Shape Context Matching For Efficient OCR

Sudeep Pillai

May 18, 2012
# Contents

1 Introduction ................................................. 2  
  1.1 Motivation .............................................. 2  
  1.2 Background ............................................. 2  
    1.2.1 MNIST data set .................................... 2  
    1.2.2 OCR recognition schemes ............................ 2  

2 Recognition via Shape Contexts .................. 4  
  2.1 Overview ................................................. 4  
  2.2 Shape Context ........................................... 4  
    2.2.1 Shape context representation ....................... 4  
    2.2.2 Shape context descriptor ............................ 4  
    2.2.3 Histogram matching .................................. 4  
    2.2.4 Graph matching ..................................... 5  
    2.2.5 Shape context score ................................ 5  
    2.2.6 Normalization for invariance ....................... 6  
    2.2.7 Outlier management ................................ 6  
    2.2.8 Performance issues ................................. 6  

3 Related and Prior Work in Efficient Matching .. 7  
  3.1 Related Work ............................................. 7  
  3.2 Fast Contour Matching using Approximate Earth Mover’s Distance .... 7  
  3.3 Fast Pruning using Generalized Shape Contexts .................. 7  
    3.3.1 Generalized shape contexts ......................... 7  
    3.3.2 Fast pruning ....................................... 8  

4 Efficient Shape Context Matching ............ 9  
  4.1 Fast matching ............................................ 9  
    4.1.1 Approximate correspondences in high dimensions ....... 9  
    4.1.2 Vocabulary-guided pyramid match .................... 10  

5 Contributions ............................................. 12  
  5.1 Overview ............................................... 12  
  5.2 Investigation in Dimensionality Reduction ............. 12  
    5.2.1 Principal Component Analysis (PCA) .................. 12  
    5.2.2 k-means clustering .................................. 13  
    5.2.3 Shapemes ........................................... 13  
  5.3 Fast Object Character Recognition .................. 14  
    5.3.1 Histogram of frequencies of cluster IDs as bag-of-words representation .... 14  
    5.3.2 Vocabulary-guided matching of frequency histogram of cluster ids .... 15  
    5.3.3 Source Code ....................................... 16
Chapter 1

Introduction

There has always been a strong push in the research community for improving the OCR accuracy on datasets such as the MNIST database. However, there does not seem to be too much interest in dealing with the performance scalability of such high high-accuracy systems. In this report, I investigate possible approaches to improving performance for large databases by bringing together two key concepts, namely shape contexts and pyramid matching kernels, to enable fast retrieval/recognition/classification of objects.

Shape contexts are generic shape descriptors that are invariant to scale and translation. Each point on a shape can be described by a high-dimensional vector that encodes the distribution of every other point in the shape relative to it. This descriptor is a natural input to the pyramid matching algorithm that efficiently computes a similarity score between such descriptors. The end result is an efficient correspondence solver for real-time recognition.

1.1 Motivation

With databases growing consistently in size, there is much need for recognition systems to be able to deal with querying such large-scale databases efficiently. This problem underlies several computer vision tasks, including object character recognition, generic object recognition and image retrieval. Content-based image retrieval has become a common technique where an image is queried to identify the most relevant images/objects/classes within a database, under some pre-defined notion of similarity.

1.2 Background

1.2.1 MNIST data set

The MNIST database of handwritten digits, available from the following url http://yann.lecun.com/exdb/mnist/, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

1.2.2 OCR recognition schemes

A variety of trainable classifiers and recognition schemes have been used to train and test on the MNIST database. Figure 1.2 presents some of the most recent test results using
different classification methods. One important fact to note is that these numbers correlate to the accuracy of these classifiers in a constrained setting (images in the database have been normalized, and centered). I focus on a subset of these, specifically the nearest neighbor classifier and investigate further the performance of using such techniques for efficient matching. [deslant] indicates that the classifier was trained and tested on the deslanted version of the database. [dist] indicates that the training set was augmented with artificially distorted examples. [16x16] indicates that the system used the 16x16 pixel images. The uncertainty in the quoted error rates is about 0.1%.

Figure 1.2: Error rate on the test set (%) for various classification methods.
Chapter 2
Recognition via Shape Contexts

2.1 Overview
Belongie et. al [1] use the concept of shape contexts to perform recognition of handwritten digits from the MNIST data set. With their novel shape context descriptors and $k$-NN matching, they achieved an error rate of 0.63%. Inspired by their work, I investigate possible performance improvements to their existing framework, and suggest implementation details for scalability using these shape contexts.

2.2 Shape Context

2.2.1 Shape context representation
Shape context is a form of shape descriptor that encodes the relative position and orientation of points with respect to each other in a set. Mathematically, this can be represented as follows. For any shape, first consider a discrete set of points that represent that specific shape. For this purpose, the shape contour is uniformly sampled to retrieve a set of points that effectively describes that shape. Now consider the the set of vectors originating from these points to every other sample point on the shape. These vectors express the configuration of the entire shape relative to a reference point.

2.2.2 Shape context descriptor
It is obvious that as we sample more points from the contour of the shape, the set of vectors that represent the shape would become richer but redundant. To avoid this redundancy, a distribution over the relative positions of each point with respect to the others becomes a sufficient descriptor that is highly discriminative. For a point $p_i$ on the shape, a coarse histogram $h_i$ of the relative coordinates of the remaining $n-1$ points is computed.

$$h_i(k) = \#\{p_j \neq i, x_j-x_i \in bin(k)\}$$

The figure below shows the shape context histogram for a point on the contour of a shape. The shape context histogram of a point $p_i$ thus represents the distribution of relative coordinates that is discretized over a log-polar space.

2.2.3 Histogram matching
Now that a shape can be described by a set of shape context histograms $h_i$, we need to be able to match these histograms to associate correspondings between points from different

---

1Sampling considerations
shapes. One such metric to match histograms is to use the $\chi^2$ test statistic:

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

where $h_i(k)$ and $h_j(k)$ denote the K-bin normalized histogram at $p_i$ and $q_j$, respectively. While the cost $C_{ij}$ can include additional terms based on the local appearance similarity at the points $p_i$ and $q_j$.

### 2.2.4 Graph matching

Since $C_{ij}$ represents the cost of matching the point $p_i$ on the first shape and $q_j$ on the second shape, an optimal point pair matching would involve the minimization of all such pairings as given by the following:

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

This however as expected is constrained to a one-to-one matching, i.e. $\pi$ is a permutation. This being a weighted bipartite matching problem, the Hungarian method [8] can be used to solve it in $O(N^3)$ time.

### 2.2.5 Shape context score

When computing the cost of matching shape descriptors $C_{ij}$’s for the bipartite matching, they include representing the dissimilarity of local tangent angles as well. They defined the matching cost as $C_{ij} = (1 - \beta)C_{ij}^{sc} + \beta C_{ij}^{tan}$, where $C_{ij}^{sc}$ is the shape context cost, $C_{ij}^{tan} = 0.5(1 - \cos(\theta_i - \theta_j))$ measures the tangent angle dissimilarity, with $\beta = 0.1$. For recognition, they use a $k$-NN classifier with a distance function of

$$D = aD_{ac} + D_{sc} + bD_{be}$$
with \( a = 1.6 \) and \( b = 0.3 \). Here, \( D_{ac} \) pertains to the appearance cost, a similarity measure of the appearance/grayscale value at the point location. \( D_{sc} \) pertains to the shape context as explained above, using the \( \chi^2 \) test to compare histograms. The authors use a Thin-Plate-Spline fitting algorithm to estimate a mapping from one shape to another. Thus \( D_{be} \) measures how much transformation is necessary to bring the two shapes into alignment. In the case of Thin-Plate-Spline deformation model, it is referred to as the bending energy. Their error rate using 20,000 training examples (from the MNIST dataset) and 3-NN is 0.63 percent.

### 2.2.6 Normalization for invariance

In order to account for the invariance in scale and translation, the shape context vectors are normalized by the mean distance of the \( n^2 \) point pairs in the shape. No special normalization is required for invariance in translation, as all the vectors are represented with respect to each other.

### 2.2.7 Outlier management

In order to robustly handle outliers, several dummy assignments have to be made with a constant matching cost. This is to prevent low cost false matching between points, effectively acting as a minimum threshold for the matching scheme.

### 2.2.8 Performance issues

The primary issue with using the same implementation as described in the paper [1] is that they employ the hungarian algorithm for correspondence matching which has a \( O(n^3) \) time complexity. Additionally, the paper suggests artificially adding dummy features with some fixed cost in order to ensure that the algorithm does not report false positives. This seems inefficient as we add more points to the matching algorithm that has cubic time complexity to the number of features added. Furthermore, the iterative approach of fitting a thin-plate-spline model with the matched features takes considerable amount of time to converge to the desired shape for matching purposes. All these factors motivate for an improved solution that is scalable with the size of the feature database.
Chapter 3

Related and Prior Work in Efficient Matching

3.1 Related Work

A number of shape matching techniques have been proposed for improving the performance of correspondence matching including using chamfer matching, Earth Mover’s Distance, embeddings in (EMD-$L_1$), shape topologies with points and tangent lines etc. Most of these techniques try to approximate matching to enable similarity scoring and recognition at a much faster rate. Here I investigate a few relevant techniques that I shall use to build intuition for later sections.

3.2 Fast Contour Matching using Approximate Earth Mover’s Distance

Grauman et. al [3] address the complexity issue of the correspondence-based shape matching algorithm, and propose a contour matching algorithm that incorporates approximation techniques to enable fast shape-based similarity retrieval from large databases. The authors propose a novel contour matching algorithm that quickly computes the minimum weight matching between sets of descriptive local features using a low-distortion embedding of the Earth Mover’s Distance in to a normed space. These contours are then matched from a large database in sublinear time via an approximate nearest neighbor search with Locality-Sensitive Hashing (LSH). The method achieves a speedup of four orders of magnitude over the exact matching method at the cost of 4% reduction in accuracy.

3.3 Fast Pruning using Generalized Shape Contexts

In a subsequent paper by Mori et. al [7], the authors realize the prohibitive nature of using the deformable matching algorithm to a large database of models, and propose a faster recognition scheme via pruning.

3.3.1 Generalized shape contexts

The same structure of shape contexts are used here, with the additional consideration of tangent vectors along the direction of the edge at each point. The algorithm sums the tangent vectors from each of the points within a bin, and considers this as the d-dimensional representation of the point. Thus the descriptor for a point $p_i$ is the histogram
where $Q = \{ p_j \neq i, x_j - x_i \in bin(k) \}$. When comparing the descriptors for two points, the d-bin histogram is converted to a two-dimensional vector $v_i$, and normalized before comparing via the $L^2$ norm:

$$
\hat{v}_i = < \hat{h}_i^{1,x}, \hat{h}_i^{1,y}, \hat{h}_i^{2,x}, \hat{h}_i^{2,y}, \ldots, \hat{h}_i^{d,x}, \hat{h}_i^{d,y} >
$$

$$
d_{GSC}(\hat{h}_i, \hat{h}_j) = \| \hat{v}_i - \hat{v}_j \|_2,
$$

where $\hat{h}_i^{j,x}$ and $\hat{h}_i^{j,y}$ are the x and y components of $\hat{h}_i^j$, respectively.

### 3.3.2 Fast pruning

One of the techniques used in the paper [7] to speed up matching is vector quantization on the shape contexts. With $|S|$ known shapes, and shape contexts computed at $s$ sample points, the full description of a shape consists of $|S| \cdot s \cdot d$-dimensional vectors. This high dimensional vector is quantized, or in other words reduced to clusters called *Shapemes*. To obtain shapemes for a specific object, all the shape contexts of the samples from the object are used to determine a $k$-means clustering, thereby obtaining $k$ shapemes. The shapemes represent a specific view for an object, where each $d$-bin shape context is quantized to its nearest shapeme and replaced by the shapeme label, namely the cluster set $1, \ldots, k$. A known view is then generated by building a distribution over all the shapemes frequencies.
Chapter 4

Efficient Shape Context Matching

4.1 Fast matching

4.1.1 Approximate correspondences in high dimensions

While a robust representation exists for shapes, it is still unclear how to query in an efficient fashion from a large database for matching purposes. The problem gets significantly harder when the object/shape that is being matched only has a partial match with one of the existing shape descriptors on the database. Several attempts have been made to develop approximation algorithms to compute close solutions for a fraction of the computational cost [2],[6]. However, most of these algorithms suffer from distortion factors that increase linearly with the dimensions of features, and fail to take advantage of the structure in the feature space.

Figure 4.1: The pyramid match measures the partial matching similarity between sets of vectors. The approximation relies on a multi-resolution partitioning of the local descriptor feature space to efficiently form correspondences between points in two sets via weighted intersections.
4.1.2 Vocabulary-guided pyramid match

Grauman et. al[4] introduce a novel pyramid embedding based on a hierarchy of non-uniformly shaped bins that takes advantage of the underlying structure of the feature space while retaining most of the robustness in its accuracy for sets with high dimensionality. The main idea is to partition the given feature space into a pyramid of non-uniformly shaped regions based on their distribution of provided database of feature vectors.

Figure 4.2: The bins are concentrated on decomposing the space where features cluster, particularly for high-dimensional features (in this figure \(d=2\)). Features are small points in red, bin centers are larger black points, and blue lines denote bin boundaries. The vocabulary-guided bins are irregularly shaped Voronoi cells.

In our case, this would be the set of histogram vectors that represent the shape context. Feature sets are then encoded as multi-resolution histograms determined by that pyramid, and an efficient intersection-based computation between any two histograms yields an approximate matching score for the original sets.

The intuition is to start collecting groups of matched points from the bottom of the pyramid up, i.e., from within increasingly larger partitions. First we consider matching the closest points that fall together within small bins, and as we proceed to the coarser resolutions in the pyramid we allow increasingly further points to be matched. The key is that the matching cost can be measured via these histogram pyramids without even computing pairwise distances between the vectors in two point sets. The matching approximation may be used directly as a distance or similarity measure. For very large databases where it is infeasible to index with a naive linear scan of all items, it is even possible to perform sub-linear time (\(O(dmL)\), for sets with \(O(m) \mathbb{R}^d\) features and pyramids with \(L\) levels ) retrievals with local feature representations and randomized hashing techniques.

Let \(n_{ij}(X)\) denote the element \(n\) from \(<p,n,d>_j\), the \(j^{th}\) bin entry of histogram \(H_i(X)\), and let \(c_h(n_{ij}(X))\) denote the element \(n\) for the \(h^{th}\) child bin of that entry, \(1 \leq h \leq k\). Similarly, let \(d_{ij}(X)\) refer to the element \(d\) from the same triple. Given point sets \(X\) and \(Y\), we compute the matching score via their pyramids (X) and (Y) as follows:

\[
C(\Psi(X), \Psi(Y)) = \sum_{i=0}^{L-1} \sum_{j=1}^{k^i} w_{ij} \left[ \min(n_{ij}(X), n_{ij}(Y)) \right] - \sum_{h=1}^{k} \min(c_h(n_{ij}(X), c_h(n_{ij}(Y))))
\]

The outer sum loops over the levels in the pyramids; the second sum loops over the bins at a given level, and the innermost sum loops over the children of a given bin. The first min term reflects the number of matchable points in the current bin, and the second min term tallies the number of matches already counted at finer resolutions (in child bins). Note that as the leaf nodes have no children, when \(i = L\) the last sum is zero. All matches are new at the leaves. The matching scores are normalized according to the size
of the input sets in order to not bias towards (for similarity) or against (for cost) larger sets.
Chapter 5

Contributions

5.1 Overview

Using shape contexts as a fundamental descriptor for shape matching seems reasonable for the application of OCR on the MNIST database given its invariance to both translation and scale. Through this work, I bring together some of the key ideas and works from [1], [7], and [4] to suggest an improvement over the existing shape matching framework for large scale databases. There are plenty of extensions to the framework whereby learning, matching, and recognition can be bootstrapped together to build persistent systems that can learn from environments over extended periods of time.

With such persistent systems in mind, it is critical to be motivated to analyse major bottlenecks in the recognition phase, especially since we can afford to learn offline. With this report, I propose a performance motivated solution to real-time recognition with little accuracy trade-offs. Furthermore, I suggest multiple performance improvement techniques to querying (some previously suggested from papers) and making an intuitive argument towards why dimensionality reduction (i.e. in other words vector/feature quantization) can be useful for solving such large-scale image database problems.

In this report, I consider two approaches during the implementation of the fast recognition of shapes. While matching with full shape context features is possible with the approximate matching scheme, a low-dimensional feature descriptor is more desirable for tractability and to avoid approximation collisions in hash-space. I investigate the use of PCA and \( k \)-means clustering as a pre-processing step to efficient database matching, and provide some of my experimental results (extracted directly from the compiled source code).

5.2 Investigation in Dimensionality Reduction

5.2.1 Principal Component Analysis (PCA)

Given a large sample of shape context histograms in \( \mathbb{R}^{60} \), PCA is employed to project all the points from \( \mathbb{R}^{60} \) onto a smaller subspace. To allow for tractable nearest neighbor searches between these smaller dimensional vectors, the shape context histograms are projected onto \( \mathbb{R}^{3} \). I found that a very low-dimensional subspace was able to capture much of the local contour variation in these data sets. Figure 5.2 measures its expressiveness as a function of feature dimension, as compared to a higher-dimensional raw point histogram. The representation of a novel contour is determined by projecting its shape context histograms onto the low-dimensional subspace. This representation is translation
invariant, and the scale of the shape context histograms initially extracted from the data is determined from the mean inter-point distance per shape.

There are several tradeoffs that must be considered when selecting \( d \), the number of subspace dimensions to use for the shape context subspace features. The larger \( d \) is, the more exactly the projection coordinates will capture the original histogram feature extracted from the image (i.e., the smaller the PCA reconstruction error will be). However, larger \( d \) values will increase both the distortion that is induced by the embedding, as well as the complexity of computing the embedding. That said, \( k \)-NN (\( k=3 \) in this case) can now be performed on this reduced dimensional set to evaluate a similarity.

![Visualization of feature subspace constructed from shape context histograms for two different data sets. The RGB channels of each point on the contours are colored according to its histograms 3-D PCA coefficient values. Set matching in this feature space means that contour points of similar color have a low matching cost, while highly contrasting colors incur a high matching cost.](image)

5.2.2 \( k \)-means clustering

To enable building of the vocabulary-guided pyramids, certain structure in the feature database needs to be learned. This can be done via \( k \)-means clustering with either euclidean distance or the shape context distance described earlier as the distance metric for clustering. The initial corpus of features is clustered into \( k \) top-level groups where group membership is determined by the Voronoi partitioning of the feature corpus according to the \( k \) cluster centers.

5.2.3 Shapemes

I employ a similar concept to shapemes\[7\] described earlier, where the shape contexts are initially clustered in \( d \)-dimensional space and later denoted by their membership cluster or shapeme. This ties very closely to the notion of vocabulary-guided pyramid bins, where membership is determined by the point location of the query shape context in the voronoi space whose centers are the \( k \)-mean clusters.

To have an intuitive idea of what the clustering extracts, Figure 5.3 shows the cluster ID/shapemes extracted for different digits from a database of 1000 training images per digit. As the paper suggests, I’ve employed a cluster size of \( k=100 \) for this experiment.
Figure 5.2: Visualization of feature subspace constructed from shape context histograms for the MNIST database. The RGB channels of each point on the contours are colored according to its histograms 3-D PCA coefficient values. Set matching in this feature space means that contour points of similar color have a low matching cost, while highly contrasting colors incur a high matching cost.

5.3 Fast Object Character Recognition

5.3.1 Histogram of frequencies of cluster IDs as bag-of-words representation

Bag of words representation is a popular object categorization technique, whereby the object is represented as a bag of words, or in this case, cluster IDs. This representation ignores any notion of order/location of clusters, but purely relies on the existence of the cluster within a specific bag that represent the object. To understand the distribution of shapemes (unique clusters) in a specific object/digit, figure 4.5 shows the corresponding histogram frequencies of shapemes for 1000 training images, or 60,000 shape contexts.

Employing shapemes to represent characters improved performance numbers by a factor of 5 for a database of 10,000 images. The performance measure is an improvement over the standard $k$-NN search over the $d$-dimensional shape context feature with the same dataset size. It is however also important to note that with the naive $k$-NN search, $k$ needs to be a reasonable number to account for false positives. This further reduces performance for the naive case over the shapeme matching procedure where the closest cluster id is assigned to the query feature vector. Since the object is sampled densely, we expect for histogram frequencies to tackle the outlier removal.
5.3.2 Vocabulary-guided matching of frequency histogram of cluster ids

This brings us to the concept of vocabulary-guided bins and its relevance to the above mentioned histogram of frequencies of cluster IDs. I see the use of the vocabulary-guided matching as a unifying notion of shapemes obtained via $k$-means clustering and matching in multiple levels of the pyramid that corresponds to the specific cluster the query point belongs/is closest to. Unsurprisingly, as the paper suggests [4] $k$-means clustering is required as a preprocessing step before the pyramid matching scheme is performed. Naturally, we expect to see a performance improvement over the traditional $k$-NN matching of shapemes for large databases. Furthermore, the pyramid match kernel provides an approximate match score for the query provided to it, which can be used as a fast approximation before a more elaborate matching is performed for re-ranking. This re-ranking scheme proves to be very useful in order to eliminate false positives in the pyramid matching approximations.
Figure 5.4: Histograms depicting the distribution of shapeme (cluster IDs) frequencies for all shape contexts extracted from each of the contours in a dataset of 1000 images per digit. Figures (a), (b), (c), (d), (e), (f), (g), (h), and (i) correspond to the histograms of frequencies of shapemes for digits 0, 1, 2, 3, 3, 4, 5, 6, 7, 8, and 9 respectively.

5.3.3 Source Code

The following mercurial repository contains the most up-to-date version of my progress in implementing shape contexts and its integration with libpmk\(^1\), a C++ implementation of the pyramid matching algorithm as stated in [5]:

```
hg clone ssh://hg@bitbucket.org/sudeeppillai/fs_software_public\(^2\)
```

\(^1\)[http://people.csail.mit.edu/jjl/libpmk/]
\(^2\)[Requires OpenCV 2.x - http://code.opencv.org]
I have also implemented a few dimensionality reduction techniques that I employ as a preprocessing step before the pyramid match algorithm is run. Specifically, I reduce a large dataset of shape contexts (60D feature vector per point) sampled from 10,000 images (300,000 $\mathbb{R}^{60}$ feature vectors in total taking 30 samples per object) into $\mathbb{R}^3$ and plot their principal components (Figure 5.2) for efficient match similarity. Most of these implementations are derived and/or inspired works from [1].

Both the implementations of the building the shape context descriptor and evaluating their shape context distances have been completely written in C++ using the above mentioned libraries. The code also utilizes libpmk.

I also investigated the idea of shapemes [7] and its application to fast retrieval of objects from large databases. Essentially, this is also a form of dimensionality reduction technique, whereby each of the 300,000 feature vectors from the database are clustered ($k=100$) and identified to have a membership cluster. Figure 5.3, also generated through the code I’ve written, shows the IDs of the clusters to which the sample points belong.
Bibliography


