Computer Science Department
School of Sciences and Technologies
University of Crete

Deformable 2D Shape Matching based on Shape Contexts and Dynamic Programming

Master's Thesis

Oikonomidis Iasonas

March 2009
Heraklion, Greece

Τμήμα Επιστήμης Υπολογιστών
Σχολή Θετικών και Τεχνολογικών Επιστημών
Πανεπιστήμιο Κρήτης

Ταίριασμα Παραμορφώσιμων 2Δ Σχημάτων με Χρήση Τοπικών Περιγραφών Σχήματος και Δυναμικό Προγραμματισμό

Μεταπτυχιακή Εργασία που εκπονήθηκε από τον
Ιάσονα Οικονομίδη
ως μερική εκπλήρωση των απαιτήσεων για το
Μεταπτυχιακό Δίπλωμα Ειδίκευσης στην Επιστήμη Υπολογιστών

ΕΓΚΡΙΣΗ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Συγγραφέας: _____
Ιάσονας Οικονομίδης

Έγκριση Επιτροπής: _____
Αντώνης Αργυρός
Αναπληρωτής Καθηγητής, Επόπτης Εργασίας

_____
Πάνος Τραχανιάς
Καθηγητής

_____
Ξενοφώντας Ζαμπούλης
Ερευνητής ΙΙΙ-ΙΤΕ

Έγκριση Τμήματος: _____
Πάνος Τραχανιάς
Καθηγητής
Διευθυντής Μεταπτυχιακών Σπουδών

23 Μαρτίου 2009
Ηράκλειο, Ελλάδα

Αφιερωμένο

σε όλους εκείνους

που με εμπνέουν

Abstract

Shape matching is an important problem of computer vision and pattern recognition which can be defined as the establishment of a similarity measure between shapes and its use for shape comparison. A byproduct of this task might also be the estimation of point correspondences between shapes. The problem has significant theoretical interest. Producing shape matching results that are intuitively correct for humans is a demanding problem that remains unsolved in its full generality. Applications of shape matching include but are not limited to object detection and recognition, content based retrieval of images, and image registration.

In this work, it is assumed that a 2D shape is well described as a single closed contour. We present a method for deformable shape matching based on shape contexts, a shape descriptor that has proven to be useful in many applications. Under the mentioned assumption, it is efficient to match two shapes in sub-cubic runtime using a recently published algorithm that performs cyclic string matching employing an efficient iterative scheme based on dynamic programming. The representation power of shape contexts combined with the capability of the matching algorithm to exploit the order in which points appear on a certain contour, result in an effective shape matching method.

Several experiments have been carried out to assess the effectiveness and the performance of the proposed method on several benchmark data sets. The method is quantitatively assessed through the Bull's Eye test applied to the MPEG-7 CE-shape-1 part B data set. Additional shape retrieval experiments have been carried out on the "gestures" and "marine" datasets. Additionally, the proposed shape matching method has been used to detect the articulation points of a human figure in monocular image sequences. More specifically, twenty five human postures have been manually annotated with the articulation points of the human figure. Shape matching between a segmented human figure and the prototype postures results in point correspondences between the human figure and its best matching prototype. These correspondences guide the deformation of a thin plate spline that enables the identification of the articulation points on the human figure.

Overall, the experimental results demonstrate that the proposed method performs very satisfactory in diverse shape matching applications. Additionally, its low computational complexity makes it a good candidate in shape matching applications requiring real-time performance.

Περίληψη

Το ταίριασμα σχημάτων αποτελεί ένα σημαντικό πρόβλημα της υπολογιστικής όρασης και της αναγνώρισης προτύπων το οποίο μπορεί να οριστεί ως ο προσδιορισμός ενός κριτηρίου ομοιότητας μεταξύ σχημάτων και η χρησιμοποίησή του για τη σύγκρισή τους. Πιθανό υποπροϊόν της διαδικασίας ταιριάσματος σχημάτων είναι ζεύγη αντιστοιχιών μεταξύ σημείων τους. Το εν λόγω πρόβλημα έχει σημαντικό θεωρητικό ενδιαφέρον. Το υπολογιστικό ταίριασμα σχημάτων με τρόπο που να ικανοποιεί πλήρως την ανθρώπινη διαίσθηση αποτελεί πρόβλημα που δεν έχει επιλυθεί στην πλήρη του γενικότητα. Από την οπτική γωνία των εφαρμογών, το ταίριασμα σχημάτων μπορεί να χρησιμοποιηθεί για την ανίχνευση και την αναγνώριση αντικειμένων, την ανάκληση εικόνων με βάση το περιεχόμενό τους, κ.ά.

Στην εργασία αυτή, γίνεται η υπόθεση ότι ένα 2Δ σχήμα περιγράφεται πλήρως από ένα, μοναδικό κλειστό περίγραμμα. Περιγράφεται μία μέθοδος ταιριάσματος παραμορφώσιμων σχημάτων βασισμένη στην αναπαράσταση των ès hape contextsŋ, μία τοπική περιγραφή του σχήματος που έχει αποδειχτεί αποτελεσματική σε διάφορες εφαρμογές. Κάτω από την υιοθετούμενη υπόθεση, είναι ιδιαίτερα αποδοτικό το ταίριασμα δύο σχημάτων με τη χρήση ενός πρόσφατα δημοσιευμένου αλγορίθμου για το κυκλικό ταίριασμα συμβολοσειρών. Ο χρησιμοποιούμενος αλγόριθμος χρησιμοποιεί ένα αποδοτικό, επαναληπτικό σχήμα βασισμένο στο Δυναμικό Προγραμματισμό. Η εκφραστική δύναμη των shape contexts, συνδυασμένη με την δυνατότητα του αλγορίθμου ταιριάσματος να εκμεταλλεύεται τη διάταξη με την οποία τα σημεία εμφανίζονται σε ένα περίγραμμα, έχουν ως αποτέλεσμα ένα αποδοτικό αλγόριθμο ταιριάσματος σχημάτων.

Ένα εκτεταμένο σύνολο πειραμάτων διεξάχθηκε προκειμένου να αξιολογηθεί ποσοτικά και ποιοτικά η αποτελεσματικότητα της προτεινόμενης μεθόδου σε μία σειρά από καθιερωμένα σύνολα πειραματικών δεδομένων. Η προτεινόμενη μέθοδος αποτιμάται ποσοτικά με βάση το Bull's Eye τεστ το οποίο εφαρμόζεται στο MPEG-7 CE-shape-1 part B σύνολο δεδομένων. Επιπρόσθετα πειράματα ανάκλησης σχημάτων εκτελέστηκαν με βάση τα σύνολα δεδομένων "gestures" και "marine". Η προτεινόμενη μέθοδος ταιριάσματος σχημάτων εφαρμόστηκε επίσης προκειμένου να επιλυθεί το πρόβλημα εντοπισμού των αρθρώσεων μιας ανθρώπινης φιγούρας σε ακολουθίες εικόνων μιας κάμερας. Πιο συγκεκριμένα, οι αρθρώσεις του ανθρώπινου σώματος προσδιορίστηκαν σε εικοσιπέντε μοντέλα ανθρώπινων

φιγούρων που απεικονίζουν το ανθρώπινο σώμα σε διαφορετικές στάσεις. Το ταίριασμα σχήματος μεταξύ μιας τμηματοποιημένης ανθρώπινης φιγούρας και των μοντέλων έχει ως αποτέλεσμα αντιστοιχίσεις των περιγραμμάτων της ανθρώπινης φιγούρας και του πλησιέστερου μοντέλου. Αυτές οι αντιστοιχίσεις καθοδηγούν την παραμόρφωση ενός thin plate spline η οποία επιτρέπει τον εντοπισμό των αρθρώσεων στη φιγούρα εισόδου.

Συνολικά, τα πειραματικά αποτελέσματα δείχνουν ότι η προτεινόμενη μέθοδος λειτουργεί αποτελεσματικά σε ποικιλία εφαρμογών ταιριάσματος σχημάτων. Επιπρόσθετα, οι χαμηλές υπολογιστικές της απαιτήσεις την κάνουν μία καλή επιλογή σε εφαρμογές όπου απαιτείται ταίριασμα 2Δ σχημάτων σε πραγματικό χρόνο.

Ευχαριστίες

Νιώθω την ανάγκη να ευχαριστήσω κάποιους ανθρώπους οι οποίοι, με τον δικό τους τρόπο ο καθένας, έχουνε συμβάλει στην προσπάθειά μου για την παρούσα δουλειά, αλλά και στην γενικότερη ως τώρα πορεία μου έως την απόκτηση αυτού του τίτλου.

Καταρχήν θέλω να ευχαριστήσω τους γονείς μου, Νίκο και Μαρία, που με έχουνε στηρίξει με κάθε διαθέσιμο μέσο και σε κάθε φάση της ζωής μου μέχρι σήμερα, δίνοντας μου την δυνατότητα αλλά και την παρότρυνση να κυνηγήσω και να επιτύχω τα όνειρά μου.

Δε θα μπορούσα να παραλείψω τα αδέρφια μου Χλόη, Παύλο και Χριστιάνα, με τα οποία παίξαμε, γελάσαμε, και μοιραστήκαμε τα παιδικά και εφηβικά μας χρόνια. Πάντοτε θα είναι τρεις δικοί μου άνθρωποι.

Στη σύντροφό μου Ελευθερία οφείλω ένα μεγάλο ευχαριστώ για τη στήριξη, τη συμπαράστασή της, και για το γεγονός ότι στάθηκε δίπλα μου σε όλη τη διάρκεια αυτής μου της προσπάθειας και όχι μόνο. Οι δικές της επιδιώξεις είναι πιο απαιτητικές από τις ως τώρα δικές μου, αλλά πάντα βρίσκει το κουράγιο να αντιμετωπίζει τις όποιες δυσκολίες με ένα χαμόγελο.

Οφείλω ένα θερμό ευχαριστώ στο Αντώνη Αργυρό, επόπτη αυτής της εργασίας. Αποτέλεσε μια πηγή ιδεών μέσα από ουσιαστικές συζητήσεις. Δεν περιορίστηκε μόνο στα τυπικά του καθήκοντα, είχε πάντοτε την καλή διάθεση να ακούσει οποιοδήποτε μικρό η μεγάλο πρόβλημα αντιμετώπιζα. Εκτός των άλλων, του οφείλω ένα ευχαριστώ και για την εμπιστοσύνη που έδειξε στο πρόσωπό μου, και πιστεύω ότι θα φανώ αντάξιος αυτής και στη συνέχεια της συνεργασίας μας.

Οφείλω ένα ευχαριστώ και στα άλλα δυο μέλη της τριμελούς επιτροπής μου, τον Πάνο Τραχανιά και τον Φώντα Ζαμπούλη, αν μη τι άλλο γιατί αφιέρωσαν σε εμένα χρόνο τους, χωρίς να έχουνε την τυπική υποχρέωση να το κάνουν.

Το εργαστήριο Υπολογιστικής Όρασης και Ρομποτικής αλλά και γενικότερα το Ίδρυμα Τεχνολογίας και Έρευνας είναι για μένα ένα φιλόξενο περιβάλλον. Συμφοιτητές, μέλη του εργαστηρίου, αλλά και άνθρωποι από όλο το ίδρυμα έχουνε κάνει την ύπαρξή μου σε αυτό τον χώρο κάτι παραπάνω από άνετη. Ιδιαίτερες ευχαριστίες θα πρέπει να δώσω στο Νίκο γιατί, εκτός των άλλων, είχε τη μεγαλύτερη συμβολή σχετικά με τη δουλειά αυτή. Οφείλω όμως ένα ευχαριστώ σε όλα ανεξαιρέτως τα μέλη του εργαστηρίου αλλά και σε ανθρώπους πέρα από αυτό. Για μικρότερους ή μεγαλύτερους λόγους, λιγότερο

ή περισσότερο σχετικούς με το περιεχόμενο αυτής της εργασίας, ονομαστικά πρέπει να αναφερθούν: ο Μάρκος, ο Μάνος, ο Γιώργος, ο Γιάννης, η Μαρία, ο Θωμάς, η άλλη Μαρία, ο Μιχάλης, ο άλλος Μιχάλης, ο Μανώλης, ο Άκης, ο Χάρης, ο Στάθης, η Λένα, ο άλλος Γιώργος, η Χριστίνα, η Σοφία, η Κατερίνα και ο Βασίλης.

Στην παιδεία μου έχει συμβάλλει το Πανεπιστήμιο Κρήτης, και οφείλω ένα ευχαριστώ σε μέλη του Τμήματος Επιστήμης Υπολογιστών, αλλά και του Μαθηματικού τμήματος, από το οποίο πήρα το πρώτο μου πτυχίο. Παλαιότερα ωστόσο από το Πανεπιστήμιο Κρήτης, καθοριστική για μένα υπήρξε η επιρροή τριών ανθρώπων: του Γιώργου Κώστα, του Λουκά Κώστα, και του Νικήτα Χαντζηκωνσταντίνου που μου έδωσαν γερές βάσεις για την ως τώρα σταδιοδρομία μου.

Τέλος θέλω να πω ένα ευχαριστώ σε φίλους που, με το δικό του τρόπο ο καθένας, έχουνε συμβάλλει στο να είμαι τελικά αυτός που είμαι. Καταρχήν ο Λευτέρης, που με τα κοινά μας ενδιαφέροντα αλλά και τη διαφορετική, σε αρκετές περιπτώσεις, προσέγγισή μας σε αυτά, πάντα μου δίνει μια φρέσκια και συνήθως ανατρεπτική άποψη για τα πράγματα. Ο Χρήστος πάντα είναι πηγή καλής διάθεσης, και μιας δόσης τρέλας. Ο Αντώνης είναι ένα σταθερό σημείο αναφοράς, που μπορεί να ακούσει, να καταλάβει, αλλά και να δώσει άποψη. Ο Τάσος, ο Αποστόλης κι ο Γιώργος είναι τρεις ακόμα φίλοι που ξέρω ότι μπορώ να υπολογίζω.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Shape matching is an important problem of computer vision and pattern recognition. It is an actively researched topic with a significant amount of scientific literature, facts that reflect to a steady performance improvement of the established benchmarks. This activity can be explained because of the importance of the problem due to its diverse applications including, but not limited to, object detection, classification and possibly tracking, content based retrieval for images, optical character recognition and image registration. Many methods and systems have been proposed to solve these and related problems, with diverse mathematic and computational approaches. Common characteristics of methods that can be denoted as shape matching techniques are the establishment of correspondences between two given shapes, and the computation of a similarity or 'shape distance' metric [1].

Throughout this work, only the planar case is considered. The case of 3D shapes has also significant scientific interest. Relative research on this subject is mentioned in section 1.2. Throughout this work however, the term shape always refers to a 2D shape.

[1] Here the word metric is used with a broad sense, and does not necessarily imply that this function satisfies the properties of metric functions as defined in metric spaces.

1.1 Problem definition

1.1.1 Definition of shape

Geometrically, shape is "what is left when the differences which can be attributed to translations, rotations and dilatations have been quotiented out" [19]. A way to define shape is to represent it as a simple closed curve of $\mathbb{R}^2$. Formally, a curve c is defined as a function $c : [0,1] \rightarrow \mathbb{R}^2$ with $c(t) = (c_x(t), c_y(t)), t \in [0,1]$ where c_x, c_y are continuous and, usually (but not necessarily), smooth functions. A curve on $\mathbb{R}^2$ is called simple if it does not cross itself except perhaps at the endpoints. This constraint can be formally expressed as $\forall t_1, t_2 \in [0,1]\, c(t_1) = c(t_2) \Rightarrow (t_1 = t_2 \vee (t_1 = 0 \wedge t_2 = 1) \vee (t_1 = 1 \wedge t_2 = 0))$. A curve is called closed if the endpoint coincides with the starting point or $c(0) = c(1)$.

For specific applications it is adequate to represent a closed curve as the polygon that occurs by sampling the curve at sufficient distinct positions since the coarse characteristics are preserved. This means that a shape can be described by a simple closed polygon, a definition that will be followed throughout this work[2]. A polygon is a piecewise linear curve. Each linear segment is called an edge of the polygon and a point where two consecutive edges meet is called a vertex. A polygon can be represented as an ordered set of points, namely the vertices in a counterclockwise order. To sum up, a shape sh described by a curve c will be represented as an ordered set of points in $\mathbb{R}^2$: $sh = (p_1, p_2, \ldots, p_n)$ with $p_i = c(\frac{i-1}{n-1})$ for $i = 1, 2, \ldots n$. The number n will be properly selected.

The usual way to factor out the parameter of position is to consider only polygons with barycenter at the origin. Orientation invariance can be achieved by aligning the shape according to the covariance matrix of its points. Finally for scaling, one can take the mean of all the distances among points of the shape as the scale of this shape, and normalize properly. Notice that, although these normalizations fulfill the requirement that two instances of the same shape have the same representation, no mention to the notion of "similarity" is made. These normalizations may in practice align disproportionally two shapes that a human might consider similar.

As a side note, the following is an alternative definition of shape that inherently

[2]It should be noted here that the term "silhouette" instead of the term "shape" is sometimes used for this concept in the relevant bibliography.

possesses the properties of location, scale and rotation invariance: shape can be defined as the set of all (semi[3])invertible linear transformations of a set of points on the plane [19]. In practice it is inefficient to search exhaustively in the space of all affine transforms. Also, this might not be desirable for certain cases or applications (see below).

1.1.2 Definition of shape matching

Humans intuitively understand the notion of shape matching as the establishment of correspondences between parts of two given objects. Moreover, it follows naturally to estimate a degree of similarity between the objects. The matched parts may have similar appearance, or something more complex in common. For example, it would be natural (and rather easy) for a human to match the various limbs of two human figures viewed by a different viewpoint, despite the fact that the appearance of these parts may change significantly.

No matter how complicated the process providing the answers is, it is straightforward to mathematically formulate the result. Given a pair of shapes $sh_1 = \{(x_{1\,i}, y_{1\,i}) \mid i = 1, 2, \ldots k\}$ and $sh_2 = \{(x_{2\,j}, y_{2\,j}) \mid j = 1, 2 \ldots l\}$, shape matching is defined as a process that computes a set $C = \{(p_{1\,i}, p_{2\,i}) \mid i = 1, 2, \ldots n, p_{1,i} \in sh_1, p_{2,i} \in sh2\}$ of n (with $n \leq k$ and $n \leq l$) corresponding point pairs as well as a number $d \geq 0$ that represents the similarity between these shapes. The goal is to establish matches that are intuitively acceptable, i.e. points that have a similar appearance.

A variety of properties may be desirable for a shape matching method:

- Translation invariance. The movement of a shape should not affect its correct detection and matching.
- Rotation invariance. Similarly, the rotation of a shape should not affect the matching.
- Scale invariance. Scaling of an object usually needs special handling to detect.
- General affine transform invariance. Viewing a planar object from a different viewpoint (as well as other types of affine transformations) can result in a significantly

[3]Degeneracies may of interest, as long as the whole shape does not collapse to a single point.

different shape.

- Occlusion handling. Occlusions occur regularly in real world situations and humans exhibit significant ability to ignore them.

- Noise tolerance. Occlusion can be considered as a type of noise, but usually uncorrelated white noise is treated separately.

- Viewpoint change (of a non-planar object). A combination of almost all of the above, it can be an especially demanding case.

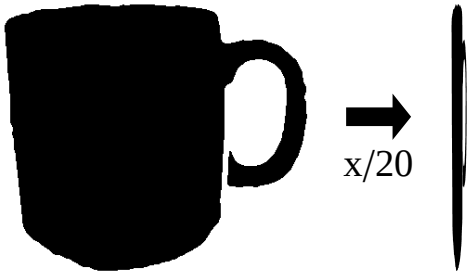


Figure 1.1: A cup (left) that undergoes a 20-fold compression on one axis (right) looks more like a pen.

However, not all of these properties may be desirable for a specific task. The needs vary depending on the target application. For example in OCR it is important to be able to discriminate between a 6 and a 9, so rotations of more than half a cycle should not be considered. Also, heavy affine transformations have the potential to introduce unexpected results: a cup can look more like a pen if it is stretched a lot along the horizontal axis as in figure 1.1. This indicates that complete affine transform invariance may introduce false matches.

1.2 Literature review

This section references selected research work on shape matching and specifically shape contexts (the method of choice for this work).

1.2.1 Shape matching

Many methods have been proposed for shape matching proposing different approaches for varying purposes. Numerous survey studies have organized and classified this significant amount of work. Loncaric in [23] adopts three different classifications proposed by Pavlidis in [29]. A brief outline of these classification follows:

- The first classification concerns the distinction between boundary or external vs global or internal methods. Methods that exploit only the shape boundary fall in the first category. This category includes most methods based on shape contexts, including the present work. Methods that consider the interior of the shapes (such as decompositions to simpler parts, articulation points etc.) fall in the second category.

- The second classification is based on the type of computational result of each method. The goal of many shape matching methods is to compute a similarity measure, represented by a single number. Thus they are named numeric methods, and they comprise the first category in this distinction. On the other hand, methods that compute an alignment of the shapes fall in the second category of non-numeric methods.

- The third classification is based on the portion of input information that each method actually utilizes. Methods that transform the boundary of the shape in a way that the original shape can be recovered, are called information preserving methods. On the contrary, the non-preserving methods are those that use incomplete descriptors, in the sense that they do not contain enough information to recover the original shape.

Veltkamp and Hagedoorn in [41] present shape matching methods that focus on the geometric aspect of the problem. They distinguish different types of problems. They formulate the dissimilarity problem in a computation and a decision version:

- compute the dissimilarity between two shapes,

- and decide whether the dissimilarity is below a given threshold.

They continue by recognizing two problems involving shape transformations and a dissimilarity measure. They formulate a decision and an optimization problem:

- decide whether there exists a transformation that transforms one of the shapes so that the two shapes are dissimilar within a given threshold,
- and find the optimal transformation between two shapes, i.e. the one that minimizes the dissimilarity between the shapes.

Finally they formulate an approximation problem: compute a transformation between two patterns that makes the dissimilarity between them close to the optimum within a specified factor.

Hoffman and Richards in [17] propose that humans have the natural tendency to split shapes into parts. They provide psychological evidence supporting the assumption that humans tend to segment 3D objects along their local curvature minima contours. They assume that the human visual system reconstructs a 3D shape of the object from the 2D image on the retinas, and then proceeds with this method to partition it.

In [5], Basri et al. propose a method to estimate shape similarity based on both part articulation and local deformation cost. They specify a set of desirable conditions for a shape distance, that implicitly take into account the part nature of objects. They proceed to define three different distance measures as well as a strategy to compute them using Dynamic Programming.

In [9], Chi and Leung propose a general approach towards object identification based on shape contours. Shapes are broken down to a set of lines and arcs. A voting scheme is applied to the extracted shape description for the purpose of robust object recognition under clutter or occlusion.

Wu et al. in [44] employ genetic algorithms to search over the space of affine transformations. They describe representation and resampling schemas suitable for the specific application, and propose variations to improve the time and accuracy of matching.

Backes et al. in [3] use as descriptor the distribution of the distances between points on the boundary of the shape. They propose two different distributions as descriptors, and use them for shape classification with the aid of Linear Discriminant Analysis (LDA). The experiments show that the method is robust to noise and other types of shape distortions.

Latecki et al. in [20] propose a variation of open shape matching based on Dynamic Programming. The method tries to exploit the strengths of both the Dynamic Time Warping [35] as well as the Longest Common Subsequence technique [42]. They use local tangent for the purpose of shape description, and provide experimental evidence showing that the method matches correctly and robustly open contours.

Adamek and O'Connnor propose a multiscale representation of shape contours in [1]. Initially, they apply different levels of smoothing on the shape contours. They further process this result by applying a transformation that detects concave and convex parts of the contour. They then proceed to match such shape descriptions with the use of an appropriate comparison distance, and Dynamic Programming.

Arica and Vural in [2] propose a simple geometric transformation for the purpose of shape description. They compute the bearing angle of three consecutive contour points for variable offsets between these points. The values obtained for each point of the contour and for various offset sizes are considered as random variable measurements. They proceed to calculate the moments of these measurements, forming the proposed descriptor. The matching of the descriptions is performed using Dynamic Programming. Despite its simplicity (or possibly because of it), this method is almost invariant to affine transformations, and performs robustly under non-rigid distortions.

Torres and Falcao in [11] extend their work with Costa in [40]. They compute image skeletons at multiple scales, and use them to detect salient points on the contour of the shape. Their extension handles better the concave points, by eliminating spurious branches on the skeletonization process. They also propose a second descriptor that can describe segments of the shape instead of single points. They match these shape descriptions with a Dynamic Programming method, the same as in [2].

Felzenszwalb et al. present in [14] a method based on an elegant and powerful idea. They propose the representation of each shape as a tree, with each level representing a different level of description. The root of the tree represents a properly selected cut on the curve while the left and right children represent cuts on the occurring subcurves. They propose an iterative matching scheme that can be efficiently solved using Dynamic Programming. They proceed to formulate an algorithm that can locate query shapes in real-world color images.

Ebrahim et al. present in [12] a method that employs as a descriptor the Hilbert curve. This curve is a fractal pattern that covers an entire square grid without crossing itself. The authors transform the raster of each shape to a one-dimensional signal according to the occurrence of shape points on a Hilbert curve. This signal is then smoothed by keeping the largest coefficients of a wavelet transform. Experimental evidence illustrates the performance of the method in several established test.

An important work is presented in [45]. It is related to shape matching although it is not a shape matching technique. Specifically, the goal of this work is to improve the performance of any shape matching method, by exploiting all the information that is available after every shape in a database has been compared to all others. In practice, different poses of an object may look quite different. However, if enough intermediate postures are present in the database, one can reach to the logical result that the shapes come from the same class by looking to the intermediate comparisons. Algorithmically this is achieved by representing the full result of the comparisons as a fully connected graph, and performing a kind of flooding on it. A significant experimental result of this method is that triangle inequalities between many triplets of shapes are restored. The mathematical definition of a metric function requires the triangle inequality. It can thus be argued that, qualitatively, this technique improves the properties of any comparison method.

1.2.2 Shape contexts

Shape context is a shape descriptor introduced by Belongie et al. in [7]. The main idea is that the local distribution of points for the purpose of local description is well captured using a log-polar histogram. In the original form, selected points from the contour of an image were used as centers, and the distribution of the other contour points around each center was used as a descriptor vector. Shape contexts capture the fact that local features play a more important role than more distant ones for the purpose of local matching. Effectively, the used logarithm weighs more the proximate features, and less the more distant ones. Shape contexts have been employed in many applications. Numerous extensions to the original method have been proposed.

Mori and Malik in [27] present an algorithm based on shape contexts that can break a visual CAPTCHA system. CAPTCHA stands for "Completely Automated Public Turing test to Tell Computers and Humans Apart". The task is artificially hard since computers should not be able to pass it. The authors employ and extend shape contexts to detect letters in the cluttered images. The proposed extension is the use of local tangent vectors as the sampled quantity instead of contour points. This allows the shape contexts to perform robustly in the cluttered environment, since no automatic segmentation method would provide acceptable contours of the shapes of interests (letters).

Mortensen et al. in [28] propose an improvement of the SIFT descriptor [24] using shape contexts. The authors coin the term global context for the variation of shape contexts that they propose. The descriptor is sampled over a larger area of the image, in contrast to most of the other applications of shape contexts. Also, the sampled quantity is not edge points. Instead, the authors propose the quantity of local curvature arguing that edge detection is more prone to noise than curvature. They concatenate the SIFT descriptor and the global context of a point and they proceed to compare this new descriptor with the standard SIFT.

The goal of the work presented in [26] by Mori and Malik is to estimate the 3D posture of humans. They use the framework that they originally proposed in [7] to establish correspondences between contours of human figures. They then proceed to reconstruct the 3D posture using the method that Taylor proposes in [39].

In [15] extensions of shape contexts to 3D shapes are discussed. The natural extension of the log-polar to the log-spherical coordinate system is proposed, and the details that need attention are outlined. Specifically, the large volume variations between the bins need careful treatment, and the extra degree of freedom with respect to rotation is another consideration. The authors proceed to propose another descriptor named harmonic shape context. This descriptor uses the aforementioned 3D histogram. A spherical harmonic transform of this histogram is computed, and a predetermined number of coefficients is discarded keeping only the largest ones in absolute value. This is effectively a low pass filter of the data, aimed at reducing noise. A useful property of this transform is rotation invariance. Both descriptors are evaluated in the retrieval of 3D shapes, and compared to others.

An important work that utilizes shape contexts is presented in [22]. The goal of this work is to exploit the articulated nature that many common shapes possess to improve shape matching. The authors suggest that the distances and angles to be sampled should be measured only inside the closed contour of a figure. This means that articulations are handled quite well by this improved descriptor. The main idea is that the inner distance of articulated shapes is invariant to articulation (in contrast to the classic Euclidean distance). Any descriptor that uses distances as input is thus suitable. The selected descriptor for all their experiments is shape contexts.

In [32], Rodriguez and Shah employ shape contexts to create an occlusion-resistant representation of segmented human figures. They use this descriptions along with already extracted human figures as their training set. They also use shape contexts in the detection stage to find possible positions of the learnt features. A voting scheme determines the final detections of human figures in the scene.

1.3 Introduction to the proposed methodology

In the following chapters, the proposed methodology and experimental results are presented and discussed.

The proposed method is thoroughly presented in chapter 2. First, computational tools used throughout this work are described. In order these are: Dynamic Programming and Dynamic Time Warping, cubic spline and thin plate spline interpolation and shape contexts. The main method for shape matching follows the methodology presented by Belongie et al. in [6]. The shape contexts are formed and compared as presented in [6]. The main difference is the next step of the method, namely the matching step. In the present work, the method used for matching is the one proposed in [36].

Experimental results of this methodology are presented in chapter 3. Qualitative tests on well-known datasets are firstly presented. Experimental comparison of the method using the well established bulls-eye test is also presented. Finally, an application that utilizes the proposed method is outlined. Objective of this application is to localize the joints of human figures. This is possible because the matching method can choose the most similar human figure from a set of models. Also, the method produces correspondences

between the observed and the model figure, enabling the estimation of a transformation between the shapes which can be used to transfer known joints positions from the model to the observed figure.

Strong points and limitations of the method are discussed in chapter 4. Issues for further investigation are finally proposed.

Chapter 2

Methodology

2.1 Necessary tools

Well established algorithmic and numerical analysis methods for the manipulation of observational data are utilized in various steps of this work. An introduction to them is provided below.

2.1.1 Dynamic Programming [1]

Dynamic Programming is a technique commonly used to accelerate the solution of optimization problems. It takes advantage of the properties of optimal substructure and overlapping subproblems that some problems exhibit.

Optimization problems are used to formulate a variety of interesting questions which cover a vast spectrum of sciences and disciplines. Defining an objective function and searching for an optimum set of parameters for it, is a useful way to formulate scientific questions. Assume a real-valued function f that takes as argument x from a subset of $\mathbb{R}^n$. Concisely $f : A \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$. The optimization problem related to f is the task of finding the global minimum of it. This means finding an $x_{min} \in A$ that satisfies $\forall x \in A f(x_{min}) \leq f(x)$. The function f is referred to as the objective or target function. Note that this formulation covers the cases when the global maximum of a function is

[1] This presentation of Dynamic Programming is loosely based on that of chapter 15 of the book "Introduction to Algorithms" by Thomas H. Cormen[10]

needed. It suffices to take the negative function as objective, and subsequently find the minimum. Optimization in the general case of a function with domain in the real numbers is studied in real analysis. The used tools for this kind of problem are differentiation and some kind of gradient following strategy, provided that f is smooth enough. This is necessary because the domain set is uncountably infinite and so it is impossible to evaluate f in each point of its domain and choose the one with the lowest value. Instead of that, only a finite subset of the domain is chosen for evaluation, and the properties of the function ensure local or even global optimality.

Dynamic Programming is applied to problems that have objective functions with finite domain sets (conventionally in the natural numbers). In these cases (unlike the continuous case) the global optimum is mathematically trivial to find, since all that is needed is to evaluate the target function in each point of the domain and keep the point with the preferred score. In practise however, this strategy can quickly become infeasible as the cardinality of the domain set increases. The problems that can benefit from Dynamic Programming are usually optimization problems with a domain set that grows exponentially with respect to the input size of the problem. This behaviour can quickly render a naive approach useless for even small instances of a problem. The properties of optimal substructure and overlapping subproblems help to find a way to reduce this large amount of computations turning an exponential time solution into a polynomial one.

An optimization problem is said to exhibit optimal substructure if the (optimal) solution to an instance of the problem 'contains' (in the sense that it is efficient to compute) solutions to smaller instances of this problem. It is then efficient to find a solution to a large instance of the problem by combining the solutions to smaller instances of this problem. In general, a problem that exhibits optimal substructure can be solved using several different methods, one of them being the technique of Dynamic Programming. A general term used for these methods is divide and conquer. By definition a divide and conquer algorithm solves the requested instance of a problem by recursively breaking it into smaller subproblems until they become simple enough to solve directly. It then constructs efficiently the solution to the more complicated problems all the way up to the start of the recursion. However not all problems with the optimal substructure property are solved using Dynamic Programming. Alternative divide and conquer tech-

niques include greedy algorithms and simple divide and conquer ones that do no fall into any subcategory. A problem that exhibits the optimal substructure property as well as overlapping subproblems will most certainly have an efficient solution based on Dynamic Programming.

The overlapping subproblems property refers to the recurrence of the subproblems into which the original problem is divided. It is useful because one can compute once the solution to each reoccurring subproblem, and then reuse the result in all the needed places. A trivial example is the computation of the Fibonacci numbers: the nth Fibonacci number is given by

$$\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$$

with initial conditions $\text{fib}(0) = \text{fib}(1) = 1$. Ignoring the fact that it is easy to compute this sequence in linear time with respect to n using just a loop and two variables, the naive recursive implementation would be:

Algorithm 1 fib(n) // naive recursive implementation

Input: $n \in \mathbb{N}$

Output: The nth Fibonacci number

```
if n < 2 then
    return 1
else
    return fib(n − 1) + fib(n − 2)
end if
```

With this approach, the call fib(5) will need fib(4) and fib(3), and each one of them would recalculate the value of fib(2). The resulting call tree is show in figure 2.1. The call fib(2) is made 3 times resulting in two unnecessary additions. This number will grow rapidly. In fact, the number of calls to fib(k) is the Fibonacci sequence delayed by k, and the total number of additions needed to compute fib(n) is fib(n)-1 (the number fib(n) is calculated by adding fib(0) and fib(1), both equal to 1, as many times as needed, which is fib(n)-1). The sequence fib(n) is closely approximated by an exponential function. This implies that the computation of fib(n) becomes infeasible for small values of n, even less

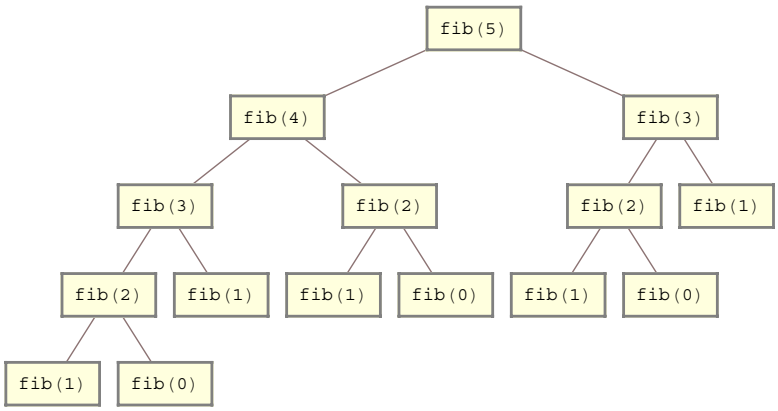


Figure 2.1: The call tree of the naive implementation of fib(n).

than 100 (fib(100) is 573147844013817084101 – more than 9000 years of computations assuming 2 billion additions per second).

The improvement that renders efficient the calculation of fib(n) – as well as many problems that exhibit the overlapping subproblems property, is the use of memoization. Memoization is the technique of keeping already computed values for subproblems, and the subsequent reuse of them. There are many ways to achieve this practically, and the design of the algorithm may actually perform it implicitly. Keeping the computed values of interest in an array for later lookup is an explicit way to perform memoization. Explicit ways are the more straightforward ones for demonstration purposes, and also for clarity of the implementation. Regarding the Fibonacci sequence example, assuming access to a map data structure (available in most modern programming languages) that can associate a key to a value, a simple trick can reduce an exponential time recursion to a linear one.

The change in the call tree is obvious. It is almost reduced to a simple chain as can be seen in 2.2.

2.1.1.1 Levenshtein distance / DTW as order-preserving matching tools

V. Levenshtein in [21] presents a way to compare two strings by counting how many operations are required to transform one string to the other. The allowed operations that were originally proposed are single character insertions, deletions and replacements. Generalizations of this method allow the cost of operations to vary (one can assume a cost

Algorithm 2 fib(n) // memoized implementation

Input: $n \in \mathbb{N}$

Output: The nth Fibonacci number

```
// assume that an empty map of name fibmap is initialized
if n < 2 then
    return 1
else
    if fibmap does not have association for n then
        associate pair (n, fib(n − 1) + fib(n − 2)) in fibmap
    end if
    return association of n in fibmap
end if
```

of 1 for each operation in the original form of the method). The result of the algorithm is the minimum transformation cost, along with the actual steps that perform this transformation. The cost of trying every possible sequence of transformations is prohibitive, but proper use of Dynamic Programming allows the calculation to be performed in $\mathcal{O}(m \cdot n)$ steps where m, n are the respective sizes of the two strings under comparison.

To solve the problem of varying speed in speech, the speech recognition community uses Dynamic Programming in a similar form under the name Dynamic Time Warping (DTW). Algorithmically the technique is a generalized Levenshtein distance with symbol matching costs representing similarities of phonemes. Some of the first works towards this direction are [35], [43] and [34]. Conventionally the result of the Levenshtein distance computation is the cost itself while the Dynamic Time Warping gives the transformation rules sequence.

The DTW is useful for the aforementioned purpose because speech is assumed to keep an order that is already known. This means that the assigned task is not one of finding the permutation of symbols that best matches the given string. Instead, the problem is to use a known order as extra information to align the given strings.

The strategy to solve the problem starts by arranging a matrix with dimensions exceeding by one the lengths of the two strings. Each row of the matrix after the first one

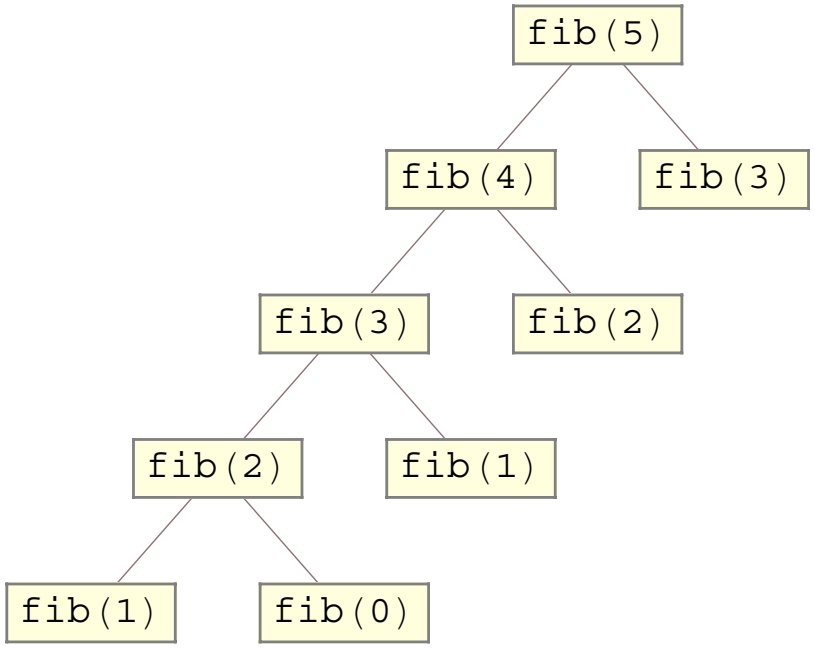


Figure 2.2: The call tree of the memoized implementation of fib(n).

corresponds to one symbol of the first string, and each column after the first one to a symbol of the second string. Each cell of the matrix then represents a subproblem of the original one, and specifically the subproblem of matching the two substrings that reach to this row and column. The necessity for an extra row and column occurs because the possible positions for insertions and deletions on a string of length n are $n + 1$, including the beginning and the end of the string. The convention followed here is that each operation takes place after the current symbol of the string, hence the need for an empty row and column at the beginning of the two strings. The reason that there needs to be an extra row and column is that it must be possible to insert and delete characters before considering the first character of a string. The originally proposed metric can be calculated with the Algorithm 3.

After the end of this algorithm, it is easy to obtain the sequence of operations that achieves the minimum. The way to do this is to backtrack on the matrix C starting from the bottom-right position. From there, one recalls the choice that lead to the current square, and reverses the movement. In the case of a tie, it suffices to choose one of the previous squares at random. At the end of the backtrack a path from the top-left

Algorithm 3 levdist(s_1, s_2)

Input: s_1 and s_2 are strings of symbols

Output: The Levenshtein distance between s_1 and s_2

Let $m = length(s_1)$, $n = length(s_2)$

Initialize a matrix C of size $(m+1) \times (n+1)$ containing zeros

$C(1:m, 0) = [1:m]$

$C(0, 1:n) = [1:n]$

for $i = 1$ to m **do**

for $j = 1$ to n **do**

$cost = ((s_1(i-1) == s_2(j-1))?0 : 1)$

$$C(i,j) = min \begin{cases} C(i, j-1) + 1 & // \text{ insertion} \\ C(i-1, j) + 1 & // \text{ deletion} \\ C(i-1, j-1) + cost & // \text{ replacement/match} \end{cases}$$

end for

end for

return $C(m, n)$

to the bottom-right is formed that consists of right, down, and diagonal (right-down) movements. Each type of movement corresponds to one type of operation as follows: a movement to the right represents an insertion, down represents a deletion, and a diagonal movement represents a match or substitution. For example, one run of the algorithm with input s_1 = "antonis" and s_2 = "markos" can be visualized by the following matrix:

		m	a	r	k	o	s
	0	**1**	2	3	4	5	6
a	1	1	**1**	2	3	4	5
n	2	2	2	**2**	3	4	5
t	3	3	3	3	**3**	4	5
o	4	4	4	4	4	**3**	4
n	5	5	5	5	5	**4**	4
i	6	6	6	6	6	**5**	5
s	7	7	7	7	7	6	**5**

The highlighted numbers denote a possible backtrack path (for this particular example it is actually the only valid path) on the cost matrix C representing a transformation sequence of length 5. For the specific example, the highlighted sequence along with the intermediate results is:

	antonis
insert 'm'	mantonis
match the 'a' 's (0 cost), replace 'n' by 'r'	martonis
replace 't' by 'k'	markonis
delete 'n'	markois
delete 'i'	markos

The Levenshtein distance can be computed using Dynamic Programming because the problem possesses the necessary properties of optimal substructure and overlapping subproblems. The problem indeed exhibits optimal substructure since each step for the best operation can be decided among three different subproblems. What is more, as these decisions ask for the solution of more and more subproblems, the same substrings reoccur, hence the property of overlapping subproblems. Memoization is explicitly performed with the use of the cost matrix C where intermediate "notes" are kept.

As mentioned earlier, a generalization of the Levenshtein distance allows for arbitrary matching costs between symbols. The implementation shown in Algorithm 3 almost supports this. The single line that has to change is the one where the variable *cost* is calculated. Instead of a simple zero-one comparison, one can replace this by a matching cost defined by an arbitrary function. The values of this function need not be integers. As long as the values are non-negative, the single constraint is that the matching values must follow arithmetically the cost of insertions and deletions. If the matching cost for any symbol is larger than the sum of an insertion and a deletion, it is obvious that substitutions will never be preferred. On the other hand, if the matching cost is too low, all symbols will match, and no insertions and deletions will be chosen.

This generalization allows the Levenshtein distance to be used in continuous signals. In these cases the continuous signal must be discretized, and the discrete parts must be compared somehow. Every symbol of the first string is compared to all the symbols

of the second string, and the results of this comparison are provided as input to the generalized Levenshtein Distance. This is the way Dynamic Time Warping uses Dynamic Programming to match varying speed signals.

2.1.1.2 Using Dynamic Programming to match cyclic strings [2]

In the case of interest for this work, the input signal is the contour of a figure. The discretization can be assumed already performed since the input image is usually represented by a matrix of pixels. The comparison of the discrete parts is possible with the use of shape contexts. All that remains is to input these matching costs to a DTW algorithm and obtain an alignment sequence. The case of contours however has a complication. The contours are circular, and there is no safe way to align them before the DTW run. The naive solution to this problem is to run many times the DTW, each with a different starting point, and keep the result with the lowest transformation cost. However this would raise the time complexity of the comparison. Specifically, assuming that each of the contours is represented by n shape contexts, the time complexity for one DTW run is $\mathcal{O}(n^2)$ and so the total runtime for all the n different initial matches would be $\mathcal{O}(n^3)$.

In [36], Schmidt et al. present a way to find this initial match and the occurring alignment sequence in $\mathcal{O}(n^2 \log(n))$. The speedup is possible through the reduction of the problem to one of shortest paths on a graph, as well as with clever reuse of the already computed intermediate costs.

Specifically, the problem of aligning two strings using DTW is equivalent to the problem of finding the shortest path on the graph G defined as follows. Let the two strings s_1 and s_2 be represented by sequences of symbols: $s_1 = (s_{1,1}, s_{1,2}, \ldots, s_{1,m-1}, s_{1,m})$ and $s_2 = (s_{2,1}, s_{2,2}, \ldots, s_{2,n-1}, s_{2,n})$. Let now $G = (V, E)$ a directed graph having nodes V, and edges $E \subseteq V \times V$. In this case $V = \{(s_{1,k}, s_{2,l}) | k = 0, 1, 2, \ldots m, l = 0, 1, 2, \ldots n\}$ with $s_{1,0}$ and $s_{2,0}$ special starting symbols. The edge set E is then defined $E = \{((s_{1,i}, s_{2,j}), (s_{1,k}, s_{2,l})) | 0 \leq k - i \leq 1, 0 \leq l - j \leq 1\}$. That is, each node represents one square of the memoization matrix, and each edge represents a possible movement between squares. It follows naturally to visualize the graph on a rectangular grid as in figure 2.3. Each horizontal edge is assigned a fixed length or cost of insertion, and each

[2] The presentation of this section is based on the respective publication [36]

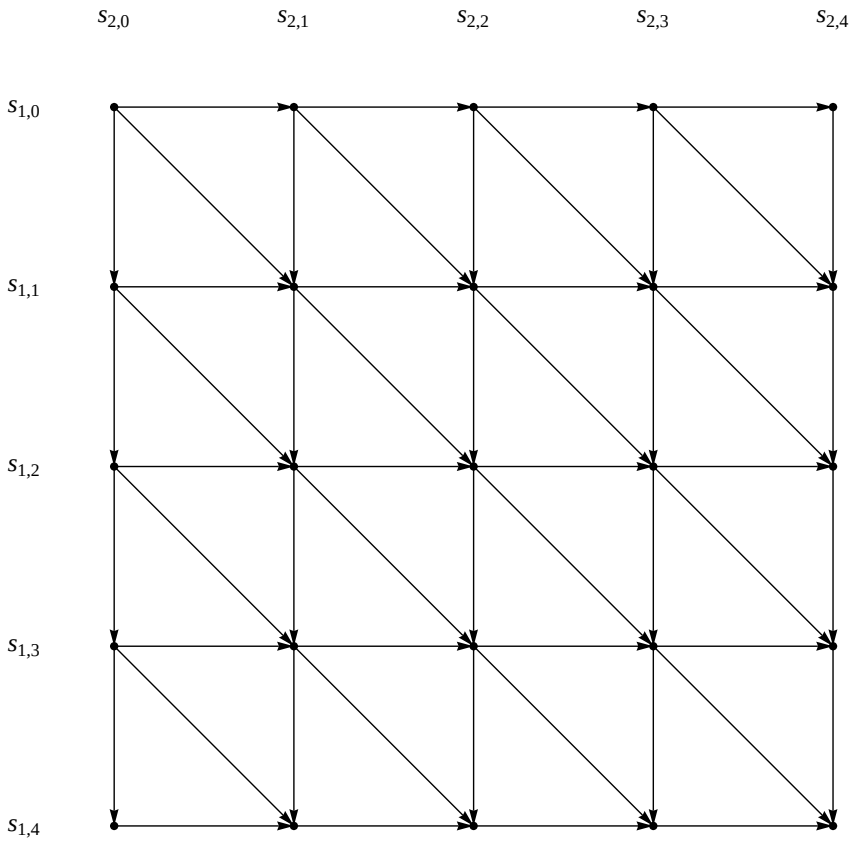


Figure 2.3: Visualization of the graph G.

vertical a cost of deletion. To keep the distance symmetric, these two costs should be equal. The diagonal edges are assigned matching costs according to the input of the algorithm. The shortest path of G between $(s_{1,0}, s_{2,0})$ and $(s_{1,m}, s_{2,n})$ represents the solution to the generalized Levenshtein distance problem.

However, in the case of cyclic strings, the situation is a little more complicated. The topology of the graph is no longer a grid, but a torus. The last symbol $s_{1,m}$ is followed by $s_{1,1}$ and the same holds for the respective symbols on s_2. One can unfold the graph to resemble that of figure 2.3 but to preserve and exploit its cyclic properties, it is more convenient to replicate another copy of it to the right, as in figure 2.4. On this new graph G', the starting symbol $s_{2,0}$ of the second string is not necessary, and all the other symbols of this string are repeated once. For the cyclic case, the assumption that both strings have length n is made. To simplify the notation, let $v_{i,j} = (s_{1,i}, s_{2,j})$. Any closed path starting at a circled node $v_{0,k}$ on the top-left of the graph and ending at the corresponding node $v_{n,n+k}$ at the bottom right of the graph represents an alignment sequence between

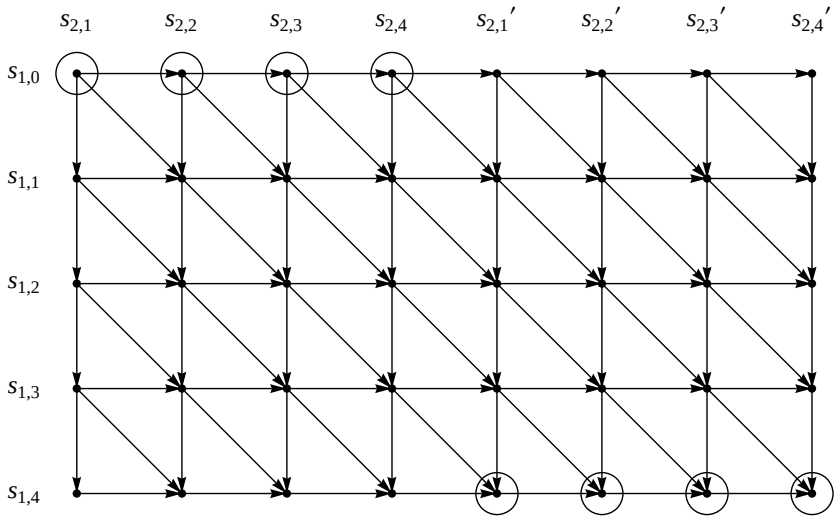


Figure 2.4: Visualization of the unfolded cyclic G'.

the two strings. The whole problem can now be formulated as follows: find the number k along with the corresponding shortest path $v_{0,k} \rightsquigarrow v_{n,n+k}$ that jointly minimize the alignment cost between the two strings.

As stated before, searching among every possible initial match has a computational complexity of $\mathcal{O}(n^3)$. An algorithm to reduce this is based on the following theorem:

Theorem 1. *Let $G = (V, E)$ with $V = \{V_i\}_i$ be a graph and let $p_1 = v_{i,1}v_{i,2}\dots v_{i,n}$ and $p_2 = v_{k,1}v_{k,2}\dots v_{k,m}$ be two minimum-cost paths. Then, if p_1 and p_2 have two nodes v_p and v_q in common, there is a path $p_2' = v_{k_1}\dots v_{k_m}$ with the same cost as p_2, which has a common sub-path $v_p \rightsquigarrow v_q$ with path p_1.*

A proof is sketched below.

Proof. The path p_2' can be constructed as $p_2' = v_{k,1} \rightsquigarrow v_p \overset{p_2}{\rightsquigarrow} v_q \rightsquigarrow v_{k,m}$ where $v_p \overset{p_2}{\rightsquigarrow} v_q$ denotes the sub-path of the path p_2.
There are two cases:

- The sub-path $v_p \rightsquigarrow v_q$ on the path p_1 coincides with that on p_2. This case needs no further examination, since the path p_2' is identical to p_2 and obviously they have the same cost.

- The sub-path $v_p \rightsquigarrow v_q$ on P_2 does not fully coincide with the respective one on p_1. Then the path $sub_2 = v_p \overset{p_2}{\rightsquigarrow} v_q$ has the same cost as $sub_1 = v_p \overset{p_1}{\rightsquigarrow} v_q$, because

any sub-path of a shortest path is itself a shortest path between the nodes that it connects. This means that any alternative path cannot have less cost than the sub-path under consideration. Since both sub_1 and sub_2 are sub-paths of shortest paths, they possess this property. This in turn means that $c(sub_1) \leq c(sub_2)$ and $c(sub_1) \geq c(sub_2)$ and hence $c(sub_1) = c(sub_2)$, where $c(.)$ denotes the cost function for a path. This concludes the proof. □

The theorem provides a way to split the problem into subproblems. Specifically, a shortest path $v_{0,k} \rightsquigarrow v_{n,n+k}$ on the rectangular grid shown in figure 2.4 defines two regions on it. If one is searching for the shortest path $v_{0,m} \rightsquigarrow v_{n,n+m}$, with $m < k$, then there is no reason to consider nodes on the right of the existing shortest path. The opposite restriction would hold for $m > k$. The algorithm is divided in five steps.

- **Step 1:** Compute the shortest path $p_l = v_{0,1} \rightsquigarrow v_{n,n+1}$ and copy it shifted by n elements to the right $p_r = v_{n,1} \rightsquigarrow v_{2n,n+1}$.

- **Step 2:** Use the paths p_l and p_r as boundaries on the $2n$ by $n+1$ matrix of costs, and compute shortest paths starting from the node $v_{(l+r)/2,0}$. Shortest paths for the whole range among the bottom nodes of p_l and p_r can be computed in one pass.

- **Step 3:** Estimate the largest k' with $l + n \leq k' \leq (l + r)/2 + n$ such that the shortest path from $v_{(l+r)/2,0} \rightsquigarrow v_{n+L,n+1}$ has a common sub-path with p_l. Similarly, estimate the lowest k'' with $l + n \leq k'' \leq (l + r)/2 + n$ for which the shortest path from $v_{(l+r)/2,0} \rightsquigarrow v_{n+k'',n+1}$ has a common sub-path with p_r.

- **Step 4:** Let the node v_a be the first of a common subpath in the order from top to bottom. Use this node as start for a DTW pass, thus finding in two passes (one for each v_a) all the shortest paths $v_{0,m} \rightsquigarrow v_{n,n+m}$ with $m \in \{1, \ldots L\} \cup \{R, ldots, n\}$. It is now possible to identify the shortest circular path in these ranges.

- **Step 5:** Recurse two times from step 2, one for $l = k' + 1$ and $r = (l + r)/2 - 1$ and one for $l = (l + r)/2 + 1$ and $r = k'' - 1$. The bounding paths that must accompany each number are already computed at the previous step.

2.1.2 Spline interpolation

Interpolation is a useful tool that allows to predict missing values of a function given some samples of this function. For the univariate case, the input (or conditions) to an interpolation method is a set of n measurements $X = \{(x_i, y_i) \subseteq \mathbb{R}^2, i = 1, 2, \ldots n\}$ with $i \neq j \Rightarrow x_i \neq x_j$, and the result is a function $f : \mathbb{R} \rightarrow \mathbb{R}$. Usually, the distinction between exact and inexact interpolation is made. In the first case it is required that the resulting function f passes through all input points (hence the term conditions), that is $f(x_i) = y_i$ for all $i = 1, 2, \ldots n$ while in the second case this restriction is relaxed. The inexact interpolation is useful to smooth out noisy data and possibly to filter outliers. The function f is usually called the interpolant. The way to compute the interpolant is to fit the data points with a predetermined model. Important classes of models used for interpolation include: piecewise constant (alternatively nearest neighbor), piecewise linear, polynomial interpolation and spline interpolation.

Spline interpolation can be described as piecewise polynomial interpolation. One of the earliest references of the word "spline" in the context of piecewise polynomial functions for the purpose of numerical approximation is found in [37]. The model is based on a mechanical one that was used by draftsmen to draw smooth curves. Thin strips of wood or other flexible material were pinned to specific points on the plane, and the shape that occurred is the one that minimizes the total bending energy of the object. For more details, the interested reader is referred to [4].

The input data to a spline interpolation defines a partition on the horizontal axis. Without loss of generality, the input can be assumed to be ordered on the horizontal axis: $x_i < x_{i+1}$ for all $i = 1, 2, \ldots n - 1$. A partition of $\mathbb{R}$ defined by these numbers is $\mathbb{R} = (-\infty, x_1] \cup (x_1, x_2] \cup \ldots \cup (x_{n-1}, x_n] \cup (x_n, \infty)$. For each interval of this partition, a polynomial of predetermined degree is fitted. Usually, the degree for these polynomials is 3 (cubic splines). This choice is common because the extra degrees of freedom allow for sufficiently smooth interpolants without the extra computational overhead that a higher degree would imply. More precisely, it is possible to obtain interpolants that have continuous second order derivatives. Cubic spline interpolation is a common choice because the method is arithmetically stable without requiring special care to the sampling

$$\begin{cases} p_i(x_i) = y_i & i = 1, 2, \ldots n-1 \\ p_i(x_{i+1}) = y_{i+1} & i = 1, 2, \ldots n-1 \\ p_i'(x_{i+1}) = p_{i+1}'(x_{i+1}) & i = 1, 2 \ldots n-2 \\ p_i''(x_{i+1}) = p_{i+1}''(x_{i+1}) & i = 1, 2 \ldots n-2 \end{cases}$$

Table 2.1: The set of standard equations for a cubic spline, ensuring continuity up to the second order derivative of the interpolant.

$$\begin{cases} p_1''(x_1) = 0 \\ p_{n-1}''(x_n) = 0 \end{cases}$$

Table 2.2: The two additional constraints for the natural spline.

points. This is in contrast to the polynomial interpolation which is rather sensitive to the position of the samples on the horizontal axis.

The following describes the implementation of natural cubic splines that was used in this work. There exist faster and more stable methods to achieve this [4], but they are less intuitive. The presented implementation however proved to be sufficient in both speed and stability for the needs of the present work as discussed at the end of the section.

For n input data points, the piecewise polynomials that have to be computed are $n-1$. This makes a total of $4(n-1)$ parameters (since each cubic polynomial has four coefficients). Let one such third degree polynomial be denoted as p_i for $i = 1, 2, \ldots n-1$. It can be written as $p_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$. The values of the parameters can be obtained by solving a linear system of size $4(n-1)$. The first $2n-2$ constraints are imposed by the values of the polynomials at the two ends of their respective segments: $p_i(x_i) = y_i$ and $p_i(x_{i+1}) = y_{i+1}$ for $i = 1, 2, \ldots n-1$. The next $n-2$ constraints are imposed by the requirement to have smooth first derivatives: $p_i'(x_i) = p_{i+1}'(x_i)$ for $i = 1, 2, \ldots n-2$. Because of the requirement for second order derivative continuity, $n-2$ more constraints are obtained: $p_i''(x_i) = p_{i+1}''(x_i)$ for $i = 1, 2, \ldots n-2$. To sum up, the constraints are listed in table 2.1. This is a total of $4(n-1)-2$ linear equations on the $4(n-1)$ unknowns. To solve this system exactly, one needs two more extra constraints. The usual choice

for these constraints involves the first or second order derivatives. Setting the second order derivatives at the edges of the interpolation interval equal to zero as shown in table 2.2 yields the so called natural spline. Other choices include predetermined values for the first or second derivatives. With the number of equations completed, all that is now needed is to solve the system and obtain the interpolant f. The form of the interpolant is

$$f(x) = \begin{cases} p_1(x) = a_1x^3 + b_1x^2 + c_1x + d_1 & \text{if } x \leq x_1 \\ p_2(x) = a_2x^3 + b_2x^2 + c_2x + d_2 & \text{if } x_1 < x \leq x_2 \\ p_3(x) = a_3x^3 + b_3x^2 + c_3x + d_3 & \text{if } x_2 < x \leq x_3 \\ \vdots & \vdots \\ p_{n-1}(x) = a_{n-1}x^3 + b_{n-1}x^2 + c_{n-1}x + d_{n-1} & \text{if } x_{n-2} < x \leq x_{n-1} \\ p_n(x) = a_nx^3 + b_nx^2 + c_nx + d_n & \text{if } x_{n-1} < x \end{cases}$$

The result of a cubic spline interpolation is displayed in figure 2.5. In this example the "unknown" quantity is the function $\sin(x)$. The input to the interpolation method is a set of seven points equally spaced on the interval $[0, 2\pi]$. This particular example is a little easier than average since both assumptions made by the natural spline are actually true: $\sin''(0) = \sin''(2\pi) = 0$. Nevertheless, the performance of the natural spline is rarely unacceptable and usually quite good. Of course, by construction, the cubic spline cannot predict discontinuities of the interpolated quantity up to second derivative.

Although the definition of cubic splines only mentions the univariate case, it is easy to extend this tool to interpolate any one-dimensional curve g embedded in an n-dimensional space $g : \mathbb{R} \to \mathbb{R}^n$. To achieve this, one has to compute n cubic splines f_i with $i = 1, 2 \ldots n$, where f_i interpolates the i-th dimension of the sample data. The interpolant is then the function $x \to (f_1(x), f_2(x), \ldots f_n(x))$. An example of this method is provided in figure 2.6. Note that the left-hand side of the linear system does not depend on the values of the function, but only depends on the function parameters which are common to all the interpolations needed in this case. This allows to compute once the inverse of the matrix, and then use the inverse with different right-hand sides to compute all the f_i.

As previously stated, the speed and stability of this method are sufficient for the

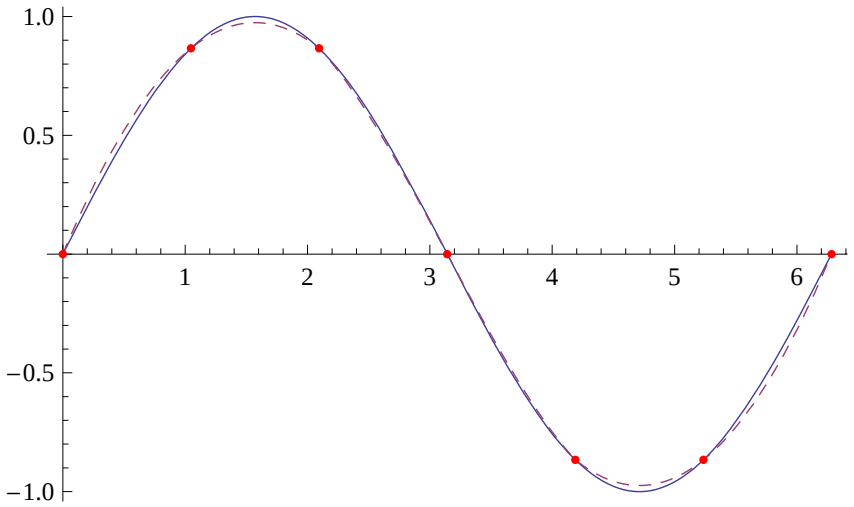


Figure 2.5: The cubic spline interpolation of the sinus function using only seven samples (shown as dots), along with the original curve (dashed).

needs of this work. Speed is not an issue because of the reuse of the inverted matrix as discussed in the previous paragraph. The matrix of the linear system is the same for all the cases of shapes with the same number of points, and needs to be inverted only once. Also, numerical stability is acceptable in practice, since the presented method works with adequate accuracy using double precision arithmetic. Specifically, the determinant of the resulting matrix is usually in the order of 10^{-40}. However the calculated inverse leaves negligible residuals in the magnitude of round-off error if multiplied by the original matrix and the result is subtracted by the identity matrix.

2.1.3 Thin plate splines [3]

Thin plate splines is a form of multidimensional interpolation. The interpolant is a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ that fits the input data using radial basis functions. The interpolation conditions are represented by n triplets of reals $\{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots (x_n, y_n, z_n)\}$ with $i \neq j \Rightarrow |x_i - x_j| + |y_i - y_j| \neq 0$. For every such triplet (x, y, z), it is required that the interpolant f satisfies $f(x, y) = z$. This restriction is relaxed in the regularized case. Similarly to the cubic spline case, the thin plate spline is based on a physical model, satisfying a solution to a minimum bending energy problem subject to the imposed con-

[3] This presentation is based on the respective part of ref. [6].

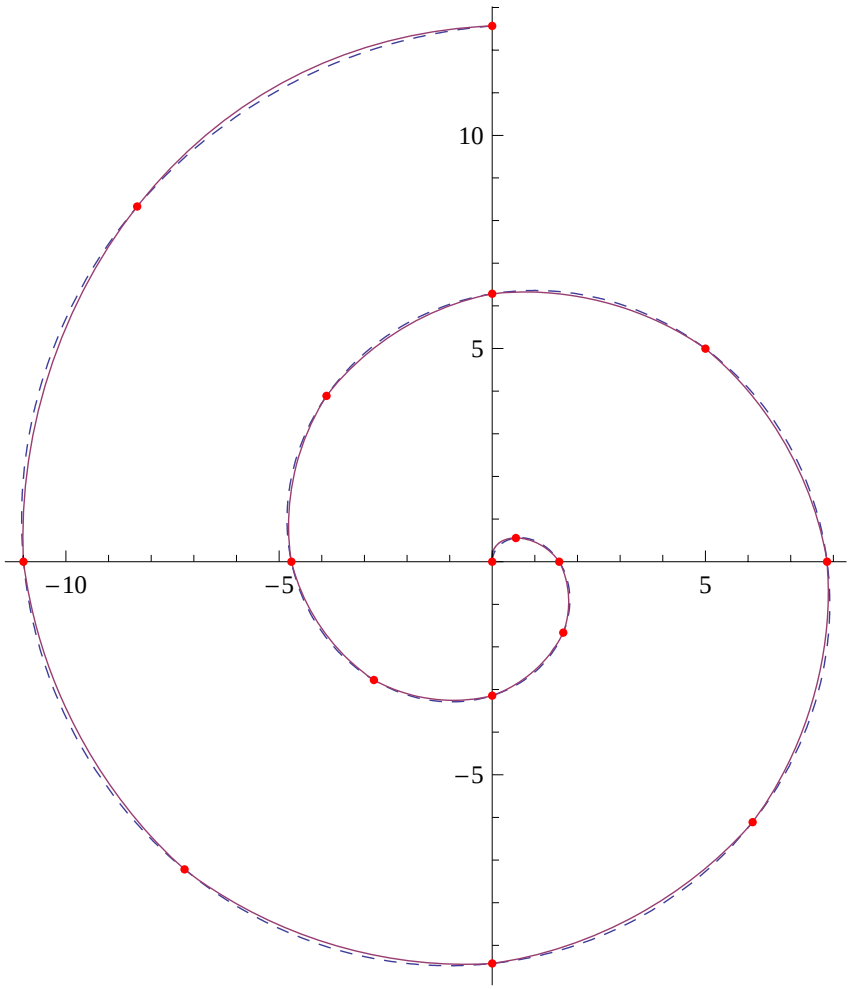


Figure 2.6: An example of interpolating a one-dimensional curve using multiple splines.

straints. Specifically, the bending energy for a function f is

$$I_f = \int\int_{\mathbb{R}^2} \left(\frac{\partial^2 f}{\partial x^2}\right)^2 + 2\left(\frac{\partial^2 f}{\partial x \partial y}\right)^2 + \left(\frac{\partial^2 f}{\partial y^2}\right)^2 dxdy$$

The parametric form of the model is:

$$f(x,y) = a_1 + a_x x + a_y y + \sum_{i=1}^{n} w_i U(\|(x_i,y_i) - (x,y)\|)$$

with the kernel function U defined as $U(r) = r^2 \log(r^2)$. The limit

$$\lim_{r \to 0} U(r) = 0$$

resolves the indeterminate form $U(0) = 0$ at $r = 0$. The number of free parameters of this model for n input points is $n + 3$. These parameters can be estimated

by solving a linear system. First, the vector of unknowns is formed as $(w \mid a)^T = (w_1, w_2, \ldots, w_n \mid a_1, a_x, a_y)^T$. On the right-hand side of the system, the vector of the z coordinates is padded with three zeroes to reach the necessary length of $n+3$: $(v \mid 0)^T = (z_1, z_2, \ldots z_n \mid 0, 0, 0)^T$. The matrix K with $K_{ij} = U(\|(x_i, y_i) - (x_j, y_j)\|)$ and the matrix P with $P_i = (1, x_i, y_i)$ complete the linear system:

$$\left(\begin{array}{c|c} K & P \\ \hline P^T & 0 \end{array}\right) \quad \left(\begin{array}{c} w \\ \hline a \end{array}\right) \quad = \quad \left(\begin{array}{c} v \\ \hline 0 \end{array}\right)$$

In the regularized case, the bending energy term is not the complete objective function. A term of square error is weighed with it, for an objective function of the form:

$$H[f] = \sum_{i=1}^{n} (z_i - f(x_i, y_i))^2 + \lambda I_f$$

thus relaxing the restriction $f(x_i, y_i) = z_i$. The parameter λ can be thought as the trade-off between smoothness of the interpolant, and exactness of the interpolation. Note that for $\lambda = 0$ the regularized case reduces to the exact one. On the other hand, for large values of λ the model degenerates to a least-squares affine one. All that one has to do to compute the parameters for this case, is to substitute the matrix K on the left-hand side of the equation by the matrix $K + \lambda I$, where I is the appropriately sized identity matrix.

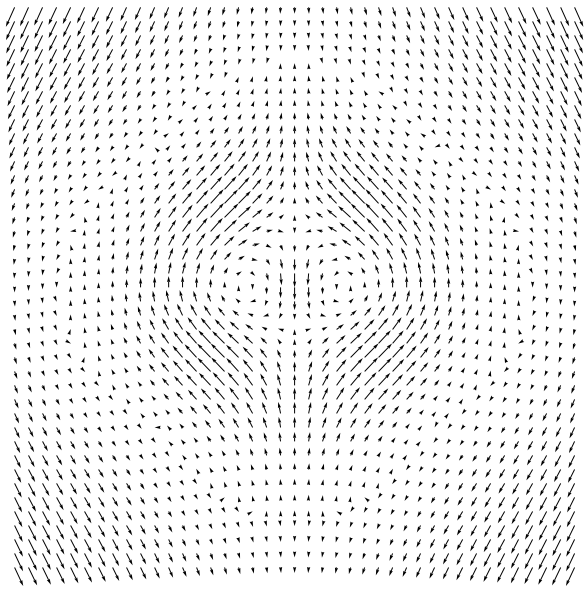


Figure 2.7: Visualization of an arbitrary plane warping using thin plate splines.

The interpolation of multivariate quantities is as straightforward as in the cubic splines case. An interpolant of the form $\mathbb{R}^2 \rightarrow \mathbb{R}^n$ is created by concatenating n thin plate splines. An example for the case $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ is shown in figure 2.7. Note that, as in the cubic spline case, it is possible to use the same matrix inverse for both dimensions.

The vector a represents a linear transformation of the $\mathbb{R}^2$ domain. The vector w represents the non-linear distortion imposed by the provided conditions. One can interpret these two sets of parameters as a rough linear alignment with an additional non-linear one. For a study on the algebra of this model in the case of $\mathbb{R}^2$ to $\mathbb{R}^2$ see [8].

2.1.4 Shape contexts

As previously stated, the shape context is a local shape descriptor proposed by Belongie et al. in [7] for the purpose of shape matching. They observe that a rather small set of points on the contour of an object can capture its shape. They thus propose that all objects are sampled, and the local distribution of these points is the information captured by shape contexts. More specifically, around each center, the distribution of all the other centers is used to calculate the description. Points that are within a maximum distance are kept, and the rest are discarded, in order to capture the local information. The distribution of the selected points is sampled in the log-polar space to emphasize on the detail closer to the center and allow more variation further away. As a side note, the actual density of the selected points is application dependent and it is ultimately a parameter to be determined primarily by trial and error. Also, this type of sampled shape satisfies the definition of shape as presented in 1.1.1.

The steps of the sampling process are shown in figure 2.8. The input (a) is a shape contour of potentially infinite accuracy (it can be defined as a continuous parametric curve). This curve is subsampled to a reasonable level as in (b) aiming to preserve the features of the shape. Around each selected point in the sampled contour, a log-polar histogram of the other points is calculated (c).

These resulting histograms can be used for the purpose of local matching. This is illustrated in figure 2.9. Similar-looking parts of two shapes have also similar shape contexts, while the opposite holds for dissimilar parts. In the figure, each shape is sampled

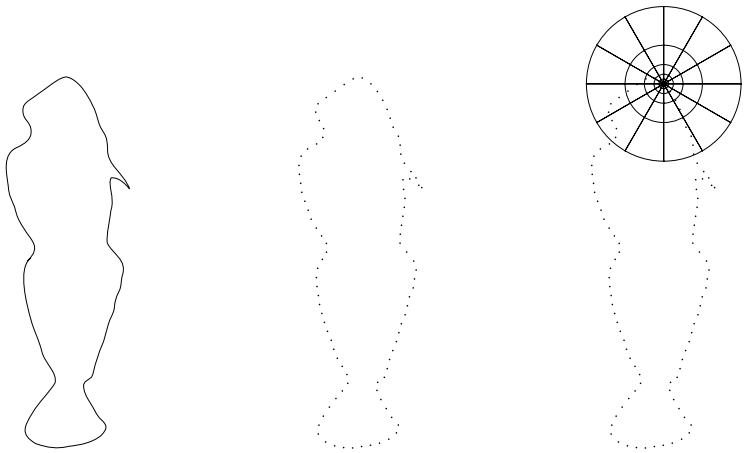


Figure 2.8: The process of sampling and calculating shape contexts.

with the bins exactly as presented. The resulting histogram is displayed below each shape. Lower values are denoted by darker color. The horizontal axis corresponds to angle, and the vertical to (increasing) logarithmic distance.

The description of each shape is formed as the (possibly ordered) set of all the computed histograms. A number of methods are available to compare and match these points. For comparison, the choice in [7] is the x^2 statistic, while any other histogram comparison statistic can be used. For matching, the method proposed in [7] is the Maximum Weighted Bipartite Matching, while the present work uses the method proposed in [36].

2.2 Method

The objective of this work is to present a shape matching technique. As stated earlier, the technique utilizes shape contexts to describe selected points on the figure contour. Each descriptor of the first shape is then compared to all the descriptors of the other, giving matching costs between them. These costs are provided as input to the cyclic DTW matching and correspondences between the shapes are established. These correspondences are used to calculate an aligning transformation (using thin plate splines) between the two shapes. The DTW cost along with the TPS transformation energy are balanced and this comprises the final distance between the shapes.

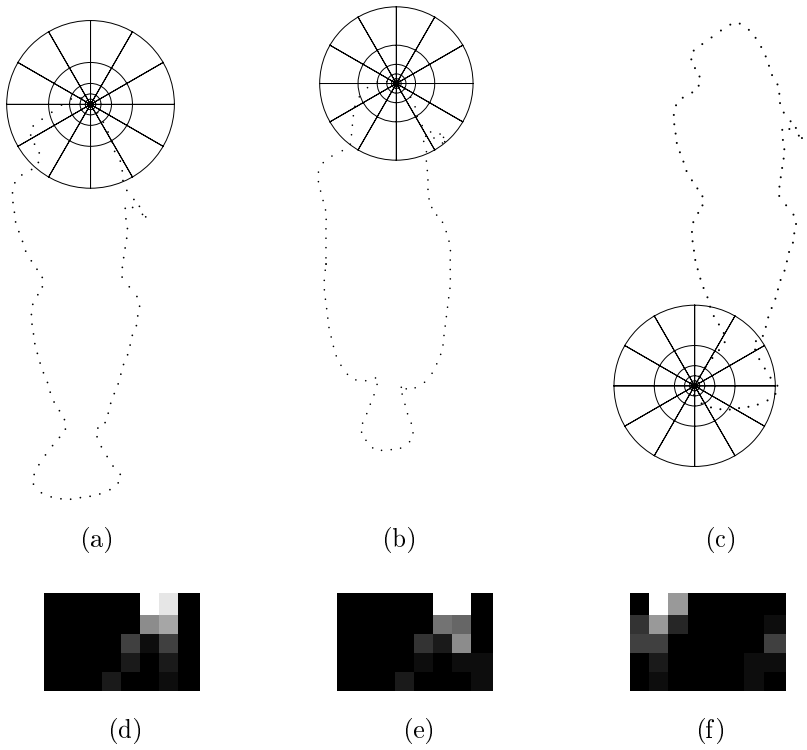


Figure 2.9: Similar and dissimilar parts with the corresponding shape contexts.

Every input figure will be assumed to have a fixed number of n points. The order of the points is also important, as it is assumed to be the natural order on the contour of the figure. Furthermore, it is important to ensure that each figure is listed in the same cyclical order, a requirement imposed by the DTW method. This order can be assumed to be counter-clockwise.

2.2.1 Scale estimation and orientation

The first step of the method is to perform a rough scale estimation of each input figure. As in [6], the mean distance between all the $(n(n-1))/2$ point pairs is evaluated and the figure is scaled accordingly. A similar possible choice for this step is the Median Absolute Deviation (MAD) discussed in [33]. This alternative was also evaluated, but the result was not significantly different, and so the less computationally expensive method was

selected. Denoting the ith input point as pt_i, the scale a is estimated as

$$a = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{2 \left\| \mathrm{pt}_i - \mathrm{pt}_j \right\|}{n(n-1)} \tag{2.1}$$

Consequently, every point of the input is multiplied by $1/a$.

During this first pass, the orientation of the contour at each point can also be determined. The order of the points is an important hint for the matching technique in use, so it is important that all the points of the shapes are listed in a fixed order. It is decided that all shapes are listed in a counter-clockwise order. To achieve this, the sign of the area of the polygon is calculated using the formula $A = \frac{1}{2} \sum_{i=1}^{n} x_i y_{i+1} - x_{i+1} y_i$ with $x_{n+1} = x_1$ and $y_{n+1} = y_1$. If the sign of A is negative, then the order of the input points is reversed.

2.2.2 Shape context computation

The next step of the method is to compute shape contexts. For each of the n (now scaled) points, a log-polar histogram of the other points' positions is computed. The number of bins in each dimension plays role in the performance of the descriptor. Generally, very dense discretization would result in aliasing artifacts (points falling in adjacent bins), while very sparse would result in reduced discriminative power. In practice it proved sufficient to use values similar to the ones proposed in [6]. For specific details on the present work see section 3.

The size of the shape contexts is another important parameter. The scale normalization on the previous step allows for choices independent of the input. The majority of the methods employing shape context use them as local feature descriptors. The partial nature of the problem is thus exploited. However, choosing too small diameter for the shape contexts can lead to increased sensitivity to noise, since the sampling may become disproportionally dense. Another issue that can arise in this case is reduced discriminative power. This may happen because all the descriptors may look similar if they include just one or two samples. Since the sampling is performed exponentially, there is also need for a minimum radius. This parameter should be chosen in a range that allows the smallest ring to actually have samples, while no significant fraction of the samples falls

in the unsampled inner circle. A rough estimate for this range is $\bar{d}/2 \leq r_{min} \leq \bar{d}$, where $\bar{d}$ denotes the average distance between consecutive points of the shape and r_{min} is the minimum radius. Again, for specific details on the choices for this work, see section 3.

Let b_r, b_θ represent the number of bins in the radial and angular dimension respectively. Also, let r_{min} and r_{max} denote the minimum and maximum sampling radius respectively. Each bin of the histogram is bound in the *log* dimension by

$$\frac{\log(r_{max}) - \log(r_{min})}{b_r}(k-1)$$

and

$$\frac{\log(r_{max}) - \log(r_{min})}{b_r}k$$

where k is the bin number ranging from 1 to b_r. In the angular dimensions, the corresponding bounds are $\frac{2\pi}{b_\theta}(l-1)$ and $\frac{2\pi}{b_\theta}l$, again with l in the range 1 through b_θ denoting the bin number. For each point of the shape, the logarithm of distance and bearing angle for all the other points is computed, and binned according to the previous bounds.

2.2.2.1 Rotation invariance

For the purposes of the present work, rotation invariance is a desirable property. The method proposed by Belongie et al. in [6] is used. Specifically, they propose the alignment of each shape context along the local tangent of the shape. The estimation of a reliable local tangent vector is not straightforward. Since the local tangent is essentially the local derivative, it is sensitive to noise. The usual way to treat noise is to smooth the input signal, something that in this methodology is already performed prior to the sampling of the contour.

In this work, the local tangent is estimated using cubic spline interpolation. First, the 2D curve is fitted by a cubic spline model as described in section 2.1.2. The next step is to compute the derivatives of the two cubic spline models at each point of interest. For each such pair of derivatives, it is now straightforward to compute the local tangent by taking the generalized arc tangent function with two arguments. This method has the advantage that the computed angles are consistently aligned not only to a good estimate of the local derivative, but also to a consistent direction. The three steps of this process are displayed in figure 2.10. In (a), the input to the process is shown, a set of points

sampled from the original input curve. The parametric curve computed by the cubic spline model is shown in (b), and the resulting local tangent estimations are shown in (c).

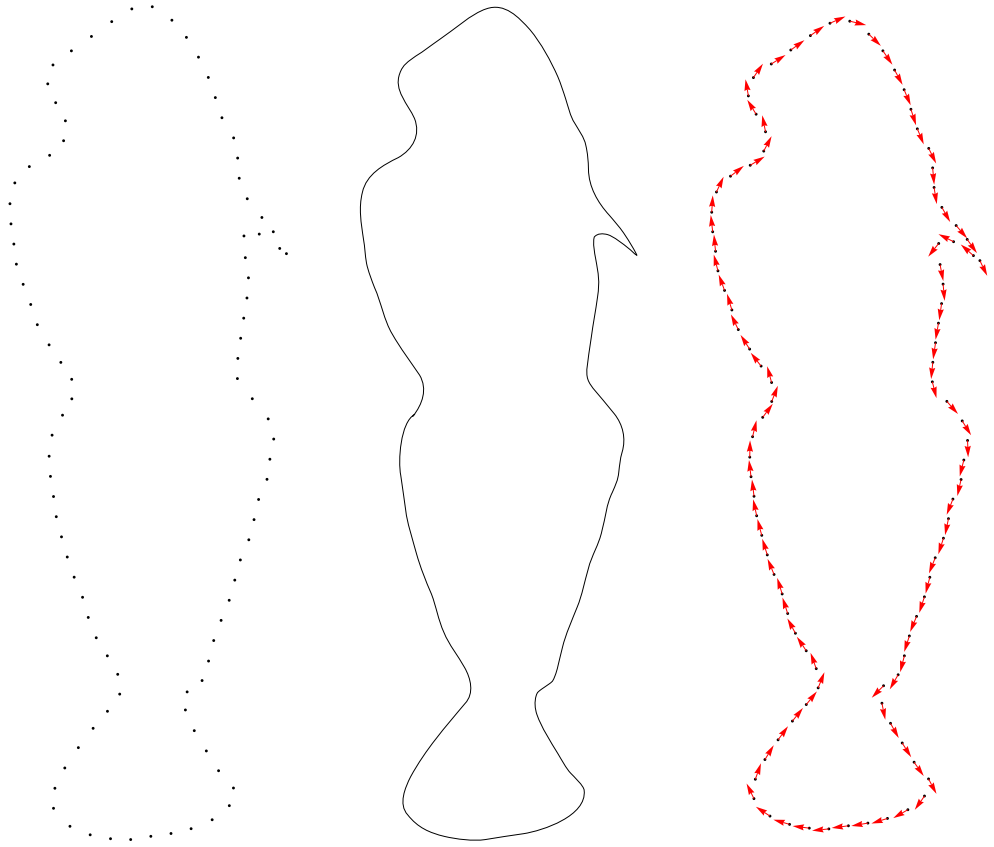


Figure 2.10: The process of sampling and computing local orientation.

After a reliable local estimation of the contour orientation has been established, it is straightforward to compute rotation invariant shape contexts. The origin of the angular dimension is determined by the local tangent, and all the computations are performed in this reference frame. The process is illustrated in figure 2.11. At the implementation level, it is sufficient to subtract the local orientation angle from each bearing in the log-polar space, before the histogram computation.

The resulting shape contexts are indeed rotation invariant, since the local tangent is dependent on the global rotation of the shape. A rotated version of an otherwise identical shape will have identical shape contexts if they are rotated according to the local

tangent. This essentially exploits the corresponding property for translation. Since global translation does not affect the distances between points of the shape, shape contexts are invariant to translation. The same holds for local tangents: their relative angles are invariant to local rotation, making the described method rotation invariant.

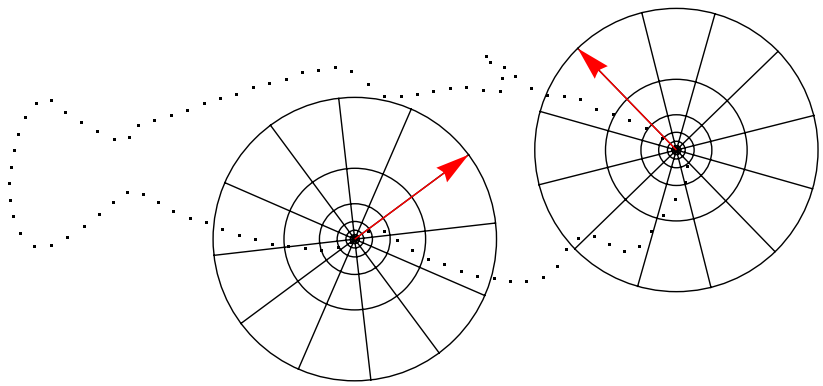


Figure 2.11: Each shape context is aligned to the local tangent estimation.

2.2.3 Shape context comparison - x^2 statistic

The method has reached to the point where a shape is described by a set of shape contexts. The next step is to compare the shape contexts of two shapes. This can be achieved with a number of different histogram comparison statistics, and the x^2 statistic is selected as in [6]. The comparison of two shapes can be represented by a matrix C. The element (i, j) of this matrix is the result of the x^2 statistic of shape context i of the first shape, and shape context j of the second shape. Specifically:

$$C(i,j) = \frac{1}{2}\sum_{k=1}^{K}\frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)}$$

where $h_m(n)$ is the shape context histogram (flatten to one dimension with K bins) of the mth shape at the nth point. An example comparison is shown in figure 2.12.

2.2.4 Cyclic matching

The local similarity of point i of the first shape with point j of the second shape is $C(i, j)$. Any such pair is a potential correspondence between the shapes. In the general case, any

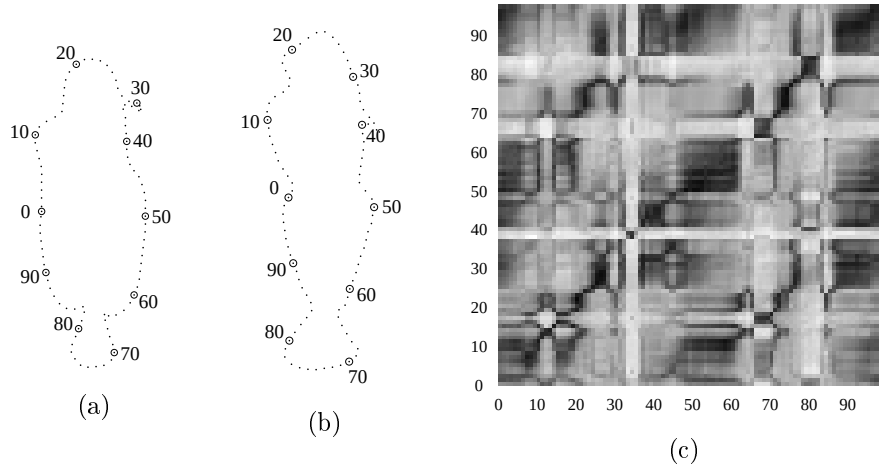
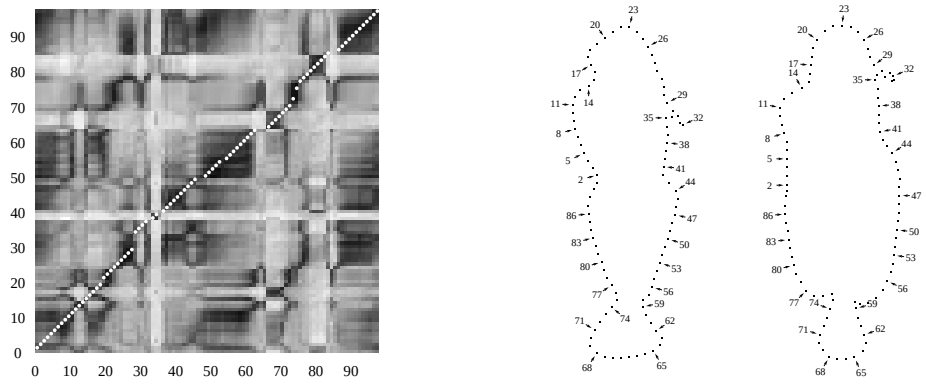


Figure 2.12: Visualization of the comparison matrix C. The shape in fig. (a) corresponds to the horizontal axis of the matrix. Dark colors denote low values i.e. similar shape contexts.

one-to-one relationship between the points of the shapes can be considered as a possible matching. Belongie et al. in [6] use a Maximum Weighted Bipartite Matching formulation to establish a set of one-to-one correspondences between the shapes. However, assuming that the points of the shapes are listed in an order that is naturally imposed by the contour, the search space can be reduced.

For the purpose of matching, the selected method in this work is [36], presented in section 2.1.1.2. The matching cost matrix needed for the cyclic matching as input, is naturally the matrix C of x^2 comparisons. Along with the matching pairs, a total matching cost c_m is calculated as the sum of all the operations that were used. The result of this matching step can be visualized in 2.13. In the figure, the selected pairs are shown in (a), and some of them are displayed over the shapes in (b). The specific example is not the general case, since the resulting matches are close to the main diagonal of the matrix. In the general case, the result will look like a cyclic shift of the displayed one.

The cyclic matching method has a single parameter, which is the insertion and deletion cost (the case where these costs are different is not considered). The insertion and deletion cost is manually set to a level that experimentally gives acceptable results. There is no reliable method to automatically determine this value. It might be possible to determine



(a) (b)

Figure 2.13: The matched path on the cost matrix is shown in (a), and some of the pairs are shown in (b)

it on a per-case basis, as a function of the matching cost matrix.

At the end of this step, the result is a set of matched points on the shapes. This matching is the result required by the definition in 1.1.2. Also, the total matching cost c_m can be used as a distance measure between the shapes.

2.2.5 Thin plate spline computation

The final step of the presented technique is the computation of the planar deformation that aligns one shape to the other. The alignment is performed using thin plate splines, as presented in 2.1.3. Input is the result of the previous step, i.e. a set of pairs of 2D points, and the output is a deformation of the plane, as well as a deformation cost. This cost is properly weighted along with the cost of the previous step to form the final matching cost or distance between the shapes.

The regularized version of the thin plate spline model is used, with parameter λ as discussed in 2.1.3. This parameter acts as a smoothness factor. The model tolerates higher noise levels for higher values of λ and vice versa. Since the scale of all shapes is roughly estimated at the first step of the methodology, the value of λ can be uniformly set to compensate for a fixed amount of noise. For all the experiments, λ is fixed to 1 as in [6]. The resulting transformation of the shapes is displayed in figure 2.14. Notice that the regularized model preserves the characteristics of the shape even after the transformation.

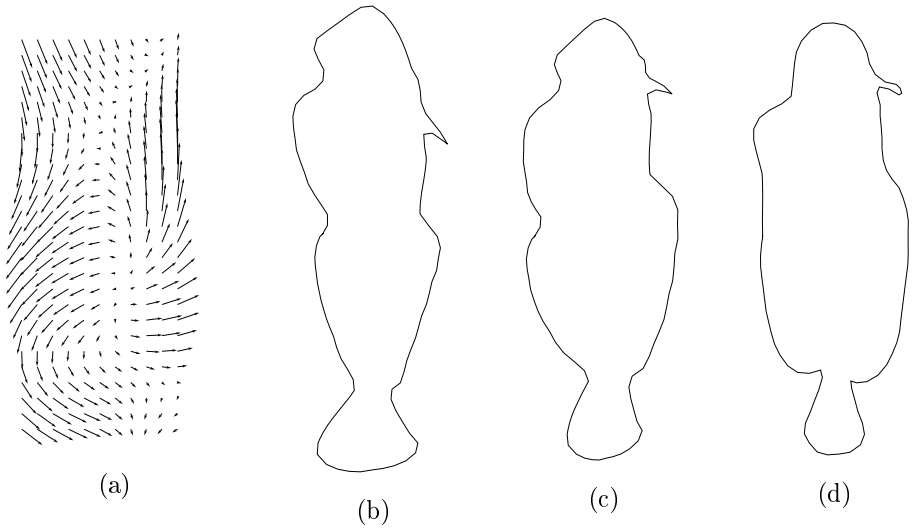


Figure 2.14: The calculated thin plate spline model visualized in (a) warps the shape displayed in (b) to the one displayed in (c). This result is quite similar to the compared shape, displayed in figure (d).

The tail and fin of the fish in (c) share common appearance with those of the original in (b), despite the fact that their relative positions follow the respective ones of the matched fish in (d).

The final outcome of the presented method is the warping between the compared shapes, along with the total matching cost d. This total cost is a weighted average of the cyclic matching cost c_m and the thin plate spline bending cost c_b: $d = l_1\, c_m + l_2\, c_b$ for $l_1 > 0$ and $l_2 > 0$. The weight factor between the costs is manually chosen.

Belongie et al. in [6] propose an iterative scheme after the warping of the shape. The new shape is again compared to the other, in an attempt to use the new information contained in the warped shape in order to match more points. This proved to be rather useless and so it was not included in the proposed method.

2.3 Method Summary

This concludes the proposed methodology. The methodology is summarized below. The input to the algorithm is a pair of shapes, and the output a matching cost and a plane-to-plane deformation that best aligns the first shape to the second. The algorithm has

a number of parameters which are application dependent. These are: The number n of points per shape contour, the number of bins in the radial and angular dimensions b_r and b_θ respectively, the minimum and maximum sampling radius r_{min} and r_{max} respectively, the insertion/deletion cost for the cyclic DTW, the regularization parameter λ of the TPS model and the final weighting parameters l_1 and l_2.

Algorithm 4 shapepreprocess(pts)

Input: pts $\in (\mathbb{R}^2)^n$

Output: Scale and rotation invariant shape contexts of the points pts

$a = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{2\left\|\text{pts}_i - \text{pts}_j\right\|}{n(n-1)}$ // scale estimation

pts $= \frac{1}{a}$pts // and normalization

$A = \frac{1}{2} \sum_{i=1}^{n} x_i y_{i+1} - x_{i+1} y_i$ (addition is modulo n) // signed area computation

if sign$(A) < 0$ **then**

// the shape is in clockwise order

reverse the order of the points pts

end if

// compute local tangents

$f_x(t) =$ cubicspline(pts_x)

$f_y(t) =$ cubicspline(pts_y)

tangents $=$ atan2$(f_y'(t), f_x'(t))$, $t = 1, 2, \ldots, n$

// compute shape contexts

for each point $p \in$ pts, create a histogram for the values

$(\log_2 \|p - p_i\|, \text{bearing}(p, p_i) - \text{tangent of } p)$, $p_i \in \text{pts} - p$

the bins have log boundaries:

low: $\frac{\log(r_{max}) - \log(r_{min})}{b_r}(k-1)$

high: $\frac{\log(r_{max}) - \log(r_{min})}{b_r} k$, $k = 1, 2, \ldots, b_r$

polar boundaries:

low: $\frac{2\pi}{b_\theta}(l-1)$

high: $\frac{2\pi}{b_\theta} l$, $l = 1, 2, \ldots, b_\theta$

Algorithm 5 shapematching($\text{pts}_1, \text{pts}_2$)

Input: $\text{pts}_1, \text{pts}_2 \in (\mathbb{R}^2)^n$

Output: The matching cost d, along with the TPS model T

// pre-process the shapes

shapepreprocess(pts_1)

shapepreprocess(pts_2)

compute x^2 statistic matrix C

$C(i,j) = \frac{1}{2} \sum_{k=1}^{K} \frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)}$

cyclic DTW using C as input

output of DTW: a matching consisting of pairs between the points of the shapes

also output of DTW: matching cost c_m

computation of the TPS transform T with input the pairs of the previous step

the bending cost of the transformation T is c_b

final results: T and $d = l_1\, c_m + l_2\, c_b$

Chapter 3

Results

Experimental results evaluating the proposed method are presented in this chapter. The first section presents a qualitative assessment of the method in datasets that are frequently used for the purpose of shape matching. The second section is dedicated to the quantitative performance of the presented method. Specifically, the results of the method for the bulls-eye test are presented. An application of the method for the localization of joints in human figures is described in the next section. Some implementation notes in the last section conclude the chapter.

3.1 Marine - Gestures

Experiments on two datasets are presented in this section. These are the "marine" and "gestures" datasets used in [31]. They were acquired from [13], and the "marine" dataset is also referred to as the SQUID dataset from the respective work [25]. The first dataset consists of 1100 marine creature figures, and the second consists of 980 synthetically generated gestures. In figures 3.1 and 3.2, the first row depicts the query shape, and in each column the results in order of relevance are displayed. The retrieved shapes are in most cases very similar to the input, while in every case there is some resemblance. It is experimentally exhibited with these results that the method is suitable for varying tasks.

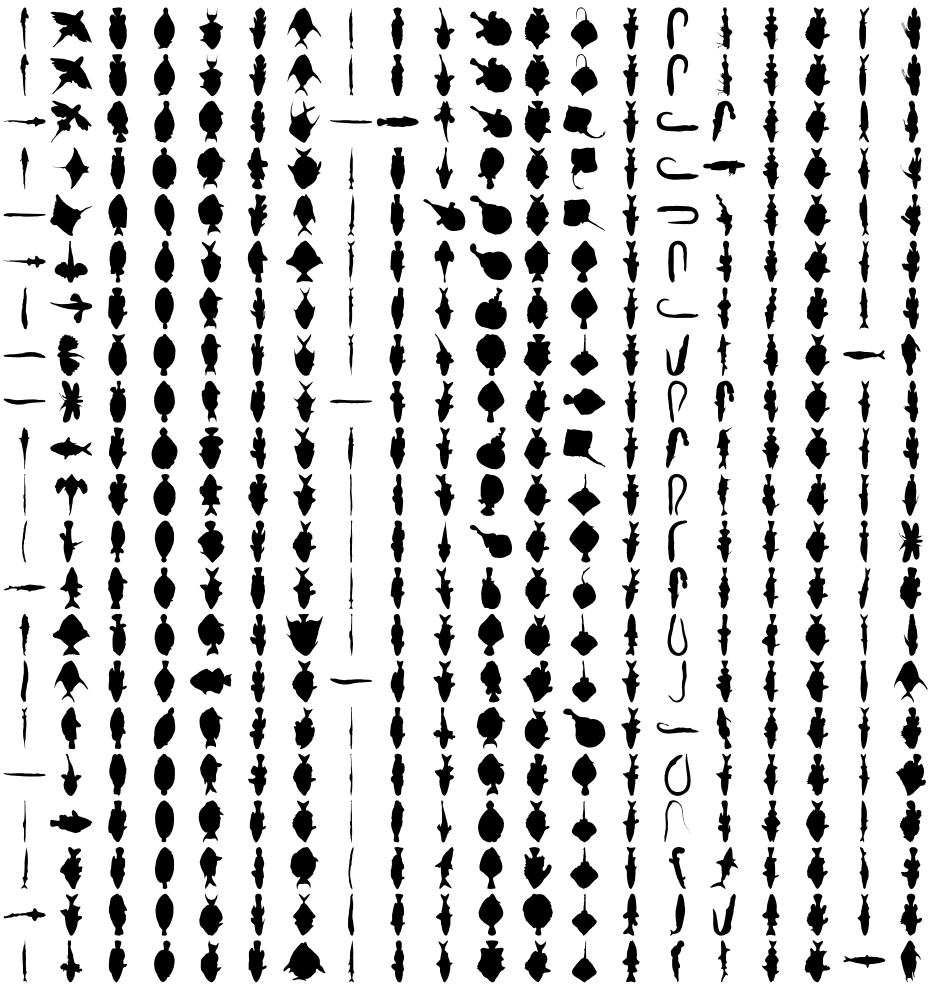


Figure 3.1: Results for the marine database.

3.2 Bulls-eye test

Apart from the qualitative assessment presented in the previous section, more experiments quantitatively assessing the method were conducted. The presented methodology was evaluated quantitatively by performing the bulls-eye test on the MPEG-7 CE-shape-1 part B dataset [18], a widely used evaluation method for shape matching methods. The shape database for this experiment consists of 70 categories of 20 shapes each, for a total of 1400 shapes. There are many types of shapes including faces, household objects and other human-made objects, animals, and some more abstract shapes.

The mpeg7 bulls-eye test is performed as follows: it is assumed that the shape matching method can rank all the shapes in the database according to their similarity with a

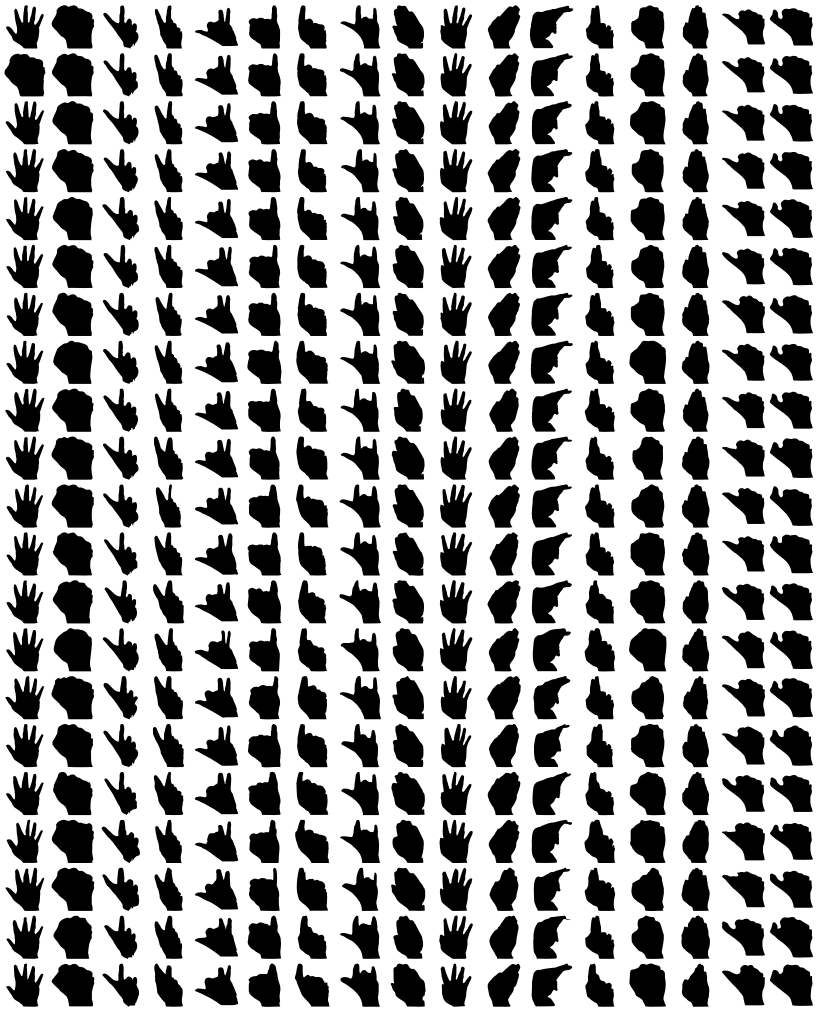


Figure 3.2: Results for the gestures database.

given query shape. Each one of the shapes in turn plays the role of the query, so the full all-to-all comparison must be evaluated. Among the first forty results of each ranking, the number of objects similar to the query is counted. The percentage of the correct retrievals in these places is the final bulls-eye score.

The result of the present method for the bulls-eye test is 72.35%. The method parameters are: 100 points per shape, 12 bins in the angular and 5 in the radial dimension, shape context radius 2 (this refers to sizes after the scale normalization) and small radius .125, thin plate spline regularization parameter $\lambda = 1$ as stated earlier and insertion/deletion cost for the cyclic matching manually chosen to be 1.5. The cyclic matching cost is only taken into account, not utilizing the TPS cost ($l_1 = 1$ and $l_2 = 0$). This set of param-

eters is used throughout all the experiments of this work. The presented method does not natively handle mirroring, so the minimum of the costs to the original or mirrored shape is used where necessary. Using the graph transduction method [45] with the values proposed by the authors, the percentage is increased to 75.42%. As a comparison, the claimed performance for [7] is 76.45%.

The result of the proposed method without graph transduction is presented in figures 3.6 and 3.7. The shapes at the top row are the queries, and for each query the first forty retrieved shapes are displayed in the column in order of relevance.

A way to visualize the bulls-eye test results is presented in figure 3.3. This graph essentially turns the rather arbitrary limit of forty best results discussed earlier into a variable. The horizontal axis of the graph is this variable recall length, and the vertical axis is the percentage of correct results among the examined ones. The first twenty points of the curve follow a downward trend, which is then reversed. This happens because after the twentieth point the denominator of the percentage (the number of possible correct answers) remains constant, while the numerator still increases, since many shapes that were not matched absolutely correctly in the first twenty places still happen to occur close to this range. The reported score of the test is the fortieth point of this curve.

The lowest of the three dashed profiles corresponds to the results of the proposed method. The first point of this profile denotes a performance of 100%, since each shape has zero matching cost to itself. The second point denotes a performance of 98.5%, meaning that the second best match for each shape (that is, after itself) is at almost all the cases from the same category. However this number quickly drops to 66% at the first 20 matches, and only increases to the reported value of 72.35% for the first 40 matches. This is in contrast with the other dashed profile corresponding to transduction results: the first twenty shapes already contain most of the relevant shapes, since the curve after this point is almost flat.

Similar behaviour but with lower overall performance is observed when substituting the cyclic string matching algorithm of the proposed method with the Maximum Weighted Bipartite Matching (MWBM). This is an attempt to evenly compare the presented method with the method proposed in [7]. The iterative scheme proposed in [7] is not used in any of these results. The lower performance of this method (the proposed

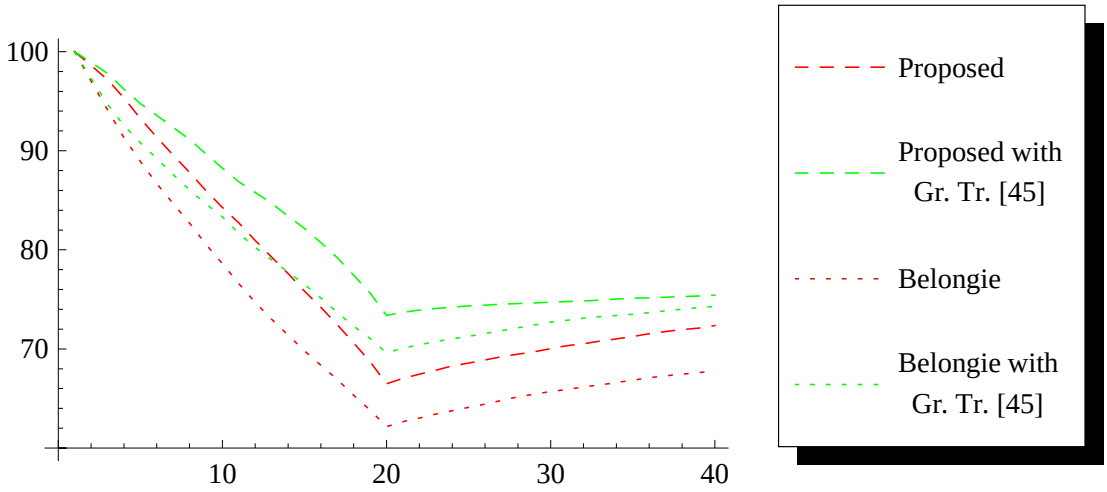


Figure 3.3: The bulls-eye performance of the method on the mpeg7 dataset as a function of the recall depth. The lowest of the dashed profiles corresponds to the original method, while the other corresponds to the result after graph transduction [45]. The dotted profiles correspond to the same method with the bipartite matching algorithm utilized in [7].

with MWBM instead of cyclic matching) can be partly attributed to the fact that this algorithm cannot match correctly shape contexts that have been rotated according to the local tangent as described in section 2.2.2.1. As shown in the graph, the bulls-eye score of the proposed method with MWBM instead of cyclic matching is 67.8%, a figure that increases to 74.3% after graph transduction.

3.3 Detection of articulation points in human figures

For the purpose of human joint detection, the system uses a set of model figures. Each one of them is accompanied by hand-labeled points on it, indicating the positions of joints. This constitutes the initialization of the system. Each figure that is provided to the system for detection undergoes a comparison with each one of the model figures. The model with the lowest score is picked as corresponding to the input. The TPS transformation between the model and the input is subsequently used to warp the labeled control points on the input image. This concludes the outline of the joint detection method, enabling the

estimation of a stick-figure that follows the pose of the observed human figure.

For experimental purposes with real data, images acquired by a camera in a realistic situation should be used. However, the input of the methodology presented in section 2 is a shape as defined in section 1.1.1. The process that outputs shapes given raw images as input is described in the following.

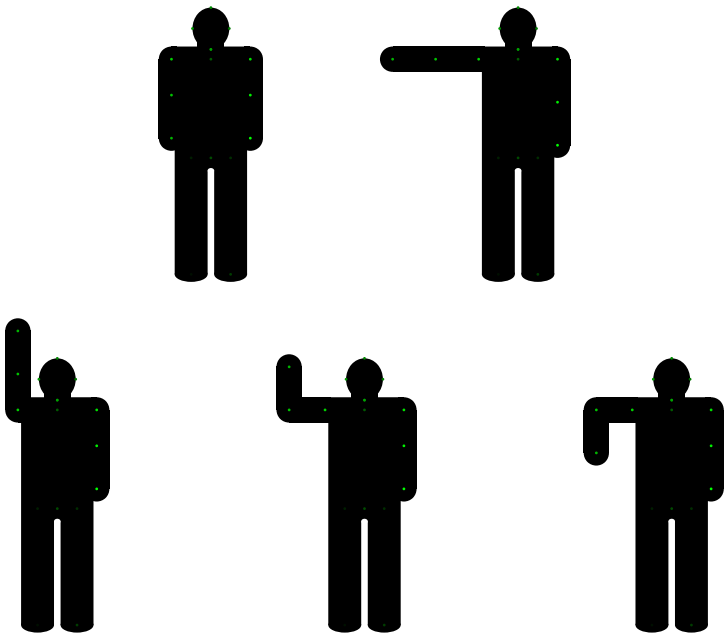


Figure 3.4: The five different model configurations for the right hand.

3.3.1 Preprocessing for figure extraction

Throughout all the joint detection experiments presented below, the image acquisition was performed using the Dragonfly 2 camera by Point Grey Research [30]. The first processing step, right after image acquisition is to segment the foreground objects from the background, thus forming figures for further processing. For this work, the image sequences were segmented using the method presented in [46]. The input to this method is an image sequence, and the output is an image sequence with foreground marked as white (grey level 255) and background as black (grey level 0). The method has the ability to use another grey level value (in the implementation the value 127 is used) to indicate

shadows, and so anything that is not white is not of interest for the purpose of figure extraction.

Briefly, this method utilizes a separate background model for each pixel. This model uses Mixtures of Gaussians with a varying number of distinct Gaussian kernels. For each frame, the mixture model is updated so as to "forget" old observed values, and adapt to new permanent situations. The update rules for the Gaussian Mixture Model follow the method presented in [38] and [16]. Also, in every update, the number of samples that support each Gaussian curve are taken into account and the less supported ones are discarded.

3.3.2 Edge extraction - Point selection

Each image of the sequence is treated separately, first by extracting edges. Although in general the process of edge extraction requires time proportional to the size of the image (one has to look at each pixel at least once), in the case of binary images this time can be reduced to be proportional to the number of edge pixels (edgels). This number is usually considerably smaller than the total number of pixels in the image. This speedup can be achieved by exhaustively searching the pixels for an edgel, and then following this edge. In the case that there exist more than one connected components on the segmented image, the one with the largest area is selected for further processing. After a list of edge pixels is formed, it is easy to select a fixed number of roughly equidistant pixels. These pixels are the actual input to the method of chapter 2.

3.3.3 Model creation

As previously stated, the joint detection system needs labeled models. For this purpose, a simple figure generator with four parameters was created. The four parameters are the angles at each shoulder and elbow. Labeled points are automatically generated for each figure. There are sixteen labeled points for each figure: ankles, hips, waist, neck, shoulders, elbows, wrists and four control points for the head. Figures produced by this generator are displayed in figure 3.4 along with marked points on them.

Figure 3.5: Results of the joint detection method.

3.3.4 Matching and annotation

The implemented system is initialized with labeled human figures as previously stated. Specifically, five different poses for each hand are utilized, for a total of twenty five models. The five poses for the right hand are displayed in figure 3.4. After the initialization, the system processes each input image as already described to produce a shape. This shape is compared with all the models, and the one that better matches the observed shape according to the shape distance as defined in chapter 2 is selected. The thin plate spline transformation that was calculated for the matching step is used to calculate the positions of the labeled points of the matched model. This concludes the process, enabling the estimations shown in figure 3.5.

3.4 Implementation and practical issues

Heavy use of morphological processing was necessary because the segmentation process in some cases failed to output consistent results. Specifically, the segmented figures displayed large holes, comparable in size to the figure itself, something that is evident in image 3.5, even after the processing. The morphological operations that were used are equivalent to heavy smoothing. However, smoothing is generally not necessary for the proposed method, since it performs well in the presence of white noise. It can be argued for the particular case that the shape is altered by the specific type of noise. The case resembles occlusion, something that was not explicitly accounted for in the method. As a conclusion, the method exhibits moderate tolerance to occlusion, but the performance deteriorates as the amount of occlusion is increased.

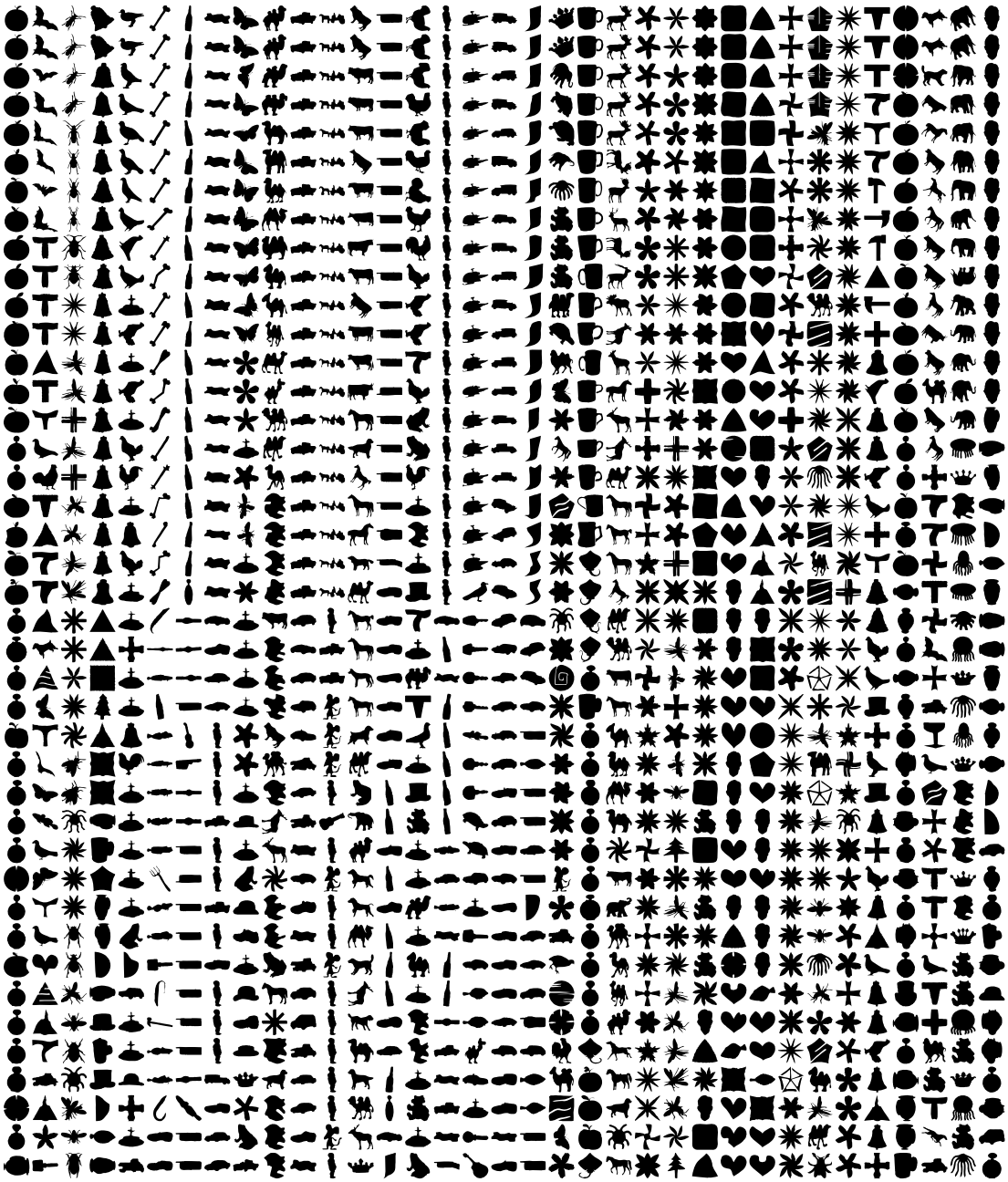


Figure 3.6: Visualization of the mpeg7 bulls-eye test results part 1

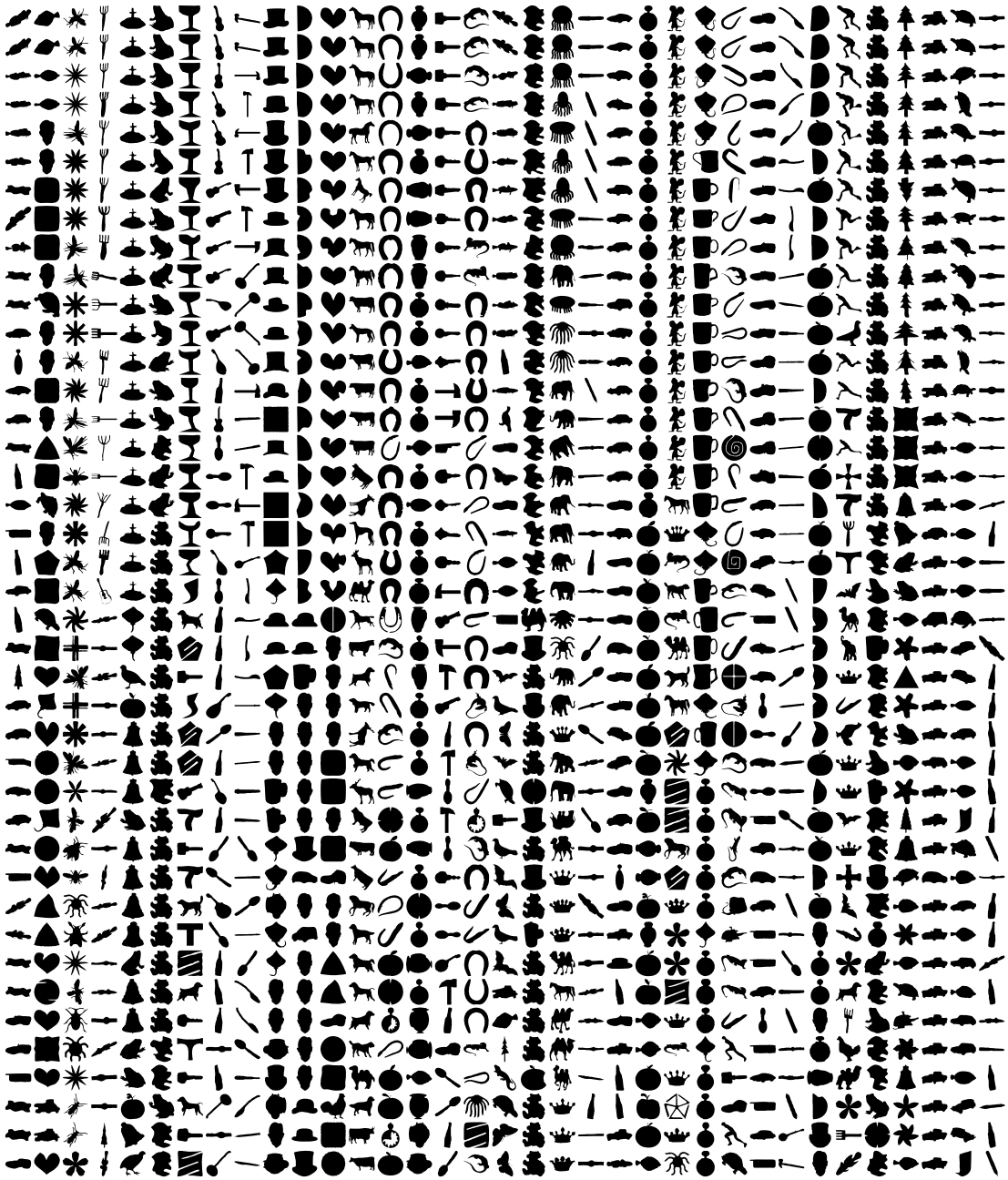


Figure 3.7: Visualization of the mpeg7 bulls-eye test results part 2

Chapter 4

Discussion

The method proposed by Belongie et al. in [7] utilizes the information that shape contexts capture regarding local shape similarity. Under the assumption that the shapes of interest are well described by a single closed contour, an extra piece of information is available for use. This is the global order of the points on the contour. Belongie et al. use Maximum Weighted Bipartite Matching (MWBM) as the matching method [7], which does not take into account any ordering information. Enforcing the constraint that the points of each shape are matched according to their relative order demands a different matching method. As already described, the cyclic string matching method presented in [36] fulfills the requirements.

This decision reflects the basic assumption made about shapes, namely that each shape consists of a single closed contour. The method performs well when this assumption holds, but the shapes handled correctly using MWBM may consist of more contours. As exhibited in the experimental sections of [6], this feature is useful since many real-world shapes consist of more than one contour.

The computational complexity of the presented method is dominated by the matching step. Assuming that the number of points of each shape is n[1], the computations of the shape context descriptors and of the cost matrix C are both $O(n^2)$. The cyclic string matching employed in this work has a computational complexity of $O(n^2 \log(n))$, while the MWBM is at the order of $O(n^2 \log(n) + nE) = O(n^3)$ (for the case in study, the number E of edges in the graph is $O(n^2)$). The final warping step has also a complexity of $O(n^2)$.

[1] The extraction of these points is also a runtime consideration as discussed in sections 3.3.1 and 3.3.2.

This difference of complexities is justified since the MWBM has a larger search space than the cyclic string matching. Notice also that these complexities are not necessarily lower bounds for the respective problems.

The theoretical time complexity advantage of the method can also be a practical one. The implementation used for the presented results is not optimized for speed, yet the system can perform joint detection at a rate that was higher than one frame per second, that is, 25 comparisons per second. The processing time for the edge extraction at each frame was comparable to the time required for all the comparisons. This means that an implementation explicitly designed to perform this task in real time is feasible.

The properties of the discussed method are:

- Translation invariance. The method is fully translation invariant since all measured distances are relative to the position of a point on the shape.

- Rotation invariance. The method is robustly rotation invariant for the same reason as above: all measured angles are relative to the local tangent.

- Scale invariance. The method is partially scale invariant, however there is room for improvement in this domain.

It should be noted that the method is mathematically invariant to these three transformations (similarity transformations). A set of n points undergoing only similarity transformations will have precisely 0 distance from the original shape (using the present method). Practically, even in the case of "similar" (in the sense of similarity transformations) shapes, the actual compared points may be slightly different if they occur as a sampling of an edge map. In any case however, the distance between "similar" shapes will be quite small. In practice, for the cases of interest, similarity transformations are not the only type of distortion that is observed. In the presence of more complex distortions, the method is invariant to translation, robust to rotation, but has room for improvement in the scaling transformation.

- Affine transformation invariance. The general projective transformation is not explicitly handled, however the nature of the descriptor makes the method robust to small amounts of affine transformation.

- Occlusion handling. Small amounts of occlusion are handled adequately, however the performance quickly deteriorates as the occlusion size grows. This type of deformation essentially affects the rough scale computation and the contour sampling process. These estimations are the basis of the whole method, and so the method is sensitive to occlusions.

- Noise handling. The nature of the descriptors makes the method robust to small amounts of noise, and the performance deteriorates gracefully with the amount of noise.

- The method does not match mirrored shapes. If this is desired, one has to compare the shape two times: one time as usual, and one after mirroring one of the shapes. The final match will then be the one with minimum distance between the shapes.

- It should be also noted that the proposed method cannot be used for partial matching. Even if a subset of the points has identical shape (under similarity transformations), the method will not be able to locate this similarity unless the common subset is actually the biggest part in both shapes.

In the introductive chapter of this work, the issue of affine transform is discussed. The presented method exhibits reasonable robustness but is sometimes prone to the problem exhibited in figure 1.1. The output of the proposed method contains an estimation of the affine transformation that roughly aligns the shapes (the linear part of the TPS transformation). It is possible to decompose this linear map and use the ratio of its eigenvalues as an extra measure of similarity. The ratio will be close to one for similar shapes, but large for spurious cases like the one of figure 1.1. Also, in the case that full rotation invariance is undesirable, one can either omit the derotation step in section 2.2.2.1, or extract and penalize the rotation from the linear part of the TPS transformation.

The properties of the proposed method as a comparison function are discussed below. One can view the result of the proposed method (or any other shape matching method that estimates a distance between the compared shapes) as a mapping from the space of pairs of shapes to the positive reals. Using the definitions of this work, the mapping can

be written as a function *comp*:

$$comp : \mathbb{R}^{2n} \times \mathbb{R}^{2n} \rightarrow \mathbb{R}_+$$

while by restricting the first argument of this function to a given shape sh_1 the function $comp_1(sh) = comp(sh_1, sh)$

$$comp_1 : \mathbb{R}^{2n} \rightarrow \mathbb{R}_+$$

is derived. This function represents the final value d that will be the result of the process of comparing a given shape sh with the specified shape sh_1 and can be studied for its properties. An important property of this function is the level set of $comp_1$ for varying values of the distance d. Level set of a function f on a parameter vector x for a value l is defined to be the set $\{x_l | f(x_l) = l\}$, essentially denoting the contour line(s) of the function f (given that f is smooth enough to form lines). In the case under study, the level set for a given value d is the set of all shapes that yield distance d to the shape sh_1. This set is interesting because it allows us to understand the behavior of the comparison method under consideration.

The case $d = 0$ defines the set of all shapes that are "identical" to the shape sh_1. As already stated, this set contains all the shapes that are rotated, translated and scaled versions of sh_1. For small perturbations of the points that comprise the shape, it is possible that every shape context does not change at all (for every computed shape context, the points that fall in each bin remain the same). The matching cost will then be again 0, as in the case of identical shapes. However, the TPS energy will not be 0 any more, and so, unless the parameter l_2 is set to 0, the method will yield 0 distance only in the case that the compared shape sh is the shape sh_1 under a similarity transformation.

Another interesting case occurs for small values of the distance d, defining a level set of shapes that are close to sh_1 but not identical under similarity transforms. The function $cost_1$ is not smooth, since it has a discrete step, namely the establishments of correspondences between the compared shapes. This makes the analysis of the level set under consideration non-trivial. However, it is reasonable to assume that locally transformed versions of sh_1 with different specific transformations can belong in the set under discussion.

The case of large values for d does not pose so much interest, since the set of shapes

that are dissimilar to sh_1 is very large. Many different shapes with no apparent similarities can belong in a given level set of sh_1.

4.1 Future work

An important issue that remains to be resolved is the automatic determination of the insertion/deletion cost. A fixed value was experimentally estimated for all the experiments. As already mentioned, it might be possible to devise a method that adjusts this cost as a function of the matching matrix. However, it remains to be tested whether such a scheme can result in better matches and ultimately a better overall performance.

Another issue directly related to the previous one is that of the contour sampling. It proved sufficient to extract the contour of a shape, and then select the required fixed number of points at roughly equal distances. However, one can reasonably argue that such a selection is not optimal. The same holds for the case of the MWBM, but it is even more important in the cyclic DTW case. Assume that a similar part of two shapes is for some reason unequally sampled (bad scale estimation and occlusion can actually have this effect). The cyclic matching algorithm will have to use insertion and/or deletion operations to match these similar parts, adding unnecessary cost to this matching. It can be argued that this matching may still have the lowest cost, but the nature of the problem makes it difficult to be certain about such a claim.

Bibliography

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

[2] N. Arica and F. Vural. A perceptual shape descriptor. *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, 3:375–378 vol.3, 2002.

[3] A. R. Backes, D. Casanova, and O. M. Bruno. A complex network-based approach for boundary shape analysis. *Pattern Recognition*, 42(1):54 – 67, 2009.

[4] R. H. Bartels, J. C. Beatty, and B. A. Barsky. *An introduction to splines for use in computer graphics & geometric modeling.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[5] R. Basri, L. Costa, D. Geiger, and D. Jacobs. Determining the similarity of deformable shapes. *Vision Research*, 38:135–143, 1998.

[6] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:509–522, 2002.

[7] S. Belongie, G. Mori, and J. Malik. Matching with shape contexts. In *IEEE Workshop on Content-based access of Image and Video-Libraries*, page 20, 2000.

[8] F. L. Bookstein. Principal warps: thin-plate splines and the decomposition of deformations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(6):567–585, 1989.

[9] Y. Chi and M. K. H. Leung. A general shape context framework for object identification. *Comput. Vis. Image Underst.*, 112(3):324–336, 2008.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2 edition, September 2001.

[11] R. da S. Torres and A. Falcγo. Contour salience descriptors for effective image retrieval and analysis. *Image and Vision Computing*, 25(1):3 – 13, 2007. SIBGRAPI.

[12] Y. Ebrahim, M. Ahmed, W. Abdelsalam, and S.-C. Chau. Shape representation and description using the hilbert curve. *Pattern Recogn. Lett.*, 30(4):348–358, 2009.

[13] Euripides Petrakis. Shape Datasets and Evaluation of Shape Matching Methods for Image Retrieval, January 2009. `http://www.intelligence.tuc.gr/ petrakis/`.

[14] P. Felzenszwalb and J. Schwartz. Hierarchical matching of deformable shapes. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.

[15] A. Frome, D. Huber, R. Kolluri, and T. Bulow. Recognizing objects in range data using regional point descriptors. In *in Proceedings of the European Conference on Computer Vision (ECCV)*, pages 224–237, 2004.

[16] E. Hayman and J. olof Eklundh. Statistical background subtraction for a mobile observer. In *In Proceedings ICCV*, pages 67–74, 2003.

[17] D. D. Hoffman, W. Richards, A. Pentl, J. Rubin, and J. Scheuhammer. Parts of recognition. *Cognition*, 18:65–96, 1984.

[18] S. Jeannin and M. Bober. Description of core experiments for mpeg-7 motion/shape, 1999.

[19] D. G. Kendall. Shape manifolds, procrustean metrics, and complex projective spaces. *Bull. London Math. Soc.*, 16(2):81–121, March 1984.

[20] L. J. Latecki, V. Megalooikonomou, Q. Wang, and D. Yu. An elastic partial shape matching technique. *Pattern Recogn.*, 40(11):3069–3080, 2007.

[21] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, 1966.

[22] H. Ling and D. Jacobs. Shape classification using the inner-distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):286–299, February 2007.

[23] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31:983–1001, 1998.

[24] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 1150, Washington, DC, USA, 1999. IEEE Computer Society.

[25] F. Mokhtarian, S. Abbasi, and J. Kittler. Robust and efficient shape indexing through curvature scale space. In *In Proceedings of British Machine Vision Conference*, pages 53–62, 1996.

[26] G. Mori and J. Malik. Estimating human body configurations using shape context matching. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III*, pages 666–680, London, UK, 2002. Springer-Verlag.

[27] G. Mori and J. Malik. Recognizing objects in adversarial clutter: breaking a visual captcha. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 1:I–134–I–141 vol.1, June 2003.

[28] E. Mortensen, H. Deng, and L. Shapiro. A sift descriptor with global context. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 1:184–190 vol. 1, June 2005.

[29] T. Pavlidis. A review of algorithms for shape analysis. *Computer Graphics and Image Processing*, 7(2):243–258, 1978.

[30] Point Grey Research, Inc. Imaging Products - Dragonfly2, January 2009. `http://www.ptgrey.com/products/dragonfly2/index.asp`.

[31] Z. Rao, E. Petrakis, and E. Milios. Efficient retrieval of deformed and occluded shapes. volume 4, pages 67–71 vol.4, 2000.

[32] M. D. Rodriguez and M. Shah. Detecting and segmenting humans in crowded scenes. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, pages 353–356, New York, NY, USA, 2007. ACM.

[33] P. J. Rousseeuw and C. Croux. *Journal of the American Statistical Association*, 88(424):1273–1283, December 1993.

[34] H. Sakoe. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978.

[35] H. Sakoe and S. Chiba. A dynamic programming approach to continuous speech recognition. In *Proceedings of the 7th International Congress on Acoustics, Budapest*, 1971.

[36] F. Schmidt, D. Farin, and D. Cremers. Fast matching of planar shapes in sub-cubic runtime. *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–6, October 2007.

[37] I. J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. *Quart. Appl Math. 4*, pages 45–49 and 112–141, 1964.

[38] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. volume 2, page 252 Vol. 2, 1999.

[39] C. J. Taylor. Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *Comput. Vis. Image Underst.*, 80(3):349–363, 2000.

[40] R. Torres, A. Falcao, and L. Costa. Shape description by image foresting transform. *Digital Signal Processing, 2002. DSP 2002. 2002 14th International Conference on*, 2:1089–1092 vol.2, 2002.

[41] R. C. Veltkamp and M. Hagedoorn. State-of-the-art in shape matching. Technical report, Principles of Visual Information Retrieval, 1999.

[42] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh. Indexing multidimensional time-series with support for multiple distance measures. In *KDD '03:*

Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 216–225, New York, NY, USA, 2003. ACM.

[43] G. White and R. Neely. Speech recognition experiments with linear prediction, bandpass filtering, and dynamic programming. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 24(2):183–188, April 1976.

[44] A. Wu, P. Tsang, T. Yuen, and L. Yeung. Affine invariant object shape matching using genetic algorithm with multi-parent orthogonal recombination and migrant principle. *Applied Soft Computing*, 9(1):282 – 289, 2009.

[45] X. Yang, X. Bai, L. J. Latecki, and Z. Tu. Improving shape retrieval by learning graph transduction. In *ECCV (4)*, pages 788–801, 2008.

[46] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, 2:28–31, August 2004.