

Shape Based Object Detection and Recognition in Silhouettes and  
Real Images

---

A Dissertation  
Submitted to  
the Temple University Graduate Board

---

in Partial Fulfillment  
of the Requirements for the Degree of  
DOCTOR OF PHILOSOPHY

---

by  
Xingwei Yang  
March, 2011

©

by

Xingwei Yang

March, 2011

All Rights Reserved

## ABSTRACT

Shape Based Object Detection and Recognition in Silhouettes and Real  
Images

Xingwei Yang

DOCTOR OF PHILOSOPHY

Temple University, March, 2011

Dr. Longin Jan Latecki, Chair

Shape is very essential for detecting and recognizing objects. It is robust to illumination, color changes. Human can recognize objects just based on shapes, thus shape based object detection and recognition methods have been popular in many years.

Due to problem of segmentation, some researchers have worked on silhouettes instead of real images. The main problem in this area is object recognition and the difficulty is to handle shapes articulation and distortion. Previous methods mainly focus on one to one shape similarity measurement, which ignores context information between shapes. Instead, we utilize graph-transduction methods to reveal the intrinsic relation between shapes on 'shape manifold'. Our methods consider the context information in the dataset, which improves the performance a lot. To better describe the manifold structure,

we also propose a novel method to add synthetic data points for densifying data manifold. The experimental results have shown the advantage of the algorithm. Moreover, a novel diffusion process on Tensor Product Graph is carried out for learning better affinities between data. This is also used for shape retrieval, which reaches the best ever results on MPEG-7 dataset.

As shapes are important and helpful for object detection and recognition in real images, a lot of methods have used shapes to detect and recognize objects. There are two important parts for shape based methods, model construction and object detection, recognition. Most of the current methods are based on hand selected models, which is helpful but not extendable. To solve this problem, we propose to construct model by shape matching between some silhouettes and one hand decomposed silhouette. This weakly supervised method can be used not only learn the models in one object class, but also transfer the structure knowledge to other classes, which has the similar structure with the hand decomposed silhouette. The other problem is detecting and recognizing objects. A lot of methods search the images by sliding window to detect objects, which can find the global solution but with high complexity. Instead, we use sampling methods to reduce the complexity. The method we utilized is particle filter, which is popular in robot mapping and localization. We modified the standard particle filter to make it suitable for static observations and it is very helpful for object detection. Moreover, The usage of particle filter is

extended for solving the jigsaw puzzle problem, where puzzle pieces are square image patches. The proposed method is able to reach much better results than the method with Loopy Belief Propagation.

## ACKNOWLEDGEMENTS

I am a really lucky guy, who has obtained supports and helps from many wonderful people. I now get to formally acknowledge them. First I would like to thank my advisor, Longin Jan Latecki. I have learned so much from him over the past few years, and he has been a very patient and supportive advisor. He is a wonderful mentor. I am also in awe of his dedication to scholarship. Other professors at Temple that I have had the good fortune to learn from and interact with: Slobodan Vucetic and Haibin Ling (my committee members), Eugene Kwatny, Rolf Lakaemper, Zoran Obradovic, Arthur T. Poe, Yuan Shi. I also want to thank my collaborator and good friend, Xiang Bai! He helped me a lot for my research and careers. Also, Hairong Liu, another friend and collaborators from HUST, teaches me a lot and gives me many suggestions. Besides, I would like to thank my lab-mates, Nagesh Adluru, Suzan Koknar-Tezel, Chengen Lu, Tianyang Ma, Xinggang Wang, Meng Yi and Nan Li. The discussion and collaboration between you and me are exciting and intriguing. I really hope that we could continue cooperating with each other in future.

My deepest gratitude to many people I have worked with. Professor Zhuowen Tu at UCLA who have given me suggestions not only in research but also my careers. Professor Jianbo Shi from University of Pennsylvania (my committee member) gave me opportunity for joining the summer school at Beijing 2009, where I learnt a lot of state of art methods in Computer vision.

I thank Wenyu Liu, Professor at HUST, for providing research opportunities to me during my undergraduate, I am proud of being a undergraduate student at HUST.

My family has done so much for me, it is impossible to list everything. My love and thanks to my mother, Qin Yang, for supporting me no matter what happens; to my father Mingliang Yang , who is the most honest, principled man I know. He teaches me how to face pressure and problems; to my wife Cui Cao, who is always supporting and supervising me for my research and careers.

To my family,

Cui Cao, Qin Yang, Mingliang Yang,

Without your help, I cannot finish the journey for doctor.

I love you so much



# TABLE OF CONTENTS

<b>ABSTRACT</b>	<b>iv</b>
<b>ACKNOWLEDGEMENT</b>	<b>vii</b>
<b>DEDICATION</b>	<b>ix</b>
<b>LIST OF FIGURES</b>	<b>xii</b>
<b>LIST OF TABLES</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Shape Retrieval . . . . .	1
1.1.1 Related Work . . . . .	3
1.2 Shape Based Object Detection . . . . .	6
1.2.1 Related Work . . . . .	7
1.3 Particle Filter with State Permutations for Solving Image Jigsaw Puzzles . . . . .	8
1.3.1 Related Work . . . . .	8
<b>2 Shape Retrieval of Silhouettes</b>	<b>10</b>
2.1 Learning Context Sensitive Shape Similarity by Graph Transduction . . . . .	10
2.1.1 Introduction . . . . .	10
2.1.2 Learning New Distance Measures . . . . .	14
2.1.3 Relation to Label Propagation . . . . .	18
2.1.4 The Affinity Matrix . . . . .	21
2.1.5 Experimental Results . . . . .	22
2.2 Densifying Shape Manifold with Ghost Point . . . . .	36
2.2.1 Introduction . . . . .	36
2.2.2 Ghost points and metric embedding . . . . .	38
2.2.3 Diffusion process . . . . .	42

2.2.4	Locally Constrained Diffusion Process . . . . .	45
2.2.5	Experimental results . . . . .	48
2.3	Affinity Learning on a Tensor Product Graph with Applications to Shape and Image Retrieval . . . . .	55
2.3.1	Introduction . . . . .	55
2.3.2	Affinity Learning . . . . .	60
2.3.3	Diffusion Process on Tensor Product Graph . . . . .	62
2.3.4	Iterative Algorithm for Diffusion on TPG . . . . .	65
2.3.5	Dominant Neighbors . . . . .	68
2.3.6	Experimental Results . . . . .	71
<b>3</b>	<b>Shape Based Object Detection on Real Images</b>	<b>79</b>
3.1	Introduction . . . . .	79
3.2	Partially-Supervised Model Learning . . . . .	82
3.2.1	Part model construction . . . . .	83
3.2.2	Relation between model parts . . . . .	85
3.3	Framework for Object Detection . . . . .	87
3.4	Evaluation based on shape similarity . . . . .	92
3.5	Experimental Results . . . . .	94
3.5.1	Detection according to bounding boxes . . . . .	95
3.5.2	Localizing object boundaries . . . . .	98
<b>4</b>	<b>Particle Filter with State Permutations for Solving Image Jig- saw Puzzles</b>	<b>101</b>
4.1	Introduction and Problem Formulation . . . . .	101
4.2	Particle Filter Preliminaries . . . . .	105
4.3	Key Extension to Permuted States . . . . .	109
4.4	Particle Filter with State Permutations . . . . .	112
4.5	Implementation Details . . . . .	115
4.6	Experimental Results . . . . .	119
<b>5</b>	<b>Conclusion</b>	<b>124</b>
	<b>REFERENCES</b>	<b>128</b>

# LIST OF FIGURES

1.1	Existing shape similarity methods incorrectly rank shape (b) as more similar to (a) than (c). . . . .	3
2.1	A key idea of the proposed distance learning is to replace the original shape distance between (a) and (e) with a distance induced by geodesic paths in the manifold of know shapes. One such path is (a)-(e) in this figure. . . . .	11
2.2	The first column shows the query shape. The remaining 10 columns show the most similar shapes retrieved from the MPEG-7 data set. The first row shows the results of IDSC [63]. The second row shows the results of the proposed learned distance. . . . .	13
2.3	The pseudo-code for the proposed algorithm . . . . .	18
2.4	(a) A comparison of retrieval rates between IDSC [63] (blue circles) and the result improved by the proposed method (red stars) for MPEG-7. (b) A comparison of retrieval rates between visual parts in [57] (blue circles) and the result improved by the proposed method (red stars) for MPEG-7. (c) A comparison of retrieval rates between Gen. Model [101] (blue circles) and the result improved by the proposed method (red circles) for MPEG-7. . . . .	31
2.5	The first column shows the query shape. The remaining 10 columns show the most similar shapes retrieved by IDSC (odd row numbers) and by our method (even row numbers). . . . .	32
2.6	(a) The number of triangle inequality violations per iteration. (b) Plot of differences $  f_{t+1} - f_t  $ as a function of $t$ . . . . .	33
2.7	Sample shapes from Kimia's 99 dataset [86]. We show two shapes for each of the 9 classes. . . . .	33
2.8	A few sample image of the <i>Face (all)</i> data set. . . . .	34

2.9	(a) Conversion of the head profile to a curvature sequence. (b) Retrieval accuracy of DTW (blue circles) and the proposed method (red stars). . . . .	34
2.10	Typical images from the Swedish leaf database [92], one image per species. Note that some species are quite similar, e.g., the first, third and ninth species. . . . .	34
2.11	Retrieval accuracy of IDSC (blue circles) and the proposed method (red stars). . . . .	35
2.12	(c) The mean horse computed by averaging corresponding sample contour points of the aligned shapes in (a) and (b). . . . .	38
2.13	First row: the retrieval results of the mean horse from Fig. 2.12(c). Second row: the retrieval results of the ghost horse created by the averaging in distance space of the two shapes in Figs. 2.12(a) and (b). . . . .	38
2.14	The construction of $\rho(x, e)$ for $e = \mu(a, b)$ . . . . .	40
2.15	An example comparing the standard diffusion process (DM) to our method (LCDP). (a) is the plot of second most important eigenvector as a function of arc length. (b) shows the points color coded according to their second diffusion coordinate using DM. (c) and (d) show the same plots as (a) and (b) but using LCDP. . . . .	46
2.16	Comparison of our proposed approach to other methods using IDSC. . . . .	51
2.17	The gain in bull's-eye retrieval rates for each of the 70 shape classes of the MPEG-7 data set for IDSC [63] . . . . .	53
2.18	Retrieval curves of Swedish leaf data set . . . . .	54
2.19	First row: the query and the retrieval results with an original image similarity measure on subset of Caltech 101 dataset. Second row: the same query and the retrieval results after the proposed similarity learning. Third row: the query and the retrieval results with a shape similarity measure on the MPEG-7 shape dataset. Fourth row: the same query and the retrieval results with learned similarities. . . . .	58
2.20	First row: a classic $kNN$ for $k = 9$ of a dog. It contains two horses making it harder for any affinity learning algorithm to discriminate dogs from horses. Second row: the proposed dominant neighborhood (DN) obtained from $kNN$ in the first row. . . . .	59
2.21	(a) The blue stars show a classical $kNN$ of the point marked with the blue triangle for $k = 50$ . (b) The proposed, dominant neighborhood (DN) of the same point. It is obtained as a dominant subset form the $kNN$ in (a). . . . .	60

2.22	The example of a tensor product graph. The green circles are the vertexes and the lines are the edges. We do not show the self connections (loops) in the graphs, but each node has a loop.	62
2.23	Precision/Recall curves on MPEG-7 shape dataset. . . . .	74
2.24	Some images from Nister and Stewenius (N-S) dataset. . . . .	75
2.25	Some sample images from the selected subset of Caltech 101 dataset. Each class contains two examples. . . . .	78
3.1	Examples of two different inferred orders of detected contour parts. Colors represent the order, which is 1=red, 2=cyan, 3=blue, 4=green, 5=yellow, and 6=black. . . . .	82
3.2	(a) Six manually labeled parts on the horse in top left are marked with different colors. The point correspondence obtained by shape matching allows us to transfer the part structure to a different horse and to a giraffe. (b)The horse head and horse body shown on the left hand side are very different from our perception of a horse. Our measure of this fact is illustrated in the rest of this figure. . . . .	84
3.3	Precision-recall curve and detection rate (DR) vs false positive per image (FPPI) curve for the class Giraffes in ETHZ dataset.	96
3.4	Examples of detection results for Giraffes, horses and cows. . .	97
4.1	The goal is to build the original image (a) given the jigsaw puzzle pieces (b). The original image is not known, thus, it needs to be estimated given the observations shown in (b). The empty squares in (c) form possible locations for the puzzle pieces in (b). . . . .	102
4.2	First row: the original images. Second row: the jigsaw puzzle solutions of [16]. Third row: our solutions. . . . .	123
4.3	The reconstructed images of the best particle at different iterations. . . . .	123

# LIST OF TABLES

2.1	Retrieval rates (bull's eye) of different methods on the MPEG-7 data set. . . . .	24
2.2	Retrieval results on Kimia's 99 dataset [86] . . . . .	28
2.3	Retrieval rates (bull's-eye) of the MPEG-7 data set using Inner Distance Shape Context (IDSC). . . . .	52
2.4	Retrieval rates (bull's eye) of different context shape retrieval methods on the MPEG-7 shape dataset. . . . .	73
2.5	Retrieval results on N-S Dataset. The highest possible score is 4. . . . .	76
2.6	Retrieval rates on 12 image classes from Caltech-101. The best possible rate is 1. . . . .	78
3.1	Detection rate. . . . .	97
3.2	Accuracy of the boundary localization. . . . .	100
4.1	Experimental results on MIT Dataset. . . . .	122
4.2	Experimental results on the extended dataset. . . . .	122

# CHAPTER 1

## Introduction

### 1.1 Shape Retrieval

Shape matching/retrieval is a very critical problem in computer vision. There are many different kinds of shape matching methods, and the progress in improving the matching rate has been substantial in recent years. However, nearly all of these approaches are focused on pair-wise shape similarity measure. It seems to be an obvious statement that the more similar two shapes are, the smaller is their difference, which is measured by some distance function. Yet, this statement ignores the fact that some differences are more relevant while other differences are less relevant for shape similarity. It is not yet clear how the biological vision systems perform shape matching; it is clear though that shape matching involves the high-level understanding of

shapes. In particular, shapes in the same class can differ significantly because of in-class variation, distortion or non-rigid transformation. In other words, even if two shapes belong to the same class, the distance between them may be very large if the distance measure cannot capture the intrinsic property of the shape. It appears to us that many published shape distance measures [9, 101, 63, 57, 5, 62, 71, 28, 86, 90, 87, 38, 24, 10] are unable to address this issue. For example, based on the inner distance shape context (IDSC) [63], the shape in Fig. 1.1(a) is more similar to (b) than to (c), but it is obvious that shape (a) and (c) belong to the same class. This incorrect result is due to the fact that the inner distance is unaware that the missing tail and one front leg are less relevant than much smaller shape details like the dog’s ear and the shape of the head. No matter how good a shape matching algorithm is, the problem of more relevant and less relevant shape differences must be addressed if we want to obtain human-like performance. This requires having a model to capture the essence of a shape class instead of viewing each shape as a set of points, a parameterized function, or a manifold. In our method, each shape is considered in the context of other shapes in its class, and the class does not need to be known.

To utilize the context information, we first utilize the Label Propagation algorithm and reach excellent results [110]. Then, ghost points [111] are proposed to solve the problem of sparsity in the data manifold, which improves



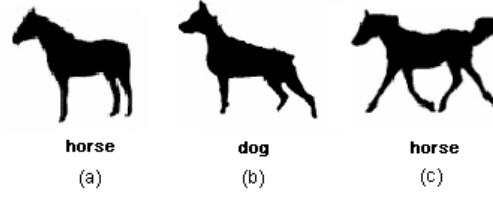


Figure 1.1: Existing shape similarity methods incorrectly rank shape (b) as more similar to (a) than (c).

the results a lot. Recently, the diffusion process on Tensor Product Graph is carried out for further improving the retrieval accuracy [114], which has also been tested on image retrieval.

### 1.1.1 Related Work

The semi-supervised learning problem has attracted an increasing amount of interest recently, and several novel approaches have been proposed. The existing approaches could be divided into several types, multiview learning [12], generative model [59], Transductive Support Vector Machine (TSVM) [49]. Recently there have been some promising graph based transductive learning approaches proposed, such as label propagation [123], Gaussian fields and harmonic functions (GFHF) [124], local and global consistency (LGC) [118], and the Linear Neighborhood Propagation (LNP) [105]. Zhou et al. [120] modified the LGC for the information retrieval. The semi-supervised learning problem is related to manifold learning approaches, e.g., [84].

The proposed method is inspired by the label propagation method [123].

The reason we choose the framework of label propagation is that it allows clamping of labels. In other words, it fixes the label of labeled data points during the propagation process. Since the query shape is the only labeled shape in the retrieval process, the label propagation allows us to enforce its label during each iteration, which naturally fits in the framework of shape retrieval. Usually, GFHF is used instead of label propagation, as both methods can achieve the same results[123]. However, in the shape retrieval, we can use only the label propagation, the reason is explained in detail in Section 2.1.2.

Since a large number of shape similarity methods have been proposed in the literature, we focus our attention on methods that reported retrieval results on the MPEG-7 shape data set (part B of the MPEG-7 Core Experiment CE-Shape-1) [58]. This allows us to clearly demonstrate the retrieval rate improvements obtained by the proposed method. Belongie et al. [9] introduced a novel 2D histograms representation of shapes called Shape Contexts (SC). Ling and Jacobs [63] modified the Shape Context by considering the geodesic distance between contour points instead of the Euclidean distance, which significantly improved the retrieval and classification of articulated shapes. Latecki and Lakämper [57] used visual parts represented by simplified polygons of contours for shape matching. Tu and Yuille [101] proposed the feature driven generative models for probabilistic shape matching. In order to avoid problems associated with purely global or local methods, Felzenszwalb and

Schwartz [28] described a dynamic and hierarchical curve matching method. Other hierarchical methods include the hierarchical graphical models in [25] and hierarchical procrustes matching [71]. Alajlan et al. proposed a mutiscale representation of triangle areas for shape matching, which also included partial and global shape information [2]. Daliri and Torre defined a symbolic descriptor based on Shape Contexts, then used edit distance for final matching in order to overcome the difficulty caused by deformation and occlusions [20]. The methods above all focused on designing improved shape descriptors for single shapes and their comparison for pairs of shapes. Although the recent methods made some progress, the improvement is not obvious as shown in Table 2.3 of Section 2.1.5. In this table, we summarize all the reported retrieval results on MPEG-7 database, and the retrieval rates of the recent publications are all around 85%. There are two main reasons that limit the progress in shape retrieval: (1) The case for large deformation and occlusions still can not be handled well. 2) The existing algorithms can not distinguish the more relevant and less relevant shape differences pointed out.

There has been a significant body of work on distance learning [116]. Xing et al. [109] propose estimating the matrix  $W$  of a Mahalanobis distance by solving a convex optimization problem. Bar-Hillel et al. [8] also use a weight matrix  $W$  to estimate the distance by relevant component analysis (RCA). Athitsos et al. [4] proposed a method called BoostMap to estimate a distance

that approximates a certain distance. Hertz’s work [42] uses AdaBoost to estimate a distance function in a product space, whereas the weak classifier minimizes an error in the original feature space. All these methods’ focus is a selection of suitable distance from a given set of distance measures. Our method aims at improving performance of a given distance measure.

## 1.2 Shape Based Object Detection

Object detection in cluttered images, with scale and intra-class variations, is one of the most difficult problems in computer vision. Appearance based methods have had remarkable success in recent years [61, 34, 89, 103]. However, in many cases, the appearance between intra-class objects varies a lot [3], which makes the appearance features not reliable. Thus, recently we have observed a significant increase in methods that utilize contour shape [30, 122, 94, 47, 67]. However, shape based methods also face many challenges, such as pose variance, missing edges, and view point changes. We propose a novel shape model learning algorithm for handling the articulation of objects and the particle filter framework is used for utilizing the model to detect objects [113].

### 1.2.1 Related Work

As there exists a lot of papers on shape based object detection and recognition, we only review the most related ones. Ferrari et al. [29] propose to use kAS, the k connected roughly straight contour segments, with Hough voting to detect objects. Later, Ferrari et al. [30] extend their work to learn the model from the image. To improve [30], Jiang et al. [47] propose to learn a shape prior model for each object class. Boundary fragments combined with classifier have also been investigated in [88]. Instead of object's contour, Trinh and Kimia [74] use skeleton-based generative shape model with modified dynamic programming to detect objects. Bai. et al. [6] also utilize skeleton to constrain the detection process. All the above methods require multiple initializations and they enumerate all possible object sizes (scales) to get the optimal results.

Ravishankar et al. [82] introduce a multi-stage contour based detection approach with dynamic programming, which is also scale independent. Zhu et al. [122] utilize Shape Context [9] to evaluate the distance between model and image segments. They formulate the shape matching of contours as a set-set matching problem and solve it by linear programming, which is fundamentally different from us. Lu et al. [67] formulate object detection as a segment correspondence problem. However, their inference framework is very different, where they utilize particle filter to solve the label assignment problem. Furthermore, they cannot detect multiple objects.

Gu et al. [40] utilize region segmentation to detect target objects. An appearance based approach was recently used by Maji and Malik [68] by integrating Hough transform based features of codebooks into kernel classifiers. To solve the problem of scales in Hough voting, Ommer and Malik [76] propose a weighted, pairwise clustering of voting lines to obtain globally consistent hypotheses. Then, a verification stage is use to re-rank the hypotheses. Unlike [76, 68, 40], we use a purely shape based method and do not utilize any classifiers like SVM to rank the hypotheses.

## 1.3 Particle Filter with State Permutations for Solving Image Jigsaw Puzzles

Particle filter nicely simulates the way human solving image jigsaw puzzles, where a starting puzzle is selected and others pieces are added one by one. Thus, we extend the particle filter framework so that it will permute different states and select the most possible ones during the process [112]. This algorithm performs well on solving image jigsaw puzzles.

### 1.3.1 Related Work

The first work on Jigsaw Puzzle Problem was reported in [32]. Since shape is an important clue for accurate pairwise relation, many methods [53, 35,

80, 108] focussed on matching distinct shapes among jigsaw pieces to solve the problem. The pairwise relations among jigsaw pieces are measured by the fitness of shapes. There also exist approaches that consider both the shape and image content [69, 75, 115]. Most methods solve the problem with a greedy algorithm and report results on just one or few images. Our problem formulation only considers the image content following Cho et. al [16].

Particle filters (PF) are also known as sequential Monte Carlo methods (SMC) for model estimation based on simulation. There is large number of articles published on PF and we refer to two excellent books [23, 65] for an overview. PF can be viewed as a powerful inference framework that is utilized in many applications. One of the leading examples is the progress in robot localization and mapping based on PF [100]. Classical examples of PF applications in computer vision are contour tracking [45] and object detection [44]. All these approaches utilize PF in the classical tracking/filtering scenario with a pre-defined order of states and observations. To our best knowledge, the proposed PF framework with state permutations is novel and has not been considered before by other authors.

# CHAPTER 2

## Shape Retrieval of Silhouettes

### 2.1 Learning Context Sensitive Shape Similarity by Graph Transduction

#### 2.1.1 Introduction

Given a database of shapes, a query shape, and a shape distance function, which does not need to be a metric, we learn a new distance function that is expressed by shortest paths on the manifold formed by the know shapes and the query shape. We can do this without explicitly learning this manifold. As we will demonstrate in our experimental results, the new learned distance function is able to incorporate the knowledge of intrinsic shape differences. It is learned in an unsupervised setting in the context of known shapes. For



example, if the database of known shapes contains shapes (a)-(e) in Fig. 2.1, then the new learned distance function will rank correctly the shape in Fig. 1.1(a) as more similar to (c) than to (b). The reason is that the new distance function will replace the original distance (a) to (c) in Fig. 1.1 with a distance induced by the shortest path between in (a) and (e) in Fig. 2.1.

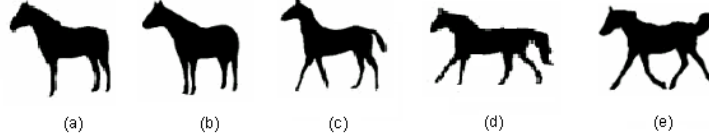


Figure 2.1: A key idea of the proposed distance learning is to replace the original shape distance between (a) and (e) with a distance induced by geodesic paths in the manifold of know shapes. One such path is (a)-(e) in this figure.

In the proposed approach, for a given similarity measure  $s_0$ , a new similarity  $s$  is learned through graph transduction. Intuitively, for a given query shape  $q$ , the similarity  $s(q, p)$  will be high if neighbors of  $p$  are also similar to  $q$ . However, even if  $s_0(q, p)$  is very high, but the neighbors of  $p$  are not similar to  $q$ , then  $s(q, p)$  will be low. Thus, the new similarity  $s$  is context sensitive, where a context of a given shape is defined by its neighbors, which are database shapes that are most similar to it. In this paper, we adopt a graph-based transductive learning algorithm to tackle this problem, and it has the following properties: (1) Instead of focusing on computing the distance (similarity) for a pair of shapes, we take advantage of the manifold formed by the existing shapes. (2) However, we do not explicitly learn the manifold nor

compute the geodesics [93], which are time consuming to calculate. A better similarity is learned by collectively propagating the similarity measures to the query shape and between the existing shapes through graph transduction.

(3) Unlike the label propagation [123] approach, which is semi-supervised, we treat shape retrieval as an unsupervised problem and do not require knowing any shape labels. (4) We can build our algorithm on top of any existing shape matching algorithm and a significant gain in retrieval rates can be observed on well-known shape datasets. (5) The learned distance by our algorithm can also be used to improve the performance of the existing shape clustering methods.

Even if the difference between shape  $A$  and shape  $C$  is large, but there is a shape  $B$  which has small difference to both of them, we still claim that shape  $A$  and shape  $C$  are similar to each other. This situation is possible for most shape distances, since they do not obey the triangle inequality, i.e., it is not true that  $d(A, C) \leq d(A, B) + d(B, C)$  for all shapes  $A, B, C$  [104]. If we have the situation that  $d(A, C) > d(A, B) + d(B, C)$  for some shapes  $A, B, C$ , then the proposed method is able to learn a new distance  $d'(A, C)$  such that  $d'(A, C) \leq d(A, B) + d(B, C)$ . Further, if there is a path in the distance space such that  $d(A, C) > d(A, B_1) + \dots + d(B_k, C)$ , then our method learns a new  $d'(A, C)$  such that  $d'(A, C) \leq d(A, B_1) + \dots + d(B_k, C)$ . Since this path represents a minimal distortion morphing of shape  $A$  to shape  $C$ , we are able to ignore less relevant shape differences, and consequently, we can

focus on more relevant shape differences with the new distance  $d'$ .

Our experimental results clearly demonstrate that the proposed method can improve the retrieval results of the existing shape matching methods. We obtained the retrieval rate of **91.61%** on part B of the MPEG-7 Core Experiment CE-Shape-1 data set [58], which is the highest ever bull’s eye score reported in the literature. We used the IDSC as our baseline algorithm, which has the retrieval rate of 85.40% on the MPEG-7 data set [63]. Fig. 2.2 illustrates the benefits of the proposed distance learning method. The first row shows the query shape followed by the first 10 shapes retrieved using IDSC only. Only two flies are retrieved among the first 10 shapes. The results of the learned distance for the same query are shown in the second row. All of the top 10 retrieval results are correct. The proposed method was able to learn that the shape differences in the number of fly legs and their shapes are not intrinsic to this shape class.

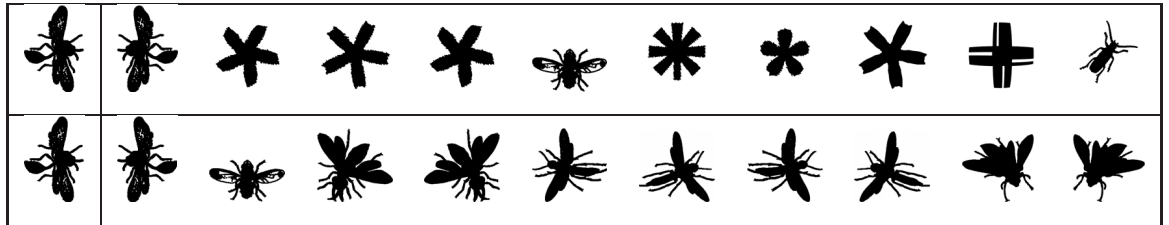


Figure 2.2: The first column shows the query shape. The remaining 10 columns show the most similar shapes retrieved from the MPEG-7 data set. The first row shows the results of IDSC [63]. The second row shows the results of the proposed learned distance.

### 2.1.2 Learning New Distance Measures

We first describe the classical setting of similarity retrieval. It applies to many retrieval scenarios like key word, document, image, and shape retrieval. Given is a set of objects  $X = \{x_1, \dots, x_n\}$  and a similarity function  $\text{sim}: X \times X \rightarrow R^+$  that assigns a similarity value (a positive value) to each pair of objects.

We assume that  $x_1$  is a query object (e.g., a query shape),  $\{x_2, \dots, x_n\}$  is a set of known database objects (or a training set). Then by sorting the values  $\text{sim}(x_1, x_i)$  in decreasing order for  $i = 2, \dots, n$  we obtain a ranking of database objects according to their similarity to the query, i.e., the most similar database object has the highest value and is listed first. Sometimes a distance measure is used in place of the similarity measure, in which case the ranking is obtained by sorting the database objects in the increasing order, i.e., the object with the smallest value is listed first. Usually, the first  $N \ll n$  objects are returned as the most similar to the query  $x_1$ .

As discussed above, the problem is that the similarity function  $\text{sim}$  is not perfect and for many pairs of objects it returns wrong results, although it may return correct scores for many pairs. We introduce now a method to learn a new similarity function  $\text{sim}_T$  that drastically improves the retrieval results of  $\text{sim}$  for the given query  $x_1$ .

Let  $w_{i,j} = \text{sim}(x_i, x_j)$ , for  $i, j = 1, \dots, n$ , be a similarity matrix, which is

also called an affinity matrix. We also define a  $n \times n$  probabilistic transition matrix  $P$  as a row-wise normalized matrix  $w$ .

$$P_{ij} = \frac{w_{ij}}{\sum_{k=1}^n w_{ik}} \quad (2.1)$$

where  $P_{ij}$  is the probability of transit from node  $i$  to node  $j$ .

We seek a new similarity measure  $s$ . Since  $s$  only needs to be defined as similarity of other elements to query  $x_1$ , we denote  $f(x_i) = s(x_1, x_i)$  for  $i = 1, \dots, n$ . A key function is  $f$  and it satisfies

$$f(x_i) = \sum_{j=1}^n P_{ij} f(x_j) \quad (2.2)$$

Thus, the similarity of  $x_i$  to the query  $x_1$ , expressed as  $f(x_i)$ , is a weighted average over all other database objects, where the weights sum to one and are proportional to the similarity of the other database objects to  $x_i$ . In other words we seek a function  $f : X \rightarrow [0, 1]$  such that  $f(x_i)$  is a weighted average of  $f(x_j)$ , where the weights are based on the original similarities  $w_{i,j} = \text{sim}(x_i, x_j)$ . Our intuition is that the new similarity  $f(x_i) = s(x_1, x_i)$  will be large iff all points  $x_j$  that are very similar to  $x_i$  (large  $\text{sim}(x_i, x_j)$ ) are also very similar to query  $x_1$  (large  $\text{sim}(x_1, x_j)$ ). Note that function  $f$  reaches equilibrium and an arbitrary function does not satisfy the equality.

The recursive equation (2.2) is closely related to PageRank. As stated in [77], a slightly simplified version of simple ranking  $R$  of a web page  $u$  in

PageRank is defined as

$$R(u) = \sum_{v \in B_u} \frac{c}{N_v} R(v), \quad (2.3)$$

where  $B_u$  is a set of pages that point to  $u$ ,  $N_v$  is the number of links from page  $v$  and  $c$  is a normalization factor.

Consequently, our equation (2.2) differs from PageRank equation (2.3) by the normalization matrix, which is defined in Eq. (2.1) in our case, and is equal to  $\frac{c}{N_v}$  for PageRank. The PageRank recursive equation takes a simple average over neighbors (a set of pages that point to a given web page), while we take a weighted average over the original input similarities. Therefore, our equation admits recursive solution analog to the solution of the PageRank equation. Before we present it, we point out one more relation to recently proposed label propagation [123].

We obtain the solution to Eq. (2.2) by the following recursive procedure:

$$f_{t+1}(x_i) = \sum_{j=1}^n P_{ij} f_t(x_j) \quad (2.4)$$

for  $i = 2, \dots, n$  and we set

$$f_{t+1}(x_1) = 1. \quad (2.5)$$

We define a sequence of newly learned similarity functions restricted to  $x_1$  as

$$sim_t(x_1, x_i) = f_t(x_i). \quad (2.6)$$

Thus, we interpret  $f_t$  as a set of normalized similarity values to the query  $x_1$ .

Observe that  $sim_1(x_1, x_i) = w_{1,i}$ .

The steps (2.4) and (2.5) are used in label propagation, which is described in Section 2.1.3. However, our goal and our setting are different. Although label propagation is an instance of semi-supervised learning, we stress that we remain in the unsupervised learning setting. In particular, we deal with the case of only one known class, which is the class of the query object. This means, in particular, that label propagation has a trivial solution in our case  $\lim_{t \rightarrow \infty} f_t(x_i) = 1$  for all  $i = 1, \dots, n$ , i.e., all objects will be assigned the class label of the query shape. Since our goal is ranking of the database objects according to their similarity to the query, we stop the computation after a suitable number of iterations  $t = T$ . As is the usual practice with iterative processes that are guaranteed to converge, the computation is halted if the difference  $\|f_{t+1} - f_t\|$  becomes very slow, see Section 2.1.5 for details.

If the database of known objects is large, the computation with all  $n$  objects may become impractical. Therefore, in practice, we construct the matrix  $w$  using only the first  $M < n$  most similar objects to the query  $x_1$  sorted according to the original distance function  $sim$ . Our experimental results in Section 2.1.5 demonstrate that the replacement of the original similarity measure  $sim$  with  $sim_T$  results in a significant increase in the retrieval rate. The pseudo-code of our algorithm is shown in Fig. 2.3.

**Input:** The  $n \times n$  row-wise normalized similarity matrix  $P$  with the query  $x_1$ ,  $f_1(x_1) = 1$ , and  $f_1(x_i) = 0$  for  $i = 2, \dots, n$ .

**while:**  $t < T$ .

**for**  $i = 2, \dots, n$ ,

$f_{t+1}(x_i) = \sum_{j=1}^n P_{ij} f_t(x_j)$

**end**

$f_{t+1}(x_1) = 1$ .

**end**

**Output:** The learned new similarity values to the query  $x_1$ :  $f_T$ .

Figure 2.3: The pseudo-code for the proposed algorithm

### 2.1.3 Relation to Label Propagation

Label propagation belongs to a set of semi-supervised learning methods, where it is usually assumed that class labels are known for a small set of data points. We have an extreme case of semi-supervised learning, since we only assume that the class label of the query is known. Thus, we have only one class that contains only one labeled element being the query  $x_1$ . In our approach, we have a sequence of labeling functions  $f_t : X \rightarrow [0, 1]$  with  $f_0(x_1) = 1$  and  $f_0(x_i) = 0$  for  $i = 2, \dots, n$ , where  $f_t(x_i)$  can be interpreted as probability that point  $x_i$  has the class label of the query  $x_1$ .

Label propagation is formulated as a form of propagation on a graph, where



node’s label propagates to neighboring nodes according to their proximity. The key idea is that its label propagates “faster” along a geodesic path on the manifold spanned by the set of known shapes than by direct connections. While following a geodesic path, the obtained new similarity measure learns to ignore less relevant shape differences. Therefore, when learning is complete, it is able to focus on more relevant shape differences. We review now the key steps of label propagation and relate them to the proposed method introduced in Section 2.1.2.

Let  $\{(x_1, y_1) \dots (x_l, y_l)\}$  be the labeled data,  $y \in \{1 \dots C\}$ , and  $\{x_{l+1} \dots x_{l+u}\}$  the unlabeled data, usually  $l \ll u$ . Let  $n = l + u$ . We will often use  $L$  and  $U$  to denote labeled and unlabeled data respectively. The Label propagation supposes the number of classes  $C$  is known, and all classes are present in the labeled data [123]. A graph is created where the nodes are all the data points, the edge between nodes  $i, j$  represents their similarity  $w_{i,j}$ . Larger edge weights allow labels to travel through more easily. Also define a  $l \times C$  label matrix  $Y_L$ , whose  $i$ th row is an indicator vector for  $y_i$ ,  $i \in L$ :  $Y_{ic} = \delta(y_{i,c})$ . The label propagation computes soft labels  $f$  for nodes, where  $f$  is a  $n \times C$  matrix whose rows can be interpreted as the probability distributions over labels. The initialization of  $f$  is not important. The label propagation algorithm is as follows:

1. Initially, set  $f_0(x_i) = y_i$  for  $i = 1, \dots, l$  and  $f_0(x_j)$  arbitrarily (e.g., 0) for

$$x_j \in X_u$$

Repeat until convergence:

2. set  $f_{t+1}(x_i) = \sum_{j=1}^n P_{ij} f_t(x_j)$ ,  $\forall x_i \in X_u$
3. set  $f_{t+1}(x_i) = y_i$  for  $i = 1, \dots, l$  (the labels of the labeled objects should be fixed).

In step 2, all nodes propagate their labels to their neighbors for one step. Step 3 is critical, since it ensures persistent label sources from labeled data. Hence instead of letting the initial labels fade way, we fix the labeled data. This constant push from labeled nodes, helps to push the class boundaries through high density regions so that they can settle in low density gaps. If this structure of data fits the classification goal, then the algorithm can use unlabeled data to improve learning.

Let  $f = \begin{pmatrix} f_L \\ f_U \end{pmatrix}$ . Since  $f_L$  is fixed to  $Y_L$ , we are solely interested in  $f_U$ . The matrix  $P$  is split into labeled and unlabeled sub-matrices

$$P = \begin{bmatrix} P_{LL} & P_{LU} \\ P_{UL} & P_{UU} \end{bmatrix} \quad (2.7)$$

As proven in [123] the label propagation converges, and the solution can be computed in closed form using matrix algebra:

$$f_U = (I - P_{UU})^{-1} P_{UL} Y_L \quad (2.8)$$

However, as the label propagation requires all classes be present in the labeled data, it is not suitable for shape retrieval. As mentioned in Section 2.1.2, for shape retrieval, the query shape is considered as the only labeled data and all other shapes are the unlabeled data. Moreover, the graph among all of the shapes is fully connected, which means the label could be propagated on the whole graph. If we iterate the label propagation infinite times, all of the data will have the same label, which is not our goal. Therefore, we stop the computation after a suitable number of iterations  $t = T$ .

#### 2.1.4 The Affinity Matrix

In this section, we address the problem of the construction of the affinity matrix  $W$ . There are some methods that address this issue, such as local scaling [117], local linear approximation [105], and adaptive kernel size selection [41].

However, in the case of shape similarity retrieval, a distance function is usually defined, e.g., [9, 63, 57, 28]. Let  $D = (D_{ij})$  be a distance matrix computed by some shape distance function. Our goal is to convert it to a similarity measure in order to construct an affinity matrix  $W$ . Usually, this can be done by using a Gaussian kernel:

$$w_{ij} = \exp\left(-\frac{D_{ij}^2}{\sigma_{ij}^2}\right) \quad (2.9)$$

Previous research has shown that the propagation results highly depend on

the kernel size  $\sigma_{ij}$  selection [105]. In [124], a method to learn the proper  $\sigma_{ij}$  for the kernel is introduced, which has excellent performance. However, it is not learnable in the case of few labeled data. In shape retrieval, since only the query shape has the label, the learning of  $\sigma_{ij}$  is not applicable. In our experiment, we use an adaptive kernel size based on the mean distance to K-nearest neighborhoods [106]:

$$\sigma_{ij} = \alpha \cdot \text{mean}(\{knnd(x_i), knnd(x_j)\}) \quad (2.10)$$

where  $\text{mean}(\{knnd(x_i), knnd(x_j)\})$  represents the mean distance of the K-nearest neighbor distance of the sample  $x_i, x_j$  and  $\alpha$  is an extra parameter. Both  $K$  and  $\alpha$  are determined empirically.

## 2.1.5 Experimental Results

### Improving MPEG-7 shape retrieval

The IDSC [63] significantly improved the performance of shape context [9] by replacing the Euclidean distance with shortest paths inside the shapes, and obtained the retrieval rate of 85.40% on the MPEG-7 data set. The proposed distance learning method is able to improve the IDSC retrieval rate to **91.61%**. For reference, Table 2.3 lists several reported results on the MPEG-7 data set. The MPEG-7 data set consists of 1400 silhouette images grouped into 70 classes. Each class has 20 different shapes. The retrieval rate is measured by

the so-called bull’s eye score. Every shape in the database is compared to all other shapes, and the number of shapes from the same class among the 40 most similar shapes is reported. The bull’s eye retrieval rate is the ratio of the total number of shapes from the same class to the highest possible number (which is  $20 \times 1400$ ). Thus, the best possible rate is 100%. From the retrieval rates collected in Table 2.3, we can clearly observe that our method made a significant progress on this database, and the second highest result is 87.70% obtained by Shape Tree [28].

In order to visualize the gain in retrieval rates by our method as compared to IDSC, we plot the percentage of correct results among the first  $k$  most similar shapes in Fig. 2.4(a), i.e., we plot the percentage of the shapes from the same class among the first  $k$ -nearest neighbors for  $k = 1, \dots, 40$ . Recall that each class has 20 shapes, which is why the curve increases for  $k > 20$ . We observe that the proposed method not only increases the bull’s eye score, but also the ranking of the shapes for all  $k = 1, \dots, 40$ .

We use the following parameters to construct the affinity matrix:  $\alpha = 0.25$  and the neighborhood size is  $K = 14$ . As stated in Section 2.1.2, in order to increase computational efficiency, it is possible to construct the affinity matrix for only part of the database of known shapes. Hence, for each query shape, we first retrieve 300 the most similar shapes, and construct the affinity matrix  $W$  for only those shapes, i.e.,  $W$  is of size  $300 \times 300$  as opposed to

Table 2.1: Retrieval rates (bull’s eye) of different methods on the MPEG-7 data set.

Alg.	CSS	Vis. Parts	Shape Contexts	Aligning Curves	Distance Set	Prob. Approach
	[73]	[57]	[9]	[85]	[39]	[70]
Score	75.44%	76.45%	76.51%	78.16%	78.38%	79.19%
Alg.	Inner	Symbolic	Hier.	Triangle	Shape	IDSC [63]
	Distance	Rep.	Procrustes	Area	Tree	+ our
	[63]	[20]	[71]	[2]	[28]	method
Score	85.40%	85.92%	86.35%	87.23%	87.70%	<b>91.61%</b>

a  $1400 \times 1400$  matrix if we consider all MPEG-7 shapes. Then we calculate the new similarity measure  $sim_T$  for only those 300 shapes. Here we assume that all relevant shapes will be among the 300 most similar shapes. Thus, by using a larger affinity matrix we could improve the retrieval rate but at the cost of computational efficiency. For each query, the average running time of our method on MPEG-7 is about 30 seconds in Matlab. For comparison the running time of the original IDSC is about one minute for each query.

In addition to the statistics presented in Fig. 2.4, Fig. 2.19 illustrates also that the proposed approach improves the performance of IDSC. A very interesting case is shown in the first row, where for IDSC only one result is correct for the query octopus. It instead retrieves nine apples as the most

similar shapes. Since the query shape of the octopus is occluded, IDSC ranks it as more similar to an apple than to the octopus. In addition, since IDSC is invariant to rotation, it confuses the tentacles with the apple stem. Even in the case of only one correct shape, the proposed method learns that the difference between the apple stem is very relevant, although the tentacles of the octopuses exhibit a significant variation in shape. We restate that this is possible because the new learned distances are induced by geodesic paths in the shape manifold spanned by the known shapes. Consequently, the learned distances retrieve nine correct shapes. The only wrong results is the elephant, where the nose and legs are similar to the tentacles of the octopus.

As shown in the third row, six of the top ten IDSC retrieval results of lizard are wrong. since IDSC cannot discover the more relevant differences between lizards and sea snakes. All retrieval results are correct for the new learned distances, since the proposed method is able to learn the less relevant differences between lizards and the more relevant differences between lizards and sea snakes. For the results of deer (fifth row), three of the top ten retrieval results of IDSC are horses. Compared to it, the proposed method (sixth row) eliminates all of the wrong results so that only deers are in the top ten results. It appears to us that our new method learned to ignore the less relevant small shape details of the antlers. Therefore, the presence of the antlers became a relevant shape feature here. The situation is similar for the bird and hat,

with three and four wrong retrieval results respectively for IDSC, which are eliminated by the proposed method.

An additional explanation of the learning mechanism of the proposed method is provided by examining the count of the number of violations of the triangle inequality that involve the query shape and the database shapes. In Fig. 7(a), the curve shows the number of triangle inequality violations after each iteration of our distance learning algorithm. The number of violations is reduced significantly after the first few hundred iterations. We cannot expect the number of violations to be reduced to zero, since cognitively motivated shape similarity may sometimes require triangle inequality violations [104]. Observe that the curve in Fig. 7(a) correlates with the plot of differences  $||f_{t+1} - f_t||$  as a function of  $t$  shown in (b). In particular, both curves decrease very slow after about 1000 iterations, and at 5000 iterations they are nearly constant. Therefore, we selected  $T = 5000$  as our stop condition. Since the situation is very similar in all our experiments, we always stop after  $T = 5000$  iterations.

Besides the inner distance shape context [63], we also demonstrate that the proposed approach can improve the performance of visual parts shape similarity [57] and feature driven generative model method [101]. We select these two methods since they are very different approach than IDSC. In [57], in order to compute the similarity between shapes, first the best possible correspondence of visual parts is established (without explicitly computing the visual parts).



Then, the similarity between corresponding parts is calculated and aggregated. The settings and parameters of our experiment are the same as for IDSC as reported in the previous section except we set  $\alpha = 0.4$ . The accuracy of this method has been increased from 76.45% to 86.69% on the MPEG-7 data set, which is more than 10%. This makes the improved visual part method one of the top scoring methods in Table 2.3. For feature driven generative model method [101], the accuracy has been increased from 80.03% to 89.29% when we set  $\alpha = 0.25$  and the other parameters are also the same as for IDSC. The detailed comparisons of the retrieval accuracy are given in Fig. 2.4(b) and Fig. 2.4(c) respectively.

Besides MPEG-7 dataset, we also present experimental results on the Kimia’s 99 dataset [86]. The dataset contains 99 shapes grouped into nine classes. In this dataset, some images have protrusions or missing parts. Fig. 2.7 shows two sample shapes for each class of this dataset. As the database only contains 99 shapes, we calculate the affinity matrix based on all of the shape in the database. The parameters used to calculate the affinity matrix are:  $\alpha = 0.25$  and the neighborhood size is  $K = 4$ . We changed the neighborhood size, since the data set is much smaller than the MPEG-7 data set. The retrieval results are summarized as the number of shapes from the same class among the first top 1 to 10 shapes (the best possible result for each of them is 99). Table 2.2 lists the numbers of correct matches of several methods. Again

Table 2.2: Retrieval results on Kimia’s 99 dataset [86]

Algorithm	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
SC [9]	97	91	88	85	84	77	75	66	56	37
Gen. Model [101]	99	97	99	98	96	96	94	83	75	48
Path Similarity [5]	99	99	99	99	96	97	95	93	89	73
Shock Edit [86]	99	99	99	98	98	97	96	95	93	82
IDSC [63]	99	99	99	98	98	97	97	98	94	79
Triangle Area [2]	99	99	99	98	98	97	98	95	93	80
Shape Tree [28]	99	99	99	99	99	99	99	97	93	86
Symbolic Rep. [20]	99	99	99	98	99	98	98	95	96	94
IDSC [63] + <b>our method</b>	99	99	99	99	99	99	99	99	97	99

we observe that our approach could improve IDSC significantly, and it yields a nearly perfect retrieval rate, which is the best result in the Table 2.2.

### Improving Face Retrieval

We used a face data set from [50], where it is called *Face (all)*. It addresses a face recognition problem based on the shape of head profiles. It contains several head profiles extracted from side view photos of 14 subjects. There exist large variations in the shape of the face profile of each subject, which is the main reason why we select this data set. Each subject is making different

face expressions, e.g., talking, yawning, smiling, frowning, laughing, etc. When the pictures of subjects were taken, they were also encouraged to look a little to the left or right, randomly. At least two subjects had glasses that they put on for half of their samples. A few sample pictures are shown in Fig. 2.8.

The head profiles are converted to sequences of curvature values, and normalized to the length of 131 points, starting from the neck area. The data set has two parts, training with 560 profiles and testing with 1690 profiles. The training set contains 40 profiles for each of the 14 classes. As reported on [50], we calculated the retrieval accuracy by matching the 1690 test shapes to the 560 training shapes. We used a dynamic time warping (DTW) algorithm with warping window [81] to generate the distance matrix, and obtained the 1NN retrieval accuracy of 88.9%. By applying our distance learning method we increased the 1NN retrieval accuracy to 95.04%. The best reported result in [50] has the first nearest neighbor (1NN) retrieval accuracy of 80.8%. The retrieval rate, which represents the percentage of the shapes from the same class (profiles of the same subject) among the first  $k$ -nearest neighbors, is shown in Fig. 2.9(b).

The accuracy of the proposed approach is stable, although the accuracy of DTW decreases significantly when  $k$  increases. In particular, our retrieval rate for  $k = 40$  remains high, 88.20%, while the DTW rate dropped to 60.18%. Thus, the learned distance allowed us to increase the retrieval rate by nearly

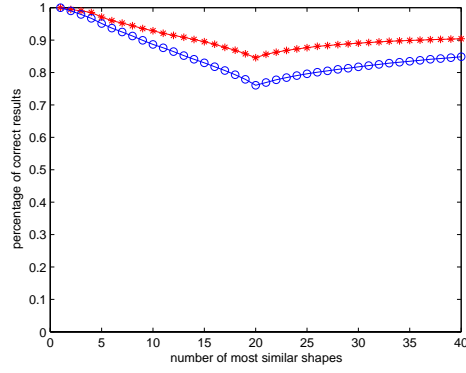
30%. Similar to the above experiments, the parameters for the affinity matrix is  $\alpha = 0.4$  and  $K = 5$ .

### Improving leaf retrieval

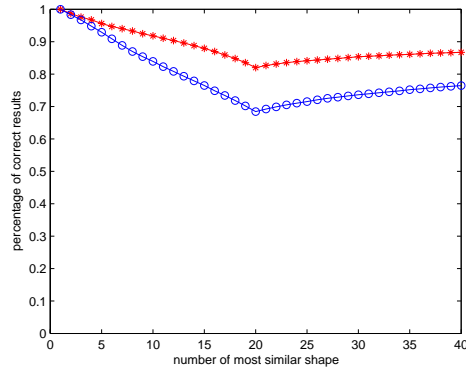
The Swedish leaf data set comes from a leaf classification project at Linköping University and Swedish Museum of Natural History [92]. Fig. 2.10 shows some representative examples. The data set contains isolated leaves from 15 different Swedish tree species, with 75 leaves per species. We followed the experimental setting for the Inner-Distance Shape Contexts used in [63], 25 leaves of each species are used for training, and the other 50 leaves are used for testing. The 1NN accuracy reported in [63] is 94.13%, but the result we obtained with their software<sup>1</sup> is 91.2%. As shown in Fig. 2.11, the retrieval rate of the Swedish leaf is improved significantly by the proposed approach, especially, the 1NN recognition rate is increased from 91.2% to 93.8%. The parameters for the affinity matrix are  $\alpha = 0.2$  and  $K = 5$ .

---

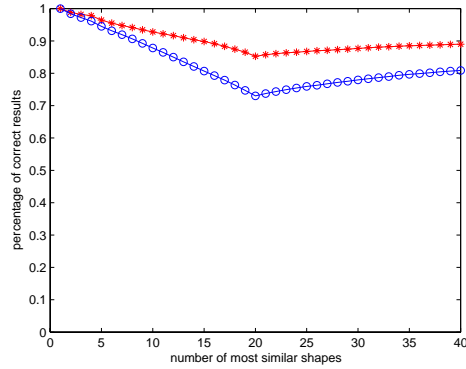
<sup>1</sup><http://vision.ucla.edu/~hbling/code>



(a)



(b)



(c)

Figure 2.4: (a) A comparison of retrieval rates between IDSC [63] (blue circles) and the result improved by the proposed method (red stars) for MPEG-7. (b) A comparison of retrieval rates between visual parts in [57] (blue circles) and the result improved by the proposed method (red stars) for MPEG-7. (c) A comparison of retrieval rates between Gen. Model [101] (blue circles) and the result improved by the proposed method (red circles) for MPEG-7.

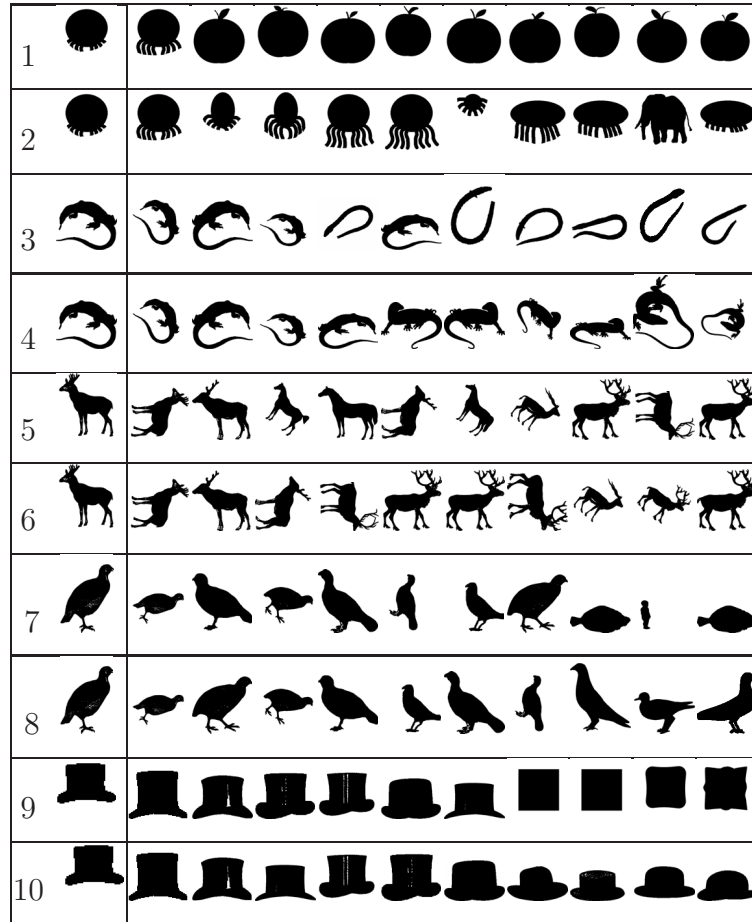
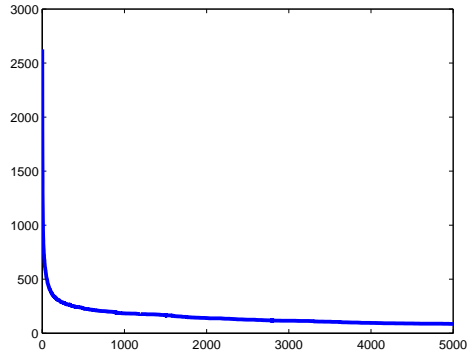
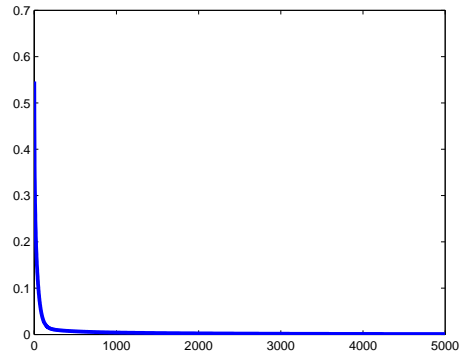


Figure 2.5: The first column shows the query shape. The remaining 10 columns show the most similar shapes retrieved by IDSC (odd row numbers) and by our method (even row numbers).



(a)



(b)

Figure 2.6: (a) The number of triangle inequality violations per iteration. (b) Plot of differences  $\|f_{t+1} - f_t\|$  as a function of  $t$ .



Figure 2.7: Sample shapes from Kimia's 99 dataset [86]. We show two shapes for each of the 9 classes.



Figure 2.8: A few sample image of the *Face (all)* data set.

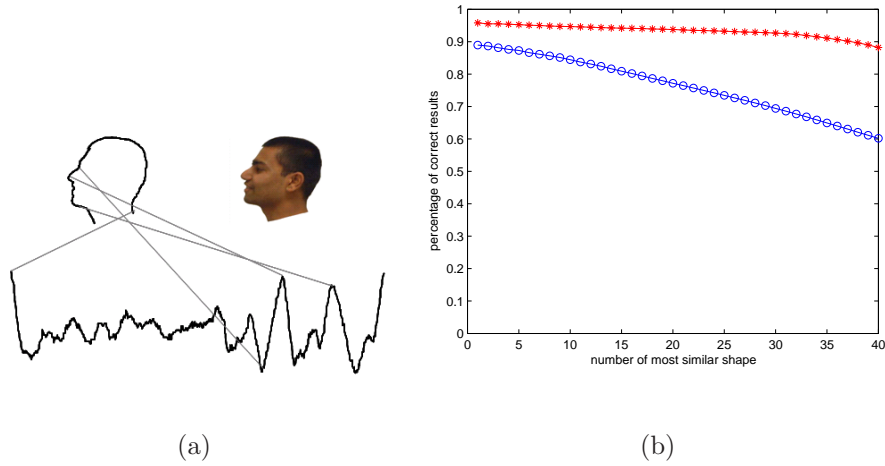


Figure 2.9: (a) Conversion of the head profile to a curvature sequence. (b) Retrieval accuracy of DTW (blue circles) and the proposed method (red stars).



Figure 2.10: Typical images from the Swedish leaf database [92], one image per species. Note that some species are quite similar, e.g., the first, third and ninth species.



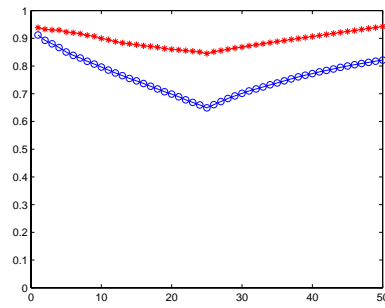


Figure 2.11: Retrieval accuracy of IDSC (blue circles) and the proposed method (red stars).

## 2.2 Densifying Shape Manifold with Ghost Point

### 2.2.1 Introduction

In our approach, the influence of other shapes is propagated as a diffusion process on a graph formed by a given set of shapes. However, as the shape space is sparse (see Sec. 2.2.2), in some cases the diffusion process can not propagate properly. It is obvious that adding more data points to the shape space would make the estimation of the data manifold more accurate. In other words, if the shape space is properly densified, a diffusion process is able to better reveal its underlying manifold structure. However, generating new data points is not always possible or may be costly. Therefore, we propose a new method to add synthetic data points to metric spaces. We introduce synthetic points with correct distances to the existing points. To the best of our knowledge, this is the first time researchers try to solve the problem of densifying the data manifold, and as our experimental results illustrate, the diffusion process performs significantly better on the densified manifold.

There have been several proposed approaches to adding synthetic examples, though these approaches try to solve the problem of balancing the number of examples in different classes, specifically over-sampling minority classes. For example, the SMOTE (Synthetic Minority Over-Sampling Technique) [14] algorithm and its variations [1, 79] have been found to be successful in clas-

sification problems but their methods are not suitable for shapes, since they work only in Euclidean space. In those methods, synthetic points are added as a weighted average of the Euclidean coordinates of two existing points. However, the Euclidean distance is known to be not suitable as a shape dissimilarity measure even if shapes are represented as vectors of their contour sample points. For example, the horse in Fig. 2.12(c) is computed as the average of the Euclidean coordinates of the two horses in Fig. 2.12(a) and Fig. 2.12(b). The Euclidean coordinates were obtained as sequences of 2D coordinates of 100 aligned contour sample points. Although the feature points of both horses correspond, it is difficult to recognize the shape in Fig. 2.12(c) as a horse. To demonstrate the problem, we submitted the average horse as a query to the MPEG-7 CE-Shape-1 part B data set [58]. The top ten retrieval results are shown in the first row of Fig. 2.13, ordered from left to right. Obviously none of the retrieval results is correct, but they are similar to the mean horse. For example, the tines of the forks are similar to the 'legs' of the average horse. The second row of Fig. 2.13 shows the retrieval results of the 'synthetic horse' generated by the proposed approach, which are all correct.

The second main idea in this paper is to replace the original diffusion process with a locally constrained diffusion process. As we will demonstrate in Section 2.2.4, it is significantly more robust to noise than the original diffusion process.

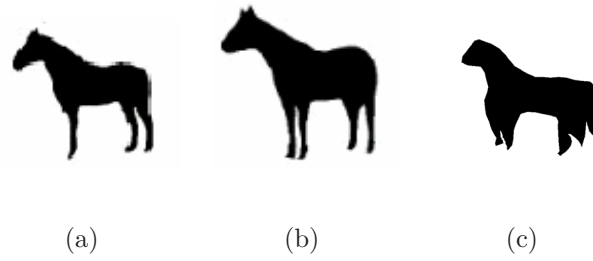


Figure 2.12: (c) The mean horse computed by averaging corresponding sample contour points of the aligned shapes in (a) and (b).

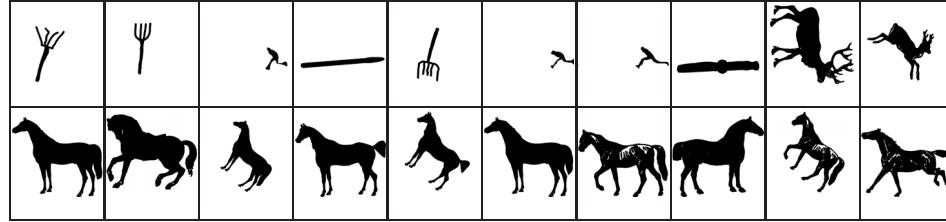


Figure 2.13: First row: the retrieval results of the mean horse from Fig. 2.12(c). Second row: the retrieval results of the ghost horse created by the averaging in distance space of the two shapes in Figs. 2.12(a) and (b).

### 2.2.2 Ghost points and metric embedding

In this paper, we view shape space as a set  $X$  and a distance function  $\rho : X \times X \rightarrow \mathfrak{R}$ , where  $\mathfrak{R}$  denotes real numbers. We require only that  $\rho(x, y) \geq 0$  for all  $(x, y) \in X \times X$  and  $\rho(x, y) = 0$  if  $x = y$ . Clearly, we would like  $\rho$  to be as close as possible to a metric, but this is not always possible, since there are clear arguments from human visual perception that the distance between shapes does not always satisfy the triangle inequality and the symmetry conditions. In any case, for theoretical reasons, we assume in Section 2.2.2 that  $\rho$  is a metric. However, as we will demonstrate in our experimental results, this assumption is not necessary for practical applications.

By embedding a metric space into a Euclidean space, we add new synthetic points to the shape space. We can do this so that the new points have correct distances to all existing points. Thus, the new points augment the shape space  $X$  but we cannot visualize them, which is the reason we call them ghost points.

### Metric embedding

The goal of metric embedding is to embed a metric space into a Euclidean space so that the distances between points are preserved. A distance preserving mapping between two metric spaces is called an isometry.

It is known that not every four point metric space can be isometrically embedded into a Euclidean space  $\mathbb{R}^k$ , e.g., see [48]. However, every three point metric space can be isometrically embedded into the plane  $\mathbb{R}^2$ . Let  $(\Delta, \rho)$ , where  $\Delta = \{x, a, b\} \subseteq X$ , be a metric space with three distinct points. Then it is easy to map  $\Delta$  to the vertices of a triangle on the plane. Let  $h : \Delta \rightarrow \mathbb{R}^2$  be the isometric embedding, which means that for any two points  $y, z \in \Delta$ ,  $\rho(y, z)^2 = \|y - z\|^2$ , where  $\|\cdot\|$  is the standard  $L_2$  norm that induces the Euclidean distance on the plane.

Let  $\mu(a, b)$  denote the mean of two points  $a, b$ . If  $a, b \in \mathbb{R}^2$ , then we have the usual formula  $\mu(a, b) = \frac{1}{2}(a + b)$  (see Fig. 2.14, where  $e = \mu(a, b)$ ).

Our first key contribution is the definition of  $\mu(a, b)$  for any two points  $a, b$  in a metric space  $X$ . To define  $\mu(a, b)$  in a metric space  $X$ , we need to

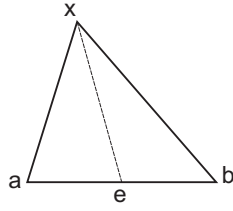


Figure 2.14: The construction of  $\rho(x, e)$  for  $e = \mu(a, b)$ .

specify  $\rho(x, \mu(a, b))$  for every  $x \in X$ . We first isometrically embed the three point metric subspace  $\Delta = \{x, a, b\} \subseteq X$  into the plane  $\mathfrak{R}^2$  by  $h$ . We define  $\mu(a, b) = h^{-1}(\frac{1}{2}(h(a) + h(b)))$ . Since  $h(\Delta)$  defines vertices of a triangle on the plane, we can easily derive that

$$\begin{aligned} & \|h(x) - \frac{h(a) + h(b)}{2}\|^2 = \\ & \frac{\|h(x) - h(a)\|^2}{2} + \frac{\|h(x) - h(b)\|^2}{2} - \frac{\|h(a) - h(b)\|^2}{4} \end{aligned}$$

Since  $h$  is an isometry and  $\mu(a, b) = h^{-1}(\frac{1}{2}(h(a) + h(b)))$ , we obtain (see Fig. 2.14)

$$\rho(x, \mu(a, b))^2 = \frac{1}{2}\rho(x, a)^2 + \frac{1}{2}\rho(x, b)^2 - \frac{1}{4}\rho(a, b)^2 \quad (2.11)$$

Consequently, Eq. 2.11 defines the distance of every point  $x \in X$  to the new point  $\mu(a, b)$ , which we call the mean of  $a$  and  $b$ . By computing the distances of  $\mu(a, b)$  to all points in  $X$ , we define a new point  $\mu(a, b)$ , and the augmented set  $X' = X \cup \{\mu(a, b)\}$  is also a distance space. We stress that to add a new point  $\mu(a, b)$  to  $X$  we do not need to compute the embedding  $h$ . We use  $h$  only to derive Eq. 2.11. Moreover, since the embedding  $h$  is an isometry, Eq. 2.11

defines correct distances from  $\mu(a, b)$  to all points in  $X$ . This fact is illustrated in the second row of Fig. 2.13, where we see the sorted 10 closest shapes to  $\mu(a, b)$  with  $a$  and  $b$  being the two shapes in Figs. 2.12(a) and (b). As shown in the first row of Fig. 2.13, simple averaging in Euclidean space may not produce correct distances, since the Euclidean distance is not adequate for shape similarity. We used Inner Distance Shape Context (IDSC) [63] as our shape distance function  $\rho$  in this example.

If the space  $X$  is finite, i.e.,  $X = \{x_1, \dots, x_n\}$ , then the distance function  $\rho : X \times X \rightarrow \mathbb{R}_{\geq 0}$  is represented by a square matrix  $M_\rho(X)$ . Each row of the square distance matrix  $M_\rho(X)$  is the distance of one shape  $x$  to all shapes in the data set, i.e., for all  $y \in X$ ,  $M_\rho(x, y) = \rho(x, y)$ . The matrix for  $X \cup \{\mu(a, b)\}$  is obtained by simply adding one row and one column to  $M_\rho(X)$ , with each entry computed using Eq. 2.11.

### Strategies for adding ghost points

There are many possible strategies for adding ghost points. Our strategy is very simple. We add to  $X = \{x_1, \dots, x_n\}$  a point  $\mu(x, NN_1(x))$  for each  $x \in X$ , where  $NN_1(x)$  is the first nearest neighbor of  $x$  different from  $x$ , i.e.,  $NN_1(x) = \operatorname{argmin}_{y \in S} (\rho(x, y))$  for  $S = X \setminus \{x\}$ . However, if  $y = NN_1(x)$  and  $x = NN_1(y)$ , this strategy would insert the same ghost point twice. Therefore, we need to take care to not add duplicate ghost points. After adding the ghost

points, we obtain a new shape space  $X'$ . As we will show in the experimental results, the augmented space  $X'$  densifies the original shape space  $X$  in such a way as to make the estimation of the data manifold more accurate.

This densification of space  $X$  is performed in the unsupervised setting, since we do not assume any knowledge of the class labels of points in  $X$ . To augment  $X$  in a supervised setting, we add ghost points to  $X$  as described above with the one exception that the first nearest neighbors are computed within the class of a given point, i.e., instead of  $S = X \setminus \{x\}$ , we define  $S = \{y \in X \mid \text{class}(y) = \text{class}(x) \text{ and } y \neq x\}$

We use the augmented shape space  $X'$  in the diffusion process (Sec. 2.2.3) to influence the shape similarity measures between the query shape and all other shapes. After the diffusion process is run, we exclude the ghost points and calculate our retrieval and classification rates based on only the original shape data set to allow for a fair comparison to existing methods.

### 2.2.3 Diffusion process

Given a set of data points  $X = \{x_1, \dots, x_n\}$ , we consider a fully connected graph  $G = (X, E)$ . The vertices of  $G$  are the data points and each edge  $E$  is labeled with the strength of the connection  $E(i, j) = k(x_i, x_j)$ , where  $k$  is a kernel function that is symmetric and positivity preserving. In this paper, given two shapes  $x_i$  and  $x_j$ ,  $k(x_i, x_j)$  is defined by applying a Gaussian to the



shape distance  $\rho(x_i, x_j)$ .

From the symmetric graph defined by  $(X, E)$ , one can construct a reversible Markov chain on  $X$ . This is a classic technique in many fields. The degree of each node is defined as

$$D(x_i) = \sum_{j=1}^n k(x_i, x_j)$$

and the transition probability is defined as

$$P(x_i, x_j) = \frac{k(x_i, x_j)}{D(x_i)}.$$

It is obvious that the transition matrix  $P$  inherits the positivity-preserving property, but it is no longer symmetric. However, we have gained a conservation property:

$$\sum_{j=1}^n P(x_i, x_j) = 1$$

From a data analysis point of view, the reason for studying this diffusion process is that the matrix  $P$  contains geometric information about the data set  $X$ . Indeed, the transitions that it defines directly reflect the local geometry defined by the immediate neighbors of each node in the graph of the data. In other words,  $P(x_i, x_j)$  represents the probability of transition in one time step from node  $x_i$  to node  $x_j$  and it is proportional to the edge-weight  $k(x_i, x_j)$ . For  $t \geq 0$ , the probability of transition from  $x_i$  to  $x_j$  in  $t$  time steps is given by  $P^t(x_i, x_j)$ , which is the  $t$ th power  $P^t$  of  $P$ . One of the main ideas of the diffusion framework is that the chain running forward in time, or equivalently, taking

larger powers of  $P$ , allows us to integrate the local geometry and therefore reveals relevant geometric structures of  $X$  at different scales, where  $t$  plays the role of a scale parameter. In [17], the data points can be embedded into Euclidean space by diffusion maps (DM), which can then reorganize the data points according to their geometric relation revealed by the diffusion process.

Ideally, diffusion coordinates generated by diffusion maps should reveal the intrinsic geometric structure of the underlying data manifold. However, as we illustrate by the following example, the diffusion process is still sensitive to noise. Our example illustrates that the diffusion process may fail to capture the correct topology if the actual topology of the data manifold is changed because of noise or outliers. Since noise and outliers can influence the distribution of data points, low density areas may become high density areas or vice versa, which will make the transition probability of the diffusion process incorrect. In Fig. 2.15, the samples are taken from a spiral as a function of arc length  $l$  with added Gaussian noise and a noise 'bridge' between inner and outer samples. Since the underlying manifold has a 1D structure, we would expect the diffusion process to be able to recover it when we use the coordinates of the second most important eigenvector, as described in [83, 98].

In Figs. 2.15(a) and (c), we plot the coordinates of the second most important eigenvector as a function of arc length (measured as point index). As can be clearly observed in Fig. 2.15(a), the function from arc length to the

second diffusion coordinate is not one-to-one, which means that the intrinsic 1D structure of the spiral has not been recovered by the standard diffusion process. Correspondingly, in Fig. 2.15(b), the order of points according to their second diffusion coordinate is color coded. Points with similar color have similar second diffusion coordinates. The fact that the 1D structure is not recovered is shown by the yellow colored points that are present in the bottom left as well as in the top right parts of the spiral. As shown in Figs. 2.15(c) and (d), the proposed locally constrained diffusion process (Sec. 2.2.4) is able to recover the 1D structure of the spiral. The graph in (c) does jitter a bit since we approximate the arc length coordinates of the spiral with the point index.

### 2.2.4 Locally Constrained Diffusion Process

As the diffusion process can be influenced even by moderate noise and outliers, in order to reduce the effect of noisy data points we introduce in this section a locally constrained diffusion process.

In the classical diffusion map setting, all paths between nodes  $x_i$  and  $x_j$  are considered when computing the probability of a walk from  $x_i$  to  $x_j$ . If there are several noisy nodes, the paths passing through these nodes will affect this probability as we demonstrated in Fig. 2.15.

A solution is introduced in [96] to solve this problem. It restricts the

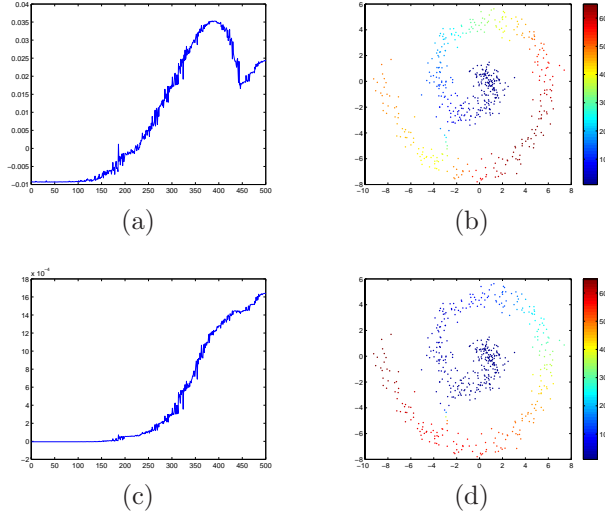


Figure 2.15: An example comparing the standard diffusion process (DM) to our method (LCDP). (a) is the plot of second most important eigenvector as a function of arc length. (b) shows the points color coded according to their second diffusion coordinate using DM. (c) and (d) show the same plots as (a) and (b) but using LCDP.

random walk to the  $K$  nearest neighbors of the data points by replacing the original graph  $G$  with a  $K$  nearest neighbor (KNN) graph  $G_K$  that has the edge weights defined as follows:  $E_K(i, j) = k(x_i, x_j)$  if  $x_j$  belongs to the KNNs of  $x_i$  and  $E_K(i, j) = 0$  otherwise. Then, the one-step transition probabilities  $P_K(x_i, x_j)$  from  $x_i$  to  $x_j$  are defined

$$P_K(x_i, x_j) = \frac{E_K(i, j)}{\sum_j E_K(i, j)}.$$

Through replacing the  $P$  in Section 2.2.3 by  $P_K$ , the effect of noise may be reduced, but it is still not robust enough to noise. The relation between the  $KNN(x_i)$  and  $KNN(x_j)$  is too hard and too narrow. It counts a data point  $x_k$  only if  $x_k$  is a KNN of both  $x_i$  and  $x_j$ . This causes problems if both points

$x_i$  and  $x_j$  belong to the same dense cluster, in which case they may have no common KNNs although they are very similar. In other words, although  $x_i$  and  $x_j$  are very similar to each other and there are many short paths connecting them in graph  $G$ , they may have no common neighbor in  $G_K$ .

In order to solve this problem, we consider the paths between KNNs of  $x_i$  and KNNs of  $x_j$ , which can be viewed as a soft measure of their KNNs' compatibility. The probability of transition from node  $x_i$  to  $x_j$  is high if all the paths between points in  $KNN(x_i)$  and in  $KNN(x_j)$  are short. We define

$$P_{KK}^{t+1}(x_i, x_j) = \sum_{k \in KNN(x_i), l \in KNN(x_j)} P(x_i, x_k) P_{KK}^t(x_k, x_l) P(x_l, x_j) \quad (2.12)$$

Eq. 2.12 can be viewed as a symmetric version of the approach in [96]. In addition, it can be expressed as matrix multiplication

$$P_{KK}^{t+1} = P_K P_{KK}^t (P_K)^T.$$

The embedding results of our proposed approach on the noisy spiral data are shown in Figs. 2.15(c) and (d). These figures demonstrate that the proposed locally constrained diffusion process (LCDP) is able to recover the intrinsic geometric structure of the spiral. The number of nearest neighbors  $K$  is very crucial. If it is too large, the effect of noise cannot be reduced efficiently; if it is too small, it will correspond to the most likely transition probability,

which is also easily affected by the 'noise bridge'.

### 2.2.5 Experimental results

In this section, we demonstrate the validity of our approach for shape retrieval on two standard data sets, MPEG-7 and Swedish Leaf. We compare the Locally Constrained Diffusion Process (LCDP) to three closely related methods: diffusion process based on Locally Appropriate Metric (LAM) [96]; diffusion distances after embedding by Diffusion Maps (DM) [55]; and the Label Propagation (LP) approach in [110]. We show also the positive effect of adding ghost points in both unsupervised and supervised settings.

In all of the following experiments, the  $\sigma$  for the Gaussian Kernel function follows the approach in [110]. The number of  $K$  nearest neighbors is 20 for the MPEG-7 data set and 40 for the Swedish Leaf data set. The number of iterations of the diffusion process,  $t$ , is set empirically.

#### MPEG-7 data set

First we show the experimental results on the MPEG-7 CE-Shape-1 part B data set [58]. MPEG-7 is a standard data set and is widely used to test shape classification and retrieval methods. It contains 1400 binary images divided into 70 shape classes of 20 images each. Every shape in the data set is compared to all other shapes, and the number of shapes from the same class

among the 40 most similar shapes is reported. The bull’s-eye retrieval rate is the ratio of the total number of shapes from the same class to the highest number possible (which is  $1400 \times 20$ ), thus the best possible score is 100%. To show that the proposed approach can improve shape retrieval results on existing shape distance measures, we choose the well-known shape similarity method, Inner Distance Shape Context (IDSC) [63], to compute the pairwise distances between the shapes. The bull’s-eye scores of the proposed approaches and the other approaches using IDSC are shown in Table 2.3, and the retrieval scores (the ratio of the number of correct shapes among the first  $k$  shapes for  $k = 1, \dots, 40$ ) is shown in Fig. 2.16. The lowest overall retrieval results of DM illustrate the fact that embedding the shape space into low dimensional Euclidean space may lead to significant loss of information. This is the only method that performs worse than the original IDSC pairwise distance measure.

Although the accuracy of LAM is higher than IDSC, it is still significantly lower than the proposed LCDP. Even without ghost points, LCDP increases the bull’s-eye score to 92.36%, which is better than the highest previously reported bull’s-eye score of 91.00% in [110] and demonstrates that our method does reduce the effect of noise and outlier shapes. By adding ghost points in an unsupervised setting, the bull’s-eye score reaches 93.31%, the highest ever reported. It is consistent with our assumption that the ghost points densify the data space, which improves the performance of the diffusion process.

In Fig. 2.17, we report the percentage gain for each of the 70 shape classes in the MPEG-7 data set obtained by LCDP with unsupervised ghost points when compared to IDSC. We observe that the bull’s-eye retrieval rate was improved by over 20% on 9 shape classes. This demonstrates the ability of the proposed approach to learn object appearance in the context of other shapes. But as learning involves generalization, there is always a danger of overgeneralization. Yet this graph demonstrates that this danger is very small for the proposed approach since the bull’s-eye score of only one class decreased significantly. Furthermore, this decrease in accuracy can be explained by the fact that this class contains shapes of spoons which are very similar to sea-snakes, pencils, and keys in the MPEG-7 data set.

From the graph in Fig. 2.16, it is clear that the retrieval rate when using the unsupervised ghost points is not always better than the other approaches. For the early nearest neighbors, i.e., when  $k$  is small, it is worse than the other methods because in the unsupervised setting we assume that the local structure of each data point is correct; that is, that the first nearest neighbor of each of the data points should be from the same class as the data point itself. However, since IDSC can not attain 100% accuracy when finding the first nearest neighbors, a few inter-class ghost points will be generated. This reduces the accuracy of the retrieval rates for small  $k$ . However, since most of the ghost points generated are intra-class (and this is what we want), the



retrieval rates for later  $k$  improves significantly, and the bull’s-eye score reaches 93.31% for  $k = 40$ . To solve this problem of generating inter-class ghost points, we also generate ghost points in a supervised setting. With supervision, only intra-class ghost points are created and this gives us the bull’s-eye score of 97.21%. Furthermore, the retrieval curve is always above the curves of the other approaches. Hence we can conclude that the performance gain in the retrieval rates is optimal when the shape space is densified in a supervised setting. We want to stress that this scenario is realistic, since we usually know the class labels of the database objects.

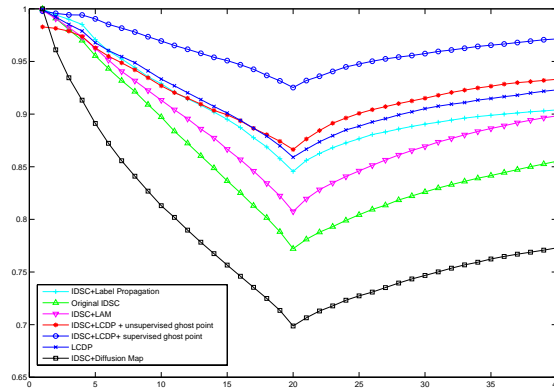


Figure 2.16: Comparison of our proposed approach to other methods using IDSC.

### Swedish Leaf data set

We also test our approach on the Swedish leaf data set, which comes from a leaf classification project at Linköping University and the Swedish Museum of

Table 2.3: Retrieval rates (bull’s-eye) of the MPEG-7 data set using Inner Distance Shape Context (IDSC).

Alg.	IDSC [63]	IDSC + LAM	IDSC +DM	IDSC +LP[110]	IDSC +LCDP	IDSC +LCDP +unsupervised GP	IDSC +LCDP +supervised GP
Score	85.40%	89.00%	78.56%	91.00%	<b>92.36%</b>	<b>93.31%</b>	<b>97.21%</b>

Natural History [92]. The data set contains images of leaves from 15 different Swedish tree species, with 75 leaves per species, for a total of 1125 images. Previous work focused on 1-nearest-neighbor (1NN) classification [63, 92]. In this paper, in addition to our results for 1NN classification, we also show the retrieval results as the ratio of correct shapes among the first  $k$  shapes for  $k = 1, \dots, 75$ . Again, we use IDSC to find the distances between the shapes of the data set and we compare our approach to two of the three methods discussed above (there are no reported results on this data set for LP [110]). The results for the different methods are shown in Fig. 2.18. Once again, the retrieval results for DM are significantly worse than the other approaches. Although LAM’s performance is quite good, it is still worse than LCDP. LCDP without ghost points improves the retrieval results significantly with the 1NN classification rate increasing from 94.12%[63] to 98.2%, the highest score in the literature.

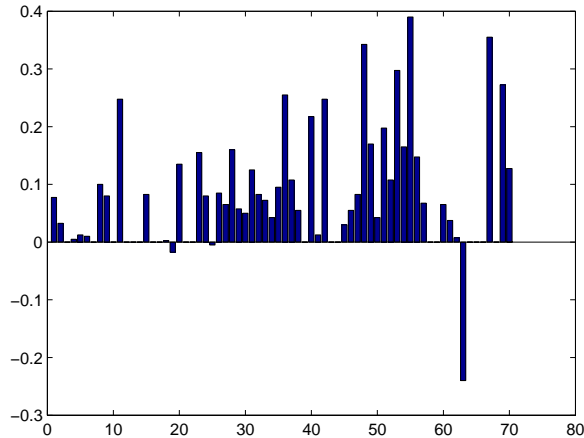


Figure 2.17: The gain in bull’s-eye retrieval rates for each of the 70 shape classes of the MPEG-7 data set for IDSC [63]

Consistent with the results obtained on the MPEG-7 data set, the addition of unsupervised ghost points does not improve the retrieval rates of LCDP for small  $k$ , but does improve them for larger  $k$ . Adding ghost points in the supervised setting achieves the best performance of all. The 1NN classification rates are 97.6% and 99.3% for unsupervised and supervised ghost points respectively.

The reason for the difference between the results for unsupervised and supervised ghost points is that the data set contains several classes that are very similar to each other and thus some of the ghost points added in the unsupervised setting are inter-class and we have the same problem as we discuss in Sec. 2.2.5. The addition of ghost points in the supervised setting solves this problem by generating only intra-class ghost points, and the retrieval rate

increases significantly.

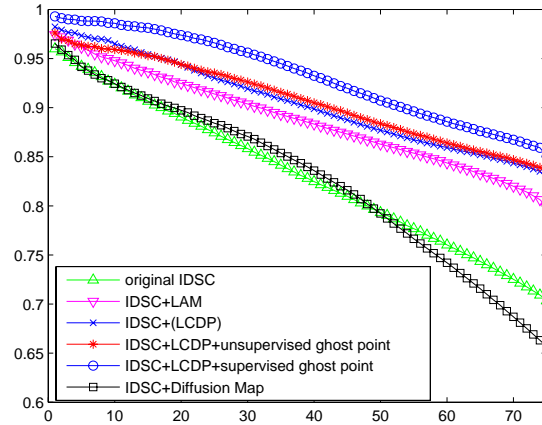


Figure 2.18: Retrieval curves of Swedish leaf data set

We can make some conclusions based on the experimental results. First, LCDP performs better than LP and LAM, which is consistent with the discussion in Sec. 2.2.4. Second, the effect of adding unsupervised ghost points depends greatly on the accuracy of the original shape similarity measure. If the assumption of local structure of the data points can be satisfied, adding unsupervised ghost points can achieve an enormous improvement. Conversely, if we can not assume local structure of the data points, adding unsupervised ghost points may actually cause the  $k$  nearest neighbor retrieval rate to decline for small  $k$  (though there still may be a substantial improvement for large  $k$ ). Third, the ghost points generated in a supervised setting consistently and significantly improve the retrieval rates for all  $k$ . Although it is not

fair to compare adding supervised ghost points to unsupervised ghost points in shape retrieval, the excellent performance shows the possible application of ghost points in other areas of supervised and semi-supervised data mining.

## **2.3 Affinity Learning on a Tensor Product Graph with Applications to Shape and Image Retrieval**

### **2.3.1 Introduction**

Image and shape retrieval belong to central topics in computer vision. Similar to other ranking/retrieval tasks, once a query object is given, the goal is to retrieve the most similar objects in the database. Traditionally, the performance of the retrieval is decided by the similarity/dissimilarity measure, which separately compares the query to each database object. However, these pairwise comparisons ignore the structure of the data manifold determined by similarities between the database objects. As shown by a sequence of recent papers, [120, 111, 54, 7], considering the data manifold structure significantly improves the performance of ranking/retrieval. The basic idea is inspired by the success of google PageRank ranking. The data manifold is represented as a graph with edge weights determined by the initial pairwise similarity values.

Then the pairwise similarities between the query and each database object are reevaluated in the context of other database objects, where the context of each object is a set of other objects most similar to it and the reevaluation is obtained by propagating the similarity information following structure of the weighted edge links in the graph. The reevaluation is closely related to random walks on the graph, e.g., [96, 118].

Compared to the existing methods, our approach differs in two main aspects. First, instead of propagating the similarity information on the original graph, we propose to utilize the tensor product graph (TPG) obtained by the tensor product of the original graph with itself. Since TPG takes into account a higher order information compared to the original methods, it comes at no surprise that we obtain better retrieval performance. Higher order information has been utilized in many applications before, e.g., [43, 119], but it comes at the price of higher order computational complexity and storage requirement. The key feature of the proposed approach is that the information propagation on can be computed with the same computational complexity and the same amount of storage as the propagation on the original graph. We utilize a graph diffusion process to propagate the similarity information on TPG, but we never compute it directly on TPG. Instead, we derive a novel iterative algorithm to compute it directly on the original graph, which is guaranteed to converge. After its convergence we obtain new edge weights that can be interpreted as

new, learnt similarities. They are then used for final retrieval ranking.

Fig. 2.19 compares the retrieval results of the learnt similarities to those of original similarities. The queries are shown in the first column. The first row shows retrieval results of an original image similarity measure on a subset of Caltech 101 dataset. The second row shows the retrieval results after learning the similarities. The third row shows retrieval results of an original shape similarity measure [64] on the MPEG-7 dataset. The results after learning the similarities with the proposed method are shown in the fourth row. As can be seen the proposed similarity learning is able to correct wrong retrieval results of the original similarities.

Second, it has been noticed that if the pairwise similarities are not accurate, the full graph contains too much noise, which hurts the affinity propagation [55, 96]. Thus, it is natural to constrain the relation from each element to its neighbors [96]. This significantly reduces the amount of noisy pairwise similarities, since the pairwise similarities are more accurate for close neighbors. A common practice to achieve this is to keep only the edge weights of  $k$  nearest neighbors ( $kNN$ ) of each object and zero out the other edge weights, i.e., remove the corresponding edges. However, the selection of  $kNN$  is also easily influenced by errors in the pairwise similarities and a "good" number of nearest neighbors  $k$  may be different for different objects. To better define the neighbors of a point, we propose a novel way to construct the neighborhood

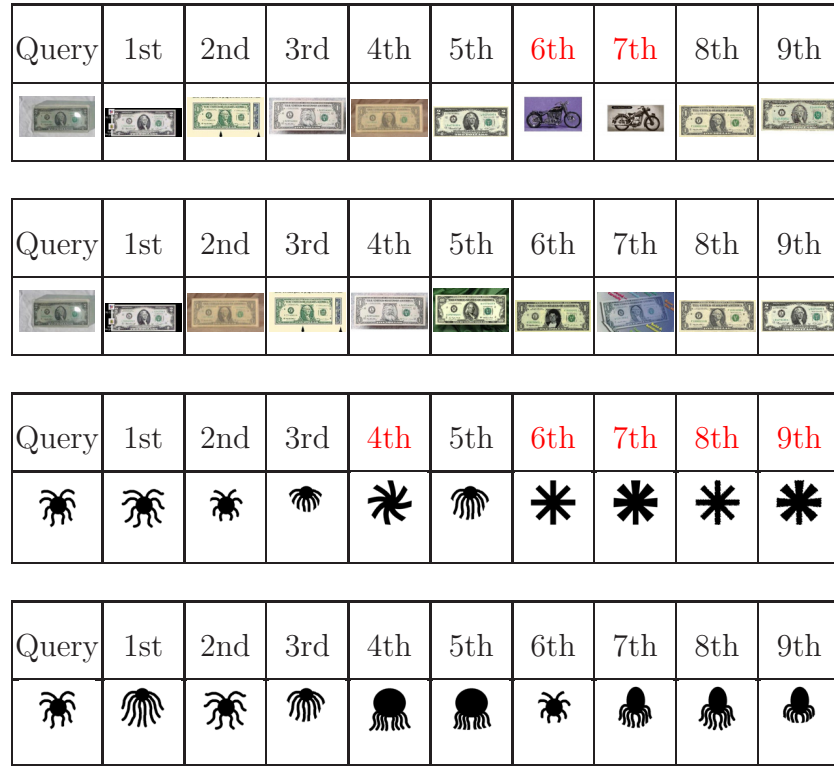












Figure 2.19: First row: the query and the retrieval results with an original image similarity measure on subset of Caltech 101 dataset. Second row: the same query and the retrieval results after the proposed similarity learning. Third row: the query and the retrieval results with a shape similarity measure on the MPEG-7 shape dataset. Fourth row: the same query and the retrieval results with learned similarities.

structure, which is called Dominant Neighborhood (DN). Like a dominant set defined in [78], DN considers the affinities among the neighbors to determine the best neighborhood structure, which makes it more robust to errors and outliers in pairwise similarities. Another advantage of DN is that it automatically determines the optimal number of neighbors. This solves one of the serious problems of  $kNN$ . If  $k$  is too large for a given point, its  $kNN$  includes points that are not its true neighbors. This fact is illustrated for binary shapes



Query	1st	2nd	3rd	4th	5th	6th	7th	8th	9th
									









Query	1st	2nd	3rd	4th	5th	6th	7th
							

Figure 2.20: First row: a classic  $kNN$  for  $k = 9$  of a dog. It contains two horses making it harder for any affinity learning algorithm to discriminate dogs from horses. Second row: the proposed dominant neighborhood (DN) obtained from  $kNN$  in the first row.

in Fig. 2.3.1. The fact that  $kNN$  for  $k = 9$  of the dog contains two horses (first row), may cause any affinity learning algorithm to lose the distinction between dogs and horses. The  $DN$  of the dog in the second row correctly contains only dogs, making it easier to learn the distinction between dogs and horses. To illustrate the problem of  $kNN$  from the point of view of a data manifold, we show a toy example in Fig. 2.21(a). The point marked with a triangle incorrectly contains points of two classes in its  $kNN$  for  $k = 50$ . As shown in Fig. 2.21(b), the dominant neighborhood of this point only contains points from its class.

In §2.3.2, the distance learning algorithm on TPG is introduced in details. The construction of DN is described in §2.3.5. The experimental results are given in §2.3.6.

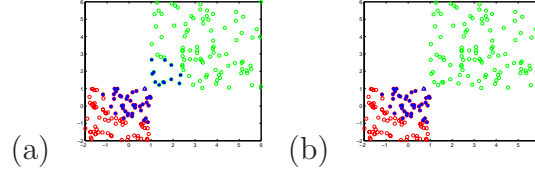


Figure 2.21: (a) The blue stars show a classical  $k$ NN of the point marked with the blue triangle for  $k = 50$ . (b) The proposed, dominant neighborhood (DN) of the same point. It is obtained as a dominant subset form the  $k$ NN in (a).

### 2.3.2 Affinity Learning

In this section we describe a novel context-sensitive affinity learning algorithm. It is introduced as a diffusion process on a Tensor Product Graph (TPG). However, the size of TPG is quadratic as compared to the original graph, which makes the diffusion on the TPG impractical on large datasets due to both high computation time and high memory requirement. To solve this problem, we propose a novel iterative algorithm on the original graph (Section 2.3.4), and prove that it is equivalent to the diffusion process on TPG. Consequently, both time complexity and memory requirements of the iterative algorithm are comparable to other affinity learning methods like diffusion on the original graph [96] or LGC[120, 118].

In the paper, the data is represented as an edge-weighted graph  $G = (V, A)$ , where  $V = \{v_1, \dots, v_n\}$  is the set of vertices representing the data points and  $A$  is the graph adjacency matrix  $A(i, j) = (a_{ij})$  for  $i, j = 1, \dots, n$ , where  $a_{ij}$  presents the edge weight from  $v_i$  to  $v_j$ . We assume that  $A$  is nonnegative

and the sum of each row is smaller than one. A matrix  $A$  that satisfies these requirements can be easily created from a stochastic matrix (see Section 2.3.5).

It is well known that a graph diffusion process is able to reveal the intrinsic relation between objects [17, 96]. Probably the simplest realization of a diffusion process on a graph is by computing powers of the graph matrix, i.e., the edge weights at time  $t$  are given by  $A^t$ . Usually, the time is discrete and  $t$  corresponds to the iteration number. However, this process is sensitive to the number of iterations [55]. For example, if the sum of each row of  $A$  is smaller than one, as we assumed, then it converges to zero matrix, in which case determining a right stopping time  $t$  is critical. In order to make the graph diffusion process independent from the number of iteration, accumulation between different numbers of iterations is widely used [55]. Following this strategy, we consider the graph diffusion process defined as

$$A^{(t)} = \sum_{i=0}^t A^i \quad (2.13)$$

Our assumption that the sum of each row of  $A < 1$  is equivalent to the fact that the the maximum of the row-wise sums of matrix  $A < 1$ . It is known that the maximum of the absolute values of the eigenvalues is bounded by the the maximum of the row-wise sums. Therefore, we obtain that the maximum of the absolute values of the eigenvalues of  $A$  is smaller than one. Consequently, (2.13) converges to a fixed and nontrivial solution given by  $\lim_{t \rightarrow \infty} A^{(t)} = (I - A)^{-1}$ , where  $I$  is the identify matrix.

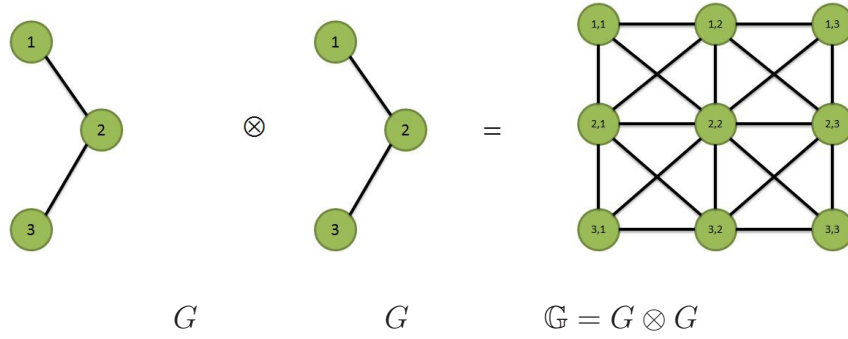


Figure 2.22: The example of a tensor product graph. The green circles are the vertices and the lines are the edges. We do not show the self connections (loops) in the graphs, but each node has a loop.

### 2.3.3 Diffusion Process on Tensor Product Graph

The Tensor Product Graph (TPG)  $\mathbb{G} = G \otimes G$  is defined as  $\mathbb{G} = (V \times V, \mathbb{A})$ . Thus, each vertex of  $\mathbb{G}$  is a pair of vertices in  $G$ , and consequently, it is indexed with a pair of indices. The adjacency matrix of  $\mathbb{G}$  is defined as  $\mathbb{A} = A \otimes A$ , where  $\otimes$  is the Kronecker product [56, 102]. In particular, for  $\alpha, \beta, i, j = 1 \dots, n$  we have

$$\mathbb{A}(\alpha, \beta, i, j) = A(\alpha, \beta) \cdot A(i, j) = a_{\alpha, \beta} \cdot a_{i, j}.$$

Thus, if  $A \in \mathbb{R}^{n \times n}$ , then  $\mathbb{A} = A \otimes A \in \mathbb{R}^{nn \times nn}$ . An example is shown in Fig. 2.22.

We define the **diffusion process on TPG** as

$$\mathbb{A}^{(t)} = \sum_{i=1}^t \mathbb{A}^i. \quad (2.14)$$

Since the edge weights of TPG relate 4 tuples of original vertices,  $\mathbb{G}$  contains high order information than the input graph. The higher order information is

helpful for revealing the intrinsic relation between objects, which is obtained by the diffusion process on TPG.

As is the case for (2.13), the process (2.14) also converges to a fixed and nontrivial solution

$$\lim_{t \rightarrow \infty} \mathbb{A}^{(t)} = \lim_{t \rightarrow \infty} \sum_{i=1}^t \mathbb{A}^i = (I - \mathbb{A})^{-1}. \quad (2.15)$$

To show this, we only need to show that the sum of each row of  $\mathbb{A}$  is smaller than 1, i.e.,  $\sum_{\beta, j} \mathbb{A}(\alpha\beta, ij) < 1$ , where  $\beta, j$  both range from 1 to  $n$ . This holds, since

$$\sum_{\beta j} \mathbb{A}(\alpha\beta, ij) = \sum_{\beta j} a_{\alpha\beta} a_{ij} = \sum_{\beta} a_{\alpha\beta} \sum_j a_{ij} < 1. \quad (2.16)$$

Consequently, (2.15) provides a closed form solution for the diffusion process on TPG. However, our goal was to utilize TPG to learn new affinities on the original graph  $G$ . i.e., to obtain a new affinity matrix  $A^*$  of size  $n \times n$ . The matrix  $A^*$  containing the **learned affinities** is defined as

$$A^* = \text{vec}^{-1}((I - \mathbb{A})^{-1} \text{vec}(I)), \quad (2.17)$$

where  $I$  is an  $n \times n$  identity matrix and  $\text{vec}$  is an operator that stacks the columns of a matrix one after the next into a column vector. Formally, for a given  $m \times n$  matrix  $B$

$$\text{vec}(B) = (b_{11}, \dots, b_{m1}, b_{12}, \dots, b_{m2}, \dots, b_{1n}, \dots, b_{mn})^T.$$

Since  $\text{vec} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{mn}$  is an isomorphism, its inverse exists, and we denote it with  $\text{vec}^{-1}$ .

To motivate the definition (2.17), we observe that the following equation holds

$$\text{vec}(A) = \mathbb{A} \text{vec}(I) = (A \otimes A) \text{vec}(I). \quad (2.18)$$

Hence, we can obtain the original matrix  $A$  from  $\mathbb{A}$  by

$$A = \text{vec}^{-1}(\mathbb{A} \text{vec}(I)). \quad (2.19)$$

We can see that (2.17) is like (2.19) but applied to the diffused  $\mathbb{A}$ , i.e., to the solution of (2.15).

To summarize, for the input affinity matrix  $A$ , the new learned affinities are given by matrix  $A^*$  defined in (2.17). However, the affinity learning with the proposed diffusion process on TPG (2.14) is impractical for large graphs due to high storage and computing cost. The diffusion on the original graph  $G$  requires  $O(n^2)$  storage (number of the matrix elements) and its computation cost is determined by the cost of matrix inversion, which is  $O(n^3)$  for Gauss-Jordan elimination or about  $O(n^{2.4})$  for the Coppersmith-Winograd algorithm. In contrast the diffusion on TPG requires  $O(n^4)$  storage and its computation cost is  $O(n^6)$  for Gauss-Jordan elimination or about  $O(n^{4.8})$  for the Coppersmith-Winograd algorithm. Therefore, we propose a novel iterative algorithm in Section 2.3.4 to compute (2.14). Its storage and computation cost

is comparable to the diffusion on the original graph, since it is executed on the original graph.

### 2.3.4 Iterative Algorithm for Diffusion on TPG

We define  $Q^{(1)} = A$  and

$$Q^{(t+1)} = A Q^{(t)} A^T + I, \quad (2.20)$$

where  $I$  is the identity matrix. We iterate (2.20) until convergence. Let us denote the limit matrix by  $Q^* = \lim_{t \rightarrow \infty} Q^{(t)}$ . The proof of the convergence of (2.20) and a closed form expression for  $Q^*$  both follow from the following key equation

$$\lim_{t \rightarrow \infty} Q^{(t)} = Q^* = A^* = \text{vec}^{-1}((I - \mathbb{A})^{-1} \text{vec}(I)). \quad (2.21)$$

The remainder of this section is devoted to prove this equation. Since  $Q^* = A^*$ , we obtain that the iterative algorithm on the original graph  $G$  defined by (2.20) yields the same affinities as the TPG diffusion process on  $\mathbb{G}$ .

In order to prove (2.21), we first transform (2.20) to

$$\begin{aligned} Q^{(t+1)} &= A Q^{(t)} A^T + I = A(A Q^{(t-1)} A^T + I)A^T + I \\ &= A^2 Q^{(t-1)} (A^T)^2 + A I A + I = \dots \\ &= A^t A (A^T)^t + A^{t-1} I (A^T)^{t-1} + \dots + I \\ &= A^t A (A^T)^t + \sum_{i=1}^{t-1} A^i I (A^T)^i \end{aligned} \quad (2.22)$$

Since (by our assumption) sum of each row of  $A < 1$ , we have  $\lim_{t \rightarrow \infty} A^t A (A^T)^t = 0$ , and consequently,

$$Q^* = \lim_{t \rightarrow \infty} Q^{(t+1)} = \lim_{t \rightarrow \infty} \sum_{i=1}^{t-1} A^i I (A^T)^i \quad (2.23)$$

We observe that the following identity holds

$$\text{vec}(A S A^T) = (A \otimes A) \text{vec}(S) = \mathbb{A} \text{vec}(S), \quad (2.24)$$

where we recall that  $\otimes$  is the Kronecker product. As a consequence we obtain for every  $i = 1, 2, \dots$

$$\text{vec}(A^i I (A^T)^i) = \mathbb{A}^i \text{vec}(I). \quad (2.25)$$

Our proof of (2.25) is by induction. Suppose

$$\text{vec}(A^k I (A^T)^k) = \mathbb{A}^k \text{vec}(I)$$

holds for  $i = k$ , then for  $i = k + 1$  we have

$$\begin{aligned} \text{vec}(A^{k+1} I (A^T)^{k+1}) &= \text{vec}(A (A^k I (A^T)^k) A^T) \\ &= \mathbb{A} \text{vec}(A^k I (A^T)^k) = \mathbb{A} \mathbb{A}^k \text{vec}(I) = \mathbb{A}^{k+1} \text{vec}(I) \end{aligned}$$

From (2.25) and from the fact that  $\text{vec}$  of a sum of matrices is sum of their  $\text{vec}$ 's, we obtain

$$\text{vec}\left(\sum_{i=1}^{t-1} (A)^i I ((A)^T)^i\right) = \sum_{i=1}^{t-1} \mathbb{A}^i \text{vec}(I). \quad (2.26)$$



Finally from (2.23) and (2.26), we derive

$$\begin{aligned}
\text{vec}(Q^*) &= \lim_{t \rightarrow \infty} \text{vec} \left( \sum_{i=1}^{t-1} A^i I (A^T)^i \right) = \lim_{t \rightarrow \infty} \sum_{i=1}^{t-1} (\mathbb{A}^i \text{vec}(I)) \\
&= \left( \lim_{t \rightarrow \infty} \sum_{i=1}^{t-1} \mathbb{A}^i \right) \text{vec}(I) = (I - \mathbb{A})^{-1} \text{vec}(I)
\end{aligned} \tag{2.27}$$

This proves our key equation (2.21). Hence the iterative algorithm (2.20) on  $G$  yields the same affinities as the TPG diffusion process on  $\mathbb{G}$ .

Since our iterative algorithm works on the original graph  $G$ , both its storage and computational cost requirements are significantly lower than those of the TPG diffusion process. It requires  $O(n^2)$  storage and its computation cost is determined by the cost of matrix multiplication, which is  $O(n^3)$  for direct implementation or about  $O(n^{2.4})$  for the Coppersmith-Winograd algorithm. Consequently, if the number of iterations is  $t = T$ , then its computational cost is  $O(Tn^3)$  or  $O(Tn^{2.4})$ , correspondingly.

Graph  $G$  in Fig. 2.22 provides a simple example to illustrate the fact that the diffusion on the TPG considers the information from more edge weights than the diffusion on the original graph. For simplicity we compare only the second iteration, i.e., we compare  $A^{(2)}$  to  $Q^{(2)}$  and focus on the edge weight between 1 and 3. Since there is no edges between 1 and 3 in  $G$ , we have  $a_{13} = a_{31} = 0$ . Therefore, in  $A^{(2)}$  we have  $a_{13}^{(2)} = a_{12} \cdot a_{23}$ . The corresponding weight of the edge between 1 and 3 in  $Q^{(2)}$  is given by

$$q_{13}^{(2)} = a_{12} \cdot a_{23} \cdot (a_{11} + a_{22}) + a_{12} \cdot a_{33} \cdot a_{33}.$$

While  $a_{13}^{(2)}$  only depends on the edge weights  $a_{12}$  and  $a_{23}$ ,  $q_{13}^{(2)}$  also depends on the self similarities  $a_{11}, a_{22}, a_{33}$ . In particular, we can have  $a_{13}^{(2)} < q_{13}^{(2)}$  if  $a_{11} + a_{22} > 1$ , but we can also have  $a_{13}^{(2)} > q_{13}^{(2)}$ . Thus, TPG diffusion utilizes more information to determine the strength of the connection between 1 and 3 than just the connections  $a_{12}$  and  $a_{23}$  considered by the diffusion on the original graph. The difference in the number of connections considered is even more dramatic for  $t > 2$ . TPG diffusion also utilizes the self-reinforcement in that the strength of the connections depends on the ratio between the similarity of each database object to itself and the sum of its similarities to other objects.

### 2.3.5 Dominant Neighbors

The derivations in the previous section depend on the assumption that the affinity matrix  $A$  of graph  $G$  is nonnegative and the sum of each row is smaller than one. However, the original affinity matrix of graph  $G$ , let us call it  $W$ , usually does not satisfy these assumptions. In this section we propose a particular way to transform  $W$  to a matrix  $A$  that satisfies them.

In the case of retrieval and ranking,  $W$  contains pairwise similarities between the database objects and between the query and the database objects. Therefore, we can assume that all entries in  $W$  are positive. It is also natural to assume that for each object  $i$  the self similarity of  $i$  to itself is the largest, i.e.,  $\forall i \forall j \neq i (w_{ii} > w_{ij})$ . We also can assume that  $W$  is symmetric, since if

this is not the case we can replace  $W = \frac{1}{2}W W^T$ .

As we observed in the introduction, the pairwise similarities are not accurate, and consequently, the graph contains many noisy similarities. Since the pairwise similarities are more accurate for close neighbors, the amount of noisy similarities is significantly reduced if we set to zero all edge weights except the  $k$  nearest neighbors ( $kNN$ ) of each object.

However, the selection of  $kNN$  is also easily influenced by errors in the pairwise similarities and the number of suitable nearest neighbors  $k$  may be different for different objects. Therefore, to better define the neighbors of a point, we propose a novel way to construct the neighborhood structure, which is called Dominant Neighborhood (DN). The main idea is that the dominant neighborhood  $DN(i)$  of a vertex  $i$  should correspond to a maximal clique that satisfies  $DN(i) \subseteq kNN(i)$ . As stated in [78], a maximal clique in a weighted graph, which is called a dominant set, is a subset of  $V$  with maximal average affinity between all pairs of its vertices, which is equivalent to the fact that the overall similarity among internal elements is the highest in that adding any new element would lower it.

As is the case for  $kNN(i)$ , we do not want to include vertex  $i$  in its  $DN(i)$ , therefore, we set the diagonal entries of  $W$  to zero and obtain a matrix  $W_0$ . In order to select vertices that belong to a dominant set, we introduce an indicator vector  $\mathbf{x} = (x_1, x_2, \dots, x_n)$  over the vertices  $V$  of  $G$ . A vertex  $j \in V$

is selected as belonging to a maximal clique if and only if  $x_j > 0$ . As shown in [78], each dominant set can be obtained as a local maximizer of the following quadratic program

$$\begin{aligned} & \text{maximize } f(\mathbf{x}) = (\mathbf{x})^T W_0 \mathbf{x} \\ & \text{subject to } \mathbf{x} \in \Delta = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \geq 0, \sum_{j=1}^n x_j = 1\}. \end{aligned} \quad (2.28)$$

Pavan and Pelillo [78] also provide an iterative method to compute local maximizers of (2.28). Given an initialization  $\mathbf{x}(1)$ , the corresponding local solution  $\mathbf{x}^*$  of (2.28) can be obtained by the replicator equation [107]:

$$x_j(t+1) = x_j(t) \frac{(W_0 \mathbf{x}(t))_j}{\mathbf{x}(t)^T W_0 \mathbf{x}(t)} \quad j = 1, \dots, n \quad (2.29)$$

It is easy to see that  $\mathbf{x}(t) \in \Delta$  with increasing  $t$ , which means that every trajectory starting in  $\Delta$  will remain in  $\Delta$ . Moreover, since  $W_0$  is symmetric, the target function  $f(\mathbf{x}) = (\mathbf{x})^T W_0 \mathbf{x}$  is strictly increasing for a given initial vector  $\mathbf{x}(1)$  and is guaranteed to converge.

In order to obtain a dominant neighborhood  $DN(i)$  of vertex  $i$ , we initialize (2.29) with the classical  $kNN(i)$ . More precisely, we set  $x_j(1) = \frac{1}{k}$  if  $j \in kNN(i)$  and  $x_j(1) = 0$  otherwise. After (2.29) converged to the corresponding local solution  $\mathbf{x}^*$ , a vertex  $j \in DN(i)$  if and only if  $x_j^* > 0$ .

As discussed above, the dominant neighborhood of  $DN(i)$  is determined not only by the pairwise relation of  $i$  to other objects, but also the relation between the other objects, which makes  $DN(i)$  more robust to noisy pairwise

similarities than  $kNN(i)$ . Thus, we use it to first refine the matrix  $W$  to  $W^*$  so that the neighbors of each data is robust to noise and outliers. The matrix  $W^*$  is obtained from  $W$  by setting  $w_{ij}^* = w_{ij}$  if  $j \in DN(i)$  and  $w_{ij}^* = 0$  otherwise. Then,  $W^*$  is transformed into a stochastic matrix.  $A$  is derived from  $W^*$ , where  $a_{ij} = w_{ij}^*$  if  $j \in KNN(i)$  and  $a_{ij} = 0$  otherwise.

### 2.3.6 Experimental Results

To demonstrate the advantages of our approach, we test our algorithm on both shape and image retrieval tasks. On all test datasets, the proposed method achieves excellent results, which are better than the state-of-art methods. Since our iterative algorithm to compute the TPD diffusion is guaranteed to converge, we only need to ensure that the number of iterations is not too small. It is set to 200 for all test datasets.

If pairwise distances are provided for a given dataset, we transform the distances to similarities with the method introduced in [106]. Once we obtain a similarity matrix  $W$ , we first use DN to obtain the matrix  $A$ . Then, we run the proposed, iterative algorithm to compute the TPD diffusion. It returns the new affinity matrix  $A^*$  representing the learned similarities, which are then used for ranking, i.e., if vertex  $i$  represents the query objects, the most similar objects to it are obtained by sorting in descendent order the row  $i$  of matrix  $A^*$ .

## MPEG-7 Dataset

The proposed framework is tested for shape classification on a commonly used MPEG7 CE-Shape-1 part B database [57]. The dataset contains 1400 silhouette images from 70 classes, where each class has 20 different shapes (some shapes are shown in Figs. 2.19 and 2.3.1). The retrieval rate is measured by the bull’s eye score: every shape in the database is submitted as a query and the number of shapes from the same class in the top 40 is counted. The bull’s eye score is then defined as the ratio of the number of correct hits to the best possible number of hits (which is  $20 \times 1400$ ).

As shown in Table 2.4 the proposed affinity learning method can successfully improve on the state-of-the-art methods. We selected two different shape similarity methods: Aspect Shape Context (ASC) [64] and Articulated Invariant Representation (AIR) [36] as the input pairwise distance measure.  $kNN$  with  $k = 13$  was used to initialize (2.29). We observe that the affinities learned by our method improve the original retrieval score of ASC by over 8%. We reach nearly perfect bull’s eye score 99.99% on MPEG7 Dataset by using AIR for shape similarity. This is the best ever reported score on this popular shape dataset.

In order to visualize the gain in retrieval rates (precision) by our method, we plot the percentage of correct results among the first  $k$  most similar shapes for  $k = 1, \dots, 40$  in Fig. 2.3.6, where we use ASC for shape similarity. We

Table 2.4: Retrieval rates (bull’s eye) of different context shape retrieval methods on the MPEG-7 shape dataset.

IDSC	IDSC	IDSC	Perc. R.	Perc. R.
	+LP	+Mutual graph		+ LCDP
[63]	[7]	[54]	[97]	[97]
85.40%	91.61%	93.40%	88.39%	95.60%
ASC	ASC	ASC	AIR	AIR
	+ LCDP	+ DN		+ DN
[64]	[64]	+ TPG Diffusion	[36]	+ TPG Diffusion
88.30%	95.96%	<b>96.47%</b>	93.67%	<b>99.99%</b>

observe that not only does the proposed method increase the bulls eye score, but also consistently achieves the best retrieval rates. Recall that each class has 20 shapes, which is the reason for the precision curves to increase for  $k > 20$ . In order to illustrate the problem with the stopping time of the graph classical diffusion computed by matrix power, we show two curves for LCDP [111], one when it is stopped after 7 iterations and the second one when it is stopped after 100 iterations, which clearly illustrates the problem of diffusing relevant information. In contrast, the proposed algorithm is robust to the number of iterations due to its guaranteed convergence proved in Section 2.3.2.

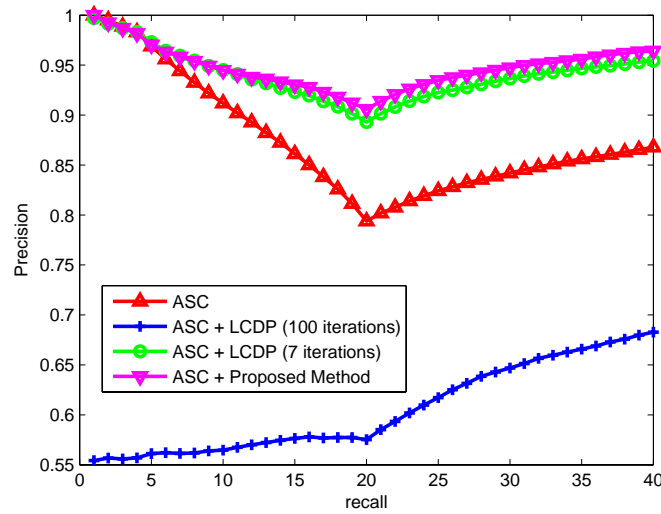


Figure 2.23: Precision/Recall curves on MPEG-7 shape dataset.

### Nister and Stewenius (N-S) dataset

In this section, we demonstrate the performance of the proposed approach on image retrieval. We compare it to other diffusion based methods and to a recently proposed method, Contextual Dissimilarity Measure (CDM) [46], which can significantly improve the similarity computed by bag-of-features. CDM learns affinities following a different principles than the proposed method. CDM is motivated by an observation that a good ranking is usually not symmetrical in image search. CDM makes two images similar when they both obtain a good ranking position when using each other as a query.

We selected the Nister and Stewenius (N-S) dataset [95] composed of 10,200 images. A few example images from N-S dataset are shown in Fig. 2.24. The





Figure 2.24: Some images from Nister and Stewenius (N-S) dataset.

N-S dataset consists of 2,550 objects or scenes, each of which is imaged from 4 different viewpoints. Hence there is only 4 images in each class and total of 2,550 image classes, which makes this dataset very challenging for any manifold learning approach, and in particular, for any diffusion based approach.

To obtain the pairwise distance relation between images for our algorithm, we implemented a baseline method described in [46]. The image descriptor is a combination of Hessian-Affine region detector [72] and SIFT descriptor [66]. A visual vocabulary is obtained using the k-means algorithm on the sub-sampled image descriptors.

The results are shown in Table 2.5. The retrieval rate is measured by the average number of correct images among the four first images returned. Thus, the maximum value is 4 and the higher the value the better is the result. Each image has been submitted as a query. The fact that our method can significantly improve the retrieval result of the baseline method (from 3.22 to 3.61) clearly shows the benefits of utilizing higher order relations by the TPG diffusion. We also observe that the result of our method is better than CDM.

Table 2.5: Retrieval results on N-S Dataset. The highest possible score is 4.

Baseline	Classic	Classic	Diffusion	CDM	TPG	TPG
	Diffusion	Diffusion	Maps		Diffusion	Diffusion
[46]	with $t = 2$	with $t = 5$	[17]	[46]	with Classic kNN	with DN
3.22	3.42	0.245	1.01	3.57	3.58	<b>3.61</b>

Finally, the usage of DN improves on the result obtained with classic  $kNN$ . We did not have much choice to set the neighborhood size  $k$  for this dataset.  $kNN$  with  $k = 3$  was used to initialize (2.29).

Since each image class has only 4 images, it is very difficult to correctly propagate the similarity relations. Therefore, the classic diffusion [17] can only improve the baseline result for a very small number of iterations. The best retrieval rate of the classic diffusion is for  $t = 2$ , i.e., when the original similarity matrix is raised to power  $t = 2$ . Already for  $t = 5$ , the retrieval rate is much lower than the rate of the baseline. We also report the retrieval result obtained after embedding the data by Diffusion Maps [17], which are significantly lower than the rate of the baseline. This justifies our observation that although Diffusion Maps are excellent for embedding into Euclidean spaces, the distances obtained after the embedding cannot be used for retrieval tasks.

## Caltech 101 dataset

Besides N-S dataset, we also test our algorithm on a well known Caltech 101 dataset [26]. The Caltech-101 dataset contains 101 classes (including animals, vehicles, flowers, etc.) with high shape variability. The number of images per category varies from 31 to 800. Most images are medium resolution, i.e. about  $300 \times 200$  pixels. We selected 12 classes from Caltech-101, which contain total 2788 images. Example images are shown in Fig.2.25. Different from experiments on N-S dataset, we just use pure SIFT descriptor [66] to calculate the distance between images. The SIFT features are extracted from  $16 \times 16$  pixel patches densely sampled from each image on a grid with step size of 8 pixels. To get the codebook, we use standard K-means clustering and fix the codebook size to 2048. Each image is represented by multiple assignment [46] and Spatial Pyramid Matching [60]. The distance between two images is obtained by the  $\chi^2$  distance between the two vectors.

The results are shown in Table 2.6. It is clear that with adjusted number of iterations according to the ground truth, which is  $t = 5$ , the classic diffusion process is able to reveal the relation between images. However, as discussed above, it is very sensitive to number of iterations, which we illustrate with its retrieval rate for  $t = 50$ . Besides, the results of Diffusion Maps [17] demonstrates that the relation between objects after embedding by Diffusion Maps [17] is not suitable for retrieval. In our algorithm, to initialize (2.29),



Figure 2.25: Some sample images from the selected subset of Caltech 101 dataset. Each class contains two examples.

Table 2.6: Retrieval rates on 12 image classes from Caltech-101. The best possible rate is 1.

Baseline	Classic Diffusion with $t = 5$	Classic Diffusion with $t = 50$	Diffusion Maps [17]	TPG Diffusion with DN
0.801	0.859	0.267	0.534	<b>0.903</b>

$kNN$  with  $k = 400$  was used. Again TPG diffusion is able to significantly improve the retrieval rate of the input pairwise distance measure. In particular, this demonstrates that TPG diffusion is robust to large variance in the number of images in each class.

# CHAPTER 3

## Shape Based Object Detection on Real Images

### 3.1 Introduction

we propose a single layer fully connected graph to model shape of deformable objects. Each node in the graph is a state variable, which consists of the position and the corresponding part. The relation between nodes is long range and not limited to direct spatial proximity. Our model can be interpreted as a generative prior for the configuration of the state variables. Since our graph is fully connected, we do not need to learn its structure, which simplifies the learning significantly. We only need to learn representation of the nodes and their pairwise relations. Since the number of pairwise rela-

tions is large, and most of them are not used in our inference process, we do not learn the pairwise relations explicitly. Instead, we learn a representation that allows us to dynamically construct the pairwise relations needed in the inference process.

In our model graph, the nodes represent contour parts and their position in a given shape class. They are learned automatically with partially-supervised learning. While many state-of-the-art approaches construct part models manually [67, 122], we limit manual labeling to a single contour. In our approach, only one silhouette is manually decomposed into visual parts in advance. Then, the part decomposition is automatically transferred to silhouettes not only in the same class but also in different classes with similar shape by shape matching. To deal with non-rigid objects, we use Inner Distance Shape Context (IDSC) introduced in [63]. The constructed part bundles (see §3.2) with proper position in the exemplar shapes form the nodes in the model graph. The relations between the nodes represent the spatial layout between parts. It is described by nonparametric density estimation, which has better discriminative power than methods based on unimodal distributions modeled as Gaussians, e.g., [27, 94]. To make the learnt model graph representative, we use the well designed exemplar based clustering by Affinity Propagation [33] to select a set of candidate silhouettes as exemplars for our model learning approach.

According to [121], there are no known algorithms for performing inference for densely connected flat models, e.g., the performance of Belief Propagation (BP) is known to degrade for representations with many closed loops. To address this issue, we propose a Markov chain Monte Carlo (MCMC) approach that is able to efficiently infer the values of the state variables representing nodes of our fully connected model graph. The proposed MCMC approach is based on Particle Filter (PF), but it differs fundamentally, since unlike the standard PF framework, our PF framework can infer an order of random variable (RVs). The inferred order follows the most informative paths in the graph. Thus, we use PF to linearize the structure of the graph, which allows us to avoid the problem of loops. Each particle may explore a different node order in this linearization, which corresponds to the order of contour parts. This fact is illustrated by two different detection examples shown in Fig. 3.1, where the PF order of detected contour parts is color coded. This property makes our algorithm different from other PF based method. As can be seen by examining the relative position of consecutive parts, the proposed inference is not limited to direct spatial proximity of the parts. This fact sets our approach apart from existing approaches, e.g., [121, 52].

In order to show the advantages of the proposed approach, we test our method on three widely used data sets, Weizmann horses [11], the ETHZ [29], and the cow dataset from the PASCAL Object Recognition Database

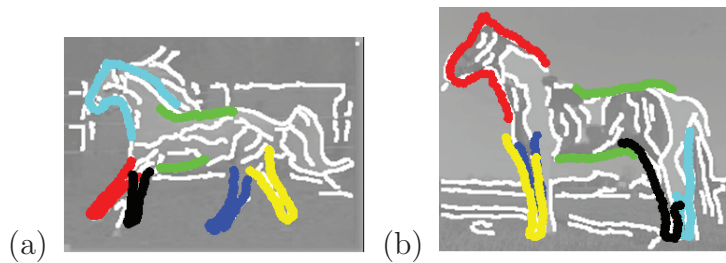


Figure 3.1: Examples of two different inferred orders of detected contour parts. Colors represent the order, which is 1=red, 2=cyan, 3=blue, 4=green, 5=yellow, and 6=black.

Collection (TU Darmstadt Database [61]). Our results measured by bounding box intersection are comparable to state-of-the-art methods. Also, we perform very well in the accuracy of boundary localization, which is evaluated by a recently proposed measure in [30].

## 3.2 Partially-Supervised Model Learning

Our approach only requires marking object parts on one exemplar. We then transfer this knowledge to other contours not only in the same shape class but also to similar shape classes. Thus, our approach is able to construct the part models for different classes of objects starting with only one exemplar contour. The constructed model can describe a wide range of objects with different poses.

As we learn the model from some exemplars, the first issue is which ones should be chosen from a given training data set. We use Affinity Propagation to select the exemplars, which are cluster centers in AP. These cluster centers



are representative, so that they can describe most of the poses of objects. The input pairwise distance between shapes is obtained by Oriented Chamfer Matching (OCM).

### 3.2.1 Part model construction

We now describe a way to automatically decompose the exemplars  $E = \{E_1, \dots, E_{N_e}\}$  into meaningful parts. We first manually segment one selected silhouette, say  $E_1$  into  $m$  different meaningful parts  $S = \{s_1, \dots, s_m\}$ . For example, for horse, we have six parts: head, two front legs, two back legs, and the body, shown in different colors in top left of Fig. 3.2(a). We then use shape matching with IDSC [63] to transfer the parts to other exemplars  $E_2, \dots, E_{N_e}$ , e.g., to the second horse in Fig.3.2(a). The corresponding points carry over the part decomposition. To ensure that the part decomposition is transferred correctly, we require that the number of corresponding points for a given contour part  $s_i$  is larger than a given threshold, e.g. 80% of the total number of points in the contour part. If this is not the case, the corresponding part is removed from the model.

We define part bundle  $B_i$  as a set composed of part  $s_i$  on  $E_1$  and all corresponding parts on  $E_2, \dots, E_{N_e}$  transferred by the IDSC matching for  $i = 1, \dots, m$ . Each part bundle  $B_i$  has at most  $N_e$  contour parts. We obtain a set of  $m$  part bundles  $B = \{B_1, B_2, \dots, B_m\}$  that defines the nodes of our part

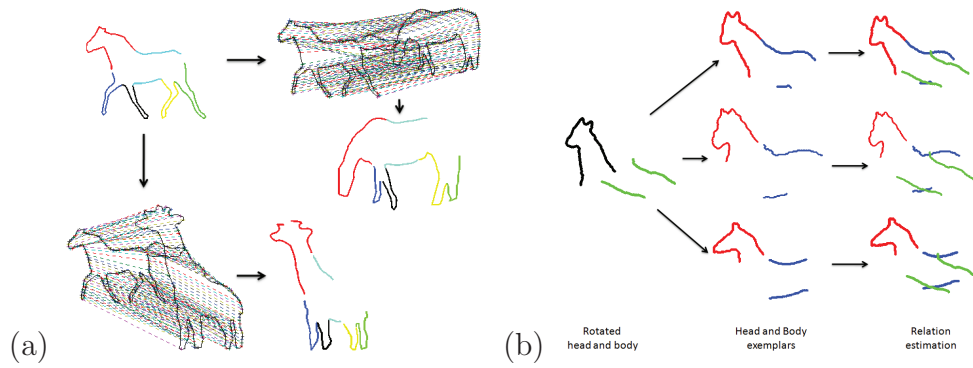


Figure 3.2: (a) Six manually labeled parts on the horse in top left are marked with different colors. The point correspondence obtained by shape matching allows us to transfer the part structure to a different horse and to a giraffe. (b) The horse head and horse body shown on the left hand side are very different from our perception of a horse. Our measure of this fact is illustrated in the rest of this figure.

model graph.

We can also employ shape matching to transfer the part structure to different but similar object classes. As illustrated in Fig. 3.2(a), our part decomposition of the horse contour transfers easily to contours of giraffes. As long as the objects in different classes have similar structure, the proposed approach can transfer the structure knowledge from the known class to the other classes and obtain the part bundle models. There are three advantages of the proposed approach: 1) It requires very little manual labeling. 2) The constructed model composed of part bundles can handle the intra-class variations as long as the training silhouettes can represent the possible poses of objects. 3) The structural knowledge can be easily transferred to different classes.

### 3.2.2 Relation between model parts

After learning the model from silhouettes, in order to make the model more flexible, we permit the rotation for each part and also some shift. However, with the increasing flexibility, the obtained model can be very different from shapes in a given object class. To reduce the negative effect of flexible models, we propose a soft way to constrain the flexibility. We allow the flexibility in a range determined by shape similarity to example shapes in a given object class. Here the shape similarity is described by spatial layout of model parts, i.e., a new rotated spatial layout of parts is allowed if it is similar to a layout previously seen for this class. An example is shown in Fig. 3.2(b). The horse head and horse body shown on the left hand side are very different from our perception of a horse. The head and body are too far away from each other and their arrangement due to rotation is really strange. With the method described below, we can offer a soft constraint on possible spatial layout of parts.

The key idea is to construct a distribution describing the spatial layout between different parts. In particular, given a part bundle  $B_i$ , the spatial relation between it and another part bundle  $B_j$  forms a distribution. This kind of distribution has been used in object detection to help describe the model [94, 27], but the distribution is assumed to be Gaussian, whose parameters can be easily learned from training samples. However, obviously, the distribution

of part relation is very complex and expressing it as Gaussian or any other parametric distribution does not seem to be a good approximation. Instead, we propose to learn the underlying distribution in a non-parametric setting.

We employ kernel density estimation, which is one of the most popular non-parametric methods. Given are two rotated parts  $p'_i$  and  $p'_j$  that come from different part bundles  $B_i$  and  $B_j$  respectively. Our goal is to find how is  $p'_j$  located with respect to  $p'_i$ . For example, we want to find out how well the green body is positioned with respect to the black horse head in Fig. 3.2(b). For part  $p'_i$ , we use  $OCM_{p'_i}$  to find the top  $k$  most similar exemplar parts  $(p_i(1), \dots, p_i(k))$  in part bundle  $B_i$  (the bundle of  $p'_i$ ). For these original parts in  $B_i$ , we know the exemplar contours they came from. From these contours, we extract parts  $(p_j(1), \dots, p_j(k))$  that belong to the same bundle as  $p'_j$ , i.e., to part bundle  $B_j$ . In Fig. 3.2(b), OCM retrieves the 3 red horse heads  $(p_i(1), p_i(2), p_i(3))$  as most similar to the black head, which in turn carry over from their original contours 3 blue horse bodies  $(p_j(1), p_j(2), p_j(3))$ . Finally, we measure the spatial layout between parts  $p'_i$  and  $p'_j$  by estimating the fitness of  $p'_j$  to the distribution described by  $(p_j(1), \dots, p_j(k))$ :

$$f(p'_j|p'_i) = \frac{1}{C_c} \sum_{t=1}^k \frac{1}{h} K\left(\frac{OCM_{p'_j}(p_j(t))}{h}\right) \quad (3.1)$$

where  $K$  is a kernel function with bandwidth  $h$ , which is Gaussian in the paper and  $C_c$  is a constant value. The computation of  $f(p'_j|p'_i)$  in our example is illustrated in the right column of Fig. 3.2(b). It is a function of the OCM

distance between the green horse body and the 3 blue horse bodies.

### 3.3 Framework for Object Detection

Our goal is to infer the maximum of a posterior distribution  $p(B_1, \dots, B_m \mid Z)$ , where  $(B_1, \dots, B_m)$  is a vector of random variables (RVs) representing part bundles, which are nodes of our shape model graph (§3.2). In our application  $Z = (I, C)$  is a set of observations, where  $I$  is a RV ranging over binary edge images and  $C$  ranges over classes of target objects including background. Thus,  $Z$  is static, since the target edge image and the class of object are fixed for a given detection process. The possible values of each RV  $B_i$  are vectors of two elements, one is the location  $x_i$  in the image and the second is the part  $s_i$  chosen from the part bundle  $B_i$  in the model. In the case of a correct detection, we expect part  $s_i$  to be located at  $x_i$  in the image. We stress that even though each part bundle has many parts, only one of them is chosen for a given location in the image. To simplify the notation, we use  $b$  to represent the pair of values  $(x, s)$  for each random variable, i.e.,  $b_l = (x_l, s_l)$ . Consequently, our goal is to find value assignments to RVs  $B_t = b_t$  for  $t = 1, \dots, m$  that maximize the posterior

$$\hat{b}_{1:m} = \operatorname{argmax}_{b_{1:m}} p(b_{1:m} \mid Z), \quad (3.2)$$

where  $b_{1:m}$  is a shorthand notation for  $(b_1, \dots, b_m)$ . We will achieve our goal by approximating the posterior distribution with a finite number of particles in the framework of Particle Filter (PF). Besides, only a small subset of the search space is considered in the framework, which reduces the complexity significantly compared to exhaustive search with sliding windows, e.g., [88].

Unlike the standard PF framework, the observations  $Z$  in our approach do not arrive sequentially, but are available at once, i.e.,  $Z$  is static. Therefore, the observations have no natural order. Consequently, the states  $b_{1:m}$  also do not have any natural order, i.e., the order of indices  $1, \dots, m$  does not have any particular meaning. Therefore, we need to extend the PF framework to infer an order of RVs, which may be different for each particle. Intuitively, we want to determine such an order of RVs so that the corresponding order of observations is most informative, which makes the particle reaches optimal solution faster and more accurate. This makes the proposed PF fundamentally different from classical PF. To represent the order of RVs we need a symbol of a bijection (onto and one-to-one function)  $< \cdot >^{(i)}: \{1, \dots, m\} \rightarrow \{1, \dots, m\}$ . Although we may have a different bijection for each particle ( $i$ ), we will drop the index ( $i$ ) from  $< 1 : t >^{(i)}$ , since the state variables already carry the particle index. For example, we denote  $(b_4^{(i)}, b_5^{(i)}, b_2^{(i)})$  as  $b_{<1:3>}^{(i)}$ , where  $< 1 : 3 > = (4, 5, 2)$ .

We first present the proposed PF algorithm followed by a discussion of its major differences to standard PF approaches. As it is often the case in

PF applications, we assume the proposal distribution to be  $q(b|b_{<1:t-1>}^{(i)}, Z) = p(b|b_{<1:t-1>}^{(i)})$ . For each particle  $(i)$ , where  $i = 1, \dots, N$ , the proposed PF algorithm in each iteration  $t = 2, \dots, m$  performs the following three steps:

1) **Importance sampling / proposal:** Sample followers of particle  $(i)$  for  $l \in \{1, \dots, m\} \setminus <1:t-1>$

$$b_l^{(i)} \sim p(b_l|b_{<1:t-1>}^{(i)}) \quad (3.3)$$

and set  $b_{<1:t-1>,l}^{(i)} = (b_{<1:t-1>}^{(i)}, b_l^{(i)})$ . In particular, in the first iteration ( $t = 1$ ) we generate samples from each dimension of the state space, i.e., we sample for  $l \in \{1, \dots, m\}$

$$b_{<1>}^{(i)} = b_l^{(i)} \sim p(b_l) \quad (3.4)$$

2) **Importance weighting/evaluation:** An individual importance weight is assigned to each follower of each particle by

$$w(b_{<1:t-1>,l}^{(i)}) = p(Z|b_{<1:t-1>,l}^{(i)}). \quad (3.5)$$

3) **Resampling:** At the sampling step we have generated more samples than the number of particles. Thus we have a larger set of particles  $b_{<1:t-1>,l}^{(i)}$  for  $i = 1, \dots, N$  and  $l \in \{1, \dots, m\} \setminus <1:t-1>$  from which we sub-sample  $N$  particles and assign equal weights to all of them as in the standard Sampling Importance Resampling (SIR) approach. We obtain a set of new particles  $b_{<1:t>}^{(i)}$  for  $i = 1, \dots, N$ . The resampling is not performed in the last step, i.e.,

when  $t = m$ .

**Algorithm discussion:**

1) This step provides our main extension of the classical PF framework. In the classical PF framework, followers of each particle are selected from only one conditional distribution, i.e., from the conditional distribution of RV at dimension  $t$  given by  $p(b_t | b_{1:t-1}^{(i)})$ , since the dimension index  $t$  represents a real order of RVs  $1 : t = 1, \dots, t$ . In contrast we sample the followers from each dimension  $l \in \{1, \dots, m\}$  that is not already included in  $\langle 1 : t - 1 \rangle$ .

The fact that one can consider more than one follower of each particle and reduce the number of followers by resampling is known in the PF literature and is referred to as prior boosting [37]. It is used to capture multi-modal likelihood regions. However, all followers are selected from the conditional distribution of the same RV (the same dimension  $t$ ) in the classical PF framework.

2) We take the weight formula from [67], where it has been derived for PF with static observations.

3) We stress that the resampling plays in our framework an additional and a very crucial role. It selects the the most informative random variables (i.e., state space dimensions) as followers of particles. Since the weight of  $b_{\langle 1:t-1 \rangle, l}^{(i)}$  is determined by the observations  $Z$ , and the resampling uses the weights to selects a follower  $b_{\langle t \rangle} = b_l$  from not yet considered dimensions



$l \in \{1, \dots, m\} \setminus \langle 1 : t-1 \rangle$ , the resampling determines the order of RVs, i.e., the bijection  $\langle t \rangle$  for  $t = 1, \dots, m$ . Consequently, the order of RVs is heavily determined by  $Z$ , and this order may be different for each particle ( $i$ ). This is in strong contrast to the classical PF, where observations  $Z$  have no influence on the order of RVs, which is fixed.

In order to execute the derived PF algorithm, we need to define the proposal distribution  $p(b_l | b_{\langle 1:t-1 \rangle}^{(i)})$ , and the evaluation pdf  $p(Z | b_{\langle 1:t-1 \rangle, l}^{(i)})$ . As stated in Eq. 3.4, the initial proposal distribution is defined by  $p(b_l)$ , where  $l$  is an index of a RV representing a part bundle and  $b_l = (s_l, x_l)$ . In our implementation,  $p(b_l)$  is simply the probability of finding model part  $s_l$  at location  $x_l$ , and it measures how well model part  $s_l$  fits the edges in the image. We compute it as a Gaussian of the oriented chamfer distance. Similarly,  $p(b_l | b_{\langle 1:t-1 \rangle}^{(i)})$  is the probability of finding model part  $s_l$  at the location  $x_l$ , but now the location is constrained, since parts  $s_{\langle 1:t-1 \rangle}$  have already been placed in the image. Thus, this conditional probability is picked around the expected location  $x_l$  determined by the locations  $x_{\langle 1:t-1 \rangle}$  of the previously added parts. While the initial proposal distribution is computed at every image location, the conditional proposal distribution is only computed at regions of interest determined by the previously placed model parts.

As  $Z = (I, C)$ , and  $I$  and  $C$  can be viewed as independent conditioned on

$b_{<1:t-1>,l}^{(i)}$ , we obtain:

$$p(Z|b_{<1:t-1>,l}^{(i)}) = p(I|b_{<1:t-1>,l}^{(i)})p(C|b_{<1:t-1>,l}^{(i)}) \quad (3.6)$$

We recall that in our detection framework, both  $I$  and  $C$  are instantiated, since they are given prior to the detection, i.e.,  $I = im$ , where  $im$  is a given binary edge image and  $C = 1$ , which represents the class of the target object. The first factor  $p(I = im|b_{<1:t-1>,l}^{(i)})$  in Eq. 3.6 describes the goodness of fit to the edge image  $im$  of the partial shape model determined by  $b_{<1:t-1>,l}^{(i)}$ , i.e., how likely the edges in  $im$  come from a picture of a shape like the shape of  $b_{<1:t-1>,l}^{(i)}$ . The second factor  $p(C = 1|b_{<1:t-1>,l}^{(i)})$  represents the probability of the target class given the model  $b_{<1:t-1>,l}^{(i)}$ . Hence it can be viewed as shape class constraints on the model. The conditional pdfs describing both factors are defined in § 3.4.

### 3.4 Evaluation based on shape similarity

As  $b_{<1:t-1>,l}^{(i)}$  consists of the parts  $s_{<1:t-1>,l}^{(i)}$  and their locations  $x_{<1:t-1>,l}^{(i)}$ , we construct a partial shape model  $\mu$  by putting parts  $s_{<1:t-1>,l}^{(i)}$  at locations  $x_{<1:t-1>,l}^{(i)}$  on the edge map  $im$ . The probability that the edge map  $im$  is an image of a real object looking like our partial model  $\mu$  is given by

$$p(I = im|b_{<1:t-1>,l}^{(i)}) = \exp(-\beta \cdot OCM_{im}(\mu)), \quad (3.7)$$

where  $OCM_{im}(\mu)$  returns the Oriented Chamfer distance between  $im$  and  $\mu$  and  $\beta$  is set to 10. Consequently,  $OCM_{im}(\mu)$  measures how well the constructed partial model matches to the edge map.

$p(C = 1|b_{<1:t-1>,l}^{(i)})$  expresses the probability of the target shape class given partial shape model  $\mu = b_{<1:t-1>,l}^{(i)}$ . We obtain by Bayes rule

$$p(C = 1|\mu) = \frac{p(\mu|C = 1)p(C = 1)}{\sum_{c=1,0} p(\mu|C = c)p(C = c)}. \quad (3.8)$$

$p(\mu|C = 1)$  measures the similarity between the constructed model and the target class. Similarly,  $p(\mu|C = 0)$  measures the similarity between the constructed model and the background. Eq. 3.8 helps to prevent accidental match to the background, since it eliminates shape models with both high similarity to a given object class and to the background, and favors models with high similarity to a given object class and low similarity to the background. We utilize a recursive computation in our PF framework to obtain

$$\begin{aligned} p(\mu|C = c) &= p(b_{<1:t-1>,l}^{(i)}|C = c) \\ &= p(b_l^{(i)}|b_{<1:t-1>,l}^{(i)}, C = c) p(b_{<1:t-1>,l}^{(i)}|C = c) \\ &= p(b_l^{(i)}|b_{<t-1>,l}^{(i)}, C = c) p(b_{<1:t-1>,l}^{(i)}|C = c) \\ &= f(b_l^{(i)}|b_{<t-1>,l}^{(i)}) p(b_{<1:t-1>,l}^{(i)}|C = c), \end{aligned} \quad (3.9)$$

where  $f$  is defined in Eq. 3.1, and a given shape class  $C = c$  is modeled as a set of exemplars  $E = \{E_1, \dots, E_{N_e}\}$ , which are selected from training examples by affinity propagation.  $f$  describes the pairwise relation between nodes in the

graph, which is naturally utilized in our PF framework. When  $C = 0$ , we randomly select some background edge configurations as training examples. In the transition from 2nd to 3rd row in Eq. 3.9, we make a Markov assumption that the new model part  $b_l^{(i)}$  only depends on the previously added part  $b_{<t-1>}^{(i)}$  conditioned that we know the shape class  $C = c$ . This simplifies the computation and makes the shape model more flexible in that the pose of the new model part is only evaluated with respect to the pose of previously added part. Finally,  $p(b_{<1:t-1>}^{(i)} | C = c)$  is remembered from the previous iteration of particle  $(i)$ .

### 3.5 Experimental Results

We have tested our algorithm on three widely used data sets: the extended Weizmann Horses [11, 88], the ETHZ shapes [30] and the TU Darmstadt Database [61]. During the testing for Weizmann Horses, only 12 automatically selected horse silhouettes with one hand decomposed horse are used to learn the shape model. All the other images are used for testing. The edge maps for this dataset are obtained by Canny edge detector. We also test our method on the class of giraffe in ETHZ shape dataset [30]. The reason why we only select the category giraffes from ETHZ is that our model learning method can only transfer between objects with similar structure and giraffe is the only object in ETHZ having similar structure to horse. Only one hand decomposed horse and

6 automatically selected giraffe silhouettes are used to learn the giraffe model. Further, we work on the cow dataset the TU Darmstadt Database [61], since cows have similar structure with the above two classes. It contains 111 images. Only one hand decomposed horse and 6 automatically selected cow silhouettes are used to learn the cow model. The edge maps for this dataset are obtained by Canny edge detector.

To adapt to large scale variance, we generate multiple models by resizing the original ones to 5 to 8 scales, and choose as the final result from the best score in all the scales. We not only report our results on the commonly used bounding box intersection, but also the accuracy of our boundary localization.

### 3.5.1 Detection according to bounding boxes

We first evaluate the ability of the proposed approach to localize objects in cluttered images using bounding-box intersection, which is widely used in traditional object detection task. We adopt the strict standards of PASCAL Challenge criterion: a detection is counted as correct only if the intersection-over-union ratio with the ground-truth bounding-box is greater than 50%.

Fig. 3.3 reports precision-recall (P/R) curve and detection rate vs false positive per image (DR/FPPI) curve for the class Giraffes in ETHZ dataset. In P/R, we compare to Lu et al. [67], Zhu et al. [122], Ommer and Malik [76] and Ferrari et al. [30], whose results are quoted from [67]. In DR/FPPI, as

Ferrari et al. [30, 29], Ommer and Malik [76] and Lu et al. [67] provide their results, we compare to them. As Ravishankar et al. [82] do not give their curves, we do not compare to them in Fig. 3.3. According to the curves, we are better than Lu et al. [67], Ommer and Malik [76], Ferrari et al. [30, 29] and perform equally well as Zhu et al. [122]. The performance of the proposed method illustrates its ability to cope with substantial nonrigid deformations, which are present in the class Giraffes. This is demonstrated by our example results in Fig. 3.4(a).

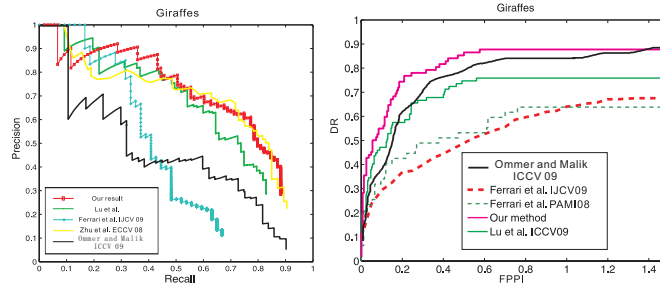


Figure 3.3: Precision-recall curve and detection rate (DR) vs false positive per image (FPPI) curve for the class Giraffes in ETHZ dataset.

Table 3.1 compares our detection rate to [121, 88] on Weizman Hores and TU Darmstadt Cows. The detection rate on horses is estimated from the DR/FPPI curve in [88]. The DR/FPPI curve for cows is not available in [88]. The method in [121] is also matching based, while [88] is a classification method. Some examples of our horse and cow detection results are shown in Fig. 3.4(b). The detection precision/recall area under curve (AUC) is a standard performance measure on the Weizmann Horses dataset. The AUC



Figure 3.4: Examples of detection results for Giraffes, horses and cows.

for our approach is 79.84%, which is comparable to the result 80.32% in Xiang et al. [6]. We compare to them as they also use the explicit shape model and matching based method for object detection. The AUC of classification based methods [88, 34] is 84.98% and 96%, respectively. We observe that classification based methods are bounding box classifiers and utilize significantly more information than matching based methods as ours. This explains why our detection rate and AUC is lower than [88, 34].

Table 3.1: Detection rate.

	Our method	Zhu et al. [121]	Shotton et al. [88]
Horses	93.97%	86.0%	95.20%
Cows	90.38%	88.6%	N/A

The proposed approach can not only succeed in extensive cluttered images, but also handles the problem of large range of scales and intra-class variability. This is demonstrated by several examples in Fig. 3.4. The images in the bottom right of Fig. 3.4(a) with red rectangles are the ones we fail to detect. The images of horses in Fig. 3.4(b) with red rectangles are false positives in the negative images provided by Shotton et. al. [88] to complement the Weizmann horse dataset. They show that the false positives in the negative set are caused by really very cluttered edges or by the structure of edges happening to match to the model very well. Interestingly, the rightmost false positive of horses is due to a camel, whose shape is very similar to horse.

### 3.5.2 Localizing object boundaries

The method presented in this paper offers one important advantage compared to texture based and classification methods like [19, 34, 22]. It can localize object boundaries, rather than just bounding-boxes.

In order to quantify how accurately the output shapes match to true boundaries, we use the coverage and precision measures defined in [30]. Coverage is the percentage of points from ground-truth boundaries closer than a threshold  $t$  to the output shapes of the proposed approach. Reversely, precision is the percentage of points from output shapes closer than  $t$  to any point of ground-truth boundaries. As in [30]  $t$  is set to 4% of the diagonal of the ground-truth



bounding box. The measures are complementary. Coverage captures how much of the object boundary has been recovered by the algorithm, whereas precision reports how much of the algorithm’s output lies on the object boundaries. These measurements are really useful and suitable for evaluating shape based approaches. In comparison, bounding-box evaluation cannot represent how accurate the detected shapes match the ground-truth boundary. It is possible to have bounding-box intersection larger than 0.5 without having correctly identified the ground-truth object boundaries. Two examples of horse detection are shown in Fig. 3.4(b) with green rectangles.

The first two columns of Table 3.2 show coverage and precision averaged over all images of the class giraffes in ETHZ dataset in comparison to the results in [30]. We measure the coverage and precision for the correct detections at 0.4 FPPI, following [30]. The coverage of the proposed approach is over 11% better than [30], which shows that our approach can efficiently recover the true boundary of objects. The precision is a little lower than [30]. More importantly, the detection rate at our 0.4 FPPI is 86.75%. However, even for 20% bounding box intersection, the detection rate at 0.4 FPPI in [30] is only around 60% , which is much less than us. It demonstrates that our approach can correctly localize object’s boundary on more images.

For horses and cows, the coverage and precision are obtained over all correct detections. The third column of Table. 3.2 shows the coverage and precision

of the proposed method on the Weizmann Horse dataset. As the edges are significantly worse than the ones provided for the giraffes, both measures are worse than the results on giraffes. The coverage and precision results for cow are shown in the fourth column of Table. 3.2. Due to less intra-shape variance, the precision is 92.02%, which is much higher than giraffes and horses. However, the coverage is only 73.86%. The main reason for the difference between these two values is that our model has a gap, since we removed the contour part representing the horse tail from the horse contour used for part decomposition. Thus, even if the model and object match perfectly, the coverage score cannot be perfect (see examples in Fig. 3.4).

Table 3.2: Accuracy of the boundary localization.

	Ours on giraffes	Results in [30] on giraffes	Ours on horses	Ours on cows
Coverage	79.4%	68.5%	77.5%	73.86%
Precision	74.6%	77.3%	61.7%	92.02%

## CHAPTER 4

### Particle Filter with State

### Permutations for Solving Image

### Jigsaw Puzzles

#### 4.1 Introduction and Problem Formulation

As shown in [21] the jigsaw puzzle problem is NP-complete if the pairwise affinity among jigsaw pieces is unreliable. Following [16], we focus on reconstructing the original image from square and non-overlapping patches. This type of puzzles does not contain the shape information of individual pieces, which is quite important to determine the pairwise affinities among them. This makes the problem more challenging, since it is more difficult to



Figure 4.1: The goal is to build the original image (a) given the jigsaw puzzle pieces (b). The original image is not known, thus, it needs to be estimated given the observations shown in (b). The empty squares in (c) form possible locations for the puzzle pieces in (b).

evaluate pairwise affinities among puzzles. This is different from most of the previous approaches [53, 35, 80, 108], where the shape of the puzzle pieces is utilized. While [16] also considers priors on the target image layout, we do not assume any prior knowledge on the image layout. Thus, only local image content information of the puzzle pieces is available in our framework, e.g., see Fig. 4.1.

Now we briefly review the PF inference. We begin with a classical tracking example. A robot is moving around and taking images at discrete time intervals. The images form a sequence of observations  $Z = (z_1, \dots, z_m)$ , where  $z_t$  is an image taken at time  $t$ . With each observation  $z_t$  there is associated a hidden state  $x_t$ . In our example the value of  $x_t$  is the robot pose (its 2D position plus orientation). The goal of PF inferences, is to derive the most likely sequence of the hidden states, i.e., to find a state vector  $x_{1:m} = (x_1, \dots, x_m)$  that maximizes the posterior  $p(x_{1:m}|Z)$ . We observe that here the observations are ordered following their time stamps. In PF inference, this order is utilized

to sequentially infer the values of states  $x_t$  for  $t = 1, \dots, m$ . Now imagine that the robot's clock broke and the time stamps are random. Thus, we are given a set of observations  $Z = \{z_1, \dots, z_m\}$ , they are indexed but their index is irrelevant. Of course, we can still associate state  $x_t$  with observation  $z_t$ , but the set of observations is not ordered, and consequently, the corresponding states  $x_t$  are not ordered. Thus, we deal with unordered observations. This is exactly the scenario of the image jigsaw puzzle problem, e.g., see Fig. 4.1. We are given  $m$  square puzzle pieces described by a set of  $m$  observations  $Z = \{z_1, \dots, z_m\}$ . Each observation  $z_t$  describes part of the original image depicted on piece  $t$  and is given by a vector of features, which are the color values of the pixels on piece  $t$  in our experimental results. The puzzle pieces are numbered with index  $t$ , but their numbering is random like the numbers in Fig. 4.1(b). The value of the state  $x_t$  of puzzle piece  $t$  is a location of an empty square in the square grid, e.g., the value of  $x_t$  is the index of an empty square in the square grid shown in Fig. 4.1(c). Our goal is to determine the state vector  $x_{1:m}$  that maximizes the posterior probability  $p(x_{1:m}|Z)$ . Since the original image is not provided, this probability is determined based on pairwise appearance consistency of the local puzzle images, i.e., the posterior distribution is a function of how well adjacent pieces fit together once they are placed on the grid. In other words, a vector of grid locations  $x_{1:m}$  maximizes  $p(x_{1:m}|Z)$  if the puzzle pieces placed at these locations form the most consistent image. We observe that

the posterior distribution  $p(x_{1:m}|Z)$  usually is very complicated and has many local maxima. This is particularly the case when the local image information of the puzzle pieces is not very descriptive.

Our main contribution is a new PF inference framework that works in this scenario. In the proposed framework we extend PF to handle the situations where we have unordered set of observations that are given simultaneously. One of our key ideas is the fact that it is possible to extend the importance sampling from the proposal distribution so that different particles explore the state space along different dimensions. Then the particle resampling allows us to automatically determine most informative orders of observations (as permutations of state space dimensions). Consequently, we can use a rich set of proposal functions in the process of estimating the posterior distribution.

The classical PF framework has been developed for sequential state estimation like tracking [51, 91] or robot localization [99, 31]. There, the observations arrive sequentially and are indexed by their time stamps, as our tracking example illustrates. It is possible to apply the classical PF framework as stochastic optimization to solve this problem by utilizing a fix order of states. However, by doing so, we would have selected an arbitrary order, and the puzzle construction may fail because of the selected order and would require extremely large number of particles. Our framework on the other hand can work with fewer particles because each particle explores different order. This gives us a

rich set of proposal distributions as opposed to having one fixed. Moreover, the observations are given simultaneously at the same time. Hence, there is no reason to favor any particular order without utilizing this fact.

In our experimental results, we compare the solutions obtained by the proposed inference framework to the solutions of the loopy believe propagation under identical settings on the dataset from [16]. In particular, we use exactly the same dissimilarity-based compatibility of puzzle pieces. The proposed PF inference significantly outperforms the loopy believe propagation in all evaluation measures. The main measure is the accuracy of the label assignment, where the difference is most significant. The accuracy using loopy believe propagation is 23.7% while that using the proposed PF inference is 69.2%.

The rest of the chapter is organized as follows. After introducing the preliminaries in §4.2, our key extensions for permuted PF are explained in §4.3 and §4.4. §4.5 provides implementation details. §4.6 shows and evaluates the experimental results not only the dataset from [16], but also an extended dataset.

## 4.2 Particle Filter Preliminaries

In this section we present some preliminary facts about Particle Filters (PFs). They will be utilized in the following sections when we introduce the proposed framework. Given is a sequence of observations  $Z = (z_1, \dots, z_m)$ ,

i.e., the observations are ordered. Our goal is to maximize the posterior distribution  $p(x_{1:m} \mid Z)$ , i.e., to find the values  $\hat{x}_t$  of states  $x_t$  such that

$$\hat{x}_{1:m} = \operatorname{argmax}_{x_{1:m}} p(x_{1:m} \mid Z), \quad (4.1)$$

where  $x_{1:m} = (x_1, \dots, x_m) \in \mathcal{X}^m$  is a state space vector and each state  $x_t$  has a corresponding observation  $z_t$  for  $t = 1, \dots, m$ .

This goal can be achieved by approximating the posterior distribution with a finite number of samples in the framework of Bayesian Importance Sampling (BIS). Since it is usually difficult to draw samples from the probability density function (pdf)  $p(x_{1:m} \mid Z)$ , samples are drawn from a proposal pdf  $q$ ,  $x_{1:m}^{(i)} \sim q(x_{1:m} \mid Z)$  for  $i = 1, \dots, N$ . Then approximation to the density  $p$  is given by

$$p(x_{1:m} \mid Z) \approx \sum_{i=1}^N w^{(i)} \delta_{x_{1:m}^{(i)}}(x_{1:m}), \quad (4.2)$$

where  $\delta_{x_{1:m}^{(i)}}(x_{1:m})$  denotes the delta-Dirac mass located at  $x_{1:m}^{(i)}$  and

$$w^{(i)} = \frac{p(x_{1:m}^{(i)} \mid Z)}{q(x_{1:m}^{(i)} \mid Z)} \quad (4.3)$$

are the importance weights of the samples. Typically the sample  $x_{1:m}^{(i)}$  with the largest weight  $w^{(i)}$  is then taken as the solution of (4.1).

Since it is still computationally intractable to draw samples from  $q$  due to high dimensionality of  $x_{1:m}$ , Sequential Importance Sampling (SIS) is usually utilized. In the classical PF approaches, samples are generated recursively following the order of dimensions in state vector  $x_{1:m} = (x_1, \dots, x_m)$ :

$$x_t^{(i)} \sim q_t(x \mid x_{1:t-1}, Z) = q_t(x \mid x_{1:t-1}, z_{1:t}) \quad (4.4)$$



for  $t = 1, \dots, m$ , and the particles are built sequentially  $x_{1:t}^{(i)} = (x_{1:t-1}^{(i)}, x_t^{(i)})$  for  $i = 1, \dots, N$ . The subscript  $t$  in  $q_t$  indicates from which dimension of the state vector the samples are generated. Since  $q$  factorizes as

$$q(x_{1:m}|Z) = q_1(x_1|Z) \prod_{t=2}^m q_t(x_t|x_{1:t-1}, Z), \quad (4.5)$$

we obtain that  $x_{1:m}^{(i)} \sim q(x_{1:m}|Z)$ . In other words, by sampling recursively  $x_t^{(i)}$  from each dimension  $t$  according to (4.4) we obtain a sample from  $q(x_{1:m}|Z)$  at  $t = m$ .

Since at a given iteration we have a *partial* state sample  $x_{1:t}^{(i)}$  for  $t < m$ , we also need an evaluation procedure of this partial state sample. For this we observe that the weights can be recursively updated according to [100]:

$$w(x_{1:t}^{(i)}) = \frac{p(z_t|x_{1:t}^{(i)}, z_{1:t-1})p(x_t^{(i)}|x_{1:t-1}^{(i)})}{q_t(x_t^{(i)}|x_{1:t-1}^{(i)}, z_{1:t})} w(x_{1:t-1}^{(i)}). \quad (4.6)$$

The above equation is derived from (4.3) using Bayes rule. Consequently, when  $t = m$ , the weight  $w(x_{1:m}^{(i)})$  of particle  $(i)$  recursively updated according to (4.6) is equal to  $w^{(i)}$  (defined in (4.3)). Hence, at  $t = m$ , we obtain a set of weighted (importance) samples from  $p(x_{1:m}|Z)$ , which is formally stated in the following theorem [18]:

**Theorem 4.2.1.** *Under reasonable assumptions on the sampling (4.4) and weighting functions (4.6) given in [18],  $p(x_{1:m}|Z)$  can be approximated with weighted samples  $\{x_{1:m}^{(i)}, w(x_{1:m}^{(i)})\}_{i=1}^N$  with any precision if  $N$  is sufficiently*

large. Thus, the convergence in (4.7) is almost sure:

$$p(x_{1:m}|Z) = \lim_{N \rightarrow \infty} \sum_{i=1}^N w(x_{1:m}^{(i)}) \delta_{x_{1:m}^{(i)}}(x_{1:m}). \quad (4.7)$$

In many applications, the weight equation (4.6) is simplified by making a common assumption that  $q_t(x_t^{(i)}|x_{1:t-1}^{(i)}, z_{1:t}) = p(x_t^{(i)}|x_{1:t-1}^{(i)})$ , i.e., we take as the proposal distribution the conditional pdf of the state at time  $t$  conditioned on the current state vector  $x_{1:t-1}^{(i)}$ . This assumption simplifies the recursive weight update to

$$w(x_{1:t}^{(i)}) = w(x_{1:t-1}^{(i)}) p(z_t|x_{1:t}^{(i)}, z_{1:t-1}), \quad (4.8)$$

and implies that the samples are generated from

$$x_t^{(i)} \sim p_t(x|x_{1:t-1}^{(i)}). \quad (4.9)$$

Analogous to (4.4)  $p_t$  in (4.9) indicates the dimension of the state space from which the samples are generated.

Now we summarize the derived **standard PF algorithm**. For every time step  $t = 1, \dots, m$  and for every particle  $i = 1, \dots, N$  execute the following three steps:

- 1) **Importance sampling / proposal:** Sample followers of particle  $(i)$  according to (4.9) (a special case of (4.4)) and set  $x_{1:t}^{(i)} = (x_{1:t-1}^{(i)}, x_t^{(i)})$ .
- 2) **Importance weighting / evaluation:** An importance weight is assigned to each particle  $x_{1:t}^{(i)}$  according to (4.8) (a special case of (4.6)).

3) **Resampling:** Sample with replacement  $N$  new particles from the current set of  $N$  particles

$$\{x_{1:t}^{(i)} | i = 1, \dots, N\}$$

according to their weights. We obtain a set of new particles  $x_{1:t}^{(i)}$  for  $i = 1, \dots, N$ , and renormalize their weights to sum to one. This procedure is a variant of Sampling Importance Resampling (SIR) [100]. It is an important part of any PF algorithm, since resampling prevents weight degeneration of particles.

### 4.3 Key Extension to Permuted States

As stated above, the standard SIS in Eq. 4.9 and particle evaluation in Eq. 4.8 utilize the sequential order of the states  $x_{1:m} = (x_1, \dots, x_m)$ . Of course, this is the best choice in many applications where the order is determined naturally by the time stamp of the observations. In contrast, the proposed approach is aimed at scenarios where no natural order of observations is given and the observations  $Z$  are initially known as in the image jigsaw puzzle problem.

The key idea of the proposed approach is not to utilize the fix order of the states  $x_{1:m} = (x_1, \dots, x_m)$  induced by the order of observations  $Z$ , but instead explore different orders of the states  $(x_{i_1}, \dots, x_{i_m})$  such that the corresponding

sequences of observations  $(z_{i_1}, \dots, z_{i_m})$  is most informative. In particular, we do not follow the order of indices of observations in  $Z$ . This way we are able to utilize the the most informative observations first allowing us to use a rich set of proposal functions. To achieve this we modify the first step of the PF algorithm so that the importance sampling is performed for every dimension not yet represented by the current particle. Intuitively, for example, if the first puzzle piece has a local image very similar to many other puzzle pieces and the second puzzle piece has a very distinctive local image that matches only a few other pieces, then our approach will first process the second puzzle piece, since it is more informative.

To formally define the proposed sampling rule, we need to explicitly represent different orders of states with a permutation  $\sigma : \{1, \dots, m\} \rightarrow \{1, \dots, m\}$ . We use the shorthand notation  $\sigma(1 : t)$  to denote  $(\sigma(1), \sigma(2), \dots, \sigma(t))$  for  $t \leq m$ . Each particle  $(i)$  now can have a different permutation  $\sigma^{(i)}$  of the puzzle pieces *in addition* to their locations. Thus the particles are now represented as  $x_{\sigma(1:t)}^{(i)}$ . We drop the superscript  $(i)$  of  $\sigma^{(i)}$  in the context of a particle which already carries the index  $(i)$ . For example, Fig. 4.1(c) shows the configuration of a particle at time  $t = 2$ , where puzzle pieces numbered 3 and 1 in Fig. 4.1(b) are placed at locations (a) and (b), correspondingly. Hence  $\sigma^{(i)}(1 : 2) = (3, 1)$  and  $x_{\sigma(1:2)}^{(i)} = (a, b)$ . Thus, a sequence of states  $x_{\sigma(1:t-1)}$  visited before time  $t$  may be any subsequence  $(i_1, \dots, i_{t-1})$  of  $t - 1$  different numbers in  $\{1, \dots, m\}$ .

We are now ready to formulate the proposed importance sampling. At each iteration  $t \leq m$ , for each particle  $(i)$  and for each  $s \in \overline{\sigma^{(i)}(1:t-1)}$ , we sample

$$x_s^{(i)} \sim p_s(x|x_{\sigma^{(i)}(1:t-1)}^{(i)}), \quad (4.10)$$

where  $\overline{\sigma^{(i)}(1:t-1)} = \{1, \dots, m\} \setminus \sigma^{(i)}(1:t-1)$ , i.e., the indices in  $1:m$  that are not present in  $\sigma^{(i)}(1:t-1)$  for  $t \leq m$ . The subscript  $s$  at the posterior pdf  $p_s$  indicates that we sample values for state  $s$ . We generate at least one sample for each state  $s \in \overline{\sigma^{(i)}(1:t-1)}$ . This means that the single particle  $x_{\sigma^{(i)}(1:t-1)}^{(i)}$  is multiplied and extended to several follower particles  $x_{\sigma^{(i)}(1:t-1),s}^{(i)}$ . Consequently, at iteration  $t < m$  particle  $(i)$  has  $m - t + 1$  followers. Each follower is a sample from a different dimension of the state (i.e., represents a location of a different puzzle piece). Going back to our toy puzzle example, we recall that the current state vector of particle  $(i)$  in Fig. 4.1(c) at time  $t = 2$  is  $x_{\sigma^{(i)}(1:2)}^{(i)} = (a, b)$ , where  $\sigma^{(i)}(1:2) = (3, 1)$ . For sampling at time  $t = 3$ , we have  $\overline{\sigma^{(i)}(1:t-1)} = (2, 4, 5, 6)$ . Consequently, we sample four followers of particle  $(i)$  in (4.10), one for each state  $s = 2, 4, 5, 6$ , where  $x_2^{(i)}$  is the sampled location of puzzle piece 2,  $x_4^{(i)}$  is the sampled location of puzzle piece 4, and so on.

In contrast, in the standard application of rule (4.9), at each iteration  $t$  particle  $(i)$  has one follower. Even when sometimes each particle  $(i)$  has many followers, all followers are samples from the same state, since there is only one unique state at time  $t$ . For our toy example, this means for particle  $(i)$ , only

locations of say puzzle piece 2 are sampled and not those of puzzle piece 4, since a fixed order of the state dimensions is followed in the classical setting.

We do not make any Markov assumption in (4.10), i.e., the new state  $x_s^{(i)}$  is dependent on all previous states  $x_{\sigma(1:t-1)}^{(i)}$  for each particle  $(i)$ .

## 4.4 Particle Filter with State Permutations

Now we are ready to outline the proposed **PF with state permutations (PFSP)** algorithm. In addition to the change is in the importance sampling step, the other two steps are also modified. For every time step  $t = 1, \dots, m$  and for every particle  $i = 1, \dots, N$  execute the following three steps:

1) **Importance sampling / proposal:** Sample followers  $x_s^{(i)}$  of particle  $(i)$  from each dimension  $s \in \overline{\sigma^{(i)}(1:t-1)}$  according to (4.10), which we repeat here for completeness,

$$x_s^{(i)} \sim p_s(x|x_{\sigma(1:t-1)}^{(i)}), \quad (4.11)$$

and set  $x_{\sigma(1:t)}^{(i,s)} = (x_{\sigma(1:t-1)}^{(i)}, x_s^{(i)})$  and  $\sigma^{(i,s)}(t) = s$ , which means that  $\sigma^{(i,s)}(1:t) = (\sigma(1:t-1), s)$ . As stated before, we drop the superscript  $(i, s)$  in  $x_{\sigma(1:t)}^{(i,s)}$ , since it is already present as the particle index.

2) **Importance weighting/evaluation:** An individual importance weight is assigned to each follower particle  $x_{\sigma(1:t)}^{(i,s)}$  according to

$$w(x_{\sigma(1:t)}^{(i,s)}) = w(x_{\sigma(1:t-1)}^{(i)})p(z_s|x_{\sigma(1:t)}^{(i,s)}, z_{\sigma^{(i)}(1:t-1)}), \quad (4.12)$$

3) **Resampling:** Sample with replacement  $N$  new particles from the current set of  $N \times (m - t + 1)$  particles

$$\{x_{\sigma(1:t)}^{(i,s)} \mid i = 1, \dots, N, s \in \overline{\sigma^{(i)}(1:t-1)}\}. \quad (4.13)$$

according to the weights. Thus, we obtain a set of new particles  $\{x_{\sigma(1:t)}^{(i)}\}_{i=1}^N$ . We also renormalize their weights to sum to one. This is a variant of the standard Sampling Importance Resampling (SIR) step [100] as in the classical PF framework.

We observe that the particle weight evaluation in (4.12) is analogous to (4.8) in that the conditional probability of observation  $z_s$  is a function of two corresponding sequences of observations and states plus the state  $x_s$ . The only difference is that the sequences are determined by the permutation  $\sigma^{(i)}(1:t-1)$ .

Sampling more than one follower of each particle and reducing the number of followers by resampling is known in the PF literature as prior boosting [37, 13]. It is used to capture multi-modal likelihood regions. The resampling in our framework plays an additional and a very crucial role. It selects the the most informative orders of states. Since the weights of  $w(x_{\sigma(1:t)}^{(i,s)})$  are determined by the corresponding order of observations  $z_{\sigma^{(i)}(1:t-1)}$ , and the resampling uses the weights to selects new particles  $x_{\sigma(1:t)}^{(i)}$ , the resampling determines the order of state dimensions. Consequently, the order of state dimensions is heavily determined by their corresponding observations, and this order may

be different for each particle ( $i$ ). This is in strong contrast to the classical PF, where observations are considered only in one order  $Z$ .

The fact that each particle explores a possibly different order of dimensions  $\sigma^{(i)}(1 : m)$  is extremely important for the proposed PFSP, since it allows for use of rich set of proposal functions with fewer number of particles. However, at  $t = m$  all state dimensions are present in each sample  $x_{\sigma(1:m)}^{(i)}$ . Hence we can reorder the sequence of state dimensions  $\sigma^{(i)}(1 : m)$  to form the original order  $1 : m$  by applying the inverse permutation  $(\sigma^{(i)})^{-1}$  and obtain  $x_{1:m}^{(i)} = x_{\sigma^{-1}\sigma(1:m)}^{(i)}$ , i.e., the state values are sorted according to the original state indices  $1 : m$  in each sample ( $i$ ). In analogy to Theorem 4.2.1, we state the following

**Theorem 4.4.1.** *Under reasonable assumptions on the sampling (4.11) and weighting functions (4.12) given in [18],  $p(x_{1:m}|Z)$  can be approximated with weighted samples  $\{x_{1:m}^{(i)}, w(x_{\sigma(1:m)}^{(i)})\}_{i=1}^N$  with any precision if  $N$  is sufficiently large. Thus, the convergence in (4.14) is almost sure:*

$$p(x_{1:m}|Z) = \lim_{N \rightarrow \infty} \sum_{i=1}^N w \left( x_{\sigma(1:m)}^{(i)} \right) \delta_{x_{1:m}^{(i)}}(x_{1:m}). \quad (4.14)$$

*Proof.* Due to Th. 4.2.1, we only need to show that  $\{x_{1:m}^{(i)}, w(x_{\sigma(1:m)}^{(i)})\}_{i=1}^N$  represent weighted samples from  $p(x_{1:m}|Z)$ .

The key observation is that  $p$  and  $q$  are probabilities on joint distribution of  $m$  random variables, and as such the order of the random variables is not relevant. This follows from the fact that a joint probability is defined as the



probability of the intersection of the sets representing events corresponding to the value assignments of the random variables, and set intersection is independent of the order of sets. Consequently, we have for every permutation  $\sigma$

$$p(x_{\sigma(1:m)}|Z) = p(x_{1:m}|Z) \quad (4.15)$$

$$q(x_{\sigma(1:m)}|Z) = q(x_{1:m}|Z) \quad (4.16)$$

According to the proposed importance sampling (4.11),  $x_{\sigma(1:m)}^{(i)}$  is a sample from  $q(x_{\sigma(1:m)}|Z)$ . Consequently, by (4.16),  $x_{1:m}^{(i)} = x_{\sigma^{-1}\sigma(1:m)}^{(i)}$  is a sample from  $q(x_{1:m}|Z)$  for each particle  $(i)$ .

By the weight recursion in (4.12), and by (4.15) and (4.16)

$$w\left(x_{\sigma(1:m)}^{(i)}\right) = \frac{p(x_{\sigma(1:m)}^{(i)}|Z)}{q(x_{\sigma(1:m)}^{(i)}|Z)} = \frac{p(x_{1:m}|Z)}{q(x_{1:m}^{(i)}|Z)}. \quad (4.17)$$

Thus  $\{x_{1:m}^{(i)}, w(x_{\sigma(1:m)}^{(i)})\}_{i=1}^N$  represent weighted samples from  $p(x_{1:m}|Z)$ .  $\square$

## 4.5 Implementation Details

In order to utilize the derived PF algorithm to solve the jigsaw puzzle problem, we need to design the proposal pdf in (4.11) and the conditional pdf of a new observation in (4.12). Both are detailed in this section.

Given are a set of  $m$  puzzle pieces  $P = \{1, \dots, m\}$  and a rectangular grid with  $m$  empty squares  $G = \{g_1, \dots, g_m\}$ , e.g., see Fig. 4.1(b,c). In order to

solve the image jigsaw puzzle we need to assign locations on  $G$  to the puzzle pieces in  $P$ . The observation associated with each puzzle piece (of size  $K \times K$ ) is the color information of the partial image depicted on it, i.e.,  $z_i$  is a  $K \times K \times 3$  matrix of pixel color values and the set of observations is  $Z = \{z_1, \dots, z_m\}$ .

A sample particle at time  $t \leq m$  is given by  $x_{\sigma(1:t)} = (x_{\sigma(1)}, \dots, x_{\sigma(t)})$ , where  $\sigma(i) \in P$  and  $x_{\sigma(i)} \in G$ . This means the puzzle piece  $\sigma(i)$  is placed on the grid square with index  $x_{\sigma(i)}$ . The corresponding observations  $z_{\sigma(1:t)} = (z_{\sigma(1)}, \dots, z_{\sigma(t)})$  represents the color information of the partial images on the puzzle pieces. In this section we drop the particle index  $(i)$ , since all definitions apply to each particle.

We now define an affinity matrix  $A$  representing the compatibility of the local images on the puzzle pieces. It is a 3D matrix of size  $m \times m \times 4$  with the third dimension being an adjacency type, since two puzzle pieces can be adjacent in four different ways: left/right, right/left, top/bottom, and bottom/top, which we denote with LR, RL, TB, and BT.

In order to be able to compare our experimental results to the results in [16] we define  $A$  following the definitions in [16]. They first define a dissimilarity-based compatibility  $D$ . Given two puzzle pieces  $j$  and  $i$ ,  $D$  measures dissimilarity between their images  $z_j, z_i$  by summing the squared  $LAB$  color differences

along their boundary, e.g., the left/right (LR) dissimilarity is defined as

$$D(j, i, LR) = \sum_{k=1}^K \sum_{c=1}^3 (z_j(k, u, c) - z_i(k, v, c))^2, \quad (4.18)$$

where  $u$  indexes the last column of  $z_j$  and  $v$  indexes the first column of  $z_i$ .

Finally, the affinity of the LR connection is given by

$$A(j, i, LR) = \exp\left(-\frac{D(j, i, LR)}{2\delta^2}\right), \quad (4.19)$$

where  $\delta$  is adaptively set as the difference between the smallest and the second smallest  $D$  values between puzzle piece  $i$  and all other pieces in  $P$ , see [16] for more details.

**Proposal and weights.** The proposal distribution  $p_s(x|x_{\sigma(1:t-1)}) : G \rightarrow \mathbb{R}$  is a discrete probability distribution of placing puzzle piece  $s$  on each grid square  $x$ .  $p_s(x|x_{\sigma(1:t-1)}) = 0$  if  $x$  is occupied or is not adjacent to any square in  $\sigma(1 : t - 1)$ . Now say  $x$  is free and is adjacent and is to the right of grid square  $x_{\sigma(j)}$  for some  $j = 1, \dots, t$ . Then

$$p_s(x|x_{\sigma(1:t-1)}) \propto A(s, \sigma(j), RL). \quad (4.20)$$

Hence this probability is proportional to the LR similarity between puzzle pieces  $s$  and  $\sigma(j)$ . The definition is analogous for the other three adjacency relations  $LR, TB, BT$ . If square  $x$  is adjacent to more than one grid squares in  $\{x_{\sigma(j)} | j = 1, \dots, t\}$ , then  $p_s(x|x_{\sigma(1:t-1)})$  is proportional to the product of the corresponding  $A$  values.

Let  $x_s$  be a sample from (4.11) at time  $t$ , and as above  $x_s$  is adjacent and is to the right of grid square  $x_{\sigma(j)}$  for some  $j = 1, \dots, t$ . The difference is that  $x_s$  is occupied now with the puzzle piece  $s$ . Then

$$p(z_s | x_{\sigma(1:t)}, z_{\sigma(1:t-1)}) \propto A(s, \sigma(j), RL). \quad (4.21)$$

The definition is analogous for the other three adjacency relations  $LR, TB, BT$ . If square  $x_s$  is adjacent to more than one grid squares in  $\{x_{\sigma(j)} | j = 1, \dots, t\}$ , then  $p(z_s | x_{\sigma(1:t)}, z_{\sigma(1:t-1)})$  is proportional to the product of the corresponding  $A$  values.

To summarize the proposal distribution is a function of how well puzzle piece  $s$  fits to the already placed pieces and assigns the probability of placing  $s$  to all grid squares, while in the evaluation we already know the grid location of puzzle piece  $s$  as well as its adjacent squares. We then use this information to compute the evaluation probability according to  $A$ . Hence, both the proposal and evaluation of a given particle are functions of how well adjacent pieces fit together following the order in which the pieces have been added.

For a given image jigsaw puzzle with  $m$  pieces, the time complexity of the proposed inference framework is  $O(m^2 N)$ , where  $N$  is the number of particles. It follows from the fact that at iteration  $t < m$  particle  $(i)$  has  $m - t + 1$  followers. We set the number of particles  $N = 10$  in all our experiments described in the next section.

## 4.6 Experimental Results

We compare the image jigsaw puzzle solutions obtained by the proposed PF inference framework to the solutions of the loopy believe propagation used in [16] under identical settings. We used the software released by the authors of [16] to obtain their results and also to compute the affinities defined in Section 4.5 used in our approach. The results are compared on the dataset provided in [16], which we call MIT Dataset. It is composed of 20 images. In addition, we also consider an extended dataset composed of 40 images, i.e., we added 20 images. As we will see below, the results of both methods on the original and extended datasets are comparable. Our implementations will be made publicly available on an accompanying website.

The experimental results in [16] are conducted in two different settings: with and without any prior on the target image layout. In [15] the prior of the image layout is given by a low resolution version of the original image. [16] weakens this assumption to a statistics of the possible image layout. We focus on the results without any prior of the image layout. Consequently, we focus on a harder problem, since we only use the pairwise relations between the image patches, given by pair-wise compatibilities of located puzzle pieces as defined in Section 4.5.

In the probabilistic framework in [16], a puzzle piece is assigned to each grid location. In our PF framework, it is more natural to assign a grid location to

each puzzle piece. The solutions of both methods are equivalent, since a final puzzle solution is a set of  $m$  pairs composed of (puzzle piece, grid location), where  $m$  is the number of the puzzle pieces. We call such pairs the solution pairs.

We use three types of evaluation methods introduced in [16]. Each method focuses on different aspects of the quality of the obtained puzzle solutions. The most natural and strictest one is **Direct Comparison**. It simply computes the percentage of correctly placed puzzle pieces, i.e., for a puzzle with  $m$  pieces, Direct Comparison is the number of correct solution pairs divided by  $m$ . A less stricter measure is **Cluster Comparison**. It tolerates an assignment error as long as the puzzle piece is assigned to a location that belongs to a similar puzzle piece. The puzzle pieces are first clustered into groups of similar pieces. Moreover, due to lack of prior knowledge of target image, the reconstructed image may be shifted compared to the ground truth image. Therefore, a third measure called **Neighbor Comparison** is used to evaluate the label consistency of adjacent puzzle pieces independent of their grid location, e.g., the location of two adjacent puzzle pieces is considered correct if two puzzle pieces are left-right neighbors in the ground truth image and they are also left-right neighbors in the inferred image. Neighbor Comparison is the fraction of correct adjacent puzzle pieces. This measurement does not penalize the accuracy as long as the adjacent patches in original image are adjacent in the

reconstructed image.

The results on the MIT Dataset are shown in Table 4.1 and on the extended dataset in Table 4.2. The proposed PF inference framework significantly outperforms the loopy believe propagation in all three performance measures. Moreover, the reconstruction accuracy (according to the most natural measure, Direct Comparison) of the original images by our algorithm is improved three times.

In order to demonstrate that the considered image jigsaw puzzle problem is also very challenging to humans, we show some example results in Fig. 4.2. There we show the original images, but we would like to emphasize that the original images are not used during the inference. Fig. 4.2 also demonstrates that the reconstructed images obtained by the proposed algorithm compare very favorably to the results of [16]. In order to demonstrate the dynamic of the proposed PF inference, we show reconstructed images of the best particle at different times (iterations) in Fig. 4.3.

Both methods are initialized with one anchor patch, i.e., with one correct (puzzle piece, grid location) pair. We always assign a correct image patch to the top left corner of the image. In all our experiments we divide each test image into 108 square patches resulting in  $m = 108$  puzzle pieces.

	[16]	Our algorithm
Direct Comparison	0.2366	0.6921
Cluster Comparison	0.4657	0.7810
Neighbor Comparison	0.6628	0.8620

Table 4.1: Experimental results on MIT Dataset.

	[16]	Our algorithm
Direct Comparison	0.2137	0.7097
Cluster Comparison	0.4500	0.8018
Neighbor Comparison	0.6458	0.8770

Table 4.2: Experimental results on the extended dataset.



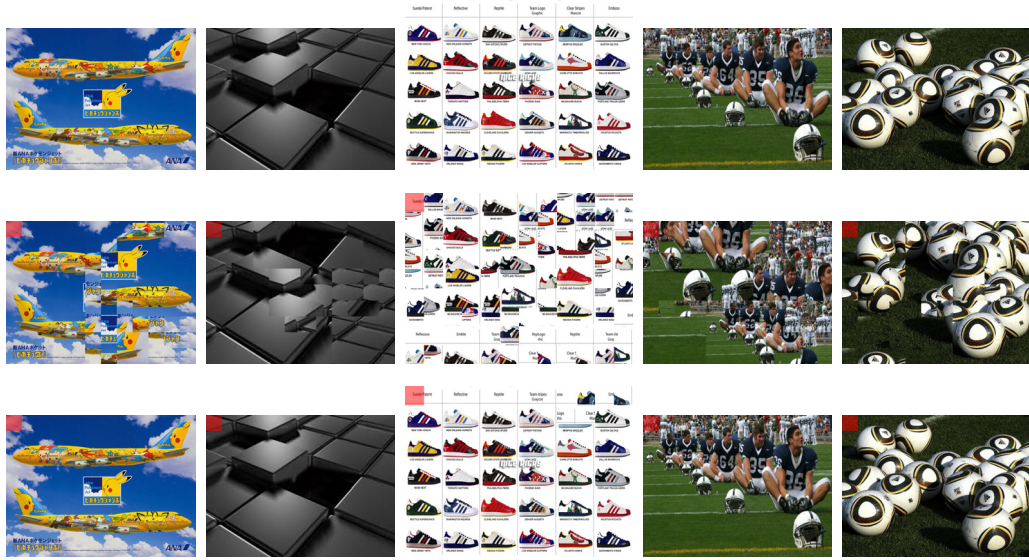


Figure 4.2: First row: the original images. Second row: the jigsaw puzzle solutions of [16]. Third row: our solutions.

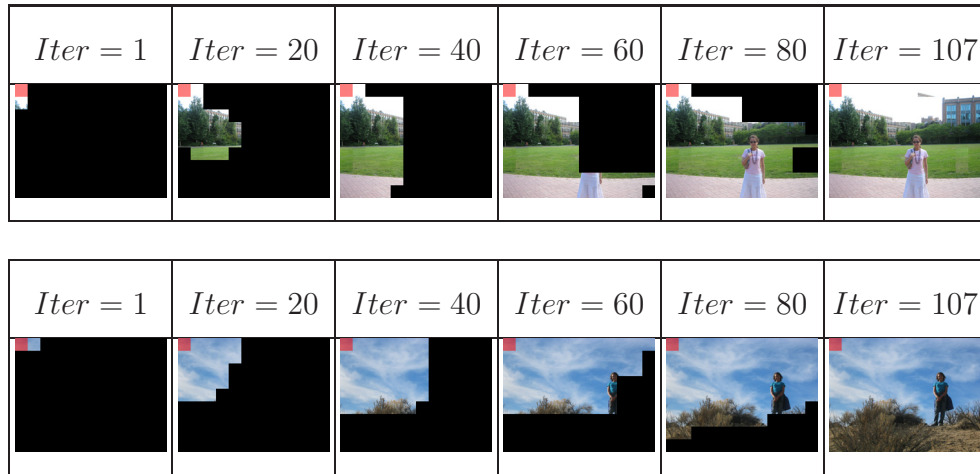


Figure 4.3: The reconstructed images of the best particle at different iterations.

# CHAPTER 5

## Conclusion

The body of work here addresses many problems in Shape retrieval, Shape Based Object Detection and Jigsaw puzzles.

Firstly, we adapted a graph transductive learning framework to learn new distances with the application to shape retrieval, shape classification, and shape clustering. The key idea is to replace the distances in the original distance space with distances induced by geodesic paths in the shape manifold. The merits of the proposed technique have been validated by significant performance gains in all presented experimental results. However, like semi-supervised learning, if there are too many outlier shapes in the shape database, the proposed approach may not be able to improve the results.

Secondly, we introduce an innovative method for inserting synthetic points into data sets. Unlike other feature based methods, our synthetic points, which

we call ghost points, are added in distance space. Using geometric analysis, we show that the distances are preserved between the ghost points the other points in the distance space. We use ghost points in different domains (mining imbalanced time series data sets and shape similarity and retrieval), and use different performance metrics to show the broad application of ghost points. For imbalanced data sets, adding ghost points to the minority class improved the overall classification accuracy of SVMs on most data sets, and significantly improved the precision and recall rate for minority classes. In shape retrieval, adding ghost points densified the underlying shape manifold, allowing graph transduction algorithms to take advantage of the densified manifold. With ghost points, we obtained the highest ever bull’s eye score on the MPEG-7 data set at that time.

Thirdly, the proposed framework for shape and image retrieval can be applied whenever original pairwise distances/similarities cannot perfectly rank the database objects. The key advantage of the proposed Tensor Product Graph diffusion is the utilization of higher order similarity relations, which are both local and long range. Usually higher order relations lead to a substantially higher computation cost. However, we are able to introduce an iterative algorithm to compute TPG diffusion that has the same space and time complexity as the classical diffusion on the original graph. We also provide a formal proof that the iterative algorithm and the TPG diffusion converge to

the same solution. Hence the proposed TPG diffusion explores the benefits of higher order relations without the price of higher computational cost.

Fourthly, the shape based object detection mainly contains two contributions: shape model learning through shape matching and a novel framework for shape based object detection. The proposed model learning method can not only learn the model for non-rigid or articulated objects with partially-supervised learning, but also transfer the structure information to different kinds of objects. More importantly, the spatial layout between parts is also modeled. We extend the classical particle filter framework in order to be able to infer an optimal label assignment to RVs whose dependencies are described by a complete graph. The values of RVs represent contour parts of our shape model and their locations. In our framework each particle explores a different order of detected contour parts, and the most informative order is selected by particle resampling. Presented experimental results demonstrate that the proposed approach can not only detect the objects but also correctly localize object boundaries, which is very crucial in many applications like pose estimation or object manipulation. Although the proposed method can perform well on three widely used datasets, it has a common problem with other shape matching based object detection methods. All of them require that edges present in the edge map contain reasonable parts of true object contours, which is still an open problem in computer vision, in particular, for low

resolution images.

Finally, we introduce a novel inference framework for solving image jigsaw puzzle problem. Our key contribution is an extension of the PF framework to work with unordered observations. Weighted particles explore the state space along different dimensions in different orders, and state permutations that yield most descriptive proposal functions are selected as new particles. By exploiting the equivalence of importance sampling under state permutations, we prove that the obtained importance samples represent samples from the original target distribution. We evaluate the performance of the proposed PF inference on a problem of image jigsaw puzzles. As the experimental results demonstrate, it significantly outperforms the loopy belief propagation. Image jigsaw puzzle problem is an instance of labeling (assignment) problem. Therefore, our future work will focus on a broader spectrum of labeling problems.

# REFERENCES

- [1] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *ECML*, pages 39–50, 2004.
- [2] N. Alajlan, M.S. Kamel, and G. Freeman. Geometry-based image retrieval in binary image databases. *IEEE Trans. on PAMI*, 30(6):1003–1013, 2008.
- [3] Axel Pinz, Andreas Opelt, and Andrew Zisserman. A boundary-fragment-model for object detection. In *ECCV*, 2006.
- [4] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. Bootmap: A method for efficient approximate similarity rankings. In *CVPR*, 2004.
- [5] Xiang Bai and Longin Jan Latecki. Path similarity skeleton graph matching. *IEEE Trans. PAMI*, 30(7):1282–1292, 2008.

- [6] Xiang Bai, Xinggang Wang, Longin Jan Latecki, and Zhuowen Tu. Active skeleton for non-rigid object detection. In *ICCV*, 2009.
- [7] Xiang Bai, Xingwei Yang, Longin Jan Latecki, Wenyu Liu, and Zhuowen Tu. Learning context sensitive shape similarity by graph transduction. *IEEE Trans. on PAMI*, 2010.
- [8] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning distance functions using equivalence relations. In *ICML*, pages 11–18, 2003.
- [9] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. PAMI*, 24:705–522, 2002.
- [10] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Trans. PAMI*, 11(6):567–585, 1989.
- [11] E. Borenstein, E. Sharon, and S. Ullman. Combining top-down and bottom-up segmentation. In *POVC*, 2004.
- [12] U. Brefeld, C. Buscher, and T. Scheffer. Multiview discriminative sequential learning. In *ECML*, 2005.
- [13] J. Carpenter, P. Clifford, and P. Fearnhead. Building robust simulation-based filters for evolving data sets, 1999.
- [14] Nitesh V. Chawla, Kevin W. Bowyer, and W. Philip Kegelmeyer. Smote:

- Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
- [15] T. S. Cho, M. Butman, S. Avidan, and W. T. Freeman. The patch transform and its applications to image editing. In *CVPR*, 2008.
- [16] Taeg Sang Cho, Shai Avidan, and William T. Freeman. A probabilistic image jigsaw puzzle solver. In *CVPR*, 2010.
- [17] R.R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21:5–30, 2006.
- [18] D. Crisan and A. Doucet. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on Signal Processing*, 50(3):736–746, 2002.
- [19] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.
- [20] M.R. Daliri and V. Torre. Robust symbolic representation for shape recognition and retrieval. *Pattern Recognition*, 41(5):1799–1815, 2008.
- [21] E. D. Demaine and M. L. Demaine. Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics*, 2007.



- [22] C. Desai, D. Ramanan, and C. Fowlkes. Discriminative models for multi-class object layout. In *ICCV*, 2009.
- [23] A. Doucet, N. D. Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2001.
- [24] I. Dryden. *Statistical shape analysis*. Wiley, 1998.
- [25] X. Fan, C. Qi, D. Liang, and H. Huang. Probabilistic contour extraction using hierarchical shape representation. In *Proc. ICCV*, pages 302–308, 2005.
- [26] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE PAMI*, 28:594–611, 2006.
- [27] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1), 2005.
- [28] Pedro F. Felzenszwalb and Joshua Schwartz. Hierarchical matching of deformable shapes. In *CVPR*, 2007.
- [29] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. From images to shape models for object detection. *IEEE Trans. PAMI*, 30(1):36–51, 2008.
- [30] V. Ferrari, F. Jurie, and C. Schmid. From images to shape models for object detection. *IJCV*, accepted.

- [31] D. Fox, S. Thrun, F. Dellaert, and W. Burgard. Particle filters for mobile robot localization. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*. Springer Verlag, New York, 2000.
- [32] H. Freeman and L. Garder. Apictorial jigsaw puzzles: the computer solution of a problem in pattern recognition. *IEEE TEC*, 13:118–127, 1964.
- [33] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.
- [34] Juergen Gall and Victor Lempitsky. Class-specific hough forests for object detection. In *CVPR*, 2009.
- [35] D. Goldberg, C. Malon, and M. Bern. A global approach to automatic solution of jigsaw puzzles. In *Symposium on Computational Geometry*, 2002.
- [36] Raghuraman Gopalan, Pavan Turaga, and Rama Chellappa. Articulation-invariant representation of non-planar shapes. In *ECCV*, 2010.
- [37] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to

- nonlinear/non-gaussian bayesian state estimation. In *Radar and Signal Processing, IEE Proceedings of*, volume 140, pages 107–113, 1993.
- [38] L. Gorelick, M. Galun, E. Sharon, R. Basri, and A. Brandt. Shape representation and classification using the poisson equation. *IEEE Trans. PAMI*, 28(12):1991–2005, 2006.
- [39] C. Grigorescu and N. Petkov. Distance sets for shape filters and shape recognition. *IEEE Trans. on Image Processing*, 12(7):729–739, 2003.
- [40] Chunhui Gu, Joseph J. Lim, Pablo Arbelaez, and Jitendra Malik. Recognition using regions. In *CVPR*, 2009.
- [41] M. Hein and M. Maier. Manifold denoising. In *NIPS*, 2006.
- [42] T. Hertz, A. Bar-Hillel, and D. Weinshall. Learning distance functions for image retrieval. In *CVPR*, pages 570–577, 2004.
- [43] Yuchi Huang, Qingshan Liu, and Dimitris Metaxas. Video object segmentation by hypergraph cut. In *CVPR*, 2009.
- [44] S. Ioffe and D. Forsyth. Probabilistic methods for finding people. *Int. J. Comput. Vision*, 43:45–68, June 2001.
- [45] Michael Isard and Andrew Blake. Contour tracking by stochastic propagation of conditional density. In *Proc. European Conf. on Computer Vision*, ECCV, pages 343–356, 1996.

- [46] Herve Jegou, Cordelia Schmid, Hedi Harzallah, and Jakob Verbeek. Accurate image search using the contextual dissimilarity measure. *IEEE PAMI*, 2010.
- [47] Tingting Jiang, Frederic Jurie, and Cordelia Schmid. Learning shape prior models for object matching. In *CVPR*, 2009.
- [48] Matousek Jiri. *Lectures on Discrete Geometry*. Springer, 2002.
- [49] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, pages 200–209, 1999.
- [50] E. Keogh. UCR time series classification/clustering page. In [http://www.cs.ucr.edu/~eamonn/time\\_series\\_data/](http://www.cs.ucr.edu/~eamonn/time_series_data/).
- [51] Zia Khan, Tucker Balch, and Frank Dellaert. An mcmc-based particle filter for tracking multiple interacting targets. In *ECCV*, 2004.
- [52] Iasonas Kokkinos and Alan Yuille. Hop: Hierarchical object parsing. In *CVPR*, 2009.
- [53] W. Kong and B. B. Kimia. On solving 2d and 3d puzzles using curve matching. In *CVPR*, 2001.
- [54] P. Kotschieder, M. Donoser, and H. Bischof. Beyond pairwise shape similarity analysis. In *ACCV*, 2009.

- [55] S. Lafon and A. B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction graph partitioning, and data set parameterization. *IEEE PAMI*, 28:1393–1403, 2006.
- [56] Peter Lancaster and Leiba Rodman. *Algebraic Riccati Equations*. Clarendon Press, Oxford, 1995.
- [57] L. J. Latecki and R. Lakämper. Shape similarity measure based on correspondence of visual parts. *IEEE Trans. PAMI*, 22(10):1185–1190, 2000.
- [58] L. J. Latecki, R. Lakämper, and U. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *CVPR*, pages 424–429, 2000.
- [59] N. D. Lawrence and M. I. Jordan. Semi-supervised learning via gaussian processes. In *NIPS*, 2004.
- [60] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.
- [61] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Proceedings of the*

*Workshop on Statistical Learning in Computer Vision*, Prague, Czech Republic, May 2004.

- [62] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *CVPR*, 2003.
- [63] H. Ling and D.W. Jacobs. Shape classification using the inner-distance. *IEEE Trans. PAMI*, 29(2):286–299, 2007.
- [64] Haibin Ling, Xingwei Yang, and Longin Jan Latecki. Balancing deformability and discriminability for shape matching. In *ECCV*, 2010.
- [65] J. Liue. *Monte Carlo strategies in Scientific Computing*. Springer Verlag, 2001.
- [66] David Lowe. Distinctive image features from scale-invariant key points. *IJCV*, 60:91–110, 2004.
- [67] Chengen Lu, Longin Jan Latecki, Nagesh Adluru, Xingwei Yang, and Haibin Ling. Shape guided contour grouping with particle filters. In *ICCV*, 2009.
- [68] S. Maji and J. Malik. Object detection using a max-margin hough transform. In *CVPR*, 2009.
- [69] M. Makridis and N. Papamarkos. A new technique for solving a jigsaw puzzle. In *ICIP*, 2006.

- [70] G. McNeill and S. Vijayakumar. 2d shape classification and retrieval. In *IJCAI*, 2005.
- [71] G. McNeill and S. Vijayakumar. Hierarchical procrustes matching for shape retrieval. In *Proc. CVPR*, 2006.
- [72] Krystian Mikolajczyk and Cordelia Schmid. Scale and affine invariant interest point detectors. *IJCV*, 60(1):63–86, 2004.
- [73] F. Mokhtarian, F. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space. *Image Databases and Multi-Media Search*, A.W.M Smeulders and R. Jain eds, pages 51–58, 1997.
- [74] BenJamin B. Kimia Nhon H. Trinh. Category-specific object recognition and segmentation using a skeletal shape model. In *BMVC*, 2009.
- [75] T. R. Nielsen, P. Drewsen, and K. Hansen. Solving jigsaw puzzles using image features. *PRL*, 2008.
- [76] Bjorn Ommer and Jitendra Malik. Multi-scale object detection by clustering lines. In *ICCV*, 2009.
- [77] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Stanford Digital Libraries Working Paper*, 1998.

- [78] M. Pavan and M. Pelillo. Dominant sets and pairwise clustering. *IEEE Trans. PAMI*, 2007.
- [79] Ronaldo C. Prati and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *Sigkdd Explorations*, 6:20–29, 2004.
- [80] G. Radack and N. Badler. Jigsaw puzzle matching using a boundary-centered polar encoding. In *CGIP*, 1982.
- [81] C. A. Ratanamahatana and E. Keogh. Three myths about dynamic time warping. In *SDM*, pages 506–510, 2005.
- [82] Saiprasad Ravishankar, Arpit Jain, and Anurag Mittal. Multi-stage contour based detection of deformable objects. In *ECCV*, 2008.
- [83] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.
- [84] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.
- [85] T. Sebastian, P. Klein, and B. Kimia. On aligning curves. *IEEE Trans. PAMI*, 25:116–125, 2003.
- [86] T. B. Sebastian, P. N. Klein, and B. B. Kimia. Recognition of shapes by editing their shock graphs. *IEEE Trans. PAMI*, 25:116–125, 2004.



- [87] A. Shokoufandeh, D. Macrini, S. Dickinson, K. Siddiqi, and S. W. Zucker. Indexing hierarchical structures using graph spectra. *IEEE Trans. PAMI*, 27(7):1125–1140, 2005.
- [88] Jamie Shotton, Andrew Blake, and Roberto Cipolla. Multi-scale categorical object recognition using contour fragments. *IEEE Trans. on PAMI*, 30(7):1270–1281, 2008.
- [89] Jamie Shotton, John M. Winn, Carsten Rother, and Antonio Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, 2006.
- [90] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker. Shock graphs and shape matching. *Int. J. of Computer Vision*, 35:13–32, 1999.
- [91] Kevin Smith, Daniel Gatica-Perez, and Jean-Marc Odobez. Using particles to track varying numbers of interacting people. In *CVPR*, 2005.
- [92] O. Soderkvist. *Computer vision classification of leaves from swedish trees*. Master’s thesis, Linkoping University, 2001.
- [93] A. Srivastava, S. H. Joshi, W. Mio, and X. Liu. Statistic shape analysis: clustering, learning, and testing. *IEEE Trans. PAMI*, 27:590–602, 2005.
- [94] Michael Stark, Michael Goesele, and Bernt Schiele. A shape-based object class model for knowledge transfer. In *ICCV*, 2009.

- [95] H. Stewenius and D. Nister. Object recognition benchmark.  
<http://vis.uky.edu/stewe/ukbench/>.
- [96] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *NIPS*, 2001.
- [97] Andrew Temlyakov, Brent C. Munsell, Jarrell W. Waggoner<sup>1</sup>, and Song Wang. Two perceptually motivated strategies for shape classification. In *CVPR*, 2010.
- [98] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.
- [99] S. Thrun. Particle filters in robotics. In *Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)*, 2002.
- [100] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press Cambridge, 2005.
- [101] Z. Tu and A. L. Yuille. Shape matching and recognition - using generative models and informative features. In *ECCV*, pages 195–209, 2004.
- [102] Charles van Loan. The ubiquitous kronecker product. *J. of Computational and Applied Math.*, 123:85–100, 2000.

- [103] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.
- [104] J. Vleugels and R. Velkamp. Efficient image retrieval through vantage objects. *Pattern Recognition*, 35 (1):69–80, 2002.
- [105] F. Wang, J. Wang, C. Zhang, and H. Shen. Semi-supervised classification using linear neighborhood propagation. In *CVPR*, 2006.
- [106] Jun Wang, Shih-Fu Chang, Xiabo Zhou, and T. C. Stephen Wong. Active microscopic cellular image annotation by superposable graph transduction with imbalanced labels. In *CVPR*, 2008.
- [107] J. Weibull. *Evolutionary game theory*. MIT Press, 1997.
- [108] H. Wolfson, E. Schonberg, A. Kalvin, and Y. Lamdam. Solving jigsaw puzzles by computer. *Annals of Operations Research*, 1988.
- [109] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *NIPS*, pages 505–512, 2003.
- [110] X. Yang, X. Bai, L. J. Latecki, and Z. Tu. Improving shape retrieval by learning graph transduction. In *ECCV*, 2008.
- [111] X. Yang, S. Köknar-Tezel, and L. J. Latecki. Locally constrained dif-

- fusion process on locally densified distance spaces with applications to shape retrieval. In *CVPR*, 2009.
- [112] Xingwei Yang, Nagesh Adluru, and Longin Jan Latecki. Particle filter with state permutations for solving image jigsaw puzzles. In *CVPR*, 2011.
- [113] Xingwei Yang and Longin Jan Latecki. Weakly supervised shape based object detection with particle filter. In *ECCV*, 2010.
- [114] Xingwei Yang and Longin Jan Latecki. Affinity learning on a tensor product graph with applications to shape and image retrieval. In *CVPR*, 2011.
- [115] F. H. Yao and G. F. Shao. A shape and image merging technique to solve jigsaw puzzles. *PRL*, 2003.
- [116] J. Yu, J. Amores, N. Sebe, P. Radeva, and Q. Tian. Distance learning for similarity estimation. *IEEE Trans. PAMI*, 30:451–462, 2008.
- [117] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *NIPS*, 2004.
- [118] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Scholkopf. Learning with local and global consistency. In *NIPS*, 2003.

- [119] D. Zhou, J. Huang, and B. Scholkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *NIPS*, 2007.
- [120] D. Zhou, J. Weston, A. Gretton, Q. Bousquet, and B. Scholkopf. Ranking on data manifolds. In *NIPS*, 2003.
- [121] Long Zhu, Yuanhao Chen, and Alan Yuille. Learning a hierarchical deformable template for rapid deformable object parsing. *IEEE Trans. PAMI*, 99(1), 2009.
- [122] Q. Zhu, L. Wang, Y. Wu, and J. Shi. Contour context selection for object detection: a set-to-set contour matching approach. In *ECCV*, 2008.
- [123] X. Zhu. Semi-supervised learning with graphs. In *Doctoral Dissertation*, pages Carnegie Mellon University, CMU-LTI-05-192, 2005.
- [124] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003.