb and Schwartz [30] wherein they utilize Procrustes matching of contour fragments for matching deformable shapes in an hierarchical fashion. Srinivasan et al. [91] pose the matching between fragments as a many-to-one matching scheme

and used a variant of the shape context shape descriptor for describing contour fragments.

The issue of partial matching of shapes was addressed by Riemenschneider et al. [80]. They use the idea of Pareto-optimality from the works of Bronstein et al. [21] and extend it to match parts of the object template with the contour fragments. They use a similar shape descriptor as the one proposed by Donoser et al. [29] to describe the contour fragments. The concept of partial shape matching was later made more robust by Ma and Latecki [61]. They augmented the shape descriptor with a distance matrix along with the purely angular descriptor similar to the ones that was used in [29] and [80]. Each contour fragment was matched to the object template at all locations and lengths. They then created a sparse affinity matrix and posed the problem of object localization as the problem of finding maximal cliques in the sparse graph. Such an approach allowed them to discard the sliding window stage of the object detector and enabled direct detection of objects. Finding all the maximal cliques in a graph is an NP hard problem and, therefore, there exist many approximate solutions to solve the same. Pavan and Pelilo [72] proposed a game-theoretic approach and showed that the local optima of the replicator dynamics constituted to identifying maximal cliques in a graph. Ma and Latecki [61] use this procedure for object localization. However, we showed in [79] that such a procedure for clustering object parts might not always lead to the best solution.

In addition, both the approaches in [80] and [61] rely on the ordered connectedness of edge fragments for generating their descriptor. To obtain the connected edge fragments, they make use of the edge linking algorithm from [50], which identifies the edge's end point, and produces a linked list of points till it reaches another end point, or a junction point. This process is repeated till all edge points are linked into their respective edge fragments. Such an approach for linking edge points does not guarantee a consistency in the ordering of the points. The generated edge fragments could be arbitrarily ordered top-to-bottom, left-to-right, clockwise or anticlockwise. The reliance on ordering of points means that the shape descriptors of the template shape (which is a closed contour), generated by [80] and [61], are either ordered in a clockwise, or anti-clockwise manner. In addition, since both these shape descriptors are not invariant to reflections, they would give incorrect matching costs while matching to true parts of the shape on the test image, but whose edge fragment has been linked in an arbitrary fashion by the edge linking

algorithm. We identified this problem while experimenting with the shape descriptors proposed by both [80] and [61]. Figure 2.5 illustrates the problem with the edge linking algorithms.

Other recent works infer the presence of object parts in a conditionally independent fashion [36, 62, 100]. Gall et al. [36] have developed a hough voting-based object detector, where the features in the image cast the votes independently of the neighboring features. The votes predict the location of the object centroid and the probability of predicting the hypothesis is computed by random forests. Since the voting method is an extension of the Hough transform, the authors name their method as Hough Forests. More recently, Wang et al. [100] proposed a fan-shaped model for identifying object parts. The parts in their model were also identified independently of the neighboring parts.

Making use of object structure is bound to provide better stability in object recognition over techniques that make inference independently of neighboring parts. However, structured prediction approaches usually come at an increased computational cost, or would need non-realistic approximations (chain models, or tree-structured models) while building the model. Most structured prediction approaches are also part-based approaches, where the different parts of an object are modeled using various structures. The most famous contemporary work in parts-based models is the pictorial structure model of Fischler and Elschlager [35]. The pictorial structure model from is shown in Figure 2.6.

Richer models such as the pictorial structures-based deformable parts model [31] have shown a fair amount of success in object detection. Motivated by the deformable parts model, Lin et al. [53] extend the max-margin framework to an AND-OR Tree representation to learn a contour fragment-based shape model. It has to be said that the work in [53] also relies on an ordered list of contour points for extracting their part features. Therefore, it is susceptible to the edge linking problem that was explained previously in this section. The area of structured-prediction approaches for object detection is still fairly new and there are lots of opportunities for improvement.

Most recently, Yarlagadda and Ommer [109] argued that the object form cannot be perceived locally and set out to learn a spatially consistent configuration of all dictionary contours using a max-margin multi instance learning framework. We believe in a similar philosophy that the labeling of an object part is influenced

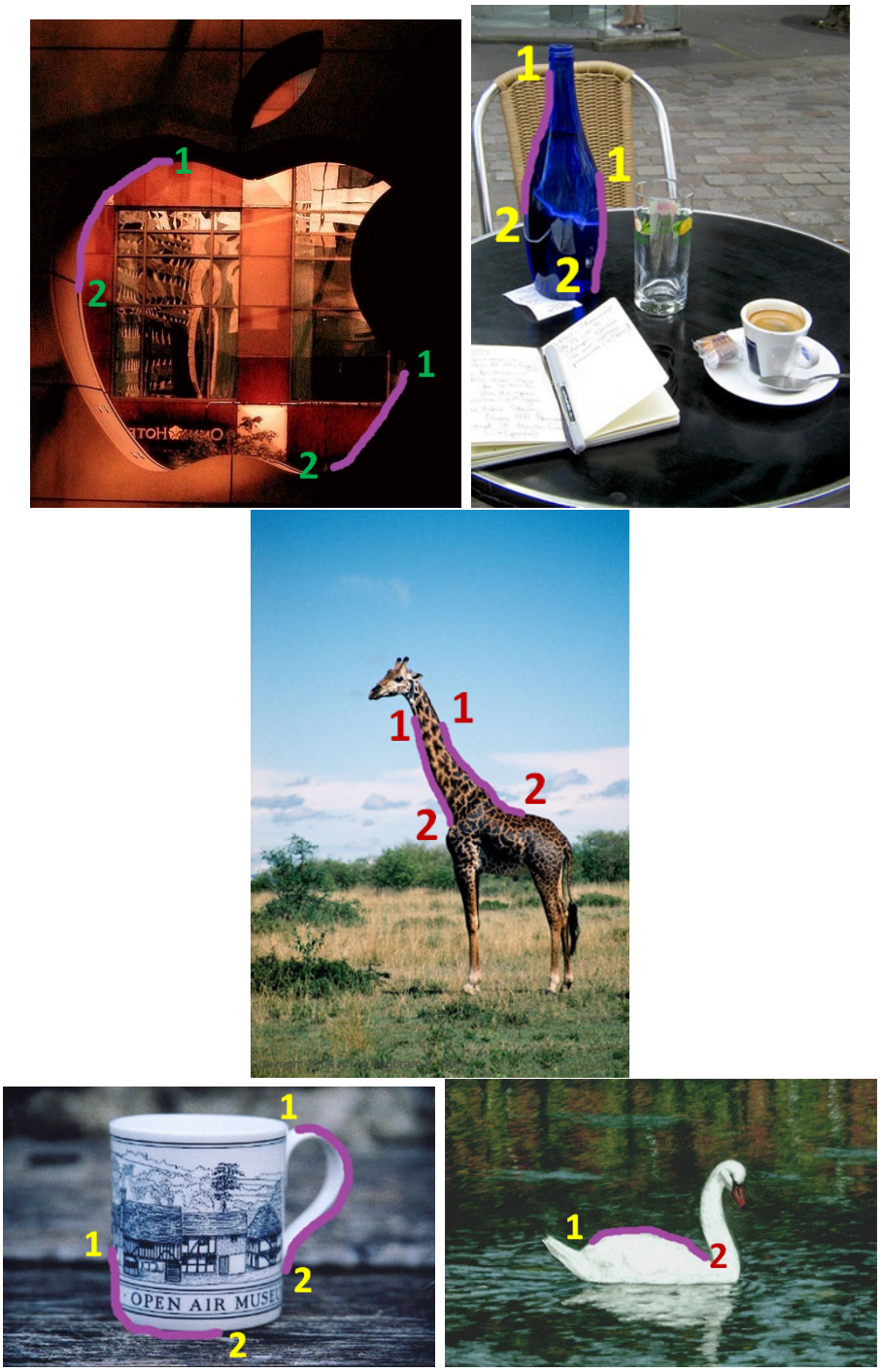


FIGURE 2.5: Figure illustrates the starting problem on an example image from each of the five different classes of the ETH-Z shape database. The purple line shows a contour fragment. The end points of the fragment are labeled either as '1', or '2'. The edge linking algorithms like the one in [50] cannot consistently select a starting point such that they all end up following a clockwise, or counter-clockwise, rule.

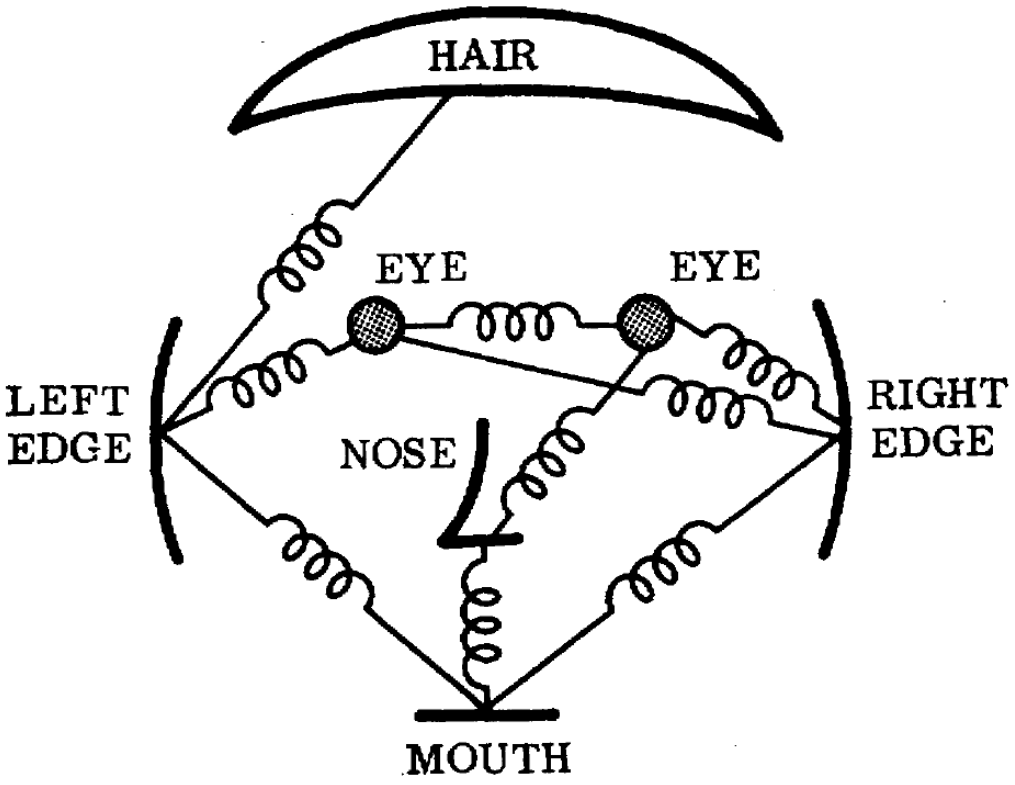


FIGURE 2.6: The figure shows the famous pictorial structures model (taken from Fischler and Elschlager [35]) for modeling deformations in the face. The various nodes in the graph correspond to semantically meaningful parts of the face. The parts are joined by 'edges' which are represented as springs, to allow deformations.

more by the labeling of the rest of the parts around it, than the local descriptor of that part. We will discuss our approach in detail in Chapter 6. Our philosophy will become apparent from our model and will also be corroborated by the use of belief propagation in the inference stage of our algorithm.

2.5.1 Summary

In this section, we have reviewed the literature in the field of shape-based object detection and have shown that there exists two major approaches for detecting objects; the HOG-based approach and the contour-based approach. As there were many approaches listed out above, we will now quickly point out the problem we intend to solve. We feel that there is a gap between these two approaches that needs to be bridged. Our work in shape-based object detection (explained in detail in Chapter 6) aims to bridge this apparent gap. We make use of the strong

discriminative capacity of HOGs and the strong localization capability of contours, and fuse them together into a single object detection framework, which also makes use of a structured prediction approach for part-labeling. In the next chapter, we explain in detail, our first contribution to this thesis, i.e., the development of a perceptually-motivated shape descriptor.

Chapter 3

A Perceptually-Motivated Shape Context

3.1 Introduction

Accurately measuring the similarity between two objects is a fundamental problem in many computer vision applications, and is still a largely unsolved problem. Many applications such as shape matching, shape-retrieval, and shape-based object detection, rely on a strong and robust similarity measure. However, coming up with such a similarity measure has proven to be a difficult task since the definition of similarity itself is rather subjective. Given two cars, one might say that their similarity should be measured based on their color, while others might argue that the make, and model, are better metrics for measuring the similarity. Given two shapes, one might justify their similarity based on the number of parts in the shape, while another might feel that the symmetry of the objects is an important criterion.

Most pattern recognition problems are required to overcome this apparent vagueness in the definition of similarity and come up with a quantitative similarity (or dissimilarity) measure between objects. Restricting ourselves to the identification of dissimilarity between shapes, given two shapes, S_1 and S_2, dissimilarity measures try to identify the cost of transforming the shape S_1 into the shape S_2. The more similar the shapes are to each other, the easier it is to transform one into another, and thus, lower is the cost of matching.

The challenge that still remains is to come up with a good measure, which can give reasonable costs between two shapes. The metric should ideally be invariant to translation, rotation, and scaling; the metric should also be able to account for non-rigid shapes i.e., it should be invariant to articulations, the metric should be robust enough to ignore noise in the shape boundary, it should be able to handle deformations, and, hopefully, it would permit partial matching of shapes. Most current-day shape matching techniques cannot handle such diversity.

The task of matching two shapes is currently being thought of as a task of matching their respective contours. A 2-D shape, S, is modeled as a planar set residing in $\mathbb{R}^2$, which has a well-defined boundary, $\mathcal{B}$. Most algorithms sample this boundary and define features at the sampled locations. A correspondence problem is then solved, and the total cost of matching the two sets of features is considered as the cost of matching the two boundaries, and therefore, the cost of matching the two shapes (Section 3.3.3 gives a detailed description of the method).

While most of the shape information can usually be extracted from just the object's contour, it is not true in cases where the objects have a strong base structure. In such cases, indentations in their boundaries have minimal effect on the human visual system. Figure 3.1 shows examples of objects that are visually similar to each other even though some of them have multiple indentations in their contours. People tend to neglect these minor (or even major) indentations while perceiving the object's shape. This is in accordance with Gestalt psychology, which maintains that the human eye sees objects in their entirety before perceiving their individual parts. The gestalt effect is the form-generating capability of our senses, particularly with respect to the visual recognition of figures, and whole forms, instead of just a collection of simple lines and curves.

Due to the Gestalt effect, approaches that perform shape matching based on part decomposition, or curve matching, will not perform well on objects such as those shown in Figure 3.1. Therefore, there is a need for the development of matching techniques that can somehow capture this Gestalt effect [27].

In this chapter, we propose a novel way of extracting the shape properties that capture the object's shape in its entirety. We show how this can help in improving the retrieval rates by testing on the well-known MPEG-7 shape database. We also show improvements in performance over other recently proposed perceptually motivated techniques.

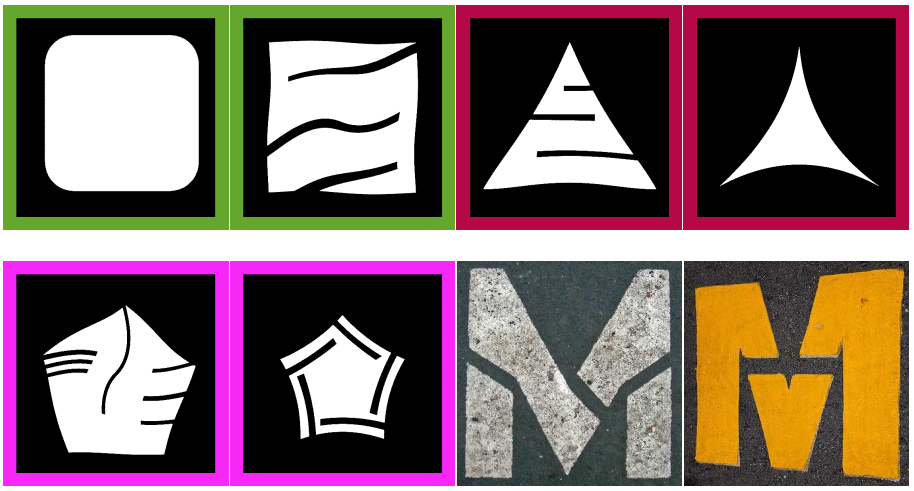


FIGURE 3.1: Figure shows examples of objects that are visually similar to each other even though they have multiple indentations (and even breaks) within their contours. Algorithms that perform contour-based matching of shapes cannot be used while matching such objects. Visually similar objects are appropriately color-coded using their bounding boxes.

The rest of the chapter is organized as follows. While we had discussed some of the previous work in the field of shape matching in Chapter 2, in Section 3.2 we discuss some of the relevant works in more detail and introduce some others that are specific to this chapter. Along with the review, we also identify some of the problems that these recent techniques face. In Section 3.3, we explain our method in detail and show how it can be used to tackle some of the problems that will be mentioned in Section 3.2. In Section 3.4, we provide results from our experiments, which shows an improvement over some of the recently proposed techniques. Finally, in Section 3.5, we summarize the chapter, with directions for future work.

3.2 Related Work

Some notable advances that have been made in this area over the past decade were discussed in Chapter 2, and is reiterated here, in more detail, to help the reader. A typical approach to measure shape similarity is through non-rigid shape deformation [30, 84]. Such methods measure the difficulty in transforming one shape into another. Geometrically, one can think of a shape S as a point on some low-dimensional manifold $\mathbb{M}$, residing in some high-dimensional space. The energy

required to transform a shape S_1 into a shape S_2 can be thought of as the geodesic distance[1] of the shortest path between the two points lying on the manifold.

Most approaches equate the task of shape matching to the matching of the respective object boundaries. The shape boundaries are discretized into a set of n landmark points, $S = \{p_1, p_2, ...p_n\}$, for easier representation and matching. Belongie et al. [13] showed that these points could be located at any place on the object boundary and that they need not be restricted to extrema points on the curve. They also proposed to describe the shape using shape contexts at each of these sampled points. The shape context at each sampled point is given by the relative distribution of the rest of the $n-1$ points, which is represented as a 2-D histogram of distances and angles.

The shape context (SC) can be made invariant to translation, rotation and scale. However, while SC matching performs well on rigid objects, it is susceptible to articulations. This is because the SC histogram is composed of Euclidean distance and angle, which cannot handle articulations. To overcome this problem, Ling and Jacobs [55] proposed a variant of SC, namely, Inner Distance Shape Context (IDSC). The IDSC uses inner distance (the length of the shortest path connecting the two points, such that the path lies completely within the shape) and inner angle, instead of Euclidean distance and angle, to generate the histograms at the sampled points. The use of this changed metric makes the descriptor invariant to articulations. Also, as suggested by Thayananthan et al. [94], they make use of the figural continuity constraints and perform the context matching using a dynamic programming scheme. Though IDSC looks at distances between points such that path connecting them lies completely within the shape's boundary, it still cannot capture the interior density of the shape. It relies completely on the distances between points that lie on the shape's boundary. We show how this important interior property can be captured in Section 3.3.1, and show how it can be used to construct meaningful shape descriptors in Section 3.3.3.

Gopalan et al. [37] identified that though the use of inner distance provided invariance to articulations, it could not be directly applied to "non-ideal" 2-D projections of 3-D objects. If the projection took place using a weak perspective, then not all parts of the 3-D model would get accurately projected onto the 2-D plane. In order to overcome this problem, they modeled an articulating object as a combination of approximate convex parts and performed affine normalization

[1] In this chapter, we will use the terms cost, distance, and energy, interchangeably

of these parts. They then use inner distance to perform shape matching on the normalized shapes. Their near-convex decomposition algorithm takes as input the contour of the object and splits the object into multiple convex parts. However, such an approach cannot be followed for shapes such as those shown in Figure 3.1, since the algorithm would split the object into multiple parts, yielding undesirable results.

The Medial Axis Transform (MAT) and its variant, shock graphs, have been used by certain authors for matching shapes [85, 89]. The medial axis, or skeleton, is the locus of the centers of all maximally inscribed circles of the object. While the MAT captures the interior properties of the shape to a certain extent, by definition, the generation of a skeleton depends on the boundary of the object. Therefore, the objects shown in Figure 3.2 will all have vastly different skeletons. Xie et al. [103], proposed to model shapes using skeletal contexts. Their contexts are calculated at the skeleton endings and the bins are populated by the non-uniformly sampled points from the boundary. Relying on the skeleton, and the boundary points, makes their method susceptible to indentations in the contour. We show, in Section 3.3, how our method does not fall prey to such boundary perturbations.

Due to the diversity involved in shape matching, it has become difficult to come up with a single measure that incorporates all the requirements. While the use of Euclidean distance is beneficial for identifying certain classes of objects, the use of inner distance favors some others. As a result, researchers have started to fuse two or more techniques while calculating the distance between two shapes. Ling et al. [57] identified that the use of inner distance was "overkill" for certain classes of objects and proposed a technique to balance deformability and discriminability. They calculate the cost between two shapes with the help of various distance measures, parameterized by an aspect weight, and retain the "best" cost. However, they still use points sampled from the contour and their algorithm would therefore be susceptible to objects with strong base structures that have indentations in their contours.

Recently, some effort has gone into the development of perceptually motivated techniques [39, 92]. These techniques tackle cases, such as those shown in Figure 3.2. To the human visual system, all objects in the figure appear to belong to the same class. However, measures that rely on the contour to obtain the object's shape properties cannot fathom this similarity.

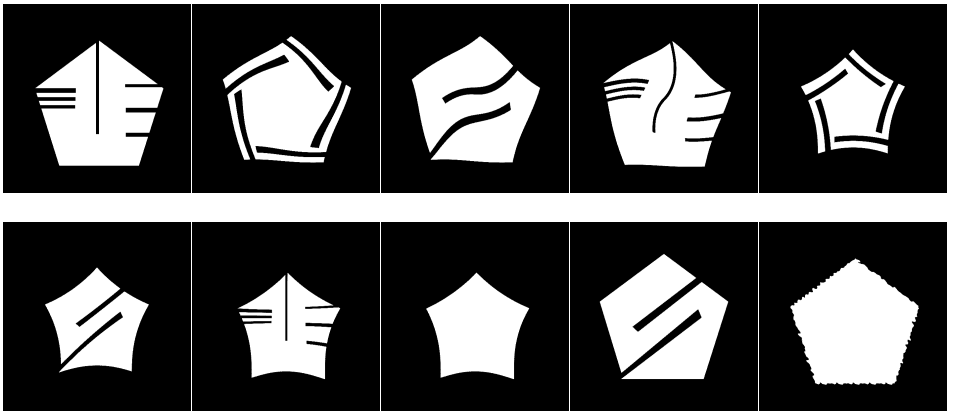


FIGURE 3.2: Here, we show more examples of a particular class of objects from the MPEG-7 database. All of the above objects have different contour properties. However, their overall visual similarity is still that of a pentagon.

Temlyakov et al. [92] propose to split the object into a base structure and multiple strand structures. They define strands as structures that are thin and long, relatively small in size, and attached to a base structure. The strands may be made of inward or outward strands. When comparing two shapes, they compare the base structures and strand structures separately. They use IDSC for comparing base structures, and for the strands, they just check if the two objects have similar number of strands, without giving much importance to the detailed geometry. Secondly, they also identify objects with a single axis of symmetry and normalize the aspect ratios of the two shapes before comparison. Fusing these two strategies along with IDSC helps them achieve better retrieval rates. Such an approach will work well if the object has a strong base structure. However, in many cases, the objects do not have a well-defined base. Even in the case of a strong base structure, multiple parameters, such as area, length, and width, have to be set to identify the strands, in the method proposed in [92]

More recently, Hu et al. [39] proposed a morphological approach to model human perceptions. To "close" the objects, they perform morphological closing on the shapes. They compare the shapes using IDSC before and after performing the morphological operation, and retain the better of the two costs. They perform the morphological operations over multiple scales. This calls for an additional scale parameter to be set. Secondly, selecting the structuring element for performing the closing operation is also a difficult task. In their experiments, they try using structuring elements of different sizes and report results from all sizes. In the following section, we explain our novel method of capturing the shape properties

in their entirety, and in Section 3.4, we show that our method can help generate better retrieval results than [92] and [39].

All the techniques described above were directed towards the development of a good distance measure between pairs of images, where the similarity of an object was influenced by just one other object. However, recent works have shown that an improvement in the retrieval performance can be achieved if other similar shapes are allowed to influence the pair-wise scores. For a given similarity measure, a new similarity measure is learned through graph transduction [6], or diffusion. Many methods that focus on improving the transduction algorithms have been proposed in the recent past [49, 106, 108].

Starting the diffusion with a good similarity matrix will lead us to obtain better similarities at the end. A good similarity matrix is one in which similar shapes have high affinity. We show that our method helps in generating a better similarity matrix after the diffusion process. We use the Locally Constrained Diffusion Process (LCDP) [107] to learn the manifold structure of the shapes and show, in Section 3.4, that our matrix is able to generate highly competitive retrieval rates.

3.3 Solid Shape Context

In the previous section, we reviewed past work in the area of shape matching and pointed to the fact that more research needs to be done in the development of perceptually motivated techniques. In this section, we introduce one such perceptually motivated technique, which can capture the shape properties in their entirety.

To motivate our work, let us go back to the examples in Figures 3.1 and 3.2. We identify that the human visual system not only recognizes shapes by their external contour, but also by their "density". We perceive a solid disc as a different object compared to a ring, though both have a circle as their outer contour. From this example, we can see that the interior solidity plays an important role in the identification of an object. We propose to utilize this important interior property of a shape by coming up with a descriptor, which is a variant of the well-known Shape Context descriptor.

To capture the interior properties of a shape, we propose to sample a set of uniformly-spaced *Dense Points* that lie within the object's body (more on this in the following sub-section). We then sample a much smaller set of points, called *Sparse Points*, where we compute the object's features (explained in detail later in this section). The computed features are our modifications of the shape context, and are described using the previously sampled *Dense Points*. Given two shapes, the contexts at their respective *Sparse Points* are used for comparison. In the following subsections, we describe in detail how we sample the dense and sparse points, and how our Solid Shape Context (SSC) is computed.

3.3.1 Dense Points

Motivated by the sampling techniques used to approximate probability density functions, we propose to approximate the interior shape of an object by sampling points lying within the object's boundary. Each part of the object is equally important in understanding the shape properties. Therefore, we use a uniform sampling scheme to sample points that lie uniformly within the shape.

The issues that we face while sampling from an arbitrary shape are similar to the issues that we face while sampling from an arbitrary distribution. Uniformly sampling from a well-known and simple shape, such as a square, rectangle, circle, or a triangle, is relatively straightforward. However, uniformly sampling a fixed set of points from a random shape is not that simple. In generating uniformly distributed samples inside a shape, we wish to generate points whose number is independent of the scale of the object. This consideration rules out simple methods such as sampling using internal pixels for cases when the shape comes as an image. Moreover, the consideration of uniform spatial distribution rules out the simple method of using sampling on a uniform grid, since there is no single grid that ensures a uniform distribution for all shapes.

One common technique that could be adopted is the rejection sampling technique. We can encompass the arbitrary shape using a well-known, and simple shape (say, circle or a square), and uniformly sample points from within it. We can then retain only those points that fall within the shape boundary and reject the rest that lie outside the shape. Figure 3.3 gives an illustration of the rejection sampling technique.

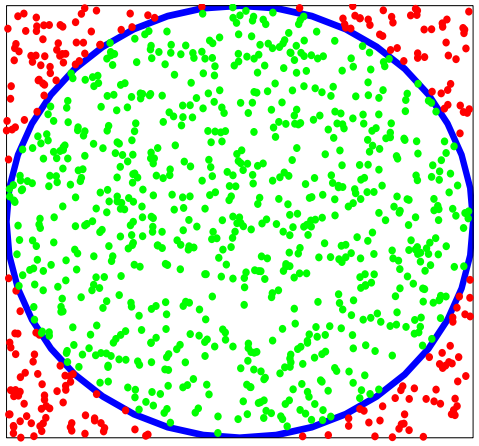


FIGURE 3.3: In order to sample points from within a circle, we uniformly sample points from a bounding square. We retain the points that fall within the circle (green) and reject the points that fall outside the circle (red).

While rejection sampling is a very simple method, there are a some issues that we encounter. It is difficult to efficiently sample a fixed number of points lying inside the shape boundary without wasting samples. For shapes with elongated parts, such as the tentacles of an octopus, the accept/reject method wastes a number of samples, which is proportional to the ratio in areas between the bounding rectangle and the object; this ratio can be quite high for objects with parts spread over a large region, such as horseshoes, octopi, or insects. Even for simple shapes such as the circle in Figure 3.3, a large number of points shown in red are wasted. Our method, which is described below, does not waste any samples, and is therefore able to maintain a constant complexity regardless of the shape of object.

The above problems are encountered because of two reasons:

1. We are trying to sample points from arbitrary shapes, for which there are no elegant sampling techniques.

2. We are not restricting ourselves to the interior of the shape, before sampling.

We wish to overcome these problems by making use of the object's boundary constraints. Firstly, we restrict our sampling area such that it lies totally within the object's boundary. Secondly, we ensure that the area we are sampling from is a simple shape, to ensure easy sampling. Below, we explain in detail how we propose to sample a fixed number of *Dense Points* without wasting any samples.

Given a shape S, we can easily extract its boundary, $\mathcal{B}^S$. We then sample a set of uniformly spaced points, $\mathcal{B}^S_P = \{\mathcal{B}^S_{p_1}, \mathcal{B}^S_{p_2}, ..., \mathcal{B}^S_{p_{|\mathcal{B}^S_P|}}\}$, that lie on the boundary of the object (Figure 3.4b). A point $\mathcal{B}^S_{p_i}$ neighbours just two other points $\mathcal{B}^S_{p_{i-1}}$ and $\mathcal{B}^S_{p_{i+1}}$ (the indices are taken modulo $|\mathcal{B}^S_P|$). We make use of this neighbourbood constraint and perform a Constrained Delaunay Triangulation (CDT) of these points. A CDT ensures that the edges specified as the constraints are retained in the triangulation process [71]. The constraints that we specify are the neighbourhood constraints i.e., our constraint ensures that a point $\mathcal{B}^S_{p_i}$ has an edge to its two neighbours $\mathcal{B}^S_{p_{i-1}}$ and $\mathcal{B}^S_{p_{i+1}}$. Once the triangulation is performed, we remove the triangles that lie in the concavities and holes of a shape [86]. This guarantees that the triangles generated from the triangulation lie totally within the object's boundary. For a given set of $N_\mathcal{B}$ points on the boundary, such a Constrained Delaunay Triangulation produces $N_\mathcal{B} - 2$ triangles, $Tri^S = \{Tri^S_1, Tri^S_2, ..., Tri^S_{N_\mathcal{B}-2}\}$. Figure 3.4c shows the output of the constrained triangulation. Notice that all the triangles now lie within the object's boundary, especially at the bottom left of the butterfly where there is a noisy indentation.

With the above formulation, it becomes easy to sample a fixed number of points such that they lie completely within the object's boundary. For any triangle, Tri^S_i, with vertices X, Y, and Z, a random point p, lying inside the triangle, can be generated using

$$p = (1 - \sqrt{r_1})X + \sqrt{r_1}(1 - r_2)Y + \sqrt{r_1}r_2Z, \tag{3.1}$$

where r_1, $r_2 \in [0, 1]$ are two random numbers, independent of each other [68]. X, Y, and Z, are 2-D vectors containing the x and y coordinates of the three vertices. The set of points lying inside Tri^S_i is given by $\mathcal{P}^{Tri^S_i} = \{p_1^{Tri^S_i}, p_2^{Tri^S_i}, ..., p^{Tri^S_i}_{|\mathcal{P}^{Tri^S_i}|}\}$. In order to generate $|\mathcal{P}^{Tri^S_i}|$ points lying inside the triangle Tri^S_i, all we have to do is generate $|\mathcal{P}^{Tri^S_i}|$ pairs of random numbers (r_1, r_2), from a uniform distribution, and use Equation 3.1 to generate the points.

In order to generate N_{DP} number of uniformly distributed *Dense Points*, lying within the object's boundary (Figure 3.4d), we generate $|\mathcal{P}^{Tri^S_i}|$ uniformly distributed points from within each triangle $Tri^S_i, \forall i \in \{1, 2, ..., N_\mathcal{B} - 2\}$, such that $|\mathcal{P}^{Tri^S_i}|$ is proportional to the area of triangle Tri^S_i, as,

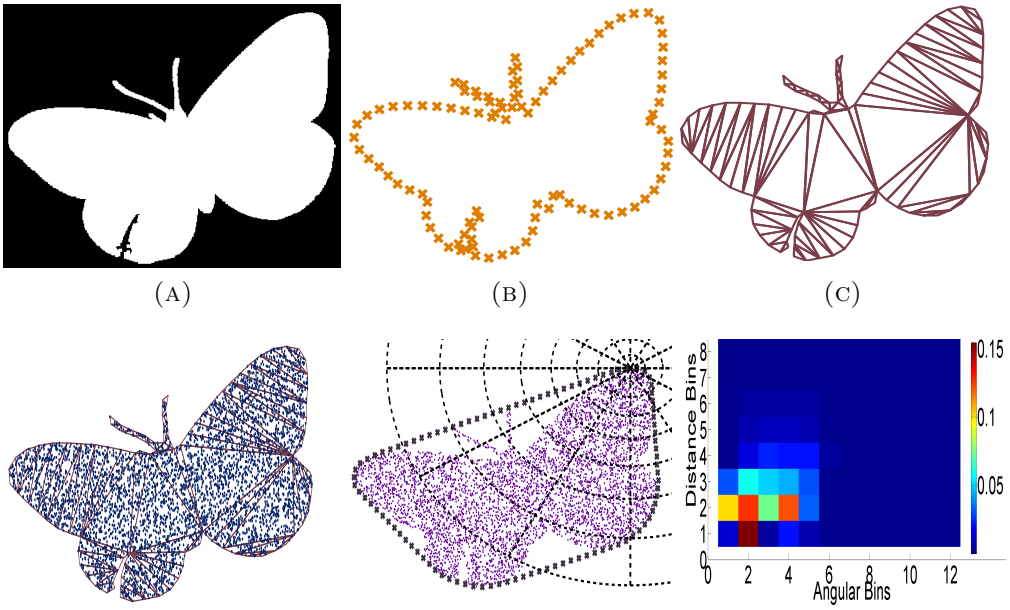


FIGURE 3.4: (a) The figure shows the silhouette of a butterfly with a noisy indent in the contour. (b) Uniformly sampled boundary points, $\mathcal{B}_P^S$, from the contour. (c) Output of the Constrained Delaunay Triangulation. The constraint is a simple neighbourhood continuity constraint of the sampled contour points. All the triangles now lie within the object's boundary. (d) *Dense Points* sampled from inside each triangle according to Equations 3.1 and 3.2. (e) *Sparse Points*, represented by crosses (zoom into the figure), are sampled from the boundary of object's convex hull. Solid Shape Context histogram is computed using log-polar bins at each *Sparse Point*. (f) A visualization of the Solid Shape Context (SSC) histogram.

$$|\mathcal{P}^{Tri_i^S}| = \frac{A_{Tri_i^S}}{\sum_{j=1}^{N_\mathcal{B}-2} A_{Tri_j^S}} N_{DP}. \tag{3.2}$$

$A_{Tri_i^S}$ is the area of triangle Tri_i^S. Therefore,

$$N_{DP} = \sum_{i=1}^{N_\mathcal{B}-2} |\mathcal{P}^{Tri_i^S}|. \tag{3.3}$$

Intuitively, we are sampling points from a particular triangle proportionate to its area relative to the area of the total object.

We have shown how we can generate a fixed number of points from within any random shape, easily. Our method overcomes the two problems that were previously listed in this section. Firstly, we restrict the sampling area to lie within the shape,

thus preventing any sampled points from being wasted. Secondly, we sample from a very simple polygon, a triangle, thus making the sampling of uniformly spaced random points quick and easy.

These densely sampled points approximate the interior density of a shape. Our shape descriptor models the shape in its entirety by making use of these *Dense Points*. The SSC shape descriptors are generated at each *Sparse Point* location. A discussion of how to select the location of these *Sparse Points* is given in the following subsection.

3.3.2 Sparse Points

A shape S is described using SSC, at N_{SP} locations, where $N_{SP} \ll N_{DP}$. Due to the fact that they are relatively less in number, compared to the *Dense Points*, we call them *Sparse Points*. It is usually enough if we generate the shape descriptors at these sparse set of locations, instead of generating them at each dense point.

The next question that arises is, how and where on the object to localise these sparse points. Ideally, we would like these feature locations to be uniformly spread across the object. We want the descriptors to describe the shape from a varied number of vantage points. One way to do this would be to generate a minimal enclosing rectangle for the object, and uniformly divide the rectangle into N_{SP} number of cells, and mark the centers of these cells as the locations of the *Sparse Points*. However, doing so would not enable us to make use of the continuity constraints while comparing the descriptors between two shapes.

Another approach could be to make use of the uniformly sampled points on the boundary, $\mathcal{B}_P^S$, as the *Sparse Points*, similar to the boundary sampling used in [13] and [55]. While this would enable us to make use of the continuity constraints that occur naturally, it would lead us to obtain certain erroneous matches, resulting in increased costs of matching two shapes. These erroneous matches would occur in cases where there are strong indentations in the boundary of the object, such as the examples shown in Figure 3.1. All the descriptors at the landmark points that lie on the indentations will have a vastly different representation of the shape compared to the descriptors that are extracted from a shape without similar (or, any) indentations. Thus, selecting the set of points, $\mathcal{B}_P^S$, as the landmark points does not seem to be a good idea.

To retain the advantage of the continuity constraints and still have *Sparse Points* that are independent of the indentations in the contour, we propose to sample the feature point locations along the boundary of the convex hull of the shape. Sampling landmark points along the convex hull gives us many advantages. Since the convex hull encloses the object completely, we retain the advantage of having the descriptors describe the object from various vantage points. Secondly, sampling from the convex hull gives us larger insensitivity to boundary perturbations. Along with the densely sampled points, which help in handling noisy indentations, sampling along the convex hull also prevents such indentations from unnecessarily affecting the landmark selection. Thirdly, sampling along the convex hull gives a better rotation invariance to the descriptor. Rotation invariance is usually added to the descriptor by tangent angle normalization, similar to the technique in Shape Context [13] and IDSC [55]. Calculating the tangent angle on the boundary of the convex hull gives better invariance to rotation than when the normalization angle is calculated using the tangent on a noisy contour. Such unwanted perturbations in the boundary would randomly skew the tangents along the boundary, thus causing large amounts of noise to be added during the angle normalization step. Finally, using the convex hull can be an advantage even when the shapes are highly concave. Since our sampling procedure ensures that the sampled points always lie inside the shape boundary, the absence of dense points in the concavities of the shape help capture the concave properties of the shape. Ex: The characteristic property of a horseshoe is its concavity, and this property is captured in our shape descriptor by means of zero height bins (see the following subsection).

Due to the above mentioned advantages, similar to $\mathcal{B}_P^S$, we obtain the set of *Sparse Points*, $\mathcal{SP}^S = \{\mathcal{SP}_1^S, \mathcal{SP}_2^S, ..., \mathcal{SP}_{N_{SP}}^S\}$, for shape S, sampled along the convex hull of the shape, and compute the SSC descriptor at each of these points. Figure 3.4e shows how the sparse points are sampled from the object's convex hull. Notice the insensitivity of the *Sparse Points* to the indentations in the butterfly's boundary.

Now that we have a set of *Dense Points* that can be used to model the interior of the shape, and a set of *Sparse Points* to represent the shape, we go on to describe how we generate our SSC descriptor using both these sets of points.

3.3.3 Solid Shape Context Descriptor

At each sparse point $\mathcal{SP}_i^S$, we generate a 2-D histogram

$$\mathcal{H}_i^S(k) = \#\{\mathcal{DP}_j^S : \mathcal{DP}_j^S \in bin(k)\}, \tag{3.4}$$

where, $\mathcal{DP}_j^S$ is the j-th *Dense Point*, k is the bin number, $i \in \{1, 2, ..., N_{SP}\}$, and $j \in \{1, 2, ..., N_{DP}\}$. Similar to [55], we use 8 distance bins and 12 angular bins to generate the log-polar histogram. We use the Euclidean distance and Euclidean angle (similar to [13]) to calculate the distance, and angle, between a *Sparse Point* and a *Dense Point*. A given shape S can now be described by a set of histograms, $\mathcal{SSC}^S = \{\mathcal{H}_1^S, \mathcal{H}_2^S, ..., \mathcal{H}_{N_{SP}}^S\}$. Similar to SC and IDSC, SSC is inherently invariant to translation. It can be made invariant to rotations, and scale, by tangent normalization, and mean distance normalization, respectively. Figure 3.4f gives a visualization of the SSC histogram for one of the sparse locations.

Given two shapes S_1 and S_2, matching them now boils down to matching their respective histogram sets, $\mathcal{SSC}^{S_1}$ and $\mathcal{SSC}^{S_2}$. The goal of the matching stage is to find a mapping function $\phi(.)$, which minimizes the total cost of matching the shape S_1 to the shape S_2, which is given by

$$\Psi_{SSC}(S_1, S_2) = \sum_{i=1}^{N_{SP}^{S_1}} \psi(\mathcal{H}_i^{S_1}, \mathcal{H}_{\phi(i)}^{S_2}). \tag{3.5}$$

The distance between two histograms, $\psi(\mathcal{H}_i^{S_1}, \mathcal{H}_{\phi(i)}^{S_2})$, is defined by the χ^2 test statistic

$$\psi(\mathcal{H}_i^{S_1}, \mathcal{H}_{\phi(i)}^{S_2}) = \frac{1}{2} \sum_{k=1}^{K} \frac{[H_i^{S_1}(k) - H_{\phi(i)}^{S_2}(k)]^2}{H_i^{S_1}(k) + H_{\phi(i)}^{S_2}(k)}, \tag{3.6}$$

where, k is the bin number and K is the total number of bins in the histogram. If the distance between the two histograms is greater than an acceptable threshold $\tau(= 0.6)$, we set the distance to equal τ, and set $\phi(i)$ to 0, which means to say that we were not able to find a suitable match for $\mathcal{H}_i^{S_1}$, in shape S_2. Similar to [55], we use a dynamic programming scheme to match the two sets of histograms.

Since the contours come in a sequential order, it is natural to utilize the ordering of the points while finding a suitable mapping $\phi(.)$. Dynamic programming helps find both the minimum cost of Equation 3.5 and the mapping $\phi(.)$. We now briefly summarize the DP procedure below. We maintain a DP matrix D of size

$N_{SP}^{S_1} \times N_{SP}^{S_2}$ where $N_{SP}^{S_1}$ is the number of *Sparse Points* from the shape S_1 and $N_{SP}^{S_2}$ is the number of *Sparse Points* from the shape S_2. The entry $D(i,j)$ of the DP matrix corresponds to the minimum cost of matching the first i points from shape S_1 to the first j points in shape S_2. Therefore, the minimum cost of matching the two shapes completely can be found in entry $D(N_{SP}^{S_1}, N_{SP}^{S_2})$. By using dynamic programming, we can fill this entry as

$$\begin{aligned} D(N_{SP}^{S_1}, N_{SP}^{S_2}) = \min\{&D(N_{SP}^{S_1}-1, N_{SP}^{S_2}-1) + \psi(\mathcal{H}_{N_{SP}^{S_1}}^{S_1}, \mathcal{H}_{N_{SP}^{S_2}}^{S_2}), \\ &D(N_{SP}^{S_1}-1, N_{SP}^{S_2}) + \tau, D(N_{SP}^{S_1}, N_{SP}^{S_2}-1) + \tau\}. \end{aligned} \tag{3.7}$$

Using the above recursive rule, we can populate all entries of the DP matrix, D, using the entries in the previous row and the previous column. For the DP algorithm to flow seamlessly, we need to initialize the first row and the first column, which is fairly straightforward.

$$D(1,1) = min\{\psi(\mathcal{H}_1^{S_1}, \mathcal{H}_1^{S_2}), \tau\} \tag{3.8}$$

$$D(1,j) = min\{\psi(\mathcal{H}_1^{S_1}, \mathcal{H}_j^{S_2}) + (j-1) * \tau, D(1,j-1) + \tau\} \tag{3.9}$$

$$D(i,1) = min\{\psi(\mathcal{H}_i^{S_1}, \mathcal{H}_1^{S_2}) + (i-1) * \tau, D(i-1,1) + \tau\} \tag{3.10}$$

The minimum cost of matching the two shapes can be found in $O(n^2)$ time.

Finally, the true cost between the two shapes S_1 and S_2 can be computed as

$$\Psi(S_1, S_2) = \min(\Psi_{IDSC}, \alpha\Psi_{SSC}), \tag{3.11}$$

where Ψ_{IDSC} is the cost of matching the two shapes using the standard IDSC method [55], Ψ_{SSC} is given by Equation 3.5, and α is a normalization constant, which is used to normalize the two costs. Fusing two or more costs to obtain the smallest cost has become popular in the recent past and is used in [57], [92] and [39].

The time complexity of SSC+DP can be split into five parts. The first part involves Constrained Delaunay Triangulation of the *Sparse Points*, the complexity of which is $O(N_{SP} log N_{SP})$. The second part involves the sampling of the *Dense Points*. Each *Dense Point* can be computed in constant time. Therefore, the total time complexity of generating N_{DP} *Dense Points* is linear in the number of *Dense Points*, i.e. $O(N_{DP})$. The third part involves the computation of the pairwise distances and angles. This can be computed in $O(N_{SP}N_{DP})$. The fourth step is the

construction of the histogram, which takes $O(N_{SP}^2)$. And finally, the computation of the minimum cost, done using dynamic programming, costs $O(N_{SP}^2)$. One has to bare in mind that the dynamic programming step assumes that the starting positions of the two sequences are fixed before hand. However, rotations of shapes do not guarantee this condition. Therefore, we have to check for all starting point locations, making the complexity of the matching step $O(N_{SP}^3)$! Thankfully, in practice, it suffices to perform the matching over a fixed set of starting locations ℓ, making the complexity $O(\ell N_{SP}^2)$. Ling and Jacobs [55] noticed that checking for more than 8 starting points usually has little impact in improving the matching cost. Therefore, with $\ell = 8$, which is a constant and which is $<< N_{SP}$, the complexity comes back down to $O(N_{SP}^2)$.

For completeness, we now provide run time statistics of SSC+DP. We perform our experiments on an Intel Core i7 machine with 8GB RAM. The CDT takes 0.5ms per object. The dense sampling takes 12ms per object. The histogram computation takes 0.7 seconds per shape. The dynamic programming step to calculate the minimum cost takes 8ms. A complete pairwise matching of two shapes takes about 1.4s in our unoptimized Matlab implementation for generating SSC.

Figure 3.4 illustrates all the steps involved in the generation of the SSC shape descriptor. In the next section, we demonstrate the effectiveness of our SSC descriptor using the results obtained from our experiments.

3.4 Experiments and Results

We use the well-known, and widely used, MPEG-7 CE-Shape-1 Part B dataset for testing our algorithm. The database consists of silhouettes of 1400 images with a wide variety among them. The database is split into 70 classes, with each class containing 20 example images. The database consists of both rigid and non-rigid objects. The objects in the database have varied levels of translations, rotations, scales, articulations, deformations and occlusions. The objects belonging to a particular class are not only similar by the contour properties, but also by their overall visual similarity. The database is considered as a challenging database as there are many instances where the inter-class object similarity is more than

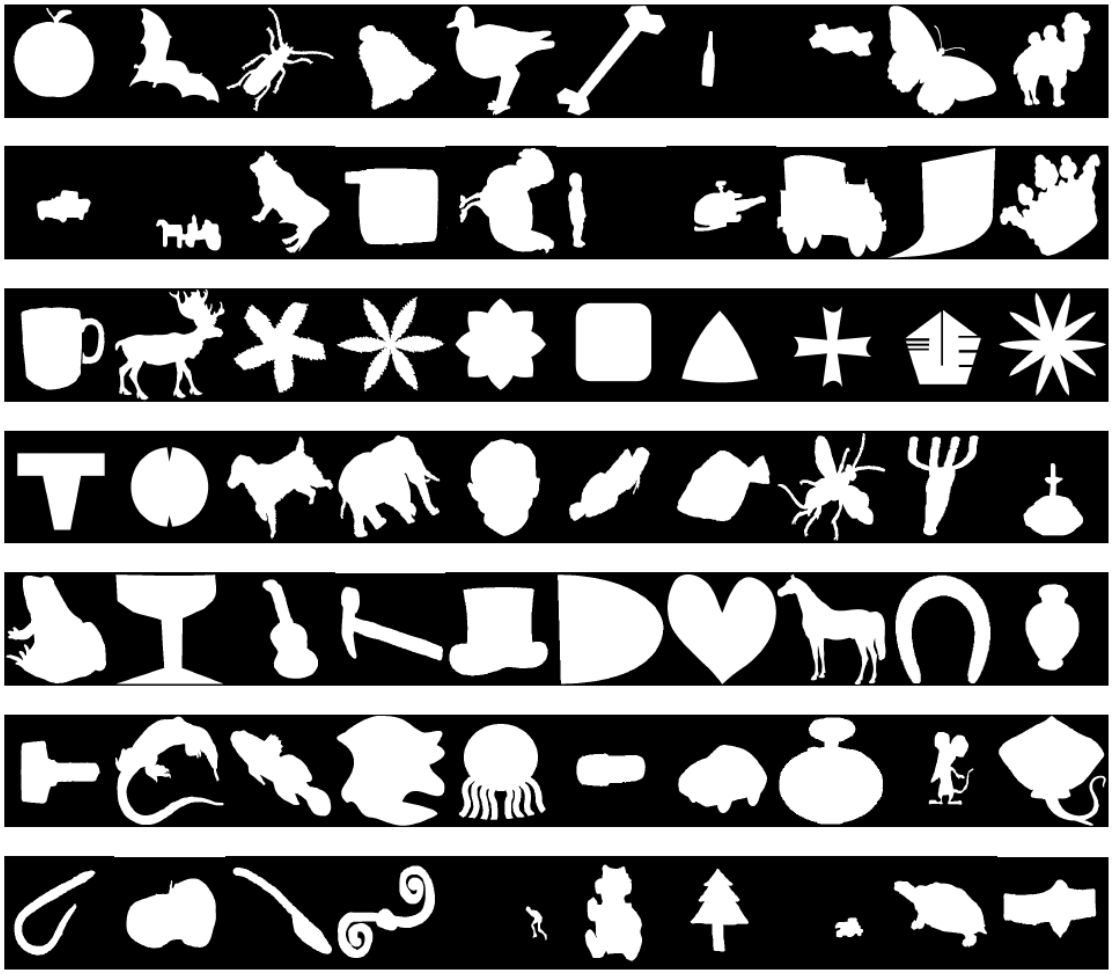
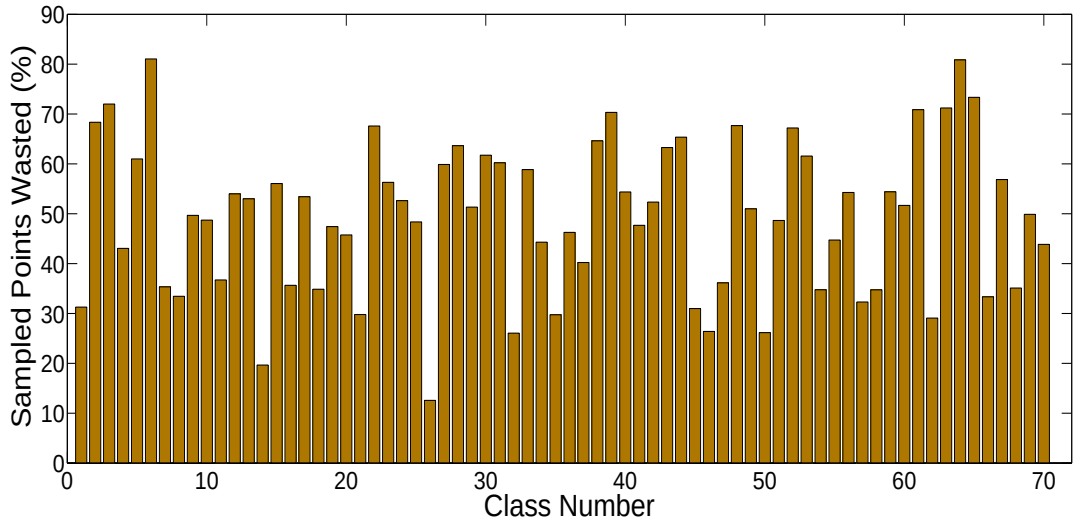


FIGURE 3.5: The figure shows example images from the MPEG-7 database. Shown above is an instance from each class. As can be seen, the database consists of images from both rigid and non-rigid objects.

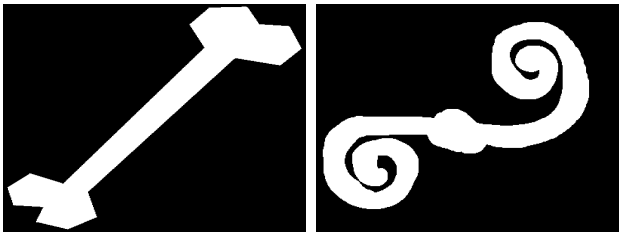
intra-class object similarity. Figure 3.5 shows an example object from each of the 70 classes.

3.4.1 Accept-Reject v. Constrained Sampling

Before reporting the image retrieval performance using the new descriptor that takes advantag of the dense interior points, we would like to show some statistics about the advantages of using our sampling procedure compared to the simple accept-reject technique. Figure 3.6 shows a bar chart of the average percentage of samples that were wasted for each of the 70 classes in order to generate 1000 uniformly spaced points from inside each object in the MPEG-7 database. As can be seen in the figure, every class requires us to sample more than 1000 points. The wastage is most apparent in classes 6 and 64, which produce the two tallest peaks



(A)



(B)

FIGURE 3.6: (a) The bar chart shows the percentage of samples that got wasted while generating 1000 uniformly spaced points lying inside the object's boundary, using the simple accept-reject sampling technique. (b) shows examples from class 6 and class 64, which represent the classes that produce the two tallest bars in the bar chart shown in (a).

in the bar chart shown in Figure 3.6a. Example images from these two classes are shown in Figure 3.6b. As mentioned in Section 3.3.1, the accept-reject technique will work well only if the shape occupies a considerable area of the bounding box from which the samples are generated. The accept-reject sampling technique required about 2200 points to be sampled in order to generate 1000 points lying inside the shape, on average, across the whole database. This shows an efficiency of less than 50% for the accept-reject sampling procedure. On the other hand, our technique of constrained sampling does not waste a single point as it is guaranteed that all the sampled points lie inside the object's boundary.

3.4.2 Image Retrieval

Now that we have verified the advantage of using our sampling technique over the accept-reject method, we proceed to demonstrate the advantage of using dense points in shape retrieval. The performance of the algorithm on the database is measured by the Bullseye score. To calculate the Bullseye score, each image is compared to every other image in the database. The top 40 best-matching images, for each image, are retained, of which at most 20 images can belong to the same class. Of the top 40 best matches, the number of objects belonging to the same class as the template image are counted. This number is divided by 20 to get the Bullseye score for the template image under consideration. The average Bullseye score over all the images in the database gives the Bullseye score for the complete database.

In our experiments, we set $N_{\mathcal{B}} = 100$ for triangulating the shape, N_{SP} to 300, N_{DP} to 2000, and α to 4. The mean of the costs obtained from IDSC was about four times the mean of all the SSC costs. The range of values was also smaller than the range of values from IDSC. Hence, the choice of $\alpha = 4$. We set N_{SP} to 300 as previous works ([57]) have used 300 feature points for shape comparison. Minor improvements were seen in the Bullseye score for $N_{SP} > 300$. No major decrease in performance was seen for $N_{SP} < 300$. A coarse representation of the shape is sufficient for interior sampling. With $N_{\mathcal{B}} = 50$, the overall shape boundary was not decipherable for some highly convoluted shapes. Thus, we increased it to 100. With further increases such as $\{200, 300, 400\}$, we did not see any major improvement in the overall results. We also tried experimenting with larger values of N_{DP} (2500, 3000, 3500 and 4000), but did not find any significant improvement in the Bullseye score. Table 3.1 lists our Bullseye score along with the Bullseye scores for various algorithms.

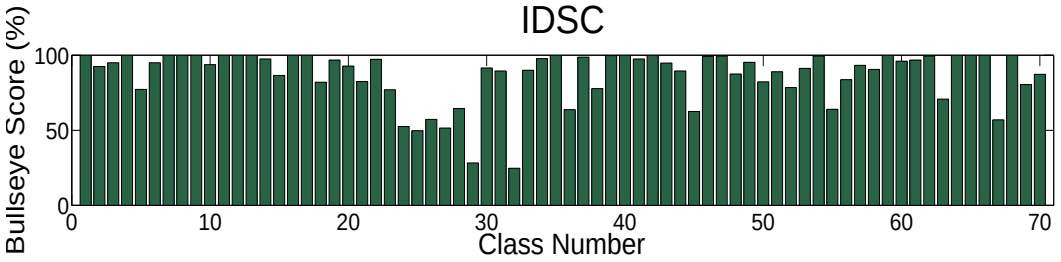
As can be seen from Table 3.1, quite a lot of work has been done in the area of shape matching. We fuse the costs from our algorithm with the costs from IDSC. Doing so significantly improves the Bullseye score from 85.40%, to 91.65%. The objects in the MPEG-7 database have different aspect ratios as well. Performing aspect normalization of shapes, as in [92], helps improve the Bullseye score further, to 91.83%. Temlyakov et al. [92] perform a similar fusion, and their algorithm helps improve the Bullseye score to 88.39%, while Hu et al.'s [39] method improves the score to 90.18%. We specifically compare our algorithm to these two methods as

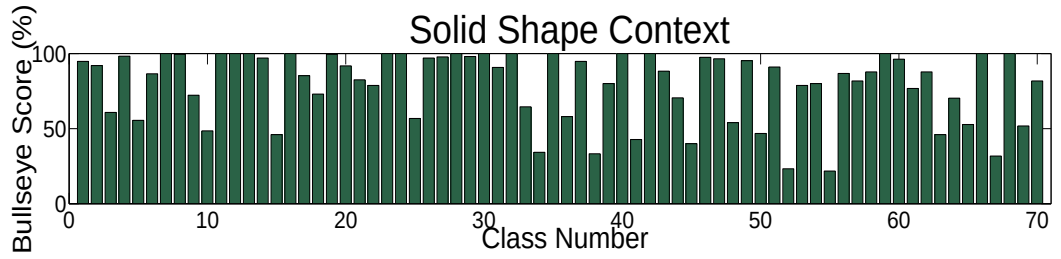
Algorithm	Bullseye Score
Visual Parts [52]	76.45%
SC+TPS [13]	76.51%
Generative Model [96]	80.03%
Curvature Scale Space [65]	81.12%
SSC	82.39%
Polygonal Multiresolution [5]	84.33%
Multiscale Representation [1]	84.93%
IDSC [55]	85.40%
Symbolic Representation [26]	85.92%
Hierarchical Procrustes Matching [64]	86.35%
IDSC(EMD) [56]	86.53%
Triangle Area [2]	87.23%
Shape Tree [30]	87.70%
ASC [57]	88.30%
IDSC+AspectNorm.+StrandRemoval [92]	88.39%
Contour Flexibility [104]	89.31%
IDSC+PMMS [39]	90.18%
IDSC+LP [106]	91.00%
IDSC+SSC	**91.65%**
IDSC+AspectNorm.+SSC	**91.83%**
IDSC+LCDP [107]	92.36%
IDSC+Affine Normalization [37]	93.67%
IDSC+AspectNorm.+StrandRemoval+LCDP [92][107]	95.60%
ASC+LCDP [57][107]	95.96%
IDSC+PMMS+LCDP [39][107]	98.56%
IDSC+SSC+LCDP	**98.85%**
IDSC+Affine Normalization+TPG [108]	99.99%

TABLE 3.1: The table gives a comprehensive list of shape matching techniques proposed in the literature, along with their respective Bullseye scores. We can see that our method helps in significantly improving the Bullseye score when fused with IDSC. Diffusion techniques, such as LCDP, further improve our Bullseye score.

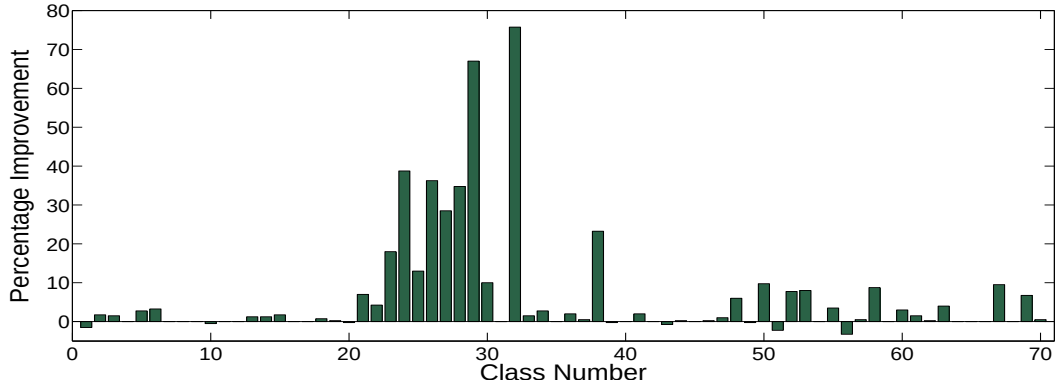
they are also perceptually motivated techniques. We would like to mention that the method in [92] requires the setting of threshold parameters for the identification of strand structures. Also, the method in [39] requires the selection of an appropriate structuring element and the identification of a proper scale at which to perform the morphological closing operation. Our method can help achieve a better Bullseye score without the requirement of such additional parameters.

Figure 3.7a shows class-specific Bullseye scores for both IDSC and SSC. We can see that the SSC performs better than IDSC for classes 21 through 32. These are the classes where there are a lot of indentations in the objects. Also, IDSC

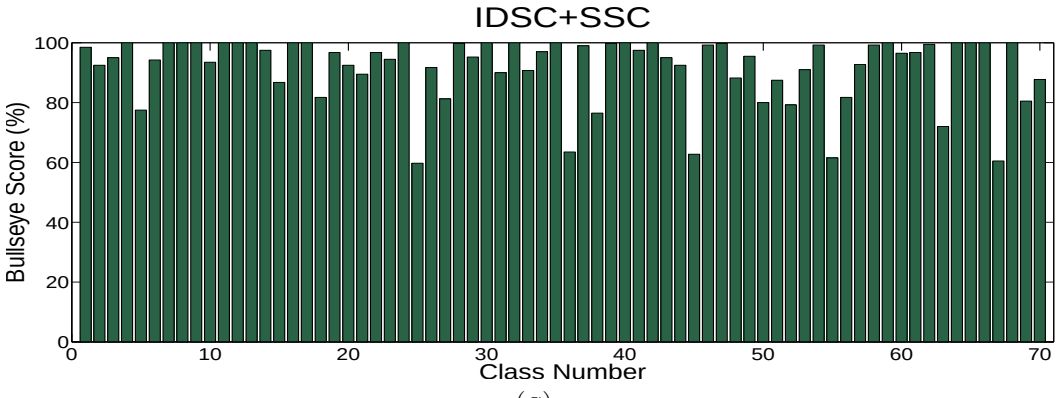




(A)



(B)



(C)

FIGURE 3.7: (a) Subfigure shows the class-specific Bullseye score for IDSC (top) and SSC (bottom). We can see that SSC complements IDSC. SSC performs better than IDSC for classes 21 through 32, while IDSC performs better than SSC for some other classes. (b) Subfigure shows the percentage gain in Bullseye score for each class, when SSC is fused with IDSC, over IDSC alone. We can see a significant improvement in the Bullseye score for the classes 21 through 32, which correspond to classes with visually similar objects, but having many indents in their contours. (c) Class-specific Bullseye score for IDSC+SSC. The bar chart shows a much more evened out score among all the classes.

performs better than SSC in some other classes. These classes correspond to the the classes of articulating objects. Ex: Lizard, Octopus, etc. From the bar chart, we can see that SSC complements IDSC well. Figure 3.7b shows the class-specific gain in Bullseye score when SSC is fused with IDSC, over IDSC alone. We can see a significant gain in the Bullseye score for a number of classes. Most of the classes that have a gain correspond to the classes where the objects have an overall visual similarity. Many objects in these classes have a number of indentations in their contours. Figure 3.7c shows the class-specific Bullseye score when IDSC is fused with SSC. We can see a much more evened out score among all the classes.

Figure 3.8 shows a comparison of the retrieval results for an example object. The first object, in each subfigure, is the query object and the rest are its top-40 best matching objects. The objects with green bounding box are correct retrievals and the objects with red bounding box are incorrect retrievals. As can be seen from the figure, IDSC retrieves just 3 correct objects, while SSC retrieves all 20 objects belonging to the same class as the query object. Moreover, while using SSC, all of the 20 objects lie closer to the top-20 locations.

In this chapter, we tackle the case where objects have minor and major indents in their contours. Our method works even when there are breaks in contours, such as the character 'M' shown in Figure 3.1. Fusing our method with the costs of IDSC means that SSC will take over whenever IDSC performs poorly, and vice-versa. So, in the cases of major protrusions from the shape's boundary, the cost from IDSC will take over. We show below, from experiments on the Kimia database, that fusing the two costs has very minimal negative effect on the overall results. To correctly match shapes with major protrusions, one might employ the strand removal method from [92], and fuse a third cost in Equation 3.11, as done by the authors of the same. No one method can correctly match all types of objects. This is why recent works (see Section 3.2) have adopted to fusing two or more costs. The results that we show in Table 3.1 are obtained by fusing just two costs. The Bullseye score will increase further if a third cost (from strand removal), or even more complementary costs [57], are fused together.

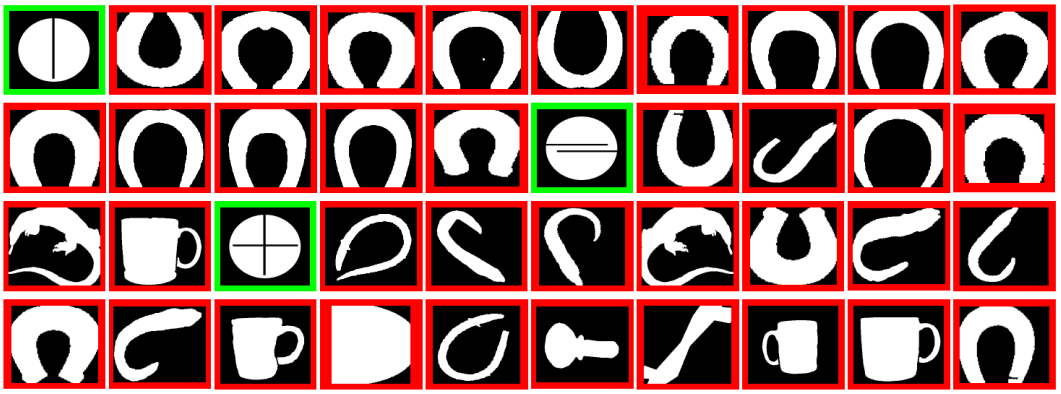
We use IDSC as the base algorithm since its code, and matrix, are easily available. From Table 3.1, we can see that [37] produces the best Bullseye score without manifold learning. However, as mentioned in Section 3.2, [37] decomposes the object into multiple convex parts and performs affine normalization of the individual parts. Doing so would cause unwanted partitioning of the objects such as those

shown in Figures 3.1 and 3.2. We would expect a high cost of matching when the top-left object in Figure 3.1 is matched with second object in the same figure, if we used the method in [37]. We believe that if SSC was combined with [37], it would improve its Bullseye score similar to how it currently improves IDSC's Bullseye score. The method of [37] is not designed for shapes with indentations (such as Figure 3.2), which our method is suited for, and therefore combining the two methods should produce a better overall cost as the two methods are otherwise compatible. SSC being complementary to IDSC, would also be complementary to the articulation invariant representation used in [37]. Thus combining SSC with [37] would help us get state-of-the-art results on the MPEG-7 database, before manifold learning.

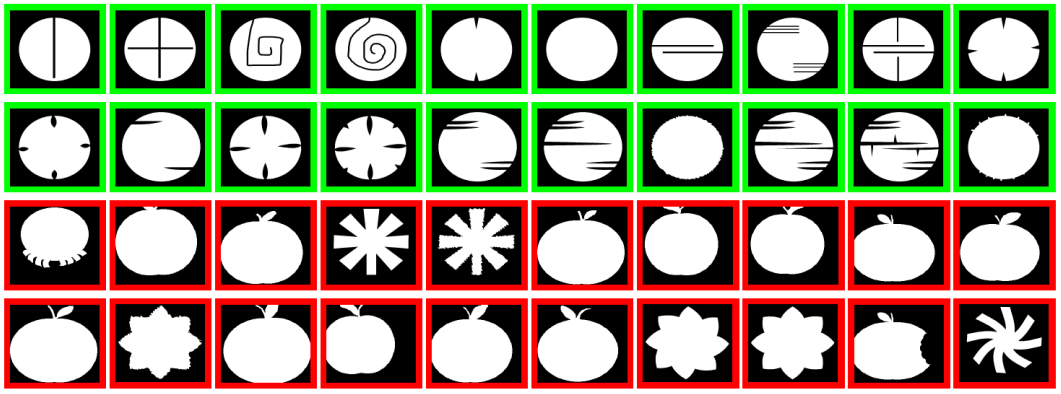
We also calculated the percentage of correct retrievals among the top-20 locations for IDSC and IDSC+SSC. When IDSC is used alone, it provides a correct retrieval percentage of 76.96%, while IDSC+SSC gives a correct retrieval percentage of 83.78%. We also calculated the average first position of a wrongly classified shape. For each shape, we find the location of the first wrongly classified shape, and take the average of this location over all shapes. The average first position of a wrongly classified shape, over all shapes, for IDSC, was found to be 14.43, and for IDSC+SSC, it was found to be 16.1371. Since there are 20 objects in each class, the best average first position of a wrongly classified shape is 21. So, the closer this number is to 21, the better. We can see that IDSC makes mistakes much earlier in the retrieval ordering when compared to IDSC+SSC.

In Section 3.2, we mentioned certain works that improved the retrieval results by allowing similar shapes to influence the pair-wise scores [6, 49, 106, 107]. These methods try to learn the underlying shape manifold structure and thus learn a better geodesic distance between two shapes. These methods take the pair-wise similarity matrix and perform diffusion on it. In order to end up with a better similarity matrix, it would be ideal if we had a good similarity matrix to start off. A good similarity matrix does not necessarily mean a matrix that produces a good Bullseye score. A good similarity matrix is one that has a low cost for similar shapes and a high cost for dissimilar shapes. This means that the costs between similar shapes, to begin with, are much more closer to the true geodesic distance on the shape manifold. Our similarity matrix does have such properties.

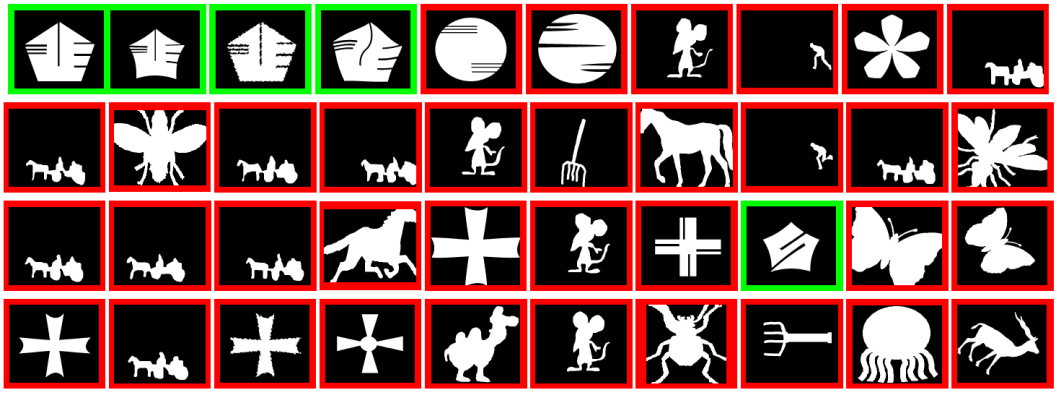
We use the Locally Constrained Diffusion Process (LCDP) [107] to perform the diffusion on our augmented matrix. The use of LCDP increases the Bullseye score



(A) IDSC



(B) SSC



(C) IDSC

(D) SSC

FIGURE 3.8: Figure shows the retrieval results for two example objects. (a) and (c) show the results obtained from IDSC, while (b) and (d) show the results obtained from SSC. The object being matched is at the top-left corner of each subfigure. All 20 objects, from the same class as the query object, get retrieved for both the examples, in case of SSC. However, IDSC is able to retrieve only 3 and 5 images, respectively.

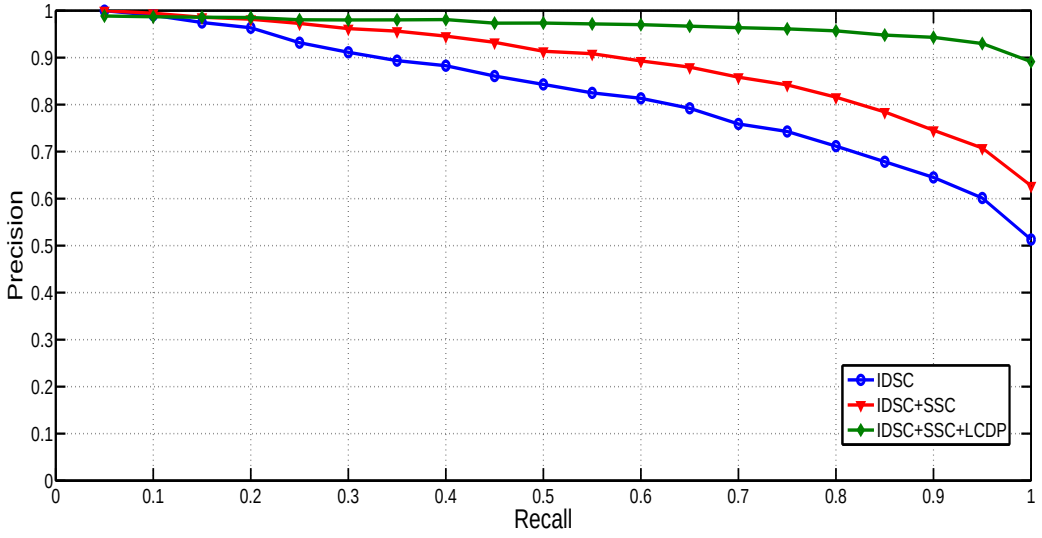


FIGURE 3.9: Precision-Recall curves for IDSC, IDSC+SSC, and IDSC+SSC+LCDP. We can see that the IDSC+SSC curve is clearly above the IDSC curve for all recalls.

of IDSC+SSC from 91.83% to 98.85%. The improvement of the Bullseye score to close to 100% shows that the matrix we started off with had good pair-wise similarity scores. The state-of-the-art results of 99.99% shown in Table 3.1 were achieved when diffusion was performed on the matrix from [37], which has a higher Bullseye score though it is not perceptually motivated. Moreover, the diffusion was performed using a Tensor Product Graph (TPG) affinity learning procedure [108], which uses higher order relations between shapes. We, on the other hand, show results that were obtained by performing diffusion using LCDP, which uses just single-order relations between shapes, and on a matrix that was obtained by augmenting the IDSC matrix. We use LCDP because it facilitates comparison with other techniques that also use LCDP [39, 57, 92, 107], and, in addition, its source code is available. However, we can use TPG for learning the shape manifold as well.

In Figure 3.9, we plot the precision-recall curves, as in [8]. The curve compares the precision of IDSC, with that of IDSC+SSC, over various recalls. From the figure, we can clearly see that IDSC+SSC has a better precision than IDSC alone, over all recalls. We also plot the curve for IDSC+SSC+LCDP.

We use IDSC as a base technique, and LCDP as the diffusion technique, since the code for both the algorithms is easily available. We stress that the costs obtained from our algorithm can be fused with the costs from any other algorithm. Finally, we tested our descriptor primarily on the MPEG-7 database as it is one of the most

challenging shape databases. Other databases such as Kimia database, Natural Silhouette Database, ETH-80 Shape Database, etc, are composed of relatively simple shapes, and have much lesser number of objects compared to the MPEG-7 database. Moreover, it is only the MPEG-7 database that has the perceptually similar shape classes with vastly different contour properties. We do, however, compare the average first position of a wrongly classified shape for the Kimia dataset [85] as well. When IDSC is used alone, the average first position of a wrongly classified shape is 11.5455, while that for IDSC+SSC is 11.3939; 12 being the ideal score, as there are 11 objects in each class of the Kimia database. This shows that even though the Kimia database does not have objects with intrusions in its contour (in fact, it has objects with major protrusions), fusing SSC with IDSC has very minimal negative effect on the overall results.

3.4.3 Extensions to 3-D objects

Not only does a dense representation of the objects help in describing 2-D objects, such a representation also helps in describing 3-D objects. Usually, the 3-D objects are represented using a triangulated mesh structure. This, however, does not guarantee that the vertices of the triangulated mesh are uniformly distributed across the surface of the mesh. In such cases of non-uniform distribution of vertices on the 3-D mesh, computing even simple metrics such as the principal components gets affected. We know that if an object is symmetric, then the plane of symmetry is along one of the principal planes of the object [23]. So, in applications that rely on the identification of symmetry, it is important that one be able to calculate the principal planes accurately. We demonstrate the usefulness of dense points in obtaining accurate symmetry planes. Given a principal plane, ρ, we calculate the symmetry error measure of the object as,

$$\Psi(\rho) = \frac{1}{|P_{left}|} \sum_{i=1}^{|P_{left}|} \min_{j \in \{1,2,\ldots,|P_{right}|\}} ||R_\rho(P^i_{left}) - P^j_{right}|| \tag{3.12}$$

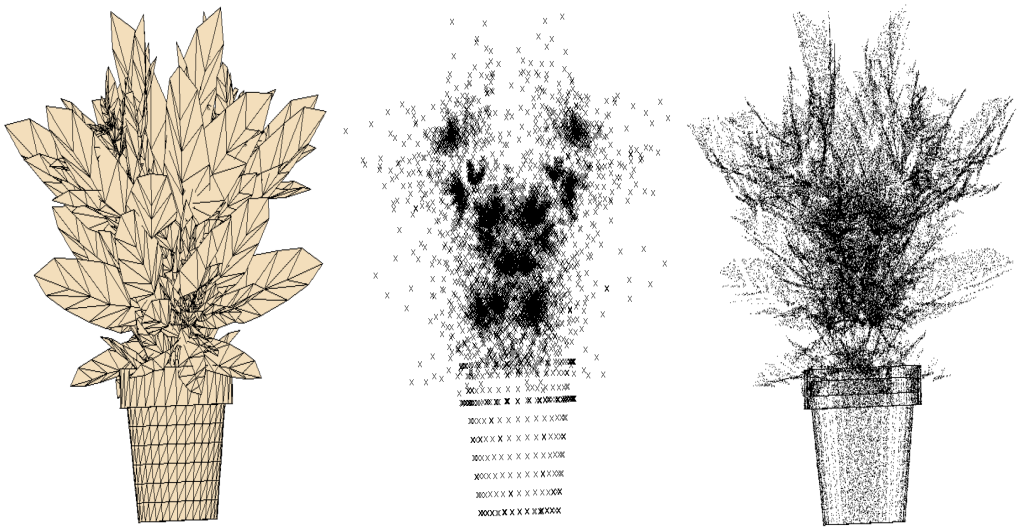
Here, $\Psi(\rho)$ is the symmetry error measure of the object across the plane, ρ, P is the set of points from the object, $|P_{left}|$ is the number of points to the left of the plane, ρ, $|P_{right}|$ is the number of points to the right of the plane, ρ, P^i_{left} is the ith point on the left side of the plane, and $R_\rho(P^i_{left})$ is the reflection of the point P^i_{left} across the plane, ρ. We calculate Ψ for all three principal planes and retain

the one with the least error measure. We obtain the symmetry error measure by computing the principal planes of the object using just the vertices of the object. We then repeat this procedure by calculating the symmetry error measure using the principal planes that were computed using the densely sampled points from the object.

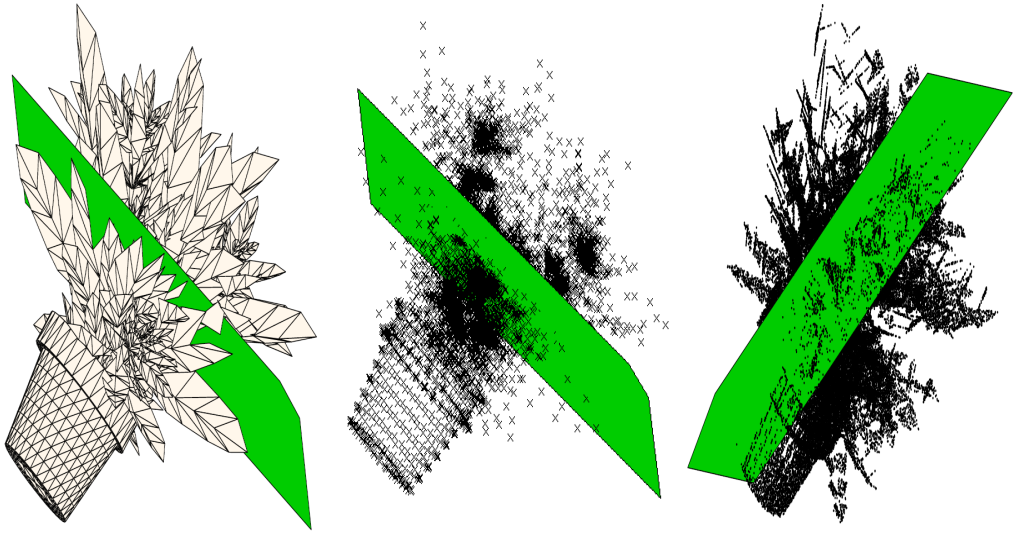
In Figure 3.10, we show a 3-D object, a potted plant, taken from the Princeton Shape Benchmark database [87]. The dominant principal component of the object is the axis going from the top to the bottom of the plotted plant. Also, the object is fairly symmetric across the plane containing the principal axis. The image on the left of Figure 3.10a is the triangular mesh, the one in the middle is the same object with just its vertices shown, and the one on the right is the point cloud obtained from the dense sampling of the mesh. Figure 3.10b shows the symmetric plane computed for the two input types (i.e., vertices and dense point cloud). We can see that the dense sampling of the points from the mesh helps in producing the correct symmetric plane (right of Figure 3.10b), while the non-uniform vertices of the triangular mesh produces an incorrect symmetric plane (left, and middle, of Figure 3.10b).

We experimented over all the shapes in the PSB database and found that the dense sampling of the points helps in calculating better symmetry planes for objects whose vertices were not spread uniformly across the surface mesh. For objects whose vertices were uniformly spread across the surface, the improvement that dense points produced, was minimal. In some cases, the error measure obtained from the point cloud was greater than the error measure obtained from using the vertices alone. However, the difference in the error measure was not large. We believe this happens because the dense points are sampled using random number generators, and that the randomization does not always lead to perfectly symmetric sampling of points across the plane of symmetry.

Equation 3.12 showed how symmetry can be detected in the spatial domain. Symmetry can also be detected in the spherical domain. The use of spherical harmonics allows us to specify the granularity of the shape while identifying the object's symmetry plane. It was recently shown in [43] that bilateral symmetry plane estimation may be carried out accurately, and efficiently, in the spherical harmonic domain. It was shown that the presence of bilateral symmetry in the 3-D shape is equivalent to a linear phase structure in the corresponding spherical harmonic coefficients.

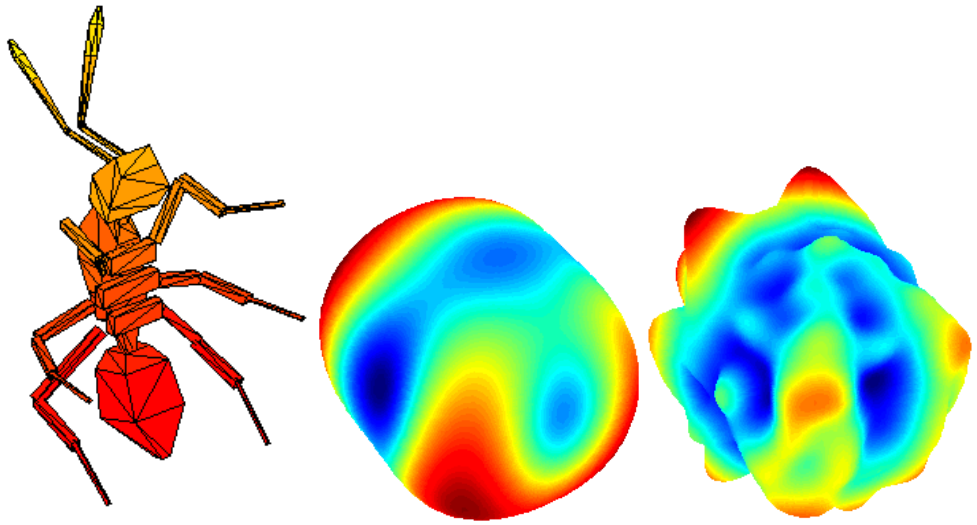


(A)



(B)

FIGURE 3.10: (a) A symmetric and elongated object, a potted-plant. The triangular mesh of the potted plant is shown on the left, the vertices of the mesh is shown in the middle, and the dense point cloud obtained using the dense sampling procedure is shown on the right. We can see that the densification brings out the shape better than the original vertices. (b) The symmetric plane calculated for the three inputs is overlaied on them. We can see that using a dense point cloud helps in properly detecting the symmetric plane.



(A) Vertex (B) L=4 (C) L=16

FIGURE 3.11: The ant model in (a) is approximated in its vertex mapping using two bandwidths in (b) and (c).

We experimented the effect of shape densification while calculating the bilateral symmetry plane in the spherical harmonics domain on the previously mentioned PSB shape dataset. Apart from the PSB dataset, we have also extensively tested on two other datasets, the SHREC Dataset [111] and the Mesh Segmentation Dataset [44]. We use a vertex mapping of the 3-D shapes to calculate obtain the spherical function, which is later used for symmetry estimation. Figure 3.11 illustrates the mapping of the vertices onto the sphere, and it also shows the mapping being performed at two different bandwidths.

From our experiments, we found that the error measure obtained while using the densified representations was always better than the error measure obtained while using the vertices alone. Table 3.2 compares the average error measure between densified models and the original models across the three databases. The data shown in the table, collected from over 5000 images in total, clearly shows that densification of shapes does indeed help in better representation. Figure 3.12 shows some other examples of symmetry detection.

3.5 Conclusions and Future Work

In this chapter, we identified certain problems that traditional contour-based shape matching techniques face while performing shape matching. We showed that the

Dataset	Error Measure (%)
PSB Dataset (Original Vertices)	8.6
PSB Dataset (Densified Mesh)	**5.6**
SHREC'10 Dataset (Original Vertices)	10.4
SHREC'10 Dataset (Densified Mesh)	**6.9**
Mesh Segmentation Dataset (Original Vertices)	9.9
Mesh Segmentation Dataset (Densified Mesh)	**6.1**

TABLE 3.2: The table compares the average error measure in finding the symmetry planes, with and without densification, across three standard 3-D shape databases. We can see that densification helps in obtaining a better error measure than the original vertices.

FIGURE 3.12: Examples of bilateral symmetry plane detection using the spherical domain.

shape interiors play an important role in object recognition, and proposed a perceptually motivated variant of the well-known Shape Context descriptor, which captures the shape properties in their entirety. We showed the benefits of modelling the interior properties of the shape using *Dense Points*. We proposed a new way for sampling from within a shape boundary, in order to capture the interior properties of the shape. We then listed out the advantages of using the convex hull of a shape to select the landmark points. We also showed how augmenting traditional shape matching techniques with the costs from our SSC descriptor can

significantly improve the retrieval rates. We hope that this work of ours would motivate other researchers in the community to take the area of shape matching to the next level, by coming up with other perceptually motivated techniques.

In addition to proposing the shape descriptor, we showed that densification not only helps in better capturing the shape properties of 2-D objects, but also helps in describing 3-D objects. This was made apparent from our experiments on 3-D objects for bilateral symmetry plane estimation, where we found that the error measure was also smaller when the shapes were densified, over the usages of vertices alone.

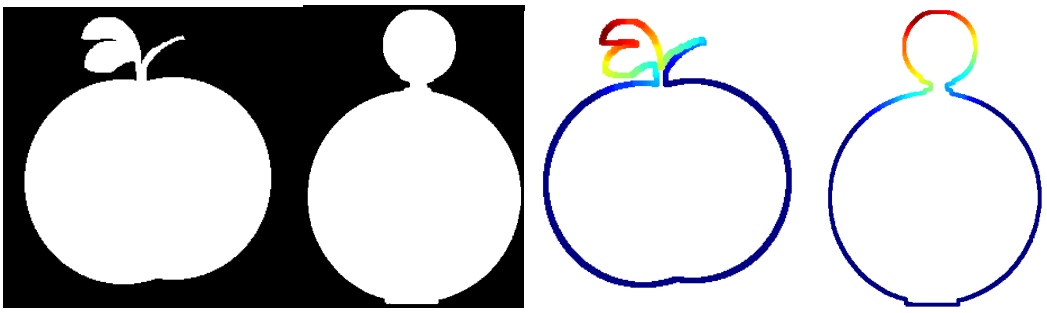
Most shape descriptors, such as the ones discussed in this chapter, make the implicit assumption that all parts of a shape are equally important. We feel that there are certain parts of the shape that have more information content in them than others. Therefore, in the next chapter, we present the concept of discriminative shape parts and propose a way to identify them.

Chapter 4

Identifying discriminative parts

While looking at images of objects, we perceive a wide variety of information such as color, shape, contrast, etc. Among these, shape seems to have a big effect on object memorability. Understanding object shapes is a key towards recognizing objects, and this has been acknowledged by the computer vision community. The object recognition/image retrieval literature has many works related to shape matching, and the development of shape descriptors [13, 55]. However, to the best of our knowledge, there is no work that looks at importance of object parts. Similar to the definition of an object shape in Section 1.1, the shape of an object part can be defined as the geometrical description of the space occupied by a portion of the object. In this chapter, we would like to ask the following questions. What parts of an object's shape make it distinctive? Are the distinctive parts consistent among all shape instances of a particular class?

Consider the example images shown in Figure 4.1. The silhouette in Figure 4.1a is that of an apple, while the silhouette in Figure 4.1b is that of a pocket watch. We have no difficulty in distinguishing between the two even though a large portion of the object is quite similar to each other. Given such shapes that have strong overall visual similarity, we tend to give more importance to the differences in the two shapes while distinguishing between them. Ex: The stem of the apple is considered to be the distinguishing part between the two objects shown. Similarly, we would like the shape matching algorithms to automatically identify the discriminative regions. Figures 4.1c and 4.1d shows the results from our algorithm, which is able to successfully identify discriminative regions.



(A) (B) (C) (D)

FIGURE 4.1: (a) and (b) show two example images that are globally similar to each other. (c) and (d) show the heat maps on the contour, which indicate the level of importance. Most important regions are depicted by red, and the least important regions are depicted by blue.

Many state-of-the-art shape matching techniques cannot make such fine-scale distinction. There is an implicit assumption that all parts of a shape are equally important. This assumption might be detrimental towards the development of better shape matching techniques in the future. Widely-used shape descriptors such as Shape Context (SC) [13], and Inner Distance Shape Context (IDSC) [55], get confused between the two classes shown in Figure 4.1. Hence, there is a need to develop techniques that can match different parts of the shape with different levels of importance.

While previous work has aimed at developing shape descriptors, and better shape matching techniques, in this work, we try to identify the "important" regions of a shape. The term "important" is quite subjective. So, we define important parts of the shape to be the discriminative parts of the shape, which helps enhance intra-class shape similarity, and reduce the inter-class shape similarity.[1]

Thus, the goal of this chapter is to introduce the concept of discriminative shape parts. To this end, we propose techniques for decomposing a shape into parts and show how to identify the discriminative parts among them. We also develop a measure of consistency of discriminative parts among various class instances.

[1]The terms important, discriminative, and distinctive, are used interchangeably throughout the chapter.

4.1 Related Work

Shape matching, being an active area of research, has numerous publications attributed to it. In this section, we refer some of the relevant and influential work from this area. Belongie et al. [13] proposed the Shape Context (SC) shape descriptor, which is a 2-D histogram of distances and angles. They also showed that good-quality matching can be done using uniformly sampled points from the shape's contour as opposed to using points of extreme curvature as landmark points. Ling et al. [55] extended this work and proposed the Inner Distance Shape Context (IDSC) to perform articulation invariant shape matching. Both these descriptors are global shape descriptors and try to match the global visual similarity between two shapes.

Bronstein et al. [21] linked the concept of Pareto-optimality to quantify matching of partial shapes. This was later used by Donoser et al. in [29], where they propose a new shape descriptor that can perform partial matching of shapes. While there has been a progress from matching shapes globally to matching partial segments of the shapes, there is still no work that looks at the importance of various parts of the shape.

Most current-day shape descriptors, whether they are global shape descriptors [13, 55, 76], or partial shape descriptors [29, 61], are obtained with the help of uniformly spaced points on the object's contour. Hence, there is an implicit assumption about the uniform importance of all parts of the shape. However, as we have seen from the example in Figure 4.1, there are some parts of the shape that need to be given more importance than others.

Xu et al. [104] measure the contour flexibility at various points on the contour. They calculate the bendable potential at each point on the contour and show that flexibility is a characteristic of the shape, which is not uniformly distributed across the complete contour. This work shows that certain parts of the contour have different characteristics compared to the rest of the contour. The bendable potential, in the work of Xu et al., was not calculated using the knowledge about the classes that the shape belongs to. We, on the other hand, exploit the information present in the class labels to identify regions of the shape that are highly distinctive when compared to others.

The closest work related to the identification of distinctiveness/uniqueness of images that we could find was not in the area of shape matching, but in the area of image memorability. The authors of [70] and [46] try to identify "memorable" regions of an image by looking at some of the intuitive features extracted from the same. They try to understand what is it about an image region that makes it memorable. In similar vein, we try to identify what is it about a shape that catches our eye, or makes it distinctive.

4.2 Discriminative Parts

4.2.1 Extracting Contour Segments

Following a learning paradigm, let us consider the set of all positive shapes to be $\mathcal{S}^+ = \{S_1^+, ..., S_{|\mathcal{S}^+|}^+\}$, and the set of all negative shapes to be $\mathcal{S}^- = \{S_1^-, ..., S_{|\mathcal{S}^-|}^-\}$. By our definition of important parts, we need to find those parts in a given shape S_i^+, which maximally discriminates it, among all other negative shapes, $\mathcal{S}^-$. In the classification literature, good decision boundaries are usually obtained by comparing positive class instances with "hard" negative examples. Hard negatives are those instances that usually turn up as false positives. We take a similar approach while comparing the shape S_i^+ to the set $\mathcal{S}^-$.

We identify the "hardest" example as,

$$S_h^- = \min_{S_j^- \in \mathcal{S}^-} \Psi_{IDSC}(S_i^+, S_j^-), \tag{4.1}$$

where S_h^- is the hardest example, $\Psi_{IDSC}(S_i^+, S_j^-)$ is the cost obtained while comparing the positive example S_i^+ with a negative example S_j^- using IDSC [55]. Considering that there are noisy instances in the negative class, we not only compare the shape S_i^+ to its hardest counterpart, but to its k hardest counterparts.

Given a shape S_i, we extract its contour $\mathcal{C}_i$. For identifying distinctive parts, we first split the contour into multiple contour segments

$$\mathcal{C}_i^{seg} = \{C_i^{seg_1}, C_i^{seg_2}, ..., C_i^{seg_p}\}. \tag{4.2}$$

Dividing a shape into multiple semantic regions is a difficult task. The usual tendency is to make cuts on the contour at regions of extreme curvature. However, one then has to find good thresholds to define "extreme curvature". In our approach, we do not make the cuts at corner points, or points of extreme curvature. We randomly split the contour into p segments, each of equal length. We repeat this procedure T times. So, we end up with T random contour cuts, with each cut dividing the contour into p parts. Randomly cutting the contour into different parts does not need any manual supervision in terms of threshold selection, or identification of "meaningful" parts. In Section 3.4, we show from examples that randomly cutting the contour into parts does not affect the identification of semantically meaningful segments.

To compare contour segments of the i-th positive shape, $\mathcal{C}_i^{seg^+}$, against the contour segments obtained from the k hard negative examples, $\mathcal{C}_{h_{(1,\ldots,k)}}^{seg^-}$, we need a descriptor to describe the segments. It is important that the shape descriptor be a partial shape descriptor and not a global shape descriptor, as we are trying to compare parts of different shapes.

We choose the shape descriptor proposed by Donoser et al. [29], as it has many good properties. Firstly, it is simple to compute the descriptor. The descriptor was defined in Section 2.3, but is redefined here for convenience. The descriptor of a contour segment with n landmark points is a matrix of size $n \times n$, with the (u, v)-th entry given as,

$$\alpha_{uv} = \angle(L_{u,v}, L_{v,v-\Delta}), \tag{4.3}$$

where u and v are two points on the contour, $L_{u,v}$ is the line segment joining the two points, and Δ is an indicator of the number of points before point v with respect to which the angle α_{uv} is calculated. Secondly, the descriptor is invariant to rotations. And, finally, this shape descriptor can be used for quick computation of distances between any two contour segments of any length. The authors of [29] make use of the integral image data structure to perform this efficient computation. More details about their descriptor can be found in their paper [29]. We use equal length segments in our experiments since the quick look-up using the integral images data structure is possible only while matching two equal length segments.

4.2.2 Distinctiveness of a Segment

The most distinctive positive segment is the one that is least likely to be found among the segments obtained from the negative classes. Therefore, we define the distinctiveness of a particular contour fragment to be inversely proportional to the likelihood of finding it, given the negative classes. The likelihood can be calculated in many ways. We can either train a max-margin classifier and find the distance of each segment to the margin, or we can model the distribution of the positive and negative segments using some prior knowledge, and calculate the likelihood of a new contour fragment belonging to either of the two classes. In this chapter, we compute the distinctiveness of a segment using the distances obtained by matching it to other negative contour segments.

Let $d(s,t)$ denote the cost of matching the s-th segment from a positive shape S_i^+, to the t-th segment from a negative shape S_j^-. $d(s,t)$ is the normalized Frobenius Norm between the respective shape descriptors. The likelihood of a segment being distinctive is then directly proportional to the distance between that segment and other negative segments. We define the log-likelihood of a segment being distinctive as

$$\log P(C_i^{seg_s^+} | \mathcal{C}^{seg^-}_{h_{(1,\dots k)}}) = \frac{1}{|\mathcal{C}^{seg^-}_{h_{(1,\dots k)}}|} \sum_j d(seg_s^+, seg_j^-), \tag{4.4}$$

where $j \in \{1, ..., |\mathcal{C}^{seg^-}_{h_{(1,\dots k)}}|\}$, and $|.|$ denotes the size of the set. To reduce notational clutter, from now on, we denote $C_i^{seg_s}$ as seg_s, $\log P(C_i^{seg_s^+} | \mathcal{C}^{seg^-}_{h_{(1,\dots k)}})$ simply as $\mathcal{L}(seg_s^+)$, and $P(C_i^{seg_s^+} | \mathcal{C}^{seg^-}_{h_{(1,\dots k)}})$ as $P(seg_s^+)$. The probability of a particular segment, seg_s^+, being distinctive can now be written as

$$P(seg_s^+) = \frac{1}{Z} \exp(\mathcal{L}(seg_s^+)). \tag{4.5}$$

Z is the normalization constant, which is calculated as

$$Z = \sum_{seg_s^+ \in \mathcal{C}_i^{seg^+}} \exp(\mathcal{L}(seg_s^+)). \tag{4.6}$$

From our experiments, we found that the log-likelihood Equation 4.4 need not be calculated as the average distance over all negative segments. It can be obtained

using the average distance over just the k closest segments to the segment under consideration.

We use the method just described to calculate the distinctiveness probability for all segments. Taking the average probability over all random cuts gives us the heat map that was shown in Figures 4.1c and 4.1d. More examples can be found later in Figure 4.3. As can be seen from these figures, our method is able to consistently identify discriminative regions that are semantically meaningful.

4.2.3 Part Consistency

The method described in the previous two subsections helps us to get the heat maps of a particular shape, which indicates the importance of a particular part. If there are many object instances in a particular class, we would like that the heat maps of different objects indicate similar importance to similar parts. In this subsection, we propose a way to perform this check.

We start off by finding correspondences between points on pairs of shapes. This can be achieved using IDSC. IDSC not only finds the cost of matching two shapes, but it also gives a correspondence between points on the shapes using dynamic programming. We randomly choose a particular shape as the reference shape, S_{ref}, and calculate the point correspondences between the reference shape and every other shape in the class. For each point on the shape, we can also obtain the distinctiveness value using the method described above. Thus, for M objects in a class, each with N points on them, we can generate an $N \times M$, matrix of distinctiveness values, which we denote as Q. The entry $Q(r, c)$ corresponds to the distinctiveness value of a point in shape S_c, whose corresponding point in the reference shape, S_{ref}, is r. We ignore occlusions while populating the matrix Q by forcing a corresponding point to exist in the second shape. In our experiments, we achieve this by setting the threshold for identifying an occluded point to infinity.

If the importance values of all the corresponding points agree with each other, then the singular values of the matrix, Q, arranged in descending order, should drop off steeply. We quantify this steep drop using the area under the curve (AUC) of the plot of the top M singular values. The larger the AUC, the more disagreement there is between the corresponding points of shapes in a class. In the

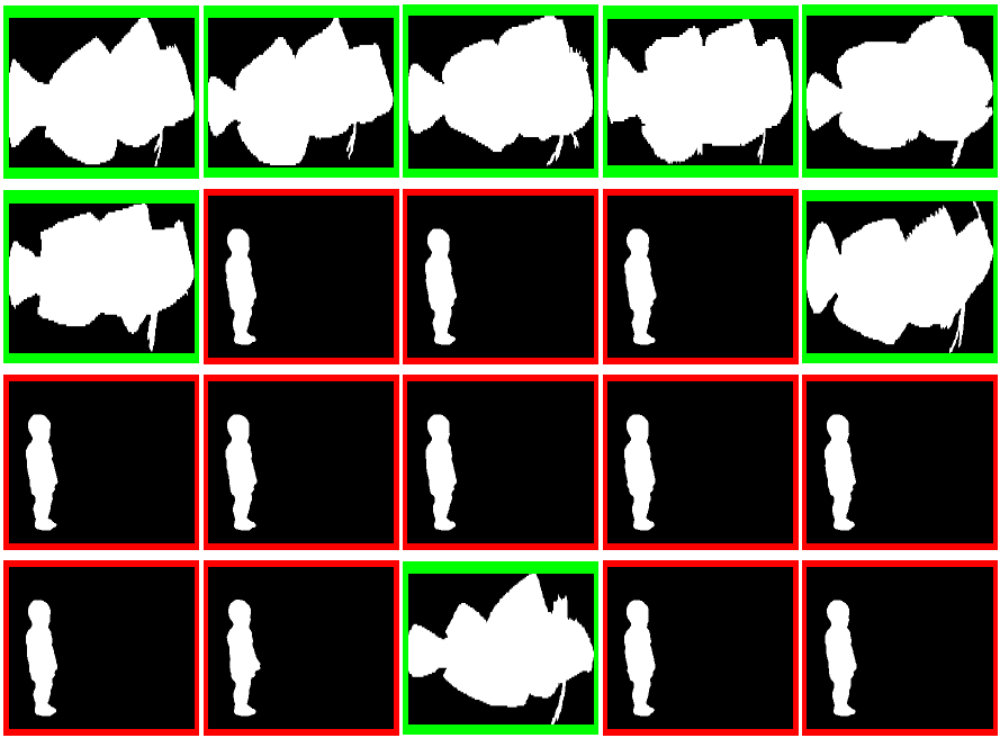
next section, we show plots of the singular values and their corresponding AUCs for some example classes.

4.3 Experiments

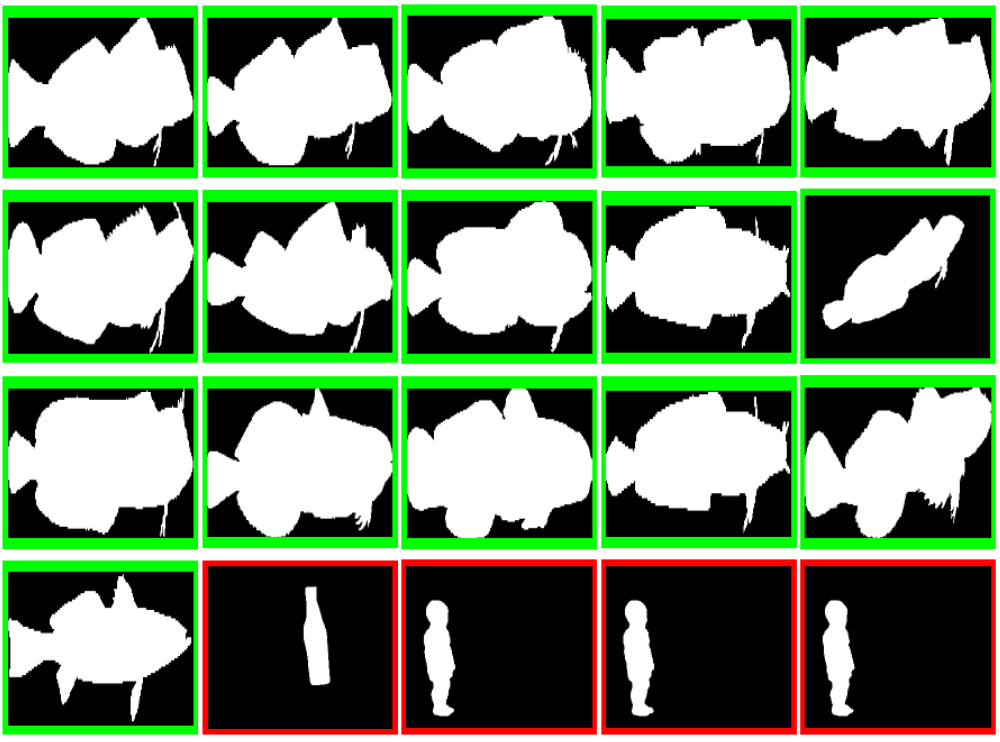
We conduct our experiments on the well-known MPEG-7 CE-Shape-1 Part B dataset. The dataset consists of 1400 images grouped into 70 different classes, with each class containing 20 example images. Most state-of-the-art shape matching techniques get confused while matching shapes that have overall visual similarity. So, we try to identify the distinguishing parts for each object in the database using the method described in Section 4.2. The top row of Figure 4.3 shows the output of our algorithm for 5 instances of a particular class, 'Apples'. We can see that even though our method uses random cuts to segment out parts, we are still able to localise on the semantically meaningful segments. Also worth noting is that our method is able to consistently identify the same segments, across all instances, as the discriminative segment of apples.

Figure 4.4 shows the plots of the singular values for some example classes. We have also shown the corresponding AUC on the graph. The agreement among what is considered as distinctive among class 'Apples' (Figure 4.3) is confirmed by the small AUC. For classes such as camels, which are articulating objects, the articulation causes different parts to be identified as distinctive. However, we found that the humps of the camel did stand out to be more distinctive than other parts (final object in row 3 of Figure 4.3). Figure 4.3 also shows more examples outputs from our algorithm.

Finally, we implemented a variant of IDSC that used non-uniform sampling as opposed to the regularly used uniform sampling of the object contours. We sampled points based on the heat values at different locations of the object. More points were sampled from regions that were considered to be more important, and less points from regions that were less important. We then generated the IDSC shape descriptor at each of the original uniformly sampled landmark points. The descriptor is now affected by the extra "important" points. We can consider the new descriptor as a weighted generalization of the uniformly weighted shape context. Figure 4.2 shows a comparison of the retrieval results obtained using the original descriptor and our importance sampling based descriptor. We can see that our



(A)



(B)

FIGURE 4.2: Figure shows the top 20 best matching objects to the fish shown in the first image. The green boxes indicate correct retrievals, while the red boxes indicate incorrect retrieval. (a) Retrieval results from uniformly sampled IDSC. (b) Retrieval results using IDSC generated using importance sampling. Importance values were obtained using the method described in Section 4.2.

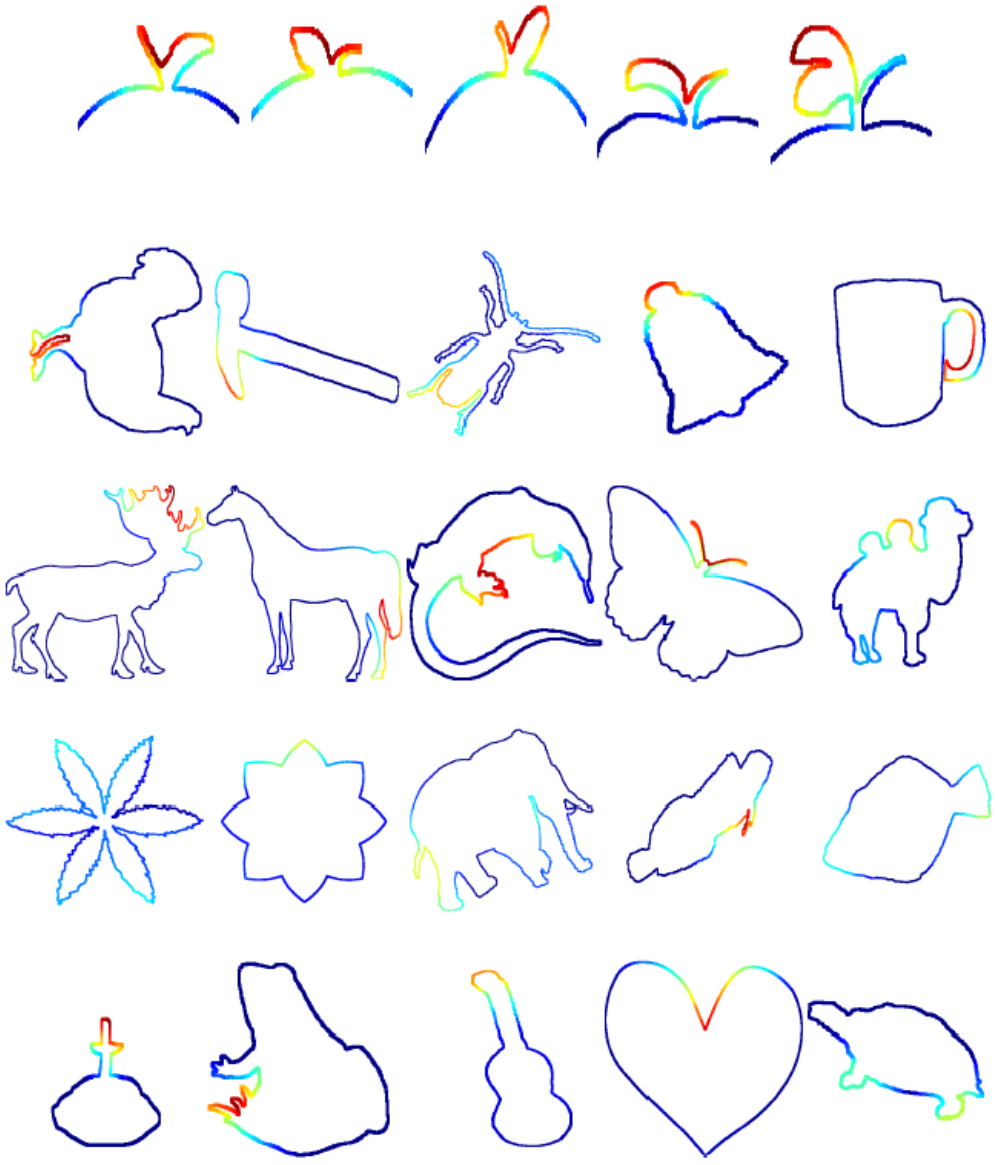


FIGURE 4.3: Figure shows the importance heat maps of some example objects generated using the method described in Section 4.2. Top row shows the cropped-out heat maps of 5 apples. Notice that our method is able to consistently identify the semantically meaningful regions.

descriptor is able to retrieve more correct examples than the original IDSC, which gets confused by globally similar objects.

Figure 4.4 also shows the precision/recall plots for two example classes. The P/R plots are calculated as in [8]. Ideally, we would like the P/R plot to be at 100% precision for all levels of recall. So, greater the area under the curve, better is the retrieval method. As can be seen from the plots, our method of generating the descriptor (using importance sampling of the contour) helps improve the precision of the retrievals.

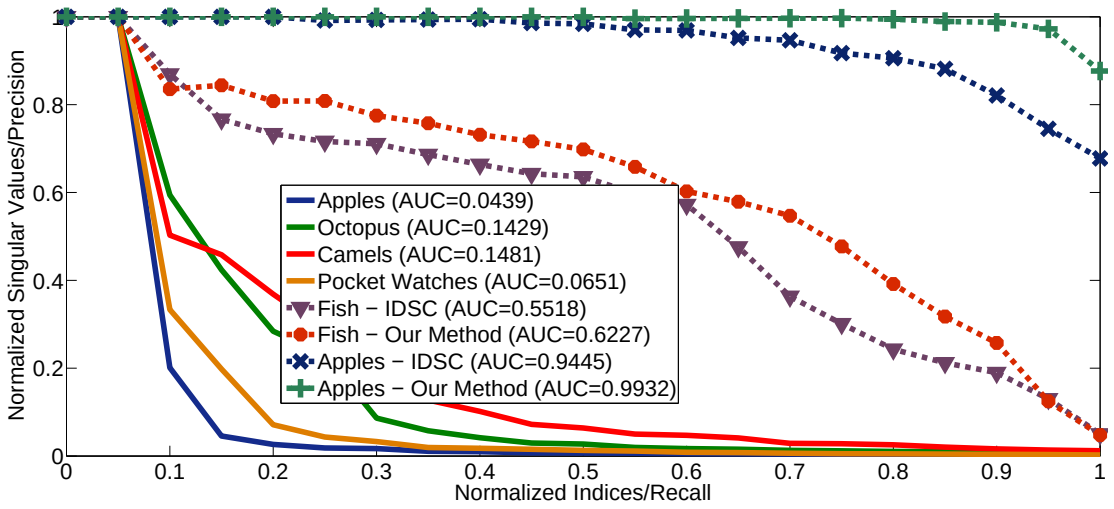


FIGURE 4.4: Bold lines show the singular value plots. The dashed lines are the Precision/Recall plots.

4.4 Conclusion

In this chapter, we have introduced the concept of distinctive parts. We have explained why it is important to look out for important regions in contours. We provide a simple way to extract the distinctive parts of any given object. Results from our experiments shows that the method is promising. As a part of future work, we would like to explore how we can make better use of the important discriminative regions to come up with more discriminative shape matching techniques.

Chapters 3 and 4 have predominantly dealt with the pairwise matching of shapes. In the next chapter, we try to go beyond pairs of shapes and make use of the complete shape database to obtain the "true" distances between shapes. We contribute to the area by proposing a simple and effective way to select a robust neighborhood for use with diffusion techniques on shape manifolds.

Chapter 5

Robust neighborhood selection in graph-based manifolds

5.1 Introduction

Using the underlying manifold structure has proven to significantly improve performance in many vision-related tasks [97, 107]. The most notable of them is the task of shape/image retrieval. The task of shape retrieval is an especially difficult task because of the vast variability of shapes even within a particular class. Given a query object, the goal of retrieval tasks is to retrieve the most similar shapes in the database. Similar objects are usually retrieved using some similarity/dissimilarity measure (ex: [13, 55]), which is computed between pairs of shapes. Many of these similarity/dissimilarity measures violate the triangle inequality and, hence, are not metrics. If the underlying manifold structure of the shapes is curved, then the Euclidean distance between shapes cannot be a good metric for shape comparison. In such cases, the geodesic distance on the shape manifold is a better metric for comparing shapes than pairwise similarity/dissimilarity measures.

Many techniques have been proposed to capture this underlying manifold structure, and hence learn the correct geodesic distances between data points that lie on the manifold. Since the manifolds are usually of a much lower dimensionality than the space in which they lie, many manifold learning algorithms make use of dimensionality reduction methods (ex: [81, 83]) to reduce the dimensionality of the feature space. Such dimensionality reduction techniques map the features

onto a lower dimensional subspace in hope that the Euclidean distance in this new lower dimensional subspace can capture geodesic distance from the original higher dimensional space.

In applications such as shape retrieval, we might not have access to the features of the data points, and might be forced to work in the data space. The shape manifold in the data space is represented as a graph with edge weights proportional to the similarity score. The true distance between two shapes can be learnt by considering the distance in context of other shapes in the neighborhood. The new distances are calculated by propagating the similarity information along the weighted edges of the graph.

Many recent papers make use of such contextual information to learn new affinity scores between pairs of data points [6, 49, 106–108]. The similarity information is usually propagated as a diffusion process on the graph. Yang et al. [107] perform diffusion on a locally constrained sparse graph, while in [108], the diffusion process is performed on a tensor product graph, thus allowing the capture of higher order information. Donoser and Bischof [28] present a generic framework for diffusion processes of which some of the above mentioned works are specific instances. The diffusion process is susceptible to noise [41], and hence the affinity propagation is not performed on a fully connected graph. Both [107] and [108] follow different styles of graph sparsification (k-nearest neighbors and dominant neighbors, respectively). The selection of a proper neighborhood is critical for diffusion to work.

This chapter addresses the question of how to build a strong local neighborhood given data in the form of a graph. In addition, we point out the drawbacks of using k-nearest neighbors (k-NN) and dominant neighbors (DN). Dominant neighbors are the nodes that form a maximal clique in a graph (Section 5.3.1 explains the process of finding the dominant neighbors in detail). We propose a new way for neighborhood selection by making use of the consensus information from various neighborhoods, and show that such a neighborhood is much more robust to parameter selections. We also show that using our consensus neighborhood information, we are able to achieve better retrieval results on standard databases (Ex: MPEG-7 shape database), as opposed to the use of k-NN or DN.

5.2 Sparse Affinity Matrix Generation

Given N shapes from a database, we can generate a $N \times N$ cost matrix by pairwise comparison of each pair of shapes s_i and s_j, $\forall i, j \in \{1, ...N\}$, using standard shape comparison methods such as [13, 55]. Ideally, we would want the shape comparison methods to generate costs such that similar shapes have less cost and dissimilar shapes have greater costs between them, and for them to obey the triangle inequality. Unfortunately, this is not true of any of the shape matching techniques. In order to learn the true geodesic distances on the shape manifold, contextual information has to be taken into consideration. Using the $N \times N$ matrix allows us to exploit much more information about the neighborhood structure of the data manifold than using just the pairwise information.

5.2.1 Affinity Matrix

The usual trend is to work with similarity scores rather than dissimilarity costs. The authors in [106] convert the cost matrix into a similarity matrix (also named as affinity matrix) and use this affinity matrix to learn the geodesics on the manifold. The affinity between a pair of shapes is calculated as,

$$A(i, j) = \exp(-D(i, j)^2/\sigma_{ij}^2). \tag{5.1}$$

Here, A is the affinity matrix, D is the distance matrix, and σ specifies the kernel size. The choice of σ is critical in generating a "good" affinity matrix. A "good" σ would help in pulling intra-class objects together and in pushing inter-class objects far away from each other. Different methods have been proposed for choosing a proper σ. Perona et al. [73] use σ_{ij} as

$$\sigma_{ij} = \sigma_i \sigma_j, where \;\; \sigma_i = D(i, K(i)), \tag{5.2}$$

where $K(i)$ is the Kth nearest shape to shape s_i, and Yang et al. [106] use σ_{ij} as

$$\sigma_{ij} = mean(K\text{–NN}(s_i), K\text{–NN}(s_j)), \tag{5.3}$$

where $mean(K\text{-NN}(s_i), K\text{-NN}(s_j))$ is the mean of the K-nearest distances of shape s_i and shape s_j. Both these methods rely on the proper choice of the kernel

neighborhood parameter, K, and choosing a "bad" K, would adversely affect the generation of a good affinity matrix.

5.2.2 Local Neighborhood Sparsification

While performing similarity propagation on a graph, it is extremely important to prune out noisy edges as the diffusion process is susceptible to noise [41]. Roweis et al. [81] assumed linearity of local neighborhoods on a manifold, and graph sparsification follows a similar principle. It is assumed that the edge weights between data points that are close to each other on the data manifold approximate the geodesic information better than the edge weights between data points that are farther away from each other. Hence, graph sparsification tries to prune out edges between nodes that are not in a local neighborhood. For a graph $G(V, E)$, we can have different variants of the neighborhood graph G', such as,

- Symmetric k-NN graph neighborhood $G'(V, E)$, where there is an edge $E(v_i, v_j)$ if $v_j \in k\text{-NN}(v_i)$ or $v_i \in k\text{-NN}(v_j)$.
- Mutual k-NN graph neighborhood $G'(V, E)$, where there is an edge $E(v_i, v_j)$ if $v_j \in k\text{-NN}(v_i)$ and $v_i \in k\text{-NN}(v_j)$.
- ϵ-Neighborhood graph $G'(V, E)$, where there is an edge between v_i and v_j, if $D(i, j) \leq \epsilon$.
- Dominant Neighborhood graph $G'(V, E)$, where there is an edge $E(v_i, v_j)$ if $v_j \in DN(v_i)$.

Of these, the ϵ-neighborhood graph is susceptible to scaling. Different clusters can have different radii. So, selecting a single ϵ for all nodes in the graph might not properly capture the neighborhood structure of the nodes. The k-NN graph neighborhoods produce a fixed-size neighborhood. However, as pointed to in [105], the k-NN graph has a tendency to include noisy edges in the neighborhood of a node. Moreover, using a fixed-size neighborhood might not adequately capture the locality in the manifold.

The need for a variable-size neighborhood for manifold structure learning was pointed to in [105] and [113]. Zhang et al. [113] propose an adaptive neighborhood

selection for manifold learning in the feature space, while Yang et al. [105] make use of dominant set computation method [72] for selecting the dominant subset of the k-nearest neighbors in the data space. The idea behind selecting a dominant neighborhood as opposed to just the k-nearest neighbors is that, the dominant neighborhood usually forms tight clusters and is therefore composed of nodes that are highly similar to each other. Therefore, dominant neighbors are less prone to noisy edges than k-nearest neighborhood.

While the dominant neighborhood graph can select variable sized neighborhoods, it is still dependent on the selection of the sparsification parameter, k[1]. Both k-NN and DN, can be adversely affected if the value of k is incorrectly chosen. In the next section, we explain in detail the problems related to the parameter selection. We will also explain how our consensus neighborhood selection strategy will help mitigate those problems.

5.3 Consensus k-NNs

Both k-NN and DN can select good local neighborhoods as long as the graph sparsification parameter, k, is properly selected. Looking back at symmetric k-NN neighborhood selection, we can see that an edge between v_i and v_j is selected if $v_j \in k\text{-NN}(v_i)$ or $v_i \in k\text{-NN}(v_j)$. This process is repeated for all nodes $i \in \{1, 2, ...N\}$ in order to obtain local neighborhoods of every node in the graph. Some of these edges might be noisy edges, while some of them are indeed accurate edges. Accurate edges are those edges that connect a particular node to other nodes, which are part of the true neighborhood, while noisy edges are those that connect a node to other nodes that are not part of the true neighborhood. As the neighborhood size, k, increases, so do the chances of adding in noisy edges. To make the neighborhood more stable even for large values of k, we propose to make use of consensus information from the multiple k-NN procedures that are applied to the graph. Consensus clustering has been previously used for other problems [51, 66] and has shown impressive results.

[1]We use upper case K to denote the kernel parameter and lower case k to denote the sparsification parameter

We define a consensus matrix, C, to keep track of the number of times a pair of nodes (v_p, v_q) appear together among all rounds of k-NN. A simple pseudo code to populate our consensus matrix is given below.

```
C = 0;
for i = 1 : N do
    S_i = k-NN(v_i);
    for p = 1 : N do
        for q = p + 1 : N do
            if p ∈ S_i and q ∈ S_i then
                C(p, q) = C(p, q) + 1;
                C(q, p) = C(q, p) + 1;
            end
        end
    end
end
```

Algorithm 1: Algorithm to collect the consensus information from multiple rounds of k-NNs.

The first advantage that we obtain from having such a consensus matrix is that it allows us to capture stronger relations between pairs of nodes. In the symmetric k-NN graph, a pair of nodes is either part of each other's neighborhood, or not. Whereas, if we use the consensus of k-NNs, we can be far more certain about the similarity, or dissimilarity, between pairs of vertices. The relation between a pair of nodes (v_p, v_q), which are a part of k-NN(v_i), is ignored in the case of symmetric k-NN (just the edges between v_i and v_p, and between v_i and v_q, were added to the graph). However, the fact that the two nodes v_p and v_q are a part of the same neighborhood, albeit some other node v_i, shows that v_p and v_q are similar to each other as well. If v_p and v_q keep appearing among the k-NNs of multiple nodes, then the chances of the two nodes being similar to each other further increases. This points to the second advantage of using consensus information.

Probabilistic Neighborhood Information: A row-normalized consensus matrix can be viewed as a probability matrix, where each value specifies the probability of that pair of nodes being similar to each other. We are not privy to such soft measures if we use the symmetric k-NN for neighborhood generation. In a symmetric k-nearest neighborhood, noisy edges have the same probability of being a part of a particular locality as accurate edges. Such noisy edges might have been included in the neighborhood by chance. The probability that such edges will be a part of multiple neighborhoods, however, is low. With the use of the consensus matrix, we can easily identify such noisy edges as those edges that have

a low probability value, and can hence be ignored. Figure 5.2b shows a consensus matrix from one of our experiments.

With the use of such probabilistic information, we get more control over neighborhood tuning. A simple way to prune out noisy edges is to select only those pairs of edges that have a probability greater than some fixed threshold. More formally, the new probabilistic neighborhood graph $G'(V, E)$, has an edge $E(v_i, v_j)$, if $C(i, j) \geq \tau$. The threshold, τ, is a global threshold that does not need to be set independently for separate nodes. Unlike symmetric k-nearest neighborhoods, even for a fixed value of τ we can get variable-sized neighborhoods that can adaptively represent the local neighborhood structure.

Pruning out noisy edges leaves us with coherent clusters, where elements of the clusters are all highly similar to each other. Yang et al. [105] also tried to obtain such a neighborhood, but they used the dominant set method for identifying such clusters. Our consensus neighborhood identification method is better in many ways compared to the dominant set extraction, especially when used for manifold learning. In the following subsection, we explain the issues that the dominant sets cannot overcome, which are especially critical for manifold learning.

5.3.1 Advantages of Consensus Neighborhood over Dominant Sets

As mentioned in the previous sections, the goal of any graph sparsification method is to produce a neighborhood that best preserves the locally linear neighborhood property of manifolds. It is assumed that coherent clusters form good local neighborhoods and, hence, clustering algorithms are used to identify local neighborhoods. The dominant set extraction method, proposed by Pavan and Pellilo [72] has shown impressive results for identifying good clusters. The authors of [72] consider clusters as dominant sets and propose an algorithm to calculate the maximal cliques in a graph. Given an affinity matrix A, and a probabilistic indicator vector $\mathbf{x}$, maximal cliques can be extracted by maximizing the following quadratic objective.

$$\begin{aligned} &\textit{maximize} && f(\mathbf{x}) = \mathbf{x}^T A \mathbf{x} \\ &\textit{subject to} && \mathbf{x} \in \Delta \end{aligned} \tag{5.4}$$

where,

$$\Delta = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x} \geq 0 \; and \; \mathbf{1}^T\mathbf{x} = 1\}. \tag{5.5}$$

The above quadratic function can be maximized using the so-called replicator dynamics, an iterative procedure, which is guaranteed to converge at optimal locations:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t)\frac{(A\mathbf{x}(t))_i}{\mathbf{x}(t)^T A\mathbf{x}(t)}. \tag{5.6}$$

Local optima: The first problem that we face with the above process is that, it can easily get stuck at local optima. The local optima, though a dominant set, might not contain the node whose neighborhood we are searching. Finding all local optima, in order to find the one with the node under consideration, is a computationally exhaustive task. To prevent the dynamics from converging at wrong optima, Yang et al. [105] restrict the search space to a local neighborhood. While finding the dominant neighborhood of v_i, they restrict the search to be among the k-nearest neighborhood of v_i, by setting $\mathbf{x}_j(1) = 1/k$, if $j \in k\text{-NN}(i)$. A correct dominant neighborhood can be obtained only if prior information is available for selecting the proper value of k.

False Neighborhoods: Secondly, even when the dynamics converges at global maxima, there is no guarantee that the node whose neighborhood we are actually searching is even part of the maximal clique! Suppose we are searching for maximal cliques in a given k-NN(i), and the maximal clique is composed of a set of nodes, which is a subset of k-NN(i), but does not include the node v_i. Then, the replicator dynamics will converge with the value for $\mathbf{x}_i(t_{end}) = 0$. If we now construct a sparse neighborhood graph $G'(V, E)$, we would end up adding edges between v_i and other nodes v_j, for which $\mathbf{x}_j(t_{end}) > 0$. Clearly, these edges do not accurately represent the true neighborhood of v_i.

Liu et al. [58] devised a means to ensure that a particular node is always a part of the final solution. They did so by forcing the node under consideration to fall within a restricted neighborhood, $\mathbf{x}_i \in [\epsilon, 1]$, where $\epsilon \in (0, 1)$. While such constraints can force the involvement of a particular node in the final solution, it still cannot guarantee that the outcome of the optimization procedure is the true local neighborhood on the manifold.

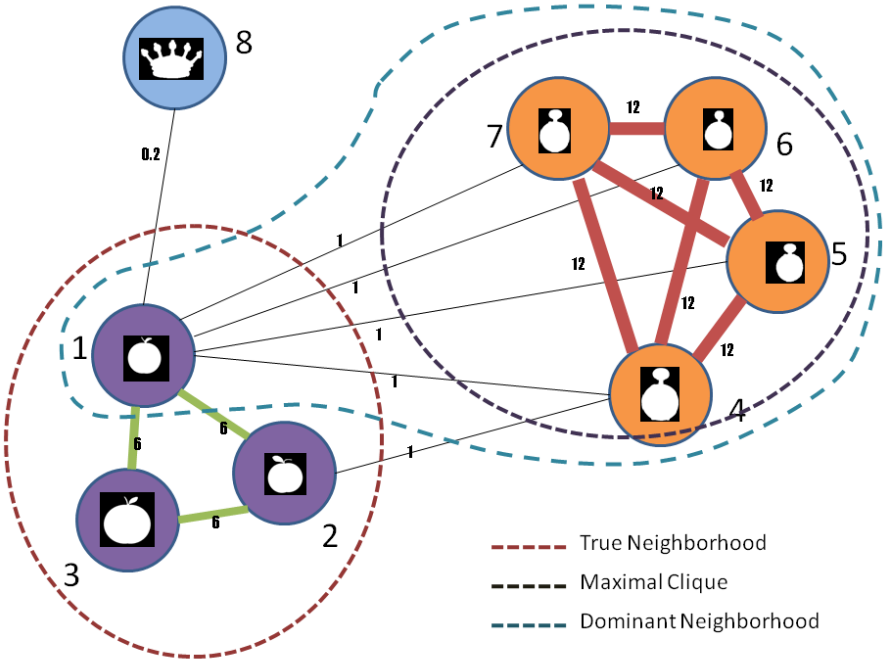
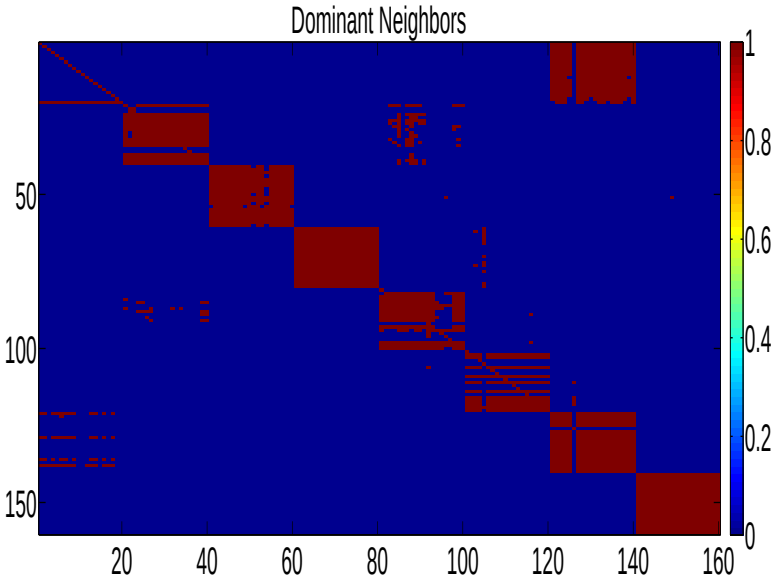


FIGURE 5.1: An example where the Dominant Neighbors of a particular node (node 1), does not include its true neighbors. Since there is less variability among pocket-watches, the affinity among them is very high. Due to the variations among the class Apples, the affinity among them is not as high as among the class of Pocket-Watches. Also, since the pocket watches do look similar to apples, they fall in the k-NN of apples. The dominant neighborhood, in this example, is a false neighborhood because, $W(S_{Pocket-Watches} \cup \{1\} > W(S_{Apples})$.

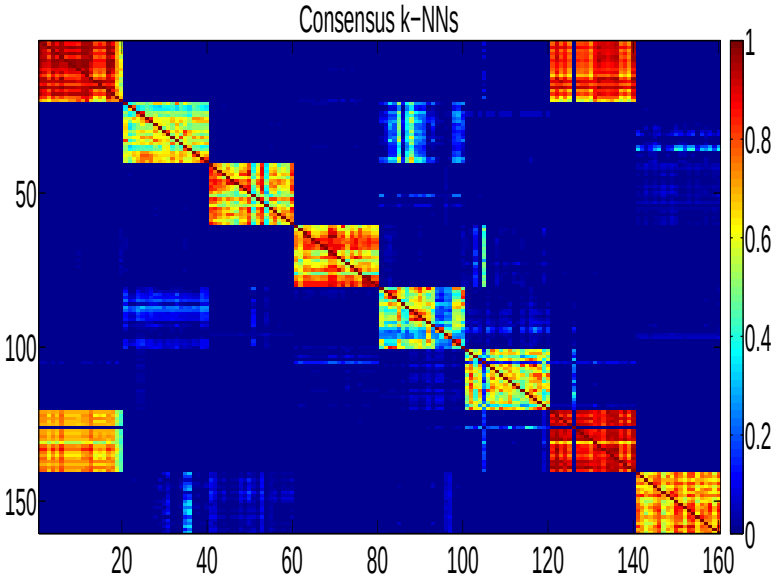
Consider the example situation shown in Figure 5.1, where the graph has two non-intersecting cliques, of which one is the true neighborhood of the node v_1, and other is "far away" from v_1. Let S_1 be the set of nodes in the true clique (nodes belonging to the class *Apples*), and S_2 be the set of nodes in the other clique (nodes belonging to the class *Pocket-Watches*). Also, let $W(S)$ denote the weight of the set S and be defined as

$$W(S) = \mathbf{x_S}^\top \mathbf{A} \mathbf{x_S} \tag{5.7}$$

where, $\mathbf{x_S}$ is a probabilistic vector where $\mathbf{x}_i > 0$ if $i \in S$, and $\mathbf{x}_j = 0$, if $j \notin S$ (see [72] for details). If $W(S_2) >> W(S_1)$, then, the dynamics (Equation 5.6) will always converge with $\mathbf{x}_j(t_{end}) > 0 \; \forall \; j \in S_2$, and, $\mathbf{x}_i(t_{end}) = 0 \; \forall \; i \in S_1$. Even when the node v_1 is forced to be present in the final solution, false neighborhoods can be obtained when $W(S_2 \cup \{1\}) > W(S_1)$. In our experiments, we noticed that the sparse graph generated by connecting a node to its dense neighbors, suffered from the above problem (see Figure 5.2).



(A)



(B)

FIGURE 5.2: Neighborhood matrix for a subset of the MPEG-7 shape database. We use $k = 50$ for both cases. (a) Figure shows the dominant neighbors, where all nodes have a binary status. Also, note that class 1 (rows 1 to 20) has chosen a neighborhood that belongs to a completely different class, class 7 (columns 121 to 140). This is because of the false neighborhood problem that was described in Section 5.3.1. (b) Figure shows a probabilistic version of the neighborhood matrix obtained from consensus of k-NNs. Note that the neighborhood of class 1 still includes others nodes from the same class, with high probability, unlike DN.

This second problem that we have just described is actually quite a serious problem, especially when the objective is to learn a local neighborhood on the manifold. The task of neighborhood selection is one in which the algorithm selects an optimal local neighborhood, which need not always be the solution that globally maximizes a function. This is a case of an ill-formed objective since the objective function does not give importance to the similarity between the node v_i and other nodes in the solution. k-NNs are not susceptible to this problem as the first goal of k-NN is to sort the rest of the nodes in decreasing order of similarity. So, we are always guaranteed to select the most similar nodes to a particular node v_i. The problem with k-NN (i.e., its tendency include noisy edges), which forced the adoption of DN, is overcome by our method of using consensus information from multiple k-NNs. Consensus information not only retains the most similar nodes, but also gives a means to prune out noisy edges. Consensus of k-NNs has the advantages that motivated the use of DN, and is not affected by the issues that affect DN.

Disconnected Graphs: The final problem that we will discuss is the case of disconnected graphs. In order to propagate similarity information between every pair of nodes, there should be at least one path connecting a node v_i to every other node v_j. This means to say that, the sparse graph should be a connected graph (need not be a fully connected graph). The graph sparsification step should not output a graph with two or more subgraphs that have no connections between them. This is critical if we wish to learn the true geodesic distance between every pair of nodes.

Dominant set extraction procedure has a tendency to output tight-knit clusters. The different optima of the optimization objective in Equation (5.4) are nodes that are subsets of V. These subsets are coherent subsets with a high degree of intra-set similarity. This causes inter-cluster edges to be pruned off, resulting in a fragmented graph with multiple connected subgraphs. If diffusion was performed on such a graph, the true geodesic distances between pairs of nodes would be learnt only among the nodes within a connected subgraph. All distances between nodes belonging to different subgraphs would end up being very large, and therefore, meaningless. In our experiments, we noticed such fragmentation of the graphs, even when k was set to moderate values ($k = 10$). This meant that, while using the dominant set for neighborhood extraction, we were never able to learn the

geodesic distance between nodes belonging to mutually disconnected subgraphs for particular values of k.

The probability of ending up with fragmented subgraphs is much less while using k-nearest neighborhoods. This is because the edges are not forced to remain only among a selected subset of nodes. We noticed fragmentation of the graph, while using k-NN, only when k was chosen to be very small ($k = \{1, 2, 3\}$). Such small neighborhoods do not provide any locality information. Hence, choosing such small k's is hardly ever the case.

Summary: To summarize, consensus of k-NNs has many advantages over dominant neighbors. Firstly, they are not prone to problems such as local optimality. While using dominant sets, one can end up with neighborhoods that arise out of locally optimal solutions. Secondly, the "neighbors" generated by the dominant sets are not guaranteed to contain the true neighbors of the node under consideration. There are no such problems while using consensus of k-NNs as k-NN guarantees that the most similar nodes to a particular node are always part of the local neighborhood. Finally, the chances of graph fragmentation is much less (in fact, hardly ever the case) in the case of consensus k-NNs when compared to the dominant neighbors. This allows similarity information to propagate between all pairs of nodes.

5.3.2 Diffusion Using Consensus Information

The consensus information just gathered can be used in two ways before performing diffusion. As mentioned in Section 5.2, there are two pre-processing steps that are performed before propagating the similarity information. The cost matrix to affinity matrix conversion stage relies on a good choice of the kernel, σ. Equation (5.3) selects σ_{ij} as the mean of the K-NN[2] distances of the two nodes v_i and v_j. Using the consensus information, the parameter can be chosen as the mean of the distances between pairs of nodes that have a probability greater than, say, τ. This would better ensure similar nodes to be grouped together much more tightly, and dissimilar nodes to be pushed much further away.

Secondly, consensus information has significant uses during the graph sparsification stage i.e., neighborhood generation stage. We have explained above, how a good

[2]We would like to remind the reader that we use upper case K to denote the kernel parameter and lower case k to denote the sparsification parameter

neighborhood graph, G', can be obtained from the consensus information. Given a neighborhood graph G' generated using consensus of k-NNs, one can obtain a probabilistic transition matrix P as

$$P(i,j) = \frac{E'(i,j)}{\sum_j E'(i,j)}, \tag{5.8}$$

where, E' is the edge set obtained from the sparse graph G'. Once P is calculated, we can now perform diffusion using any of the graph diffusion procedures (Ex: LCDP [107], or TPG [108]). In our experiments, we primarily use TPG diffusion as it takes into account higher-order similarity relations for the same space and time complexity as classical diffusion on the original graph.

5.4 Experiments

To demonstrate the stability of using consensus neighborhood, we compare the diffusion process when using k-NN, DN, and consensus of k-NNs for many different values of k. We perform our experiments on the spiral dataset and on the standard MPEG-7 shape retrieval database.

5.4.1 Spiral Data

The spiral data is obtained by generating samples from the Archimedes spiral as a function of arc length. We sample 1000 points from the spiral. Each generated point is perturbed by random noise. The spiral, which lies in the 2-D space, has an intrinsic 1-D structure. Here, we compare the performance of consensus of k-NNs with the simple k-NN, by purposely setting the value of k to be "large".

Figure 5.3 shows the output of the two approaches. The first row is obtained by using the naive k-NN, and the second row from consensus of k-NNs. The graphs on the left show the plots of the second most-important eigen vector of the final transition matrix (the first most-important eigen vector just contains the translation co-ordinates) against the arc length. The figures on the right show a color-coded spiral. Similar colors mean that the points are mapped close to each other after learning the manifold structure [81, 93]. Clearly, consensus of k-NNs has learnt a better neighborhood than the simple k-NN. From Figure 5.3a, we can

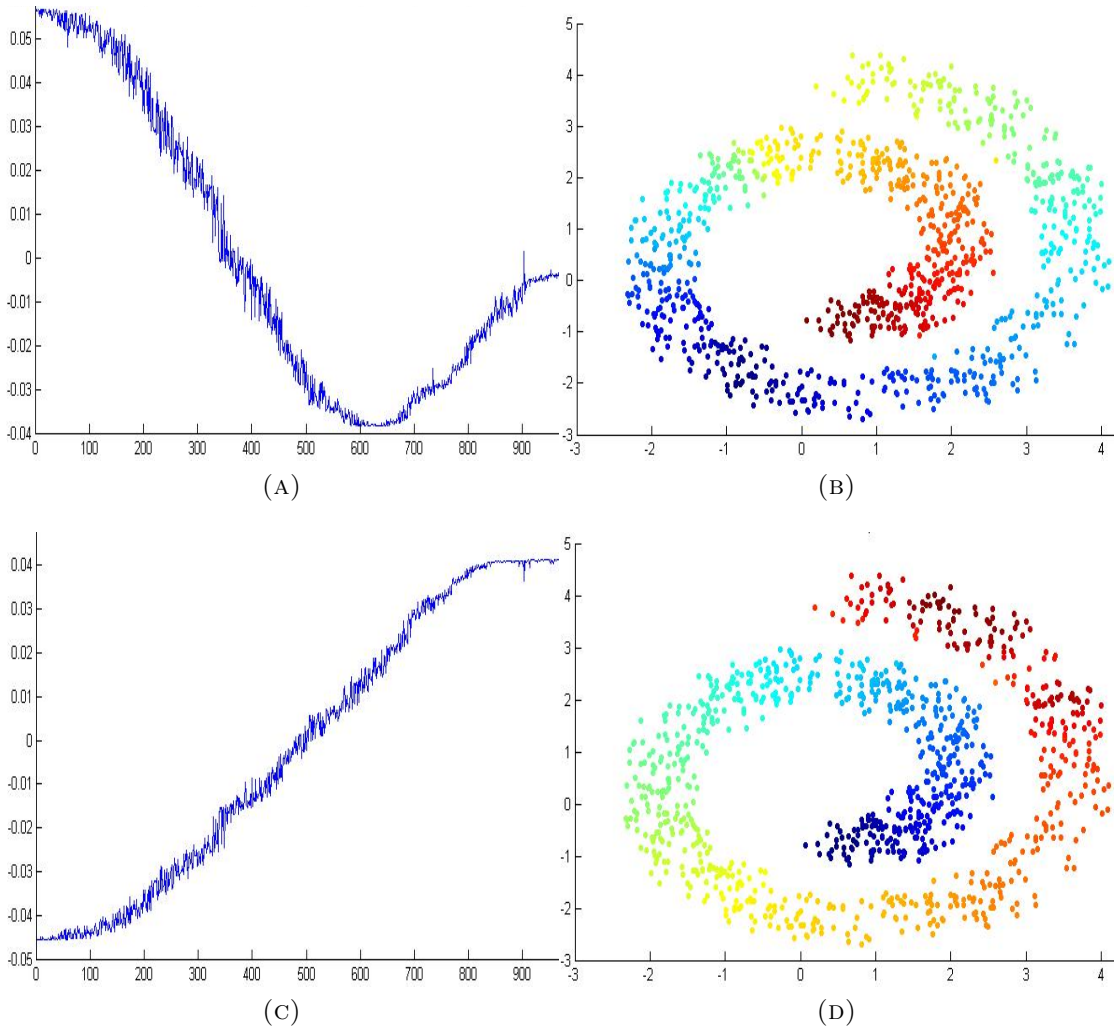


FIGURE 5.3: Top row shows the diffusion results from using simple k-NN sparse graph, and the bottom row, using the sparse graph generated using consensus of k-NNs. (a) and (c) show a plot of the mapped 1-D coordinates (second most-important eigen vector) of the points versus the arc length. (b) and (d) show a color-coded spiral. Points that are close to each other after mapping to the lower dimension are similarly colored.

see there there is no one-to-one mapping of the coordinates to the arc length. This indicates that the geodesic distances were incorrectly learnt due to the presence of noisy edges in the simple k-nearest neighborhood. On the other hand, Figure 5.3c shows a clear one-to-one mapping of the coordinates to the arc length, which shows that the neighborhood generated by consensus of k-NNs was more robust to noise. For this experiment, k is set to 15, and we use TPG to perform diffusion. We have tried for many different k's and found that consensus of k-NNs performs better than the simple k-NN.

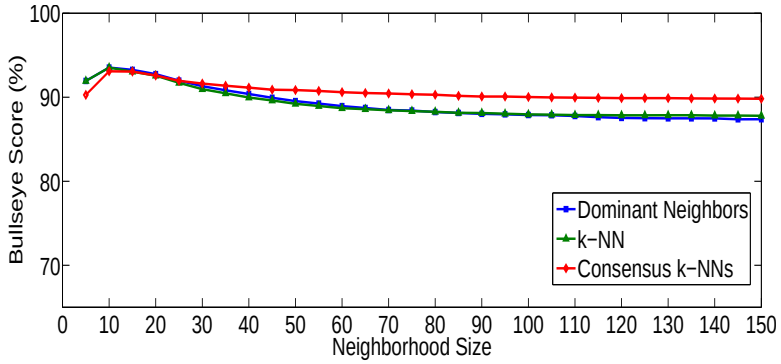
5.4.2 MPEG-7 Shape Retrieval Database

In this sub-section, we show the results from our experiments on a more challenging and real-world application i.e., image retrieval by learning the manifold of shapes. The well-known, and widely used, MPEG-7 CE-Shape-1 Part B database consists of silhouettes of 1400 images with a wide variety among them. The database is split into 70 classes, with each class containing 20 example images (see Figure 3.5). The shape-retrieval performance is measured by the so-called Bullseye score. The Bullseye score is basically the percentage of objects belonging to the same class as the query object among its top-40 best matching objects.

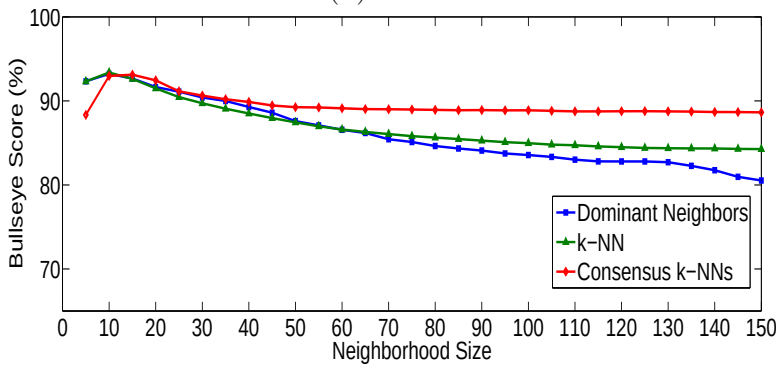
We learn the true shape manifold by starting off with the 1400×1400 pairwise dissimilarity matrix. We make use of the IDSC shape dissimilarity matrix [55], which is extensively used in the literature, for learning the shape manifold structure. The matrix has a Bullseye score of 85.40% before diffusion. While the previous trend in the literature is to carefully hand-tune a good neighborhood size, and to properly select a good number of diffusion iterations, for getting better Bullseye scores, in our experiments, we purposely select neighborhood sizes that are bound to include noisy edges and show that consensus k-NNs produce a much more stable neighborhood than k-NN or DN.

Objects belonging to the same class, usually, belong to the same neighborhood. Since the MPEG-7 shape database has 20 objects per class, it comes as no surprise that the previous state-of-the-art Bullseye scores were reported for neighborhood sizes that were purposely selected to be smaller than or equal to 20 ($k = 20$ in [107] and $k = 10$ in [108]). Such parameter selections are an example of supervised neighborhood selection. However, in a completely unsupervised setting, selecting k to be less than or equal to the number of items in a particular class is highly unlikely. Therefore, we would like our neighborhoods to be stable enough even when k is chosen to be greater than the number of examples in a particular class, and thus including more objects into the local neighborhood than there would be in the "true neighborhood".

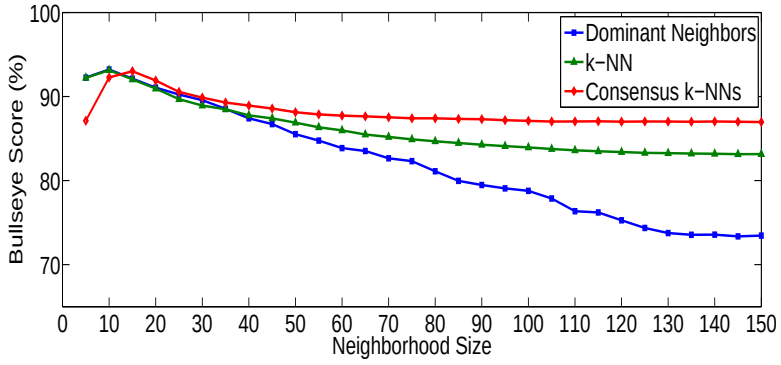
We have experimented with multiple values of k, and in Figure 5.4, we show the plots of Bullseye scores v. neighborhood sizes, when using k-NN, DN and consensus k-NN. Remember from Section 5.2 that the generation of a good sparse affinity matrix requires the choice of two neighborhood parameters: one while generating the affinity matrix and one while sparsifying the matrix. The four plots correspond



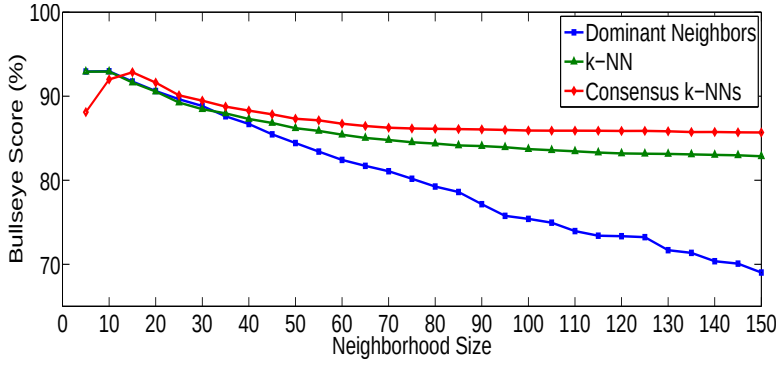
(A) K=25



(B) K=50



(C) K=75



(D) K=100

FIGURE 5.4: The figure shows plots of Bullseye score v. neighborhood size, for four different affinity matrices, while using TPG diffusion process. The affinity matrices were generated using different kernels, which are calculated using different neighborhood sizes of (a) 25, (b) 50, (c) 75, and (d) 100. For each affinity matrix, we show plots of the performance of DN, k-NN and Consensus of k-NNs. We can see that DN is better than k-NN for small values of k, but deteriorates quickly for larger k's. Consensus of k-NNs performs better than DN and k-NN for almost all neighborhood sizes. Also, the performance deterioration rate is significantly slower than both k-NN and DN for larger neighborhood sizes, thus pointing towards stable localities.

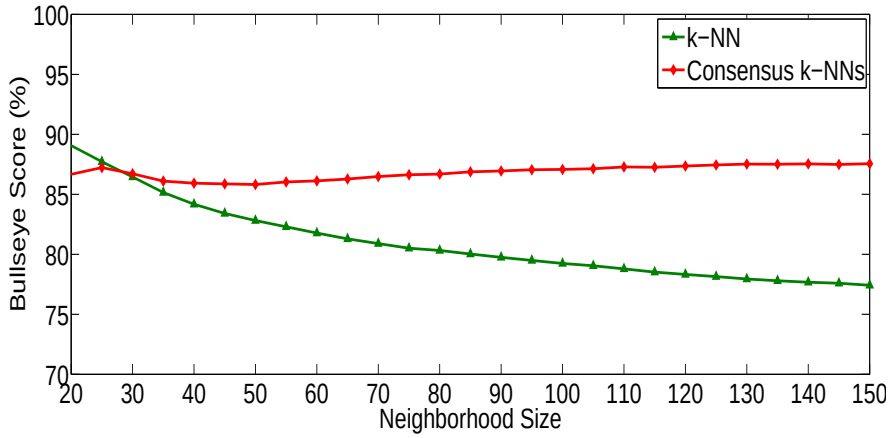


FIGURE 5.5: Bullseye score v. neighborhood size plot while using LCDP. The affinity matrix was generated using $K = 75$. We see a similar performance as in Figure 5.4. Consensus of k-NNs produces a more stable neighborhood than k-NN for larger values of k.

to four different choices of $K(= 25, 50, 75$ and $100)$ that were used to compute the kernel σ_{ij} (Equation 5.3), while generating the affinity matrix. For each of these affinity matrices, we experiment over different neighborhood sizes while sparsifying the graph. From the plots, we can see that the neighborhood generated by using consensus of k-NNs is quite stable to the neighborhood size parameter. Once the neighborhood size goes above the true neighborhood size (i.e., $k > 20$), the performance obtained from both k-NN and DN starts deteriorating at a quicker rate than consensus k-NNs. We can also see that, up to a neighborhood size of 35-40, DN outperforms k-NN, but after that, k-NN performs better. This is because, DN starts converging onto false neighborhoods because of the problems that were explained in Section 5.3.1. Moreover, even while selecting small neighborhood sizes of up to 35-40, we can see that consensus of k-NNs has learnt a better neighborhood than DN, thus producing better Bullseye scores after diffusion. For extremely small values of k, there is a relative drop in performance of our method because there is hardly any consensus information that can be extracted while using such small neighborhood sizes.

We also compared the effect of the using k-NN versus consensus of k-NNs using the Locally Constrained Diffusion Process (LCDP). We found that the neighborhood selection had a similar effect as while using TPG. Figure 5.5 shows the plot for one such trial. Thus, we see that the selection of neighborhood is independent of the diffusion process and behaves similarly across different diffusion processes.

As a final comment, we would like to point out that our method can be applied to other forms of graph sparsification techniques as well. Ex: We can generate a consensus of ϵ-neighborhood graphs while using ϵ-neighborhood sparsification. In our experiments we found that consensus of ϵ-neighborhoods outperformed the simple ϵ-neighborhood by a Bullseye score of 1-3%, when we experimented with multiple threshold values (ϵ). We do not discuss ϵ-neighborhood graphs in detail due to the lack of space and also because it is well-known that k-NN graphs are more stable than ϵ-neighborhood graphs (Section 5.2.2).

5.5 Conclusion

In this chapter, we have identified some of the problems with the currently-used neighborhood selection methods. We also propose a new way for neighborhood selection, which makes use of the consensus information from different k-NNs. We have shown that making use of such information increases the robustness of the neighbors, and thus, helps the similarity information to propagate better on the data manifolds.

In the next chapter, we move away from retrieval, and show how shape properties can be effectively used for recognizing objects in the wild.

Chapter 6

HOGs and Contours: A Combined Structured Prediction Approach For Object Detection

6.1 Introduction

Object detection is a challenging problem in computer vision and is an actively researched area. Early approaches to object detection involved appearance-based models that used bag-of-words as features [112]. However, the lack of any spatial information in such models have paved way for stronger shape-based approaches to object detection. The introduction of histograms of oriented gradients (HOGs) [25] helped in substantial advancement of the field. More recently, contour-based approaches ([33, 61, 80, 88, 91]) have come into existence because of their strong localization abilities and their direct representation of the object shapes.

Contours also have the added advantage that they are invariant to variations in the lighting conditions and color and texture of the objects. This is an important requirement for recognizing objects in the wild, especially when the images have varying illuminations, and when the objects that we are trying to recognize have a well-defined shape, but can vary in their color and texture. Most contour-based object detection approaches start off with the image edge map, group pixels using an edge linking algorithm [50], construct shape descriptors on these linked edges, and match these descriptors to the standard template descriptor.

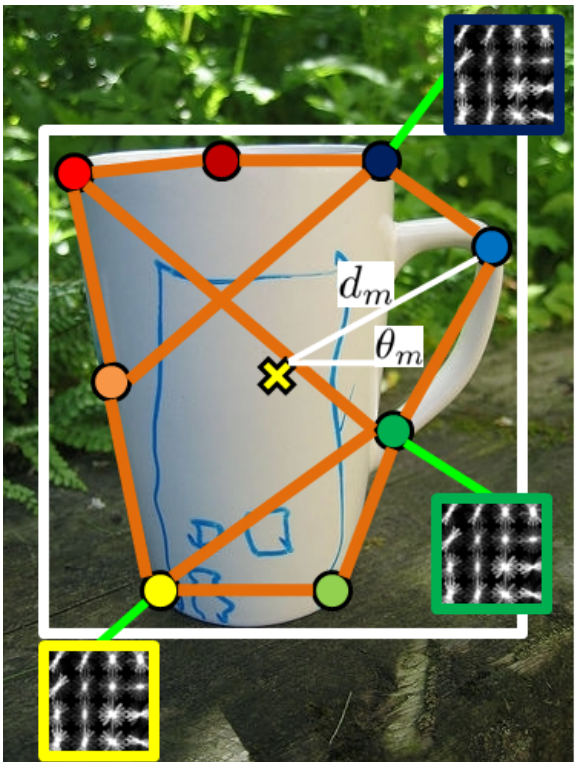


FIGURE 6.1: Our parts-based model. Unary terms are characterised by their HOG features, their distance and angle w.r.t a centroid. Different node colors indicate different parts. The pair-wise interaction graph (sparsified for better visualization) enforces structure while predicting labels.

Though significant effort has gone into the development of better edge detection algorithms, the state-of-the art methods (ex: [63, 102]) fail to produce clean, well-connected, and closed contours. On the contrary, there are breaks in the edges, there are lots of noisy edge fragments, and some edges are not even identified. These problems make it difficult to build a discriminative shape descriptor purely from edge fragments. Moreover, the contour-based approaches have had a fairly blind reliance on the ability of edge linking algorithms. The problems that might arise while linking edges have not been identified and well-understood. We identify one such problem and explain how it affect order-preserving shape descriptors.

Variations among the objects within a class, such as non-rigid deformations of objects and articulations, have posed another problem for object recognition. The development of deformable part models [31], and their success in object recognition have further fueled this field. This has renewed the interest in developing structured models that account for object deformations. However, structured prediction models that build upon contour-based approaches are not well-explored in the community. We believe that the strong localization ability of contours in

conjunction with structured prediction algorithms can help improve the field of deformable object detection by providing better localization and detection capabilities.

In this chapter, we propose a parts labeling-based object detection framework, which utilizes the advantages of the discriminative power of HOGs and the localization abilities of contours. We also emphasize a structured approach for part labeling, by allowing non-local parts to influence the labeling of a part. We design our unary terms using neighborhood gradient information and make it more discriminative by encoding its relative spatial location. Our non-local pairwise terms add structural constraints and penalize deviations from the norm, appropriately. We learn all the model parameters using training data. Moreover, the parts in our model are not manually selected to be class-specific semantic parts, thus making our approach applicable to generic objects. In addition, we make use of variational techniques for inference and explain why message-passing between nodes is an intuitively appealing way to label various parts. We report results from the ETH-Z shape database and show that our method helps achieve better results on multiple object classes. Figure 6.1 gives an illustration of our model.

6.2 Object Detection as a Labeling of Parts

6.2.1 Basic Goal

We want to detect objects in the wild by not only scoring hypothesis locations using global scores, but also by labeling the various object parts. We want to abate the spatial search for local parts by restricting the parts to be located only on the edge pixels of the contour map. This is how we take advantage of the strong localization capabilities that the edge maps provide to us.

In the previous section, we identified a problem that arises when descriptors are reliant on an ordered representation of contour points. Though some recent works like [100] do not rely on edge linking, their inference of parts takes place conditionally independent of other parts. Object shapes, being highly structured entities, call for the usage of structured prediction techniques that make inference based on random fields rather than independent inference of other parts. In our model, we follow a part-labeling style for object detection. The labeling of our parts is

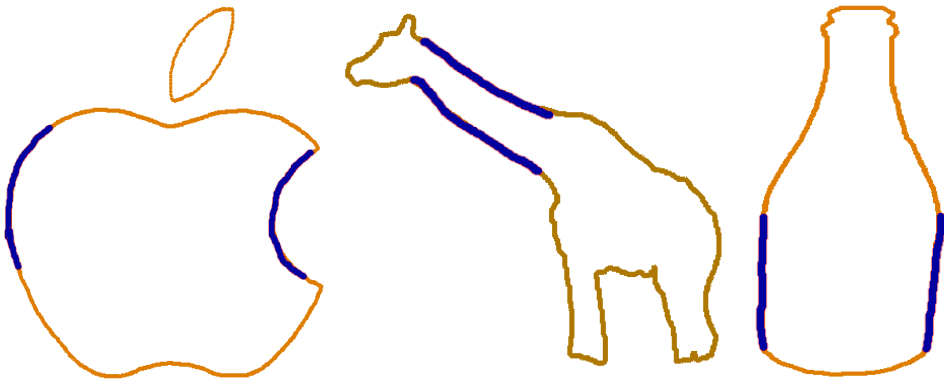


FIGURE 6.2: Three example objects from the ETH-Z database that have similar-looking parts (highlighted in blue).

influenced not only by local shape information, but also by non-local interaction terms.

As a motivation for the need of non-local terms, consider the examples shown in Figure 6.2. The figure shows three example objects that have similar-looking parts (highlighted in blue) at different locations on the contour. Looking at those contour fragments alone, it is extremely difficult for us to label whether they belong to the anterior portion of the object, or to the posterior portion. The fact that we face this difficulty in labeling the parts, based on just the local information, shows how important the surrounding contour fragments are in helping us make such decisions. If we were to adopt the partial shape matching descriptors, the highlighted fragments in Figure 6.2 would give high scores while matching both the posterior and the anterior portion of the object template, making it difficult for the contour-grouping stage to output a coherent detection. This observation motivates us to include both local and non-local terms in our objective function, and to define local descriptors relative to a reference point, and not in absolute terms.

One of our other goals is to keep our model as generic as possible. Therefore, our part labels are not class-specific semantic labels, but contains more generic and abstract labels that can be used for any shape. In the next subsection, we explain our model in more detail.

6.2.2 The Approach

With the goals laid out in the previous subsection, it is left for us to define how we model the shape, how we learn the part distributions, and how we infer various parts in a structured fashion.

Given the object template, we discretize the object's boundary into N equally spaced points. Each sampled point, which is one of our N generic parts, is appropriately labeled 1 to N. The goal of the learning step is to learn the independent distribution of each of these points (unary terms) and the distribution of the interactions between pairs of points (pairwise term). Given a test image, I, the goal of the inference step is to find the best configuration of labels, Y^*, from the set of all configurations of labels, $\mathcal{Y}$, which minimizes the energy function, $E(I, Y; \ell)$, at each image location, ℓ, i.e.,

$$Y^* = \underset{Y \in \mathcal{Y}}{\operatorname{argmin}}\, E(I, Y; \ell). \tag{6.1}$$

In the above equation, Y is a single configuration of labels where each element of Y, i.e., y_i, which is a single label from the label set $\mathcal{L} = \{1, 2, ..., N\}$, is the label assigned to a particular point $p_i, \forall i \in \{1, ..., |\Upsilon_\ell|\}$, where Υ_ℓ is the set of all contour edge points that lie in the neighborhood of the relative location ℓ. Therefore, $|Y| = |\Upsilon_\ell|$, and $|\mathcal{Y}| = N^{|\Upsilon_\ell|}$. The operator, $|.|$, is used interchangeably to denote both the cardinality of the set and the length of the vector.

The energy function, $E(I, Y; \ell)$, is defined as,

$$E(I, Y; \ell) = \sum_{p_i \in \mathcal{V}} \Phi(p_i{=}y_i; \ell) + \sum_{(p_i, p_j) \in \mathcal{E}} \Psi(p_i{=}y_i, p_j{=}y_j; \ell), \tag{6.2}$$

where, $\mathcal{V}$ is used to denote the set of nodes, and, $\mathcal{E}$, to denote the set of edges in the random field, $G = (\mathcal{V}, \mathcal{E})$. The energy term, $E(.)$, is made up of local unary terms, $\Phi(.)$, and non-local pairwise interaction terms, $\Psi(.)$. The unary and pairwise energy terms are defined as,

$$\Phi(p_i = y_i; \ell) = -\log P(p_i = y_i; \ell) \quad \text{and} \tag{6.3}$$

$$\Psi(p_i{=}y_i, p_j{=}y_j; \ell) = -\log P(p_i{=}y_i, p_j{=}y_j; \ell), \tag{6.4}$$

respectively, $\forall i, j \in \{1, ..., |\Upsilon_\ell|\}$. We now proceed to describe how we learn the distributions, in the next section.

6.3 Learning the Distributions

Given positive training images $\mathcal{I}^+ = \{I_1, I_2, ..., I_{|\mathcal{I}^+|}\}$, we have access to the ground truth bounding box of the object, and the object silhouette. However, to learn our part distributions, we do not have access to part labels. In such situations, it is a usual practice to consider the part labels as latent variables and use optimization tools such as latent structural SVMs as in [31, 53], or to manually label the parts and train in a supervised setting. To avoid the laborious process of manually labeling the parts, or to optimize over the space of latent variables, we match the shapes of the training silhouettes and get point correspondences of the parts. We make use of the dynamic programming paradigm as used in Inner Distance Shape Context (IDSC) [55] to match shapes, and to get correspondences between points.

We randomly select a particular shape from the training set as the reference shape and discretize the boundary into N equally spaced boundary points. We then match the reference shape to each of the other shapes in the training set to obtain the point correspondences. Upon doing so, we are able to transfer the labels of the discretized points in the reference shape, to all other training instances. With access to individual part labels on all training shapes, we are now free to train any classifier in a supervised manner to classify various parts.

6.3.1 Unary Part Distributions

In order to learn the part distributions over the various labels, we extract local part-specific features and train a multi-class classifier. For its impressive performance in the recent past, and its parallelization abilities, we choose random forests [3, 19] to train our part features. Moreover, we just need to train a single random forest to train the multi-class (N-parts) training dataset that we now have, unlike a binary SVM, where we might have to end up training N binary decision boundaries.

To incorporate local information into the unary term, we extract small image patches around each part point, and compute the HOG descriptor of that part.

The patches are sized such that they overlap with two other neighboring parts on either side of the point under consideration. Overlapping patches make the descriptors more robust to noise. Apart from the features around the object parts, we also train the random forest using HOG features from randomly extracted negative patches, which lie outside the ground truth bounding box.

Secondly, apart from the local gradient information surrounding each part, the relative location of each part is also extremely important while identifying shapes and labeling parts of a shape. Ex: We say that the Apple has its stem north of the centroid, the Giraffe's neck articulates between north-west to north of its centroid, or the fore legs are south-west of the centroid, and the hind legs are south-east of the centroid (visualization in Figure 6.2). Such directional information can be captured using angular features. In addition, the distance from the centroid is also quite important while classifying an object part.

To learn the distribution of the distances of a particular part, β_m, from a relative centroid, we collect the distance of the ray from the centroid of the object to the part, β_m, from all training instances, and model the distribution of these distances as a normal distribution, $\mathcal{N}_{\beta_m}(\mu_m, \sigma_m), \forall m \in \{1, ...N\}$. The angular distributions are modeled using a von Mises distribution, $\mathcal{M}_{\beta_m}(\nu_m, \kappa_m), \forall m \in \{1, ...N\}$. Such relative information was used by the authors of [100], and we make use of the same features to encode the spatial location of each part. The spatial location, in conjunction with the local patch-based gradient features, gives more discriminative capability to the local parts.

With the above description of the entities involved in the unary term of the energy function, we can write the likelihood of a part p_i taking a label $m \in \mathcal{L}$ as,

$$P(p_i{=}m;\ell){=}P_{RF}(m|\chi(p_i))\mathcal{N}_{\beta_m}(d_i^\ell;\mu_m,\sigma_m) \mathcal{M}_{\beta_m}(\theta_i^\ell;\nu_m,\kappa_m). \tag{6.5}$$

The unary energy term is the negative loglikelihood of the above term (Equation 6.3). The various symbols in the above equation are explained in detail in Section 6.4.

6.3.2 Pairwise Distributions

Local parts, on their own, do not comprise an object. The interaction between parts, and the spatial placement of various parts is what gives an object its shape and visual familiarity. Therefore, it is extremely important to impose structural constraints while composing an object from independent parts.

Star models are a particularly common form of geometric constraints-model that basically states that the parts are independent of each other given the root configuration. There also exist tree models that are relatively more generic than the star models, but still allow for efficient inference techniques. Such models have shown success while modeling deformations in face, or in human pose estimation. However, accounting for part deformations in such applications is relatively easy as these objects contain well-defined semantic parts whose functionalities are well understood. Ex: While modeling a face, the parts are defined to be eyes, ears, mouth, nose and hair; while in human pose estimation, the torso is considered as the root part in a tree with the hands, legs, and head as other semantic parts of the body.

Referring back to Section 6.2.1, one of our goals was to keep our model as generic as possible so that the same model could be applied to any object shape. This forced us to define our parts using abstract labels, and not by using class-specific semantic labels. Such an abstract labeling of parts makes it more challenging to constrain interactions between parts using one of the previously defined parts-based models. As a result, we maintain our parts-based model as a fully-connected graphical model, which is the most generic interaction graph that is possible.

The goal of a constraint model is to penalize deviations between parts that are unlikely to happen and favour part-configurations that are more likely to be present. This penalization, however, should vary for different pairs of parts. Ex: The distribution of the distances between a point on the Giraffe's neck and a point on its legs is likely to have more variance than the distribution of the distances between a pair of points, both lying on the Giraffe's neck. Therefore, we should penalize the deviation in the distance between the pair of points on the neck more severely than a similar deviation in the distance between points on the neck and leg.

We calculate the distance between each pair of parts, β_m and β_n, from all the training shapes, and model the distribution of these distances using a normal

distribution, $\mathcal{N}_{(\beta_m,\beta_n)}(\mu_{mn},\sigma_{mn}), \forall m,n \in \{1,...,N\}$. By modeling each pair of distributions separately, we are learning to appropriately penalize deviations in distances between different pairs of points. In other words, the negative loglikelihood of a particular deviation from the mean distance is smaller for a pair of parts that have a large variance, as compared to a similar deviation from the mean, for a different pair of parts, whose variance is low.

The likelihood of the pairwise distribution takes the following form,

$$P(p_i = m, p_j = n) = \mathcal{N}_{(\beta_m,\beta_n)}(d_{ij};\mu_{mn},\sigma_{mn}). \tag{6.6}$$

The pairwise energy term is the negative loglikelihood of the above term (Equation 6.4). Please refer to the following section for a description of the various symbols in the above equation.

6.4 Inference

Given the learnt distributions for the various parts and interactions between parts, the goal of the inference step is to detect the best configuration of parts in a detection window. For each location, ℓ, in the image, we identify all those contour edge points that lie within a particular radius of the centroid and ignore the rest of the edge points. These points form the neighborhood set, Υ'_ℓ. The radius is calculated by finding the maximum distance that any object part was in the training data, from the object centroid, appropriately scaled by the scaling factor.

For each of the remaining points, we extract the HOG features from the local patch around the point, and pass it down the decision trees in the random forest, which was trained in the training step. The part distribution of each contour edge point, p_i, for all labels $\gamma \in \mathcal{L} \cup \{Bg\}$, where Bg is the background label, is calculated as,

$$\widetilde{P}_{RF}(\gamma|\chi(p_i)) = \frac{1}{T}\sum_t P_t(\gamma|\chi(p_i)), \tag{6.7}$$

where, $\chi(p_i)$ is the HOG feature around point p_i, P_t is the probability obtained from the t-th tree, $\widetilde{P}_{RF}$ is the ensemble random forest probability, and T is the total number of trees in the forest. Upon collecting the distribution for each point, we ignore all points whose probability for taking on the background label is

greater than some τ_{bg}, from further consideration. The remaining points form the restricted set, Υ_ℓ''. For the rest of the points, we then compute the final marginal distribution as

$$P(p_i{=}m;\ell){=}P_{RF}(m|\chi(p_i))\mathcal{N}_{\beta_m}(d_i^\ell;\mu_m,\sigma_m) \\ \mathcal{M}_{\beta_m}(\theta_i^\ell;\nu_m,\kappa_m), \tag{6.8}$$

for all labels $m \in \mathcal{L}$, where, d_i^ℓ is the distance, and θ_i^ℓ is the angle of point p_i from the relative location, ℓ, and P_{RF} is the renormalized probability distribution after ignoring the label, Bg, from the original random forest distribution, $\widetilde{P}_{RF}$. All points p_i, whose maximum probability for taking on one of the labels, $m \in \mathcal{L}$, is less than some τ_Φ, are also ignored from further consideration. The rest of the points constitute the set, Υ_ℓ'''.

We now build a fully-connected graph using the points in Υ_ℓ''', and calculate the part-compatibility likelihood

$$P(p_i = m, p_j = n) = \mathcal{N}_{(\beta_m,\beta_n)}(d_{ij};\mu_{mn},\sigma_{mn}), \tag{6.9}$$

for all part labels $m, n \in \mathcal{L}$, where d_{ij} is the distance between points p_i and p_j. Among these remaining points, $p_i, \forall i \in \{1, ..., |\Upsilon_\ell'''|\}$, we identify all pairs of points that are highly unlikely to form any partnership in constituting the object shape and ignore the corresponding points from further consideration. More formally, if

$$\max_{m,n\in\mathcal{L}} P(p_i = m, p_j = n) < \tau_\Psi, \tag{6.10}$$

then the points p_i, p_j are ignored. We perform this part compatibility check $\forall i, j \in \{1, ..., |\Upsilon_\ell'''|\}$. The remaining points constitute our final neighborhood set, Υ_ℓ. We call these points, valid points, and they constitute the nodes, $\mathcal{V}$, and the interactions between them i.e., the edges, $\mathcal{E}$, of our random field, $G = (\mathcal{V}, \mathcal{E})$. If $|\Upsilon_\ell|$ is less than some acceptable number of parts, τ_p, say, 50% of the number of parts in our model, then, we ignore the current window location, and proceed to infer the presence of the object from the next hypothesis location.

In Section 6.2.2, we introduced the objective function, and we have now defined all parts that go into the making of that function. This, in itself, should be sufficient to discover structures in our images. However, during our experiments, we noticed that the optimization algorithm (more on that later in the section), ended up

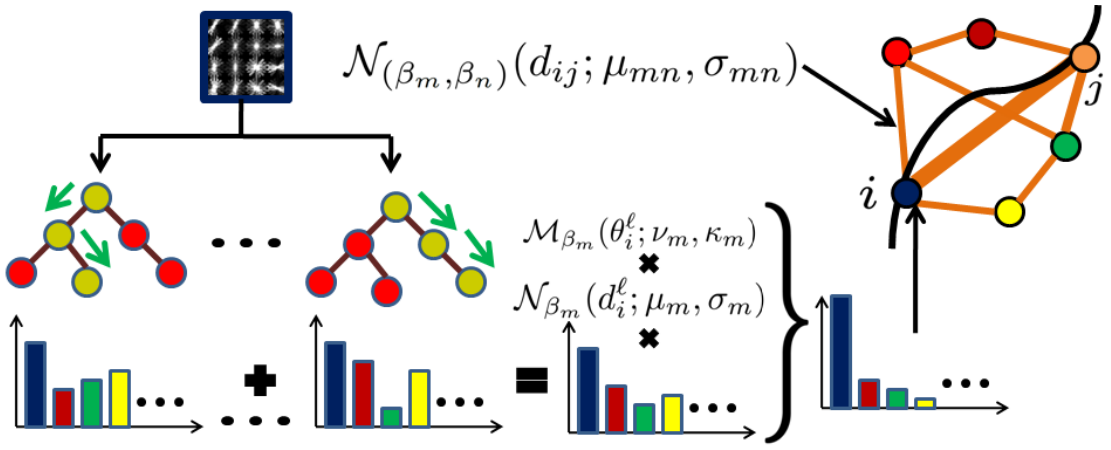


FIGURE 6.3: HOG features are passed through the random forest and combined with spatial terms to get unary distributions. The graph edges constitute pairwise terms. The edge joining nodes i and j, in the graph, have a higher weight because they both lie on the same contour fragment (shown as a black strand).

labeling noisy/background edge points that were very close to the true edge point, with the true part labels, with high probability. This made it difficult to choose one single part among two, or more, highly likely hypotheses. As a result, we added in a Potts potential [75], along with the non-local pairwise term, to enforce stronger localization, after inference.

The Potts potential, defined as,

$$\Omega(p_i = y_i, p_j = y_j) = \begin{cases} 0, & \text{if } CF(p_i) = CF(p_j) \\ \lambda, & \text{otherwise,} \end{cases} \tag{6.11}$$

is a simple penalization term, which penalizes those pairs of points that lie on different contour fragments, as opposed to lying on the same contour fragment, if the fragment can accommodate the pair of points. Here, $CF(p_i)$ basically returns the contour fragment number that the point p_i lies on. Our final objective function can now be defined as,

$$E(I, Y; \ell) = \sum_{p_i \in \mathcal{V}} \Phi(.) + \sum_{(p_i,p_j)\in\mathcal{E}} \Psi(.) + \sum_{(p_i,p_j)\in\mathcal{E}} \Omega(.). \tag{6.12}$$

Figure 6.3 gives an illustration of the various probabilities.

6.4.1 Optimization and Philosophy

The objective function is now complete, and the goal now is to minimize the overall energy in Equation 6.12 and find the best configuration of labels, Y^*, which best explains the remaining valid points. Since our part interaction model is a fully-connected graph, finding the best configuration of labels is an NP-hard problem. Therefore, we have to resort to approximation techniques.

For approximate optimization on graphs with loops, there are two approaches that can be found in the literature. The authors of [101] manually approximate their loopy graph to a tree, and then perform efficient inference on the tree. The other approach is to use message passing algorithms for converging to local optima. We follow the latter approach and use the Sequential Tree Re-Weighted Message Passing (TRW-S) [48] for optimization. Message-passing between nodes is also an intuitively appealing way to label various parts. As was motivated from our example in Figure 6.2, more often than not, it is the beliefs that the neighboring parts have about themselves, which influences the labeling of the part under consideration. The unary potentials are the marginal beliefs of each node, and the pairwise potentials are the part-consistency beliefs. By passing these beliefs as messages between various nodes, all other surrounding nodes influence the labeling of a part.

Upon convergence of the TRW-S algorithm, we are left with the final updated energies at each node. Since there might be more than N contour edge points participating in our optimization, we need a way to select the N-best contour points, all with complementary labels. We follow a non-maxima suppression-based approach for assigning the final labels. We start-off by identifying the contour edge point that has the smallest energy for any label $m \in \mathcal{L}$ and assign to it the label, say, m, for which it has the least energy. In the next iteration, we assign a label from the set $\mathcal{L}\backslash\{m\}$ to the next best contour point. We repeat this process till either all labels are exhausted, or till there are no other points left to label. If there are extra contour points that are still available to be labeled after all labels are exhausted, we consider them to be background points and ignore them from the final energy calculation. If, on the other hand, there are fewer points to be labeled than the number of available labels, it means that we have a partial match and adopt the notion of Pareto-optimality [21, 80] to score the partially-matched

object. The final part labels, corresponding to the object, $Y^*_{ob} \subseteq Y^*$, will be used in the final energy calculation.

6.5 Hypothesis Scoring

We utilize two different methods for scoring our object hypothesis. The first is a part-based score that makes use of best label configuration that was just found. Given the best object label configuration, $Y^*_{ob} \subseteq Y^*$, at location ℓ, the object score is calculated as,

$$\zeta_E = \exp(-(E(I, Y^*_{ob}; \ell) + \Lambda(|\overline{Y^*_{ob}}|)), \tag{6.13}$$

where the first term is the energy computed using the final object part labels using Equation 6.12, and the second term is the partiality penalty function, which penalizes partial matches, proportional to the length of number of parts that did not have a match. In Equation 6.13, $|\overline{Y^*_{ob}}|$ indicates the length of the complement of the object that did not have match.

The second score that we use is a global discriminative score, ζ_{SVM}, which indicates the discriminative capacity of the current object hypothesis. To calculate ζ_{SVM}, we train a boosted SVM using just the HOG features extracted from the ground-truth bounding boxes, from our positive training set, $\mathcal{I}^+$. The negative patches are randomly extracted from the images in, $\mathcal{I}^+$, such that they do not overlap with the ground-truth bounding box. Using the current decision boundary, we run a simple HOG-only-based object detector on the images in $\mathcal{I}^+$. We identify all the "hard negatives", add them to the negative instances, and retrain the SVM. ζ_{SVM} is the boosted SVM's confidence for classifying the global HOG features extracted from the bounding box, which is obtained by bounding all the points that were assigned a final part label in Y^*_{ob}.

Our final score for the current hypothesis is,

$$\zeta_{\mathcal{H}} = \zeta_E \times \zeta_{SVM}. \tag{6.14}$$

6.6 Experiments and Results

We perform our experiments on the ETH-Z shape database [32]. The database consists of five classes comprising Applelogos, Bottles, Giraffes, Mugs and Swans. The database has a total of 255 images, ranging from 32-87 images in each class. The database is a particularly challenging database as it has significant intra-class variations, illumination changes, scale changes, clutter and camouflage. Each image in the database comes with its ground-truth bounding box and the object silhouette.

For evaluating our method, we use the standard protocol described in [34], i.e., we use the first 50% of the images in each class as the training set, and test on the second half of the images in this class as the positive images, plus all other images from other classes as the negative images. We eliminate duplicate hypotheses by using non-maxima suppression. We use the PASCAL 50% overlap criterion to identify true positives and false positives.

Implementation Details: We use the Berkeley edge detector [63] to obtain the edge map. A sliding window-based approach was adopted for object detection. We use $N = 30$ parts while modeling our shapes. It is more than the 6 parts that is used by [31] and [53], but much less than the $N = 100$ sampled points that is used in [100]. Having just 6 parts, for a contour-based approach, we felt, masked many of the intra-part variations. It will be an interesting future work to see if we can learn the ideal number of parts for different objects.

For the unary HOG features, we used a small patch around each point, such that the patch included two other neighboring points on either side, and was then re-sized to a standard 48×48 pixel patch. We then extracted a 512-dimensional HOG feature vector, $\chi(.)$, comprising 32 gradient directions at each of the 4×4 grid in the local patch. To train a random forest, we use 500 randomly generated decision trees, and make the split at each node in the tree to maximize the information gain.

In order to obtain the global discriminative score, ζ_{SVM}, we resize all ground-truth bounding boxes to the mean aspect-ratio of that class, and extract a 2816-dimensional HOG feature vector, again with 32 gradient directions, and train a SVM. We would like to state that we use only the positive training images from the set, $\mathcal{I}^+$, to extract even the "hard negative" patches. We do not use negative

	Applelogos	Bottles	Giraffes	Mugs	Swans	Mean
Ours	**0.897**	0.939	0.800	**0.889**	0.819	0.868
Srinivasan et al. [91]	0.845	0.916	0.787	0.888	0.922	0.872
Felzenswalb et al. [31]	0.891	0.950	0.608	0.721	0.391	0.712
Wang et al. [100]	0.866	**0.975**	**0.832**	0.843	0.828	0.869
Ma et al. [61]	0.881	0.920	0.756	0.868	**0.959**	**0.877**
Lin et al. [53]	**0.909**	0.898	0.811	0.893	**0.964**	**0.895**
Ours Random	0.908	**0.951**	**0.821**	**0.894**	0.847	0.884

TABLE 6.1: Comparison of interpolated Average Precision (AP). Please note that the method in [53] does not follow the standard protocol given in [34]. They use a *random* 50% subset of the positive images and use negative images from other categories for training their method. Among the methods that follow the protocol, we get the best AP for Applelogos and Mugs, and comparable results for other classes.

examples from other classes for training purposes, as is done by [62], [91] and [53]. This means that we are training our model with fewer number of training images and, therefore, testing our model on more number of test images than [62], [91] and [53]. The complete training procedure takes between 10 $\sim$ 20 minutes per class, which is much quicker than the 4 $\sim$ 8 hours that the method in [53] takes. The inference stage is also extremely quick, because of our pruning step, and because of the efficient performance of the TRW-S algorithm. We search each image at 12 different scales, and the time to process each image varies from 15 seconds$\sim$4 minutes depending on the image. We also tried to search at a single scale by inferring the best scale at each hypothesis location, using the scale-voting method in [100], but found that we got better results by exhaustively searching over multiple scales.

For comparison with other methods, we plot the standard precision/recall (PR) curves in the top-row of Figure 6.6, and show the average precision (AP) in Table 6.1. We also plot the Detection Rate v. False Positives Per Image (DR/FPPI) curve in the bottom row of Figure 6.6, and show the detection rates at 0.3 and 0.4 FPPI in Table 6.2. As can be seen from the plots and the tables, our method performs well on many classes and helps achieve state-of-the-art results on many tests. We also tested the effect of training set and used a *random* 50% subset of positive images for training. The results are presented along with the results from the standard protocol. We still do not use negative images from other classes during training. Figure 6.4 shows some detection results on the ETH-Z database using our method.

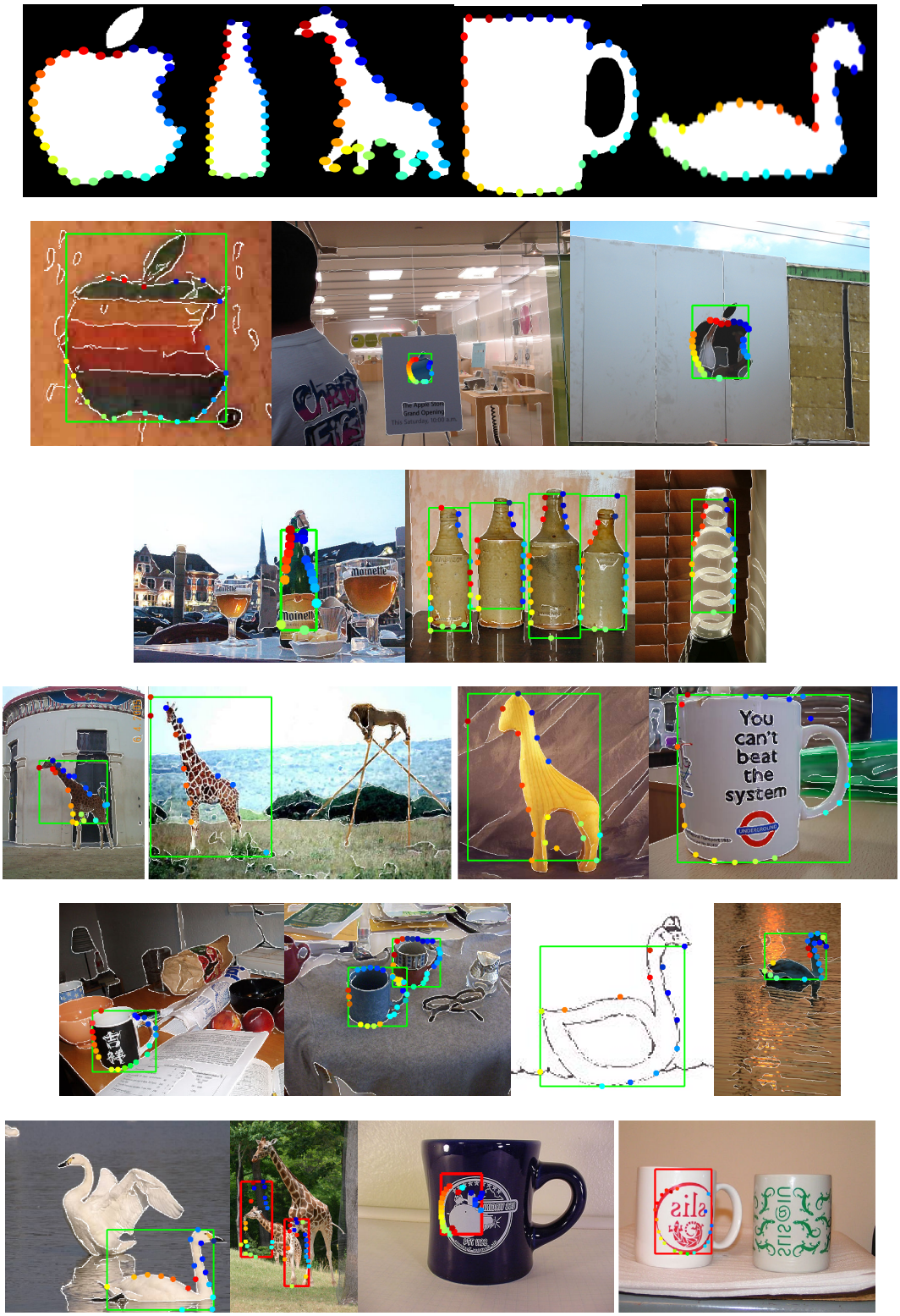


FIGURE 6.4: Some example detection results on the ETH-Z database using our method. Top row shows the part-label color code. The final three results are false positives.

	Applelogos	Bottles	Giraffes	Mugs	Swans
Ours	**0.95/ 1**	**1/ 1**	0.872/ 0.872	0.903/ 0.936	**1/ 1**
Srinivasan et al. [91]	**0.95**/ 0.95	**1/ 1**	0.872/ 0.896	0.936/ 0.936	**1/ 1**
Felzenszwalb et al. [31]	**0.95**/ 0.95	**1/ 1**	0.729/ 0.729	0.839/ 0.839	0.588/ 0.647
Wang et al. [100]	0.90/ 0.90	**1/ 1**	**0.92/ 0.92**	0.94/ 0.94	0.94/ 0.94
Yarlagadda and Ommer [109]	**0.95**/ 0.95	**1/ 1**	0.913/ 0.913	**0.967/ 0.967**	**1/ 1**
Ma and Latecki [61]	0.92/ 0.92	0.979/ 0.979	0.854/ 0.854	0.875/ 0.875	**1/ 1**
Ours Random	**0.95/ 1**	**1/ 1**	0.888/ 0.888	0.937/ 0.937	**1/ 1**

	Mean
Ours	0.945/0.961
Srinivasan et al. [91]	0.952/0.952
Felzenszwalb et al. [31]	0.821/0.833
Wang et al. [100]	0.94/0.94
Yarlagadda and Ommer [109]	**0.965/0.965**
Ma and Latecki [61]	0.926/0.926
Ours Random	0.955/**0.965**

TABLE 6.2: Comparison of Detection Rates (DR) at 0.3/0.4 FPPI. Our method gets a new best DR for Applelogos and ties with other methods for Bottles and Swans. For all the classes, we tie, or beat, the deformable part models [31]. This, we believe, is because of the localization ability that the edge maps provide to our method. Also shown, are the results from the random subset training test.

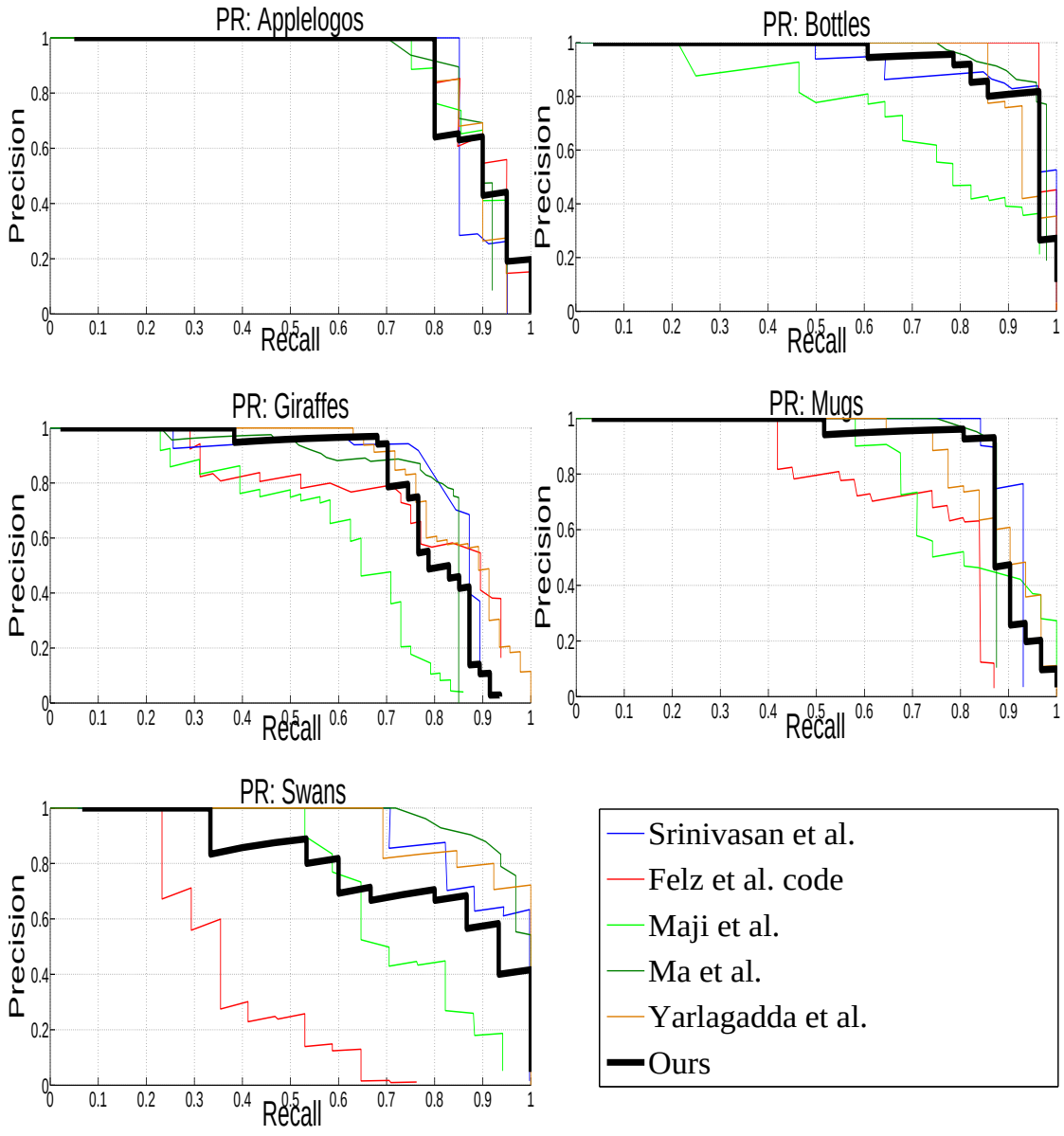


FIGURE 6.5: This figure shows the P/R plots for the 5 classes of the ETH-Z database. The legend for all plots is shown in the final subfigure.

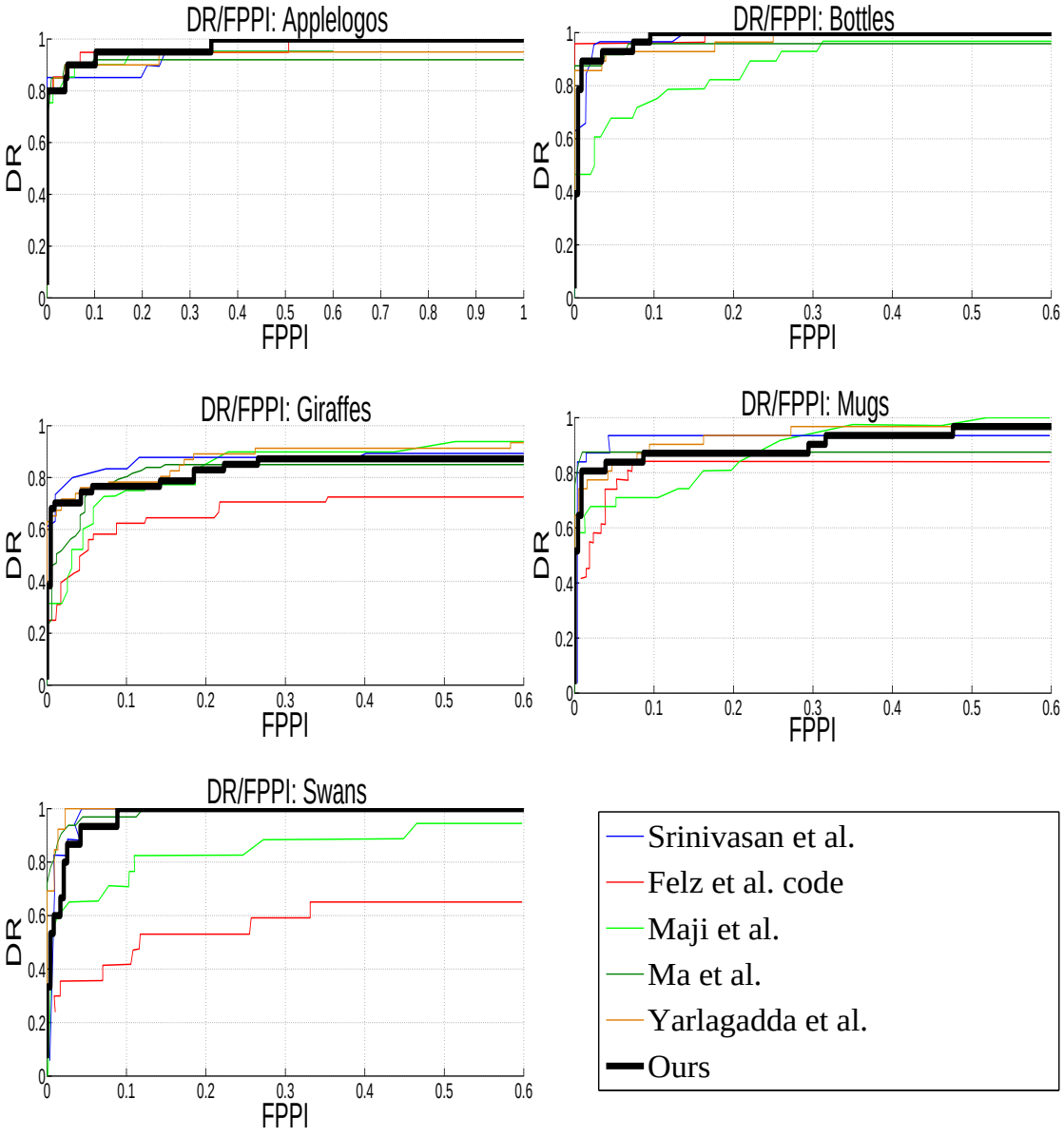


FIGURE 6.6: This figure shows the DR/FPPI plots for the 5 classes of the ETH-Z database. The legend for all plots is shown in the final subfigure.

6.7 Conclusion and Discussions

This chapter brings together various aspects of object detection and presents an unified approach towards the same. The chapter stresses the importance of enforcing a structure while recognizing objects and strives to present a generic model that can be used by any shape. The main contributions of this chapter are two fold. Firstly, we bridge the gap between HOG-based approaches and contour-based approaches by exploiting their complementary advantages. Secondly, we stress the importance of using a fully connected graph to model inter-part interactions. Experimental results on a challenging database shows the effectiveness of our approach. As with any other learning technique, the statistics of the subtle pairwise interactions that we wish to capture can be strengthened by using more of the domain knowledge and/or by learning from a larger database.

Chapter 7

Summary and Outlook

7.1 Summary

Abundance of visual data has fueled research in the area of computer vision. This thesis addressed problems specific to shape analysis and the usage of shapes in vision-based applications. We started by giving a strong motivation for the usage of shapes by citing their invariance to lighting conditions, their consistency among different object instances and the strong visual appeal they have on the human visual system. Immediate applications such as object retrieval and object detection also motivate the research in this field. To give a background and flow to the work in this thesis, we presented a historical review of some of the most important milestones in the relatively new area of shape matching. We explained how the area came into existence. The lack of progress in low-level computer vision algorithms such as the edge detectors, corner detectors, and primitive detectors, in conjunction with the development of robust histogram-based solutions that came into existence (at the same time), further fueled research in the development of better statistical shape descriptors. While the field has seen tremendous progress in terms of performance, there were still some fundamental questions and nagging issues that were left unanswered.

The assumption that all of the shape information lies only in the object's contour seemed to be inherently erroneous. We made it apparent in Chapter 3 by showing visually similar shapes that had vastly different contour properties. The identification of this problem led to the development of a perceptually-motivated variant

of the well-known shape context descriptor, which we call as the Solid Shape Context (since it make use of the interior properties of the shape). The SSC proved to be robust to indentations in the object's contour and helped in significantly improving the retrieval rates.

Research in the area of shape matching has followed a linear path where advancements tend to be extensions, or variants, of previously existing techniques. The exciting improvements that the advancements provided had somewhat blinded the community from exploring and answering other interesting questions. In Chapter 4, we ask a novel question of whether all parts of a shape are equally important. Visually looking at competing classes such as Apples and Pocket-watches points to the fact that some parts of the shape are more discriminative than others. Motivated by this finding, we developed a nearest neighbor-based algorithm to learn the importance maps of different parts of the contour. We showed that we were able to find semantically meaningful parts using the proposed method and that it even helped improve retrieval rates. Future research in this area seems to be quite promising.

We then addressed the issue of going beyond pairwise comparison of shapes and explained how making use of the shape manifold helped improve retrieval rates, significantly. However, the susceptibility of diffusion processes to noisy neighborhood connections forced manual supervision while selecting appropriate neighborhoods. Therefore, in Chapter 5, we proposed a means to select robust local neighborhood by mining information from multiple rounds of k-NNs. This not only resulted in obtaining robust neighborhoods, but also helped in providing probabilistic information about a node's edges.

Finally, in Chapter 6, we make contributions in the area of shape-based object detection as well. We explained the standard pipelines in object detection and pointed to the gulf between HOG-based object detection and contour-based object detection. In addition, we elucidated the need for structured prediction approaches since the object shapes are highly structured entities. Moreover, the object parts are more recognizable in the presence of other object parts, than independently. Thereby, we proposed a parts-based structured prediction approach, which also bridges the gap between HOG-based and contour-based object recognition approaches.

7.2 Future Work

In this thesis, we saw the benefits of using shapes for performing important tasks such as object retrieval and object detection. Building shape descriptors motivated by human perception did show significant improvements in shape matching. The lack of perceptually-motivated approaches shows the potential that this area has for future research. We hope that our work motivates other researchers to develop algorithms that are intuitively simple and perceptually-motivated. In addition, the concept of discriminative parts of a shape is a novel contribution of this thesis. Many areas of shape matching can now be explored with our introduction of shape-importance. One straightforward extension would be to come up with more sophisticated importance sampling schemes, which can aid in the development of more discriminative, and descriptive, shape descriptors.

We also showed the effect of selecting a noisy neighborhood while performing diffusion on shape manifolds. While we were able to advance the field by introducing soft measures for retaining edges while sparsifying the graph, the issue that still needs tackling is that of finding an optimal graph sparsification parameter 'k' for the graph. We would like to go a step further and say that finding adaptive 'k' across different nodes of the graph should be the focus of further research, and we believe that the soft measures that our algorithm provides is an ideal starting point to explore this area of research.

Finally, the area of object detection is far from being solved. While the ETH-Z shape database is a challenging database on its own, we are starting to see various approaches perform similarly on it, making it hard to discriminate among them. Therefore, the database is no longer considered as a state-of-the-art database. This calls for the collection of bigger shape databases (on the likes of PASCAL dataset), which incorporates bigger challenges such as significant variations in pose of articulated objects, in addition to increasing the number of object classes. In Chapter 6, we also saw the benefits of using structured prediction approaches. However, we used a fully connected graph to model all pairwise interactions. The most appealing future direction would be to come up with techniques that can learn which pairwise interactions are important and which are not, thus enabling the reduction in complexity by allowing us to prune out the less important edges.

While this thesis concentrated on showing the importance of shape properties in recognition and detection, and helped improve techniques that worked on shapes,

future research should concentrate on the development of a holistic framework that makes use of all sorts of object cues, i.e., shape, color and texture. Combining such multiple cues are not only an interesting area to work on, but also important in advancing the field.

7.3 Final Thoughts

I conclude this thesis with the following observations and thoughts. Going back to the statements that were made in Chapter 1, with billions of cameras being sold each year, and millions of photos being uploaded each day, it is clear that the research in the foreseeable future involves the processing of "Big Data". Therefore, to be a front-runner in the field, it is important for researchers to develop algorithms that can process huge amounts of data. This calls for the development of more online algorithms compared to batch algorithms.

Future research in computer vision will involve bridging the gap between low-level image primitives and high-level scene semantics. To achieve semantic understanding of scenes, algorithms that make use of contextual information will play an important role. Structured prediction tools from the area of machine learning perfectly suit this requirement. Therefore, I see that there will be an explosion in the research community in developing both generic and domain-specific structured prediction algorithms, which aim at predicting semantic labels. Training such structured prediction algorithms, which will have the capability to learn contextual image statistics, in an online fashion, will be focus of future research. All in all, this is a truly exciting time to be working in this area.

Finally, on a more generic note, the explosion in the usage of cameras and production of visual data, is bound to throw up other non-technical challenges. Similar to the issues with text data, we will face debates about privacy issues related to visual data. If history is anything to by, it is highly unlikely that the advancements in technology, and thus its use in public life, is going to wane down because of privacy issues. Therefore, we as researchers have a huge responsibility to ensure that our innovations and products are not susceptible for misuse. As computer vision researchers, we have a special responsibility towards the public, and our children, to build surveillance systems that future generations will be proud of. I feel that

there is a necessity to do research keeping in mind societal values, expecting misuse of data, and ensuring that we do everything in our capacity to protect the user's basic human rights. All of this might look like a challenging task. However, we researchers do love a challenge, do we not?

Bibliography

[1] T. Adamek and N. E. O'Connor. A multiscale representation method for nonrigid shapes with a single closed contour. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(5):742–753, 2004. 51

[2] N. Alajlan, M. S. Kamel, and G. H. Freeman. Geometry-based image retrieval in binary image databases. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):1003–1013, 2008. 51

[3] Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1300–1305, 1997. 97

[4] M. J. Atallah. A linear time algorithm for the Hausdorff distance between convex polygons. *Information Processing Letters*, 17(4):207–209, 1983. ISSN 0020-0190. 12

[5] E. Attalla and P. Siy. Robust shape similarity retrieval based on contour segmentation polygonal multiresolution and elastic matching. *Pattern Recognition*, 38(12):2229–2241, 2005. 51

[6] X. Bai, X. Yang, L. J. Latecki, W. Liu, and Z. Tu. Learning context-sensitive shape similarity by graph transduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):861–874, 2010. 38, 54, 75

[7] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *Proc. DARPA IU Workshop*, pages 21–27. Citeseer, 1978. 12

[8] E. Baseski, A. Erdem, and S. Tari. Dissimilarity between two skeletal trees in a context. *Pattern Recognition*, 42(3):370–385, 2009. 56, 72

[9] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Neural Information Processing Systems*, 14:585–591, 2001. 21

[10] S. Belongie, J. Malik, and J. Puzicha. Matching with shape contexts. URL `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/shape/sc_digits.html`. 15

[11] S. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *Neural Information Processing Systems*, 2000. 13, 14

[12] S. Belongie, J. Malik, and J. Puzicha. Matching shapes. In *International Conference on Computer Vision*, volume 1, pages 454–461, 2001. 14

[13] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 509–522, 2002. 7, 35, 43, 44, 45, 51, 63, 64, 65, 74, 76

[14] S. Belongie, G. Mori, and J. Malik. Matching with shape contexts. *Statistics and Analysis of Shapes*, pages 81–105, 2006. 14

[15] I. Biederman and G. Ju. Surface versus edge-based determinants of visual recognition. *Cognitive Psychology*, 20(1):38–64, 1988. 2

[16] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive gmmrf model. In *European Conference on Computer Vision*, pages 428–441. Springer, 2004. 2

[17] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, 1988. ISSN 0162-8828. 11

[18] M. Bray, P. Kohli, and P. Torr. Posecut: Simultaneous segmentation and 3d pose estimation of humans using dynamic graph-cuts. In *European Conference on Computer Vision*, pages 642–655. Springer, 2006. 3

[19] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 97

[20] A. M. Bronstein, M. M. Bronstein, A. M. Bruckstein, and R. Kimmel. Analysis of two-dimensional non-rigid shapes. *International Journal of Computer Vision*, 78(1):67–88, 2008. 18

[21] A. M. Bronstein, M. M. Bronstein, A. M. Bruckstein, and R. Kimmel. Partial similarity of objects, or how to compare a centaur to a horse. *International Journal of Computer Vision*, 84(2):163–183, 2009. 18, 19, 27, 65, 103

[22] J. Canny. A computational approach to edge detection. *Readings in computer vision: issues, problems, principles, and paradigms*, 184, 1987. 6

[23] S. Changming and S. Jamie. 3-d symmetry detection using the extended gaussian image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:164–168, 1997. 57

[24] F. C. Crow. Summed-area tables for texture mapping. In *ACM SIGGRAPH Computer Graphics*, volume 18, pages 207–212. ACM, 1984. 19

[25] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*, volume 1, pages 886–893. IEEE, 2005. 25, 92

[26] M. R. Daliri and V. Torre. Robust symbolic representation for shape recognition and retrieval. *Pattern Recognition*, 41(5):1782–1798, 2008. 51

[27] A. Desolneux, L. Moisan, and J. M. Morel. *From gestalt theory to image analysis: a probabilistic approach*. Springer Verlag, 2008. 33

[28] M. Donoser and H. Bischof. Diffusion processes for retrieval revisited. In *Computer Vision and Pattern Recognition*. IEEE, 2013. 75

[29] M. Donoser, H. Riemenschneider, and H. Bischof. Efficient partial shape matching of outer contours. In *Asian Conference on Computer Vision*, pages 281–292. Springer, 2009. 18, 20, 27, 65, 67

[30] P. F. Felzenszwalb and J. D. Schwartz. Hierarchical matching of deformable shapes. In *Computer Vision and Pattern Recognition*, pages 1–8, 2007. 26, 34, 51

[31] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010. 7, 28, 93, 97, 105, 106, 108

[32] V. Ferrari, T. Tuytelaars, and L. Van Gool. Object detection by contour segment networks. In *European Conference on Computer Vision*, pages 14–28. Springer, 2006. 26, 105

[33] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):36–51, 2008. 26, 92

[34] V. Ferrari, F. Jurie, and C. Schmid. From images to shape models for object detection. *International Journal of Computer Vision*, 87(3):284–303, 2010. 26, 105, 106

[35] M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 100(1):67–92, 1973. 28, 30

[36] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2188–2202, 2011. 28

[37] R. Gopalan, P. Turaga, and R. Chellappa. Articulation-invariant representation of non-planar shapes. In *European Conference on Computer Vision*, pages 286–299. Springer, 2010. 17, 35, 51, 53, 54, 56

[38] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988. 12

[39] Rong-Xiang Hu, J. Wei, Z. Yang, and G. Jie. Perceptually motivated morphological strategies for shape retrieval. *Pattern Recognition*, 45(9):3222–3230, 2012. 36, 37, 38, 46, 50, 51, 56

[40] D. P. Huttenlocher, G. A. Klanderman, and W. A. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 850–863, 1993. 11

[41] M. S. T. Jaakkola. Partially labeled classification with markov random walks. In *Neural Information Processing Systems*, volume 2, page 945. MIT Press, 2002. 75, 77

[42] C. R. Jung and R. Schramm. Rectangle detection based on a windowed hough transform. In *Brazilian Symposium on Computer Graphics and Image Processing*, pages 113–120. IEEE, 2004. 13

[43] R. Kakarala, P. Kaliamoorthi, and V. Premachandran. Three-dimensional bilateral symmetry plane estimation in the phase domain. In *Computer Vision and Pattern Recognition*, 2013. 10, 58

[44] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3d mesh segmentation and labeling. *ACM Transactions on Graphics (TOG)*, 29(4):102, 2010. 17, 60

[45] D. G. Kendall. Shape manifolds, procrustean metrics, and complex projective spaces. *Bulletin of the London Mathematical Society*, 16(2):81–121, 1984. 2, 21

[46] A. Khosla, J. Xiao, A. Torralba, and A. Oliva. Memorability of image regions. In *Neural Information Processing Systems*, Lake Tahoe, USA, December 2012. 66

[47] I. Kokkinos, M. Bronstein, R. Litman, and A. Bronstein. Intrinsic shape context descriptors for deformable shapes. In *Computer Vision and Pattern Recognition*, pages 159–166. IEEE, 2012. 17

[48] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006. 103

[49] P. Kontschieder, M. Donoser, and H. Bischof. Beyond pairwise shape similarity analysis. In *Asian Conference on Computer Vision*, pages 655–666. Springer, 2010. 38, 54, 75

[50] P. D. Kovesi. MATLAB and Octave functions for computer vision and image processing. Centre for Exploration Targeting, School of Earth and Environment, The University of Western Australia. Available from: <http://www.csse.uwa.edu.au/∼pk/research/matlabfns/>. 27, 29, 92

[51] A. Lancichinetti and S. Fortunato. Consensus clustering in complex networks. *Scientific Reports*, 2, 2012. 78

[52] L. J. Latecki and R. Lakamper. Shape similarity measure based on correspondence of visual parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1185–1190, 2000. 51

[53] L. Lin, X. Wang, W. Yang, and J. Lai. Learning contour-fragment-based shape model with and-or tree representation. In *Computer Vision and Pattern Recognition*, pages 135–142, 2012. 28, 97, 105, 106

[54] H. Ling and D. W. Jacobs. Using the inner-distance for classification of articulated shapes. In *Computer Vision and Pattern Recognition*, volume 2, pages 719–726. IEEE, 2005. 16

[55] H. Ling and D. W. Jacobs. Shape classification using the inner-distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:286–299, 2007. 16, 35, 43, 44, 45, 46, 47, 51, 63, 64, 65, 66, 74, 76, 88, 97

[56] H. Ling and K. Okada. An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):840–853, 2007. 51

[57] H. Ling, X. Yang, and L. Latecki. Balancing deformability and discriminability for shape matching. In *European Conference on Computer Vision*, pages 411–424. Springer, 2010. 36, 46, 50, 51, 53, 56

[58] H. Liu, X. Yang, L. J. Latecki, and S. Yan. Dense neighborhoods on affinity graph. *International Journal of Computer Vision*, pages 1–18, 2011. 81

[59] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision and Pattern Recognition*, volume 2, pages 1150–1157. IEEE, 1999. 7, 13

[60] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 13

[61] T. Ma and L. J. Latecki. From partial shape matching through local deformation to robust global shape similarity for object detection. In *Computer Vision and Pattern Recognition*, pages 1441–1448. IEEE, 2011. 20, 27, 28, 65, 92, 106, 108

[62] S. Maji and J. Malik. Object detection using a max-margin hough transform. In *Computer Vision and Pattern Recognition*, pages 1038–1045, 2009. 28, 106

[63] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, 26(5):530–549, 2004. 6, 26, 93, 105

[64] G. McNeill and S. Vijayakumar. Hierarchical procrustes matching for shape retrieval. In *Computer Vision and Pattern Recognition*, volume 1, pages 885–894. IEEE, 2006. 51

[65] F. Mokhtarian and M. Bober. *Curvature Scale Space Representation: Theory, Applications, and MPEG-7 Standardization*. Kluwer Academic Publishers, 2003. 51

[66] S. Monti, P. Tamayo, J. Mesirov, and T. Golub. Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. *Machine learning*, 52(1):91–118, 2003. 78

[67] G. Mori, S. Belongie, and J. Malik. Shape contexts enable efficient retrieval of similar shapes. *Computer Vision and Pattern Recognition*, 1:723–730, 2001. 14

[68] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM Transactions on Graphics (TOG)*, 21(4):807–832, 2002. 41

[69] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Dover Publications, 1998. 18

[70] D. Parikh, P. Isola, A. Torralba, and A. Oliva. Understanding the intrinsic memorability of images. *Journal of Vision*, 12(9):1082–1082, 2012. 66

[71] L. Paul Chew. Constrained delaunay triangulations. *Algorithmica*, 4(1): 97–108, 1989. 41

[72] M. Pavan and M. Pelillo. Dominant sets and pairwise clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):167–172, 2007. 27, 78, 80, 82

[73] P. Perona and L. Zelnik-Manor. Self-tuning spectral clustering. In *Neural Information Processing Systems*, volume 17, pages 1601–1608, 2004. 76

[74] E. Persoon and K. S. Fu. Shape discrimination using fourier descriptors. *IEEE Transactions on Systems, Man and Cybernetics*, 7(3):170–179, 1977. 13

[75] R. B. Potts. Some generalized order-disorder transformations. In *Proceedings of the Cambridge Philosophical Society*, volume 48, pages 106–109. Cambridge Univ Press, 1952. 102

[76] V. Premachandran and R. Kakarala. Perceptually motivated shape context which uses shape interiors. *Pattern Recognition*, 46(8):2092–2102, 2013. 10, 65

[77] V. Premachandran and R. Kakarala. Dense sampling of shape interiors for improved representation. In *Image Processing: Machine Vision Applications VI*. Proceedings of SPIE Vol. 8661, 2013. 10

[78] V. Premachandran and R. Kakarala. What parts of a shape are discriminative? In *IEEE International Conference on Image Processing*, 2013. 10

[79] V. Premachandran and R. Kakarala. Consensus of k-nns for robust neighborhood selection on graph-based manifolds. In *Computer Vision and Pattern Recognition*, 2013. 10, 27

[80] H. Riemenschneider, M. Donoser, and H. Bischof. Using partial edge contour matches for efficient object category localization. In *European Conference on Computer Vision*, pages 29–42. Springer, 2010. 27, 28, 92, 103

[81] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000. 21, 74, 77, 86

[82] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1):157–173, 2008. ISSN 0920-5691. 5

[83] B. Schölkopf, A. Smola, and K. R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998. 74

[84] T. B. Sebastian, P. N. Klein, and B. B. Kimia. On aligning curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:116–125, 2003. 34

[85] T. B. Sebastian, P. N. Klein, and B. B. Kimia. Recognition of shapes by editing their shock graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):550–571, 2004. 36, 57

[86] J. Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. *Applied Computational Geometry Towards Geometric Engineering*, pages 203–222, 1996. 41

[87] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The Princeton shape benchmark. In *Shape Modeling International*, June 2004. 58

[88] J. Shotton, A. Blake, and R. Cipolla. Multiscale categorical object recognition using contour fragments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1270–1281, 2007. 3, 11, 26, 92

[89] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 35 (1):13–32, 1999. 36

[90] S. M. Smith and J. M. Brady. SUSAN A new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997. ISSN 0920-5691. 13

[91] P. Srinivasan, Q. Zhu, and J. Shi. Many-to-one contour matching for describing and discriminating object shape. In *Computer Vision and Pattern Recognition*, pages 1673–1680. IEEE, 2010. 26, 92, 106, 108

[92] A. Temlyakov, B. C. Munsell, J. W. Waggoner, and S. Wang. Two perceptually motivated strategies for shape classification. In *Computer Vision and Pattern Recognition*, pages 2289–2296, 2010. 36, 37, 38, 46, 50, 51, 53, 56

[93] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. 21, 22, 86

[94] A. Thayananthan, B. Stenger, P. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *Computer Vision and Pattern Recognition*, volume 1, pages I:127–133. IEEE, 2003. 16, 35

[95] M. Trajkovi and M. Hedley. Fast corner detection. *Image and Vision Computing*, 16(2):75–87, 1998. ISSN 0262-8856. 13

[96] Z. Tu and A. Yuille. Shape matching and recognition-using generative models and informative features. In *European Conference on Computer Vision*, pages 195–209. Springer, 2004. 51

[97] O. Tuzel, F. Porikli, and P. Meer. Human detection via classification on riemannian manifolds. In *Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 74

[98] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004. 20, 25

[99] R. G. von Gioi, J. Jakubowicz, J. M. Morel, and G. Randall. LSD: A Fast Line Segment Detector with a False Detection Control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):722–732, 2010. ISSN 0162-8828. 13

[100] X. Wang, X. Bai, T. Ma, W. Liu, and L. J. Latecki. Fan shape model for object detection. In *Computer Vision and Pattern Recognition*, pages 151–158, 2012. 28, 94, 98, 105, 106, 108

[101] J. Xiao, B. Russell, and A. Torralba. Localizing 3d cuboids in single-view images. In *Neural Information Processing Systems*, pages 755–763, 2012. 103

[102] R. Xiaofeng and L. Bo. Discriminatively trained sparse code gradients for contour detection. In *Neural Information Processing Systems*, pages 593–601, 2012. 26, 93

[103] J. Xie, P. A. Heng, and M. Shah. Shape matching and modeling using skeletal context. *Pattern Recognition*, 41(5):1756–1767, 2008. 36

[104] C. Xu, J. Liu, and X. Tang. 2d shape matching by contour flexibility. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(1):180–186, 2009. 51, 65

[105] X. Yang and L. J. Latecki. Affinity learning on a tensor product graph with applications to shape and image retrieval. In *Computer Vision and Pattern Recognition*, pages 2369–2376. IEEE, 2011. 77, 78, 80, 81

[106] X. Yang, X. Bai, L. J. Latecki, and Z. Tu. Improving shape retrieval by learning graph transduction. *European Conference on Computer Vision*, pages 788–801, 2008. 38, 51, 54, 75, 76

[107] X. Yang, S. Koknar-Tezel, and L. J. Latecki. Locally constrained diffusion process on locally densified distance spaces with applications to shape retrieval. In *Computer Vision and Pattern Recognition*, pages 357–364. IEEE, 2009. 23, 38, 51, 54, 56, 74, 75, 86, 88

[108] X. Yang, L. Prasad, and L. J. Latecki. Affinity learning with diffusion on tensor product graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012. 23, 24, 38, 51, 56, 75, 86, 88

[109] P. Yarlagadda and B. Ommer. From meaningful contours to discriminative object shape. In *European Conference on Computer Vision*, pages 766–779. Springer, 2012. 28, 108

[110] C. T. Zahn and R. Z. Roskies. Fourier descriptors for plane closed curves. *IEEE Transactions on Computers*, C-21(3):269–281, 1972. 13

[111] J. Zhang, K. Siddiqi, D. Macrini, A. Shokoufandeh, and S. Dickinson. Retrieving articulated 3-d models using medial surfaces and their graph spectra. In *Energy minimization methods in computer vision and pattern recognition*, pages 285–300. Springer, 2005. 60

[112] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision*, 73(2):213–238, 2007. 92

[113] Z. Zhang, J. Wang, and H. Zha. Adaptive manifold learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(2):253–265, 2012. 22, 77

[114] Z. Zhenyue and Z. Hongyuan. Principal manifolds and nonlinear dimension reduction via local tangent space alignment. *SIAM Journal of Scientific Computing*, 26:313–338, 2002. 21

List Of Publications

Journals

[1] V. Premachandran and R. Kakarala. Perceptually motivated shape context which uses shape interiors. *Pattern Recognition*, 46(8):2092-2102, 2013.

[2] V. Premachandran and R. Kakarala. Measuring the effectiveness of bad pixel detection algorithms using the roc curve. *IEEE Transactions on Consumer Electronics*, 56(4):2511-2519, 2010.

Conferences

[3] V. Premachandran and R. Kakarala. What parts of a shape are discriminative? to *IEEE International Conference on Image Processing (ICIP)*, 2013. (Oral Presentation) Best Student Paper Award!

[4] V. Premachandran and R. Kakarala. Consensus of k-nns for robust neighborhood selection on graph-based manifolds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[5] V. Premachandran and R. Kakarala. Dense sampling of shape interiors for improved representation. In Image Processing: Machine Vision Applications VI, Proceedings of SPIE Vol. 8661, 2013.

[6] V. Premachandran and R. Kakarala. Improving shape context using geodesic information and reflection invariance. In Intelligent Robots and Computer Vision XXX: Algorithms and Techniques, Proceedings of SPIE Vol. 8662, 2013.

[7] A. Agrawal, V. Premachandran, R. Somavarapu, and R. Kakarala. Can relative skill be determined from a photographic portfolio? In Human Vision and Electronic Imaging XVIII, Proceedings of SPIE Vol. 8651, 2013.

[8] R. Kakarala, P. Kaliamoorthi*, and V. Premachandran*. Three-dimensional bilateral symmetry plane estimation in the phase domain. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. (* indicates equal contribution)

[9] R. Kakarala, T. Sachs, and V. Premachandran. Comparing automated and human ratings of photographic aesthetics. In *Color Imaging Conference*, 2011.