Using Shape Contexts For Shape Matching

(CS269 - Visual Modeling Course Project)

Amogh Param
Department of Computer Science
University of California,
Los Angeles
aparam@cs.ucla.edu

Deepthi N Rao
Department of Computer Science
University of California,
Los Angeles
deepthinrao@cs.ucla.edu

Saransh Gupta
Department of Computer Science
University of California,
Los Angeles
saransh@cs.ucla.edu

Abstract - Images of the same shape may vary significantly in their vector representations. To match these shapes, we compute the shape context, which is a shape descriptor that captures the relative positions of other points on the shape contours. This gives a globally discriminative characterization of the shape and not just a localized descriptor. For matching, we find the correspondence between these points by posing it as a weighted bipartite graph-matching problem (optimal assignment). Having found the correspondence, we then use regularized thin-plate splines for finding the aligning transform that best matches the shapes. Finally, we conclude by suggesting a potential application of our implementation for object recognition.

Contents

1. Introduction

With more and more images being generated in digital form, there is a growing interest in finding images from large databases of images. Retrieval of images can be done based on some descriptor such as the 'shape' - which is an important visual feature. Matching of shapes may be a trivial task for humans, but visually similar shapes may differ significantly in their vector representations. Consider the two digits in *Figure 1*. When the image intensity vectors of these two digits are compared using L2 norms, they would seem very different. But, these are two different ways of writing the number 2, i.e. they belong to the same category.

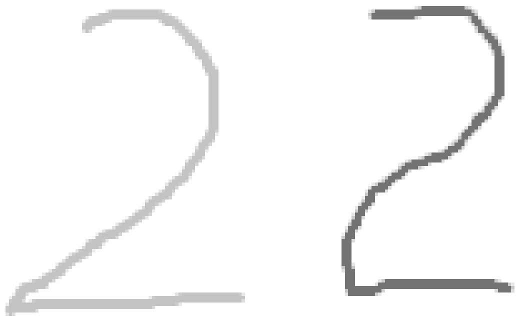


Figure 1. Examples of two handwritten representations of the same digit which are signficantly different pixel-to-pixel

In this project, we implement shape matching using shape contexts, which does the following tasks (*Figure 2*):

1. Extract a rich shape descriptor from the images of the shapes.
2. Solve the correspondence problem between two shapes.
3. Use the correspondences to find an aligning transform.
4. Compute the shape distance metric that quantifies the dis-similarity between the shapes.

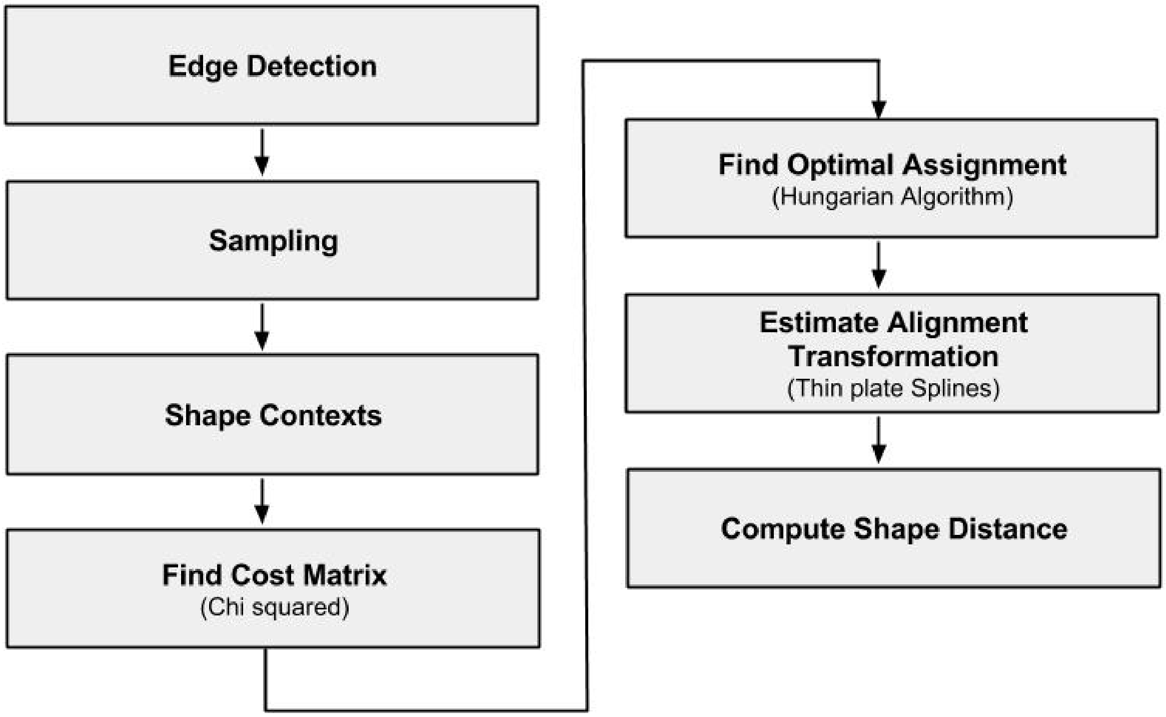


Figure 2: Processing Steps

Given the shape distance of an input image to a database of images or a set of selected prototypes that best represent each possible class of shapes, we can classify the image into the correct category by using simple algorithms like the K-nearest neighbor classifier.

2. Methodology

In this project, we used MATLAB for the implementation of the shape matching method described in [1]. In this section, we lay down the finer details and explain all the steps involved.

The first step is to extract the inner and outer boundary edges of the object. These boundary pixels are then sampled. Only a small number of points are used for purposes of computational tractability. The sample points are then used for the computation of the shape descriptor – the shape context. Since it is a matching problem, the shape contexts are computed for both the input image and the target prototype. We then estimate the best correspondence between points from the input image to the target image based on the chi-squared distance between the shape contexts of the two images. Regularized thin plate splines are then used to estimate the aligning transform that best matches the sampled points on the first image to the corresponding points on the target image. Lastly, the 'shape distance' is calculated using the weighted sum of three different measures of dissimilarity, as detailed in *Section 2.6*.

2.1 Edge Detection

In image processing, an edge of an object in an image is defined as the region where a sharp change in the image brightness value is present. These pixels point out the difference between neighboring pixels. They may or may not have loops. Any edge detector algorithm may be used to obtain edge points from the image. Contour points are obtained from edges. These pixels are closed loop of points that define the boundary or the shape of objects which can be used for matching and recognition purposes. Objects with holes have multiple sets of these shape-describing points. Contours can be obtained at different levels, with the height measured from the x-y plane.

2.2 Sampling

A shape is described by a discrete set of points sampled from the internal and external contours on the object [1]. The obtained points are then sampled to get the optimal number of points that best describe the contours of the shape. This makes the computation reliable, economical and much faster. In our experiment, we choose hundred uniformly sampled points to describe the shape of the object. Assuming that the contours are piecewise smooth, we can obtain a good approximation to the original continuous shapes.

2.3 Shape Descriptor

Given a set of sample points on an image, we wish to find a rich descriptor that encodes both the local and global information about these points. Before we describe the shape context descriptor, we briefly review one broad categorization of descriptors:

1. Brightness-based: they make direct use of the brightness of pixels.
2. Feature-based: they involve the use of spatial arrangements of extracted features such as edge elements or junctions.

2.3.1 Brightness based methods

An approach to find shape descriptors is to make direct use of the grey values within the visible portion of the object, rather than concentrating on the shape of the contour or other extracted features. One category of approaches computes the correspondences or the alignment using the grey-scale values. Other categories of brightness based methods build classifiers without explicitly finding the alignment or correspondences. Such methods rely on having a large dataset of training examples with which a learning algorithm can be trained to match shapes.

Some of the popular methods based on this approach include Principal Component Analysis (PCA), Support Vector Machines (SVM) and the LeNet classifier.

2.3.2 Feature based methods

Feature vectors that contain descriptors such as area or moments can be computed and then, these vectors between the shapes can be compared. But such techniques often discard detailed shape information in the process. There has been a lot of research in the domain of shape similarity using silhouette images. Since silhouette images have the property of not having holes or other internal markings, the resulting boundaries are represented by a single-closed curve, which can be parameterized by arc length. Since this curve is one dimensional, a convenient solution to matching would be to use dynamic programming approaches.

Since these silhouettes ignore internal contours and are difficult to extract from real images, they are limited in their application as shape descriptors for general objects. Methods that treat the shapes as sets of points in the two dimensional image are much rather preferred as they are easier to be extracted. We do not use this method in our project because it doesn’t return correspondences. Other methods to find shape similarity include casting sample points in images into finite element spring mass models, and eventually finding correspondences by comparing modes of vibration. Yet other methods use spatial configurations of a small number of key ‘landmark’ points. These methods sacrifice the shape information available in smooth portions of the object contours.

2.3.3 Shape Context

The shape context is a feature-based descriptor that is used for correspondence based matching of shapes. For each boundary point, we calculate the shape context, thereby reducing the problem of finding the correspondences between the boundary points as a problem of optimally assigning the shape contexts of these boundary points of the shapes to be matched.

Consider the set of vectors originating from one boundary point to all the other boundary points. This acts like a good descriptor of the shape relative to that point since it captures a lot of global information about the shape. However, as the number of sample boundary points increases, using all these vectors becomes unfeasible. In order to reduce the descriptor length, these distance vectors originating from point

p_i are coarsely binned in order to give a histogram $h_{i,}$ which represents the distribution of relative distance between the points.

$$h_i(k) = \#\{q \neq p_i : (q - p_i) \in bin(k)\}$$

The histograms are taken uniformly in the log-polar space. This makes the descriptor more sensitive to nearby points than those that are farther away. This makes it more robust to noise and outliers.

2.4 Cost Matrix

Once we compute the shape contexts, we consider the problem as an optimal assignment problem. As the shape-contexts are distributions represented as histograms, we use the chi-squared distance to compute the cost matrix where, $C_{ij} = C(p_i,q_j)$ represents the cost of matching a point p_i on the first shape and a point q_j on the second shape.

$$C_{ij} \equiv C(p_i, q_i) = \frac{1}{2}\sum_{k=1}^{K} \frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)}$$

Where, $h_i(k)$ and $h_j(k)$ denote the K-bin normalized histogram at p_i and q_j respectively.

2.4.1 Optimal Assignment

Using the cost matrix we try to minimize the total cost of matching, given the set of costs, C_{ij}, between all pairs of points p_i on the first shape and q_j on the second shape.

$$H(\pi) = \sum_i C(p_i, q_{\pi(i)})$$

We match based on the constraint that the matching between the points is one-to-one. This optimal assignment or bipartite graph matching problem is solved in $O(N^3)$ time using the Hungarian algorithm which provides a dynamic programming solution, where the input to the algorithm is the square cost matrix and the output is a permutation, $\pi(i)$, such that the above equation is minimized.

To handle cases where we have outliers, we can add “dummy” nodes to each point set with a constant matching cost of $\varepsilon_{d,}$ which can essentially be considered as the threshold for outlier detection. A point is matched to one of these nodes when there is no real match available at a cost lower than ε_d. This can also be applied to the case when the cost matrices of the two shapes are not equal. We can add dummy nodes to the smaller point set and hence get two square cost matrices and proceed with the matching.

2.5 Alignment Transform

Given the correspondences between the boundary points based on the chi-squared distance between the shape context descriptors, we need to find a plane transformation T: $R^2 \rightarrow R^2$ that can be used to map the points on the input image to the corresponding points on the target image.

One broad category of transformation models includes linear transformations, which include rotation, scaling, translation, and other affine transforms. Linear transformations are global in nature, thus, they cannot model local geometric differences between images.

The second category of transformations allows 'elastic' or 'non-rigid' transformations. These transformations are capable of locally warping the target image to align with the reference image.

Non-rigid transformations include the thin plate spline transformation (TPS) model. The thin plate spline is the 2D generalization of the cubic spline. In its regularized form, which is discussed below, the TPS model includes the affine model as a special case.

Given the displacement vectors of the points on the input image (x_i,y_i) to their corresponding points on the target image (x_i',y_i') with values v_i , we employ two thin plates for modeling the transformations along the x and the y directions. That is, we interpolate a planar deformation field from the sparse displacement vectors.

The 2D vector field/ transformation is given by:

$$T(x, y) = (f_x(x, y), f_y(x, y))$$

The Thin plate deformation energy for interpolation of the data along one direction (x or y) is given by:

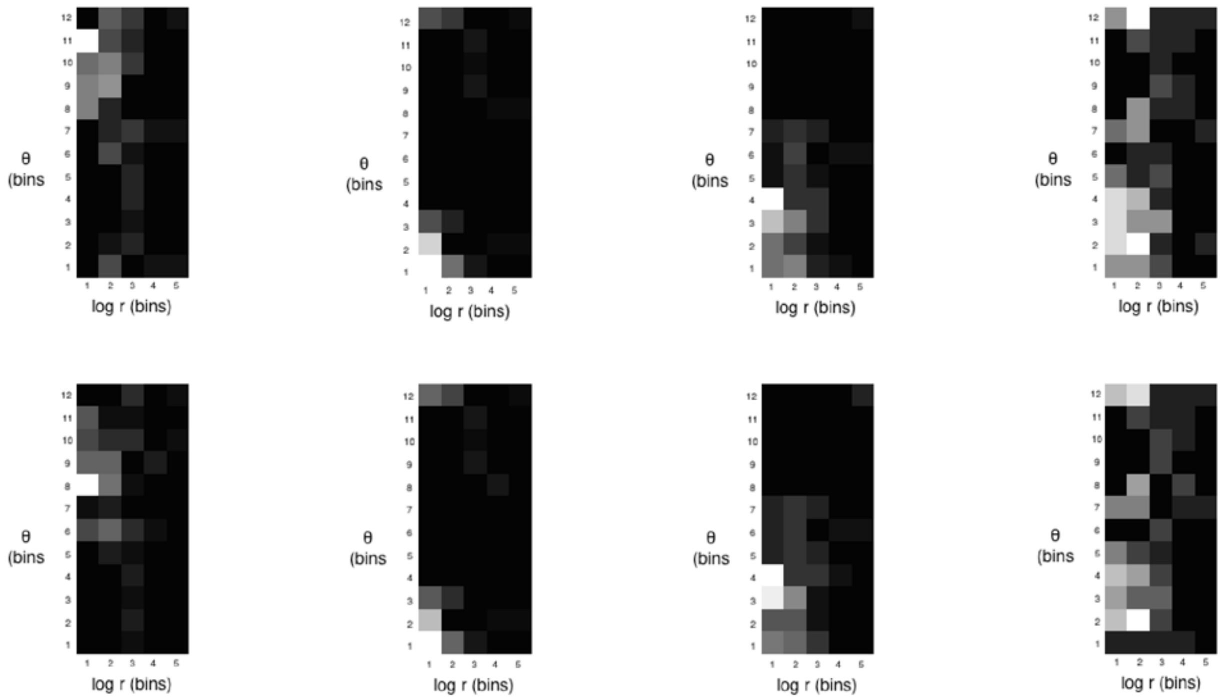


Figure 3: Log-polar histogram bins for computing Shape Context of 4 distinct sampled boundary points on the target image *(top row)* and for the corresponding points on the input image *(bottom row)*

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

This TPS model is a special case of the poly-harmonic spline and has an implicit representation in the form of the radial basis function. The interpolant *f(x,y)* is given by :

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

In the above equation, the first three terms are the affine terms that correspond to translation, scaling along *x* direction and scaling along the *y* direction respectively.

Where, $U(r) = r^2 \log r$, $f(x_i,y_i) = v_i$ and the following conditions hold true:

$$\sum_{i=1}^{n} w_i = 0 \; and \; \sum_{i=1}^{n} w_i x_i = \sum_{i=1}^{n} w_i y_i = 0$$

Thus we get a linear system of equations that can be solved for the TPS coefficients:

$$\begin{bmatrix} K & P \\ P^T & O \end{bmatrix} \begin{bmatrix} w \\ a \end{bmatrix} = \begin{bmatrix} v \\ o \end{bmatrix}$$

Where, $K_{ij} = U(\| (x_i, y_i) - (x_j, y_j) \|)$, the ith row of P is $(1, x_i, y_i)$, w and v are column vectors formed from w_i and v_i, respectively, and a is the column vector with elements a_1, a_x, a_y. We will denote the (n + 3) x (n + 3) matrix of this system by L. L is nonsingular and we can find the solution by inverting L. If we denote the upper left n x n block of L^{-1} by A, then we have

$$I_f \alpha v^T A v = w^T K w$$

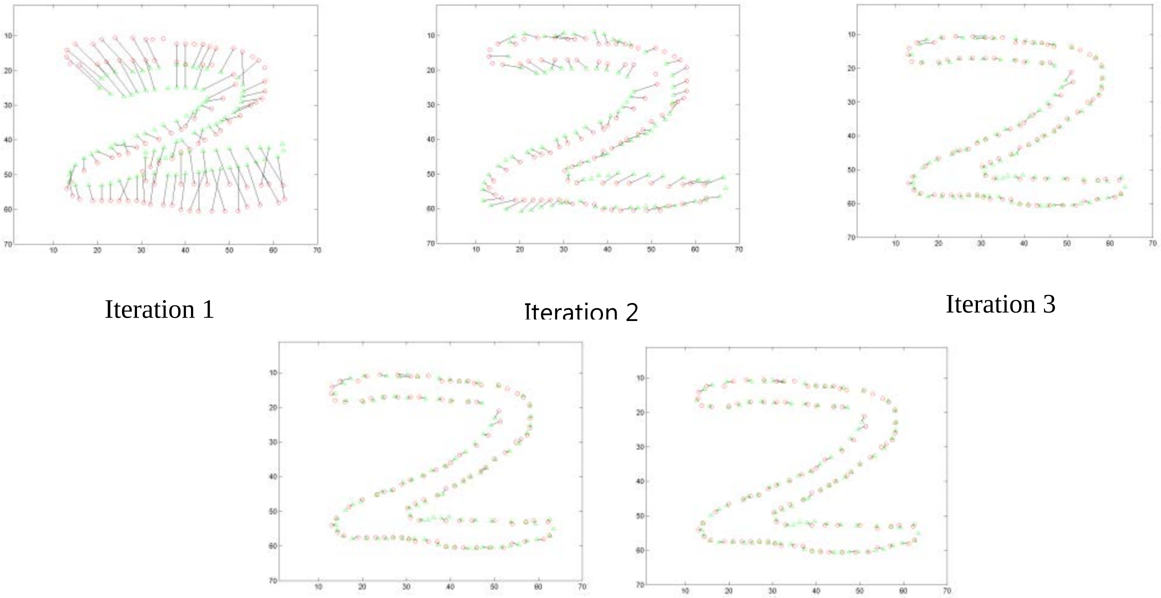
The regularized TPS energy that is minimized for estimating f is given by:

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

2.6 Shape Distance

The shape distance metric proposed by the authors is a weighted sum of three terms:

1. Shape context distance
2. Image appearance distance
3. Bending energy


Iteration 1

Iteration 2

Iteration 3

Iteration 4

Iteration 5

Figure 4: The shape matching algorithm showing the correspondences between points on the two shapes through 5 iterations of the Hungarian algorithm and TPS transformation re-estimation

The shape context distance is the chi-squared distance between the target image and the input image after the best alignment transformation is performed.

The image appearance distance (D_{ac}) is used since the shape context does not at all account for the grey-scale and texture information around the sampled boundary points. D_{ac}(P,Q) is sum of the squared differences of the pixel intensity values in Gaussian windows around the points given by:

$$\mathcal{D}_{ac}(\mathcal{P},\mathcal{Q}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{\Delta\in Z^2} G(\Delta)[I_{\mathcal{P}}(p_i+\Delta) - I_{\mathcal{Q}}\big(T\big(q_{\pi(i)}\big)+\Delta\big)]^2$$

The bending energy used is that after 3 iterations of shape context matching and TPS transformation re-evaluation.

For the images of digits the weights used are 1.6, 1 and 0.3 respectively for the three terms mentioned above.

3 Results

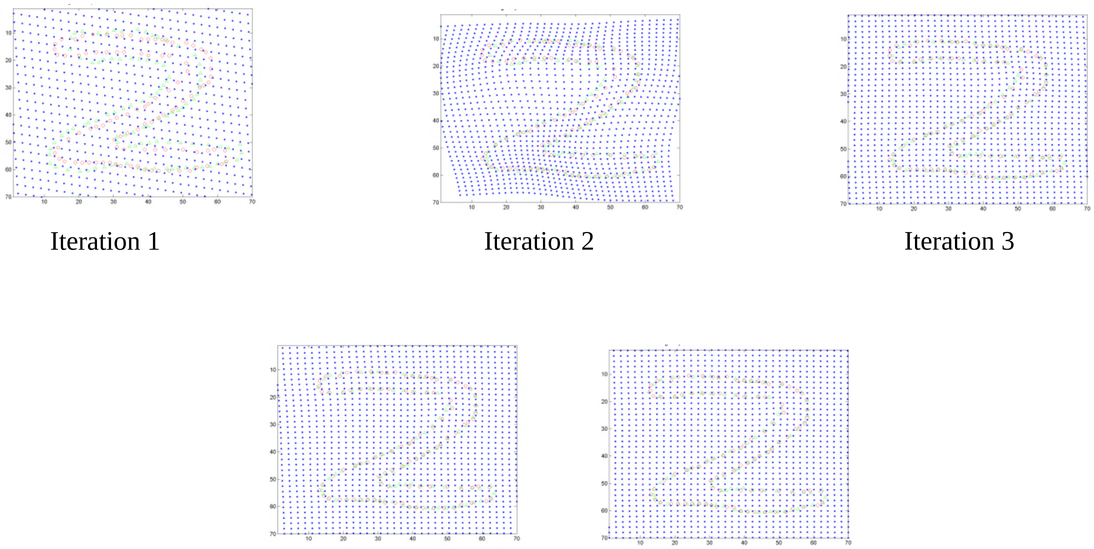
We have established some results against the MNIST data set of handwritten digits. The MNIST dataset consists of 60,000 training and 10,000 test digits.

In our experiment, we obtain the edges of the digits by using a contour detector, and sample 100 points from these edges to represent each digit. The cost computation for bipartite graph matching also accounts for dissimilarity of local tangent angles. We defined the cost as:

$$C_{ij} = (1-\beta)C_{ij}^{sc} + \beta C_{ij}^{tan}$$

Where C_{ij}^{sc} is the shape context cost and C_{ij}^{tan}, computed as $C_{ij}^{tan} = 0.5(1-\cos\big(\theta_i-\theta_j\big))$ gives the tangent angle dissimilarity.

Figure 3 shows a diagram of the log-polar histogram bins used for computing shape contexts. We have used 5 bins for log r and 12 bins for θ. Its


Iteration 1

Iteration 2

Iteration 3

Iteration 4

Iteration 5

Figure 5: Different iterations of image warping

noteworthy that the shape contexts for the corresponding points on the input and target images are quite similar, whereas the shape contexts for non-corresponding points are dissimilar.

The correspondences are calculated iteratively and the following correspondences are obtained. For one of the many test samples, the correspondences are obtained as shown in *Figure 4*.

The warping of the digits is as shown in *Figure 5*. Progressively, the images are warped in different iterations. This makes use of the thin plate spline functions for transforming coordinates.

Table 1 shows values of shape context cost and I_f for different iterations. We observe that the costs are decreasing as we converge to the optimal correspondences and alignment transform, on each iteration of the algorithm.

Iter #	**I_f**	**S.C cost**
1	1000	0.15979
2	0.061765	0.10792
3	0.013989	0.054519
4	0.0066203	0.043877
5	0.0044934	0.039482
6	0.0025944	0.036152

Table 1: I_f and SC cost values for different iterations.

4 Future Work

Given the shape distance and the optimal alignment transform, this experiment can be extended to recognize digits, 3D object in images, silhouettes, etc. We can train a k-NN classifier that uses the shape distance metric for digit recognition. In order to recognize 3D images, several images capturing an object from different views are obtained. The training data consists of equally spaced views of the object and the test set consists of remaining views. Silhouette matching uses the same approach as digit recognition. The performance of silhouette matching can be measured using the 'bull's eye test', in which each image is used as a query and the number of correct images in the top 40 matches are counted.

As proposed in [2], the TPS estimation can be approximated by one of three methods. These approximations allow significant improvements in computation speed and memory usage. This could be incorporated in our implementation to allow for a greater number of sample points with minimal impact on running times.

5 Conclusion

We have implemented the algorithm proposed in [1] for matching similar shapes using the shape contexts on a set of sampled boundary points on each of the shapes. Using the shape contexts, we get the histograms for each of the points and we use the Hungarian algorithm to find the optimal assignment permutation for these shape contexts. Thus, we get the correspondences for the shapes and then compute the alignment transform. We further compute the shape distances to quantize the measure of dissimilarity between the shapes.

6 References:

[1] S. Belongie, J. Malik and J. Puzicha, "Shape Context: A New Descriptor for Shape Matching and Object Recognition," *Advances in Neural Information Processing Systems 13: Proc. 2000 Conf.,*T.K. Leen, T.G. Dietterich, and V. Tresp, eds.. pp. 831-837, 2001.

[2] Gianluca Donato and Serge Belongie, “Approximate Thin Plate Spline Mappings”, Computer Vision – ECCV 2002

[3] Dengsheng Zhang and Guojun Lu, “Review of shape representation and description techniques”, Pattern Recognition 37 (2004) 1-19