Recognition of Graffiti Tags

Paul Schwarz

This report is submitted as partial fulfilment of the requirements for the Honours Programme of the School of Computer Science and Software Engineering, The University of Western Australia, 2006

Abstract

Graffiti tagging, the inscription of an individual graffiti writer's unique mark in public space, is considered to be an antisocial behaviour and is a criminal offense. Due to the costs involved in the removal of graffiti tags it would be desirable to deter tagging. A searchable database of graffiti images could be an integral part of the policing system aimed at deterring graffiti vandalism. In order to index the database, an algorithm must be used that is capable of describing and comparing images. There are various approaches to object recognition and shape matching. The two methods that have been examined in this thesis are the Scale-Invariant Feature Transform (SIFT) and the Shape Context. These techniques were chosen based on their suitability to deal with the types of inconsistencies that are common in graffiti tag recognition, compared with the variances that occur in generic object recognition. Shape Context was found to be better suited as it was more robust than SIFT with regard to small shape distortions that are natural in graffiti writing. An empirical measure for the correctness of a searchable graffiti database was devised. Using this measure, it was found that retrieval results using Shape Context were considerably more accurate than retrieval results using SIFT. This thesis introduces the problem of recognising graffiti tags into the areas of object recognition and shape matching. Results are presented based on a collection of photographed graffiti tags.

Keywords: graffiti tag recognition, shape matching, pattern matching, object recognition, shape context, SIFT, feature
CR Categories: I.5.5 [Pattern Recognition]: Implementation,
I.4.7 [Image Processing and Computer Vision]: Feature Measurement,
J.4 [Computer Applications]: Social and Behavioural Sciences

Acknowledgements

Thanks to my parents for their love, and for paying for me to be here.

Mark (my brother), Vuyo, Adam, and Mark G for their friendship and support during the year. And my other friends, of whom I have not seen much lately!

Dr. Luigi Barone for helping me with all the hassles of being an international student.

Dr. Ajmal Mian for the valuable discussions.

Thanks to my fellow honours students making the year memorable.

Thank you Peter for supplying the collection of photographs.

And most importantly, thank you for your excellent supervision during my honours year, Dr. David Glance and Dr. Peter Kovesi.

Contents

List of Figures

CHAPTER 1

Introduction

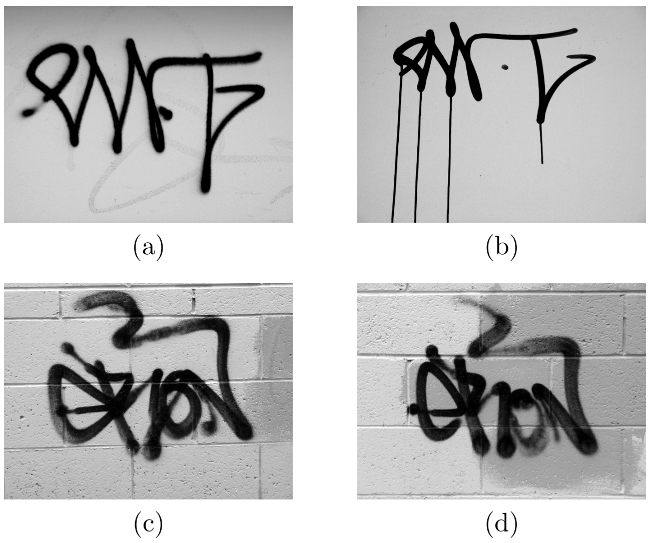


Figure 1.1: Internal and external variation. Tags (a) and (b) were created by a different graffiti writer to tags (c) and (d).

A tag is an individual graffiti writer's signature that either accompanies a work of graffiti art or is inscribed in isolation as a personal identifier. Investigations into the graffiti subculture have established that writers (as they are referred to) operate either alone or in gangs to mark territory [12]; the aim being to 'claim' public space and to command respect from rivals. Artists desire to be recognised via their tags and therefore they develop unique styles.

Developing a tagging style requires practice, in a similar way that an individual learns to write. Handwriting is a complex behaviour which is developed through repetition. The brain is trained to control the muscles which hold the writing implement. Once writing becomes routine the style tends to remain constant [17]. This project assumes that tagging styles differ significantly enough from person to person (*external variation*), and remain constant enough on an individual basis (*internal variation*), to be distinguishable by an algorithm. An example of external and internal variation is shown in Figure 1.1.

Graffiti clean-up operations involve photographing graffiti for police records and then removing the graffiti. The City of Subiaco, Perth, Western Australia, allocated $161 500 for safety and security, including graffiti management in their 2005 – 2006 budget [5].

In policing graffiti taggers, a useful resource would be a graffiti database that stores images of tags with details such as locations and dates. Police could use this information to find patterns in the behaviour of taggers. The system could play a role in forensic investigations as it may be used to extract relevant tags from a database as evidence for trials. The known presence of a fully implemented tag recognition system could deter vandals from tagging public spaces, thus saving money for city councils that employ clean-up operations.

The complete implementation would take an input image, search a database of tag images and output all similar tags along with their metadata. Therefore it could be a used to establish whether a tag is a one-off incident or one in a series of multiple offences; the latter carrying a significantly higher penalty [11].

A large photographic collection of graffiti tags would need to be stored in a database and indexed. The problem is that the indexing process could be very time consuming if conducted by a person visually inspecting each image. Therefore this thesis aims to find a suitable method for automating this process.

Many techniques exist for object recognition, shape matching, and signature and handwriting comparison. Although algorithms for recognition have not previously been used for comparing graffiti tags, these techniques are of interest to this project since they address similar problems to those of graffiti tag recognition.

Chapter 2 surveys three potential methods for matching tags:

1. Scale-invariant feature transform (SIFT) introduced by Lowe [13].

   This algorithm transforms an image into a large set of local feature vectors that describe key regions in the image called SIFT keys. These descriptors are invariant to scaling, rotation, and translation, and partially invariant to illumination changes and affine projection.

2. Shape Context (SC) introduced by Belongie, Malik, and Puzicha [2].

   Like SIFT, SC transforms an image into a set of local descriptors. It chooses a set of points sub-sampled from the contours on an object. For each of these points, it produces a *shape context* which describes the coarse arrangement of the rest of the shape with respect to the current point.

3. Matching handwritten words [10, 15, 16, 18, 19].

   These papers provide a series of approaches for word matching in handwritten documents. A scanned document is segmented into words and then features are generated to describe each word. The features are based on the profile of the word [18, 19].

An algorithm that allows a degree of leniency when taking photographs is desirable as it would eliminate the need for special photographic techniques and equipment for recording tags. Both SIFT and SC are algorithms for generic object recognition and produce robust descriptors. Unlike the approaches for matching handwritten words, the SIFT and SC methods can be used in the presence of perspective distortion. For this reason, SIFT and SC have been chosen for experiments in this thesis. Some papers that discuss the matching of handwritten words will be reviewed in Chapter 2, but will not be tested in this thesis.

In order to compare images using either SIFT or SC, a set of feature descriptors must first be extracted from image. The following steps outline the general approach:

1. Determine key locations in each image.
2. Generate a set of feature descriptors from the information extracted at each of the key locations.

Now that one set of features has been extracted for each image, the following steps deal with comparing the images:

3. For each feature in one set, find the most similar feature in the other set. The result of this step is a putative matching between the two images.
4. Model the transformation that describes the mapping of the locations of points from the one image to the other.
5. Mark points that obey this transformation as inliers. The rest of the points are disregarded.

The final correspondence score is based on the number of key point correspondences that were found to be inliers to the transformation in Step 4.

The above steps are outlined in Figure 1.2.

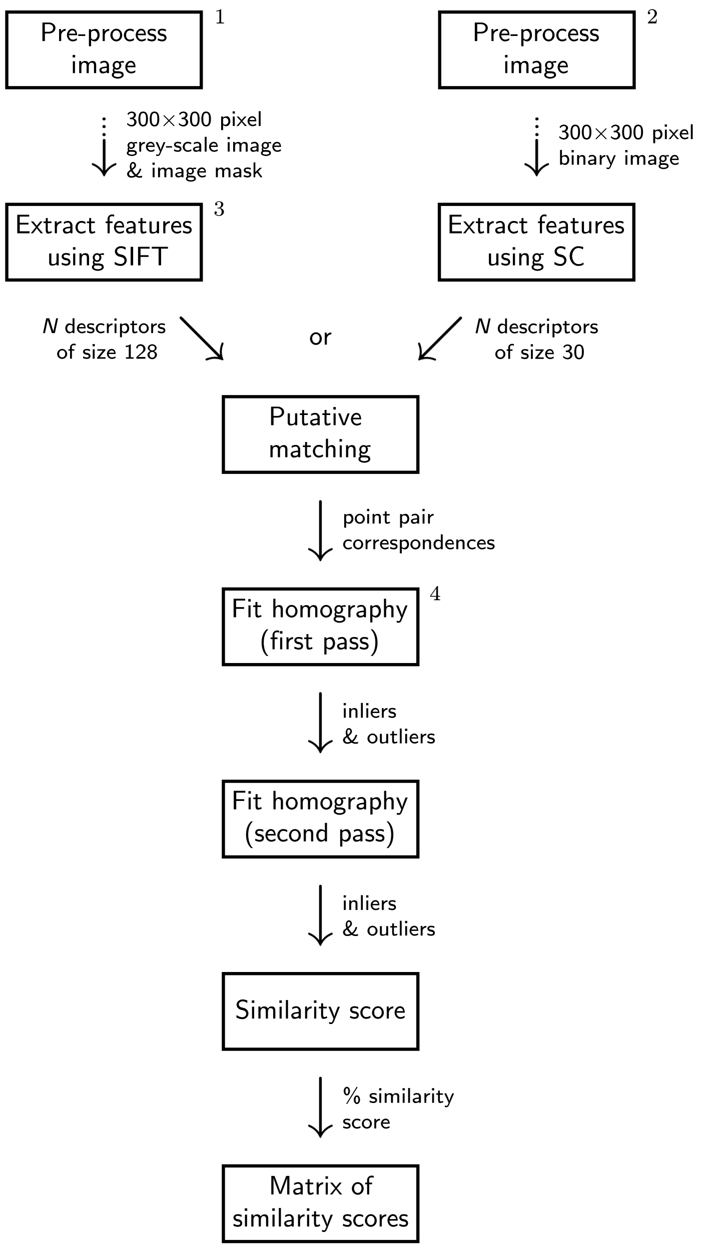

Pre-process image
1
Pre-process image
2
300×300 pixel grey-scale image & image mask
300×300 pixel binary image
Extract features using SIFT
3
Extract features using SC
N descriptors of size 128
or
N descriptors of size 30
Putative matching
point pair correspondences
Fit homography (first pass)
4
inliers & outliers
Fit homography (second pass)
inliers & outliers
Similarity score
% similarity score
Matrix of similarity scores

Figure 1.2: Flowchart for the algorithms used in this project. Extraction of descriptors is done with either SIFT or SC. The remaining steps in the algorithm are the same for both SIFT and SC. Code modules were implemented by the author, unless otherwise specified in the footnotes.

[1,2]Pre-processed manually using an external graphics application.
[3]MATLAB implementation by El-Maraghi [6].
[4]MATLAB implementation by Kovesi [9].

The flowchart in Figure 1.2 illustrates, in a modular fashion, the algorithm implemented for this project. The figure indicates which modules were written by the author for the project, and which modules consist of reused code from other authors. Chapters 3–5 discuss the implementation of these modules, and the choice of parameters used to optimise their performances. Since experiments in this project compare SIFT and SC, the implementation allows a choice of SIFT or SC to be used for feature extraction.

The aim of this thesis is to test the hypothesis that an algorithm exists that is capable of accurately grouping similar tags. Specifically, the suitability of SIFT and SC is examined, for the application of recognising graffiti tags.

This dissertation comprises the following components:

Chapter 2 Review of prior work on the subject, and research related to the techniques used in this thesis.

Chapter 3 Discussion on the choice of parameters for the SIFT algorithm.

Chapter 4 Discussion on the implementation of the SC algorithm, and the choice of parameters used.

Chapter 5 The method used to match descriptors, and an explanation of the use of RANSAC to model a transformation between sets of descriptors.

Chapter 6 The method for creating a database that facilitates quick retrieval, the scoring system to establish the correctness of the matching, experiments that find the optimal parameters for transformation fitting, and finally a comparison of the effectiveness of SIFT vs. the effectiveness of SC.

Chapter 7 Conclusion.

CHAPTER 2

Related Research

This section reviews literature that is relevant to the proposed methods for recognising graffiti tags. The discussion covers feature extraction using SIFT and SC, correspondence matching, and homographic transformation fitting using RANSAC.

Word image matching, discussed in Section 2.4, is a technique that is reviewed but not implemented for this thesis. However, it is included because the results that the technique achieved (see Sections 2.4.1–3) indicate that it should be considered for future work in this area.

2.1 Extracting feature descriptors

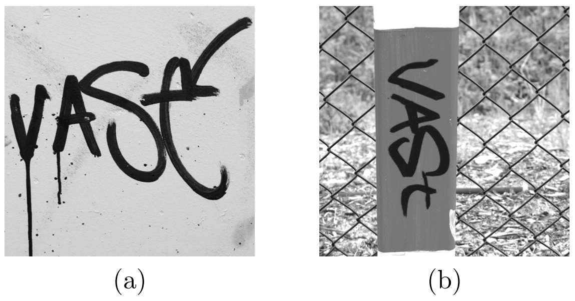


Figure 2.1: By visual inspection, tags (a) and (b) appear to belong to the same writer.

Traditionally, computer vision is concerned with recognising objects in a manner that is invariant to scale, pose, illumination, and affine distortion. Considering Figure 2.1, it becomes apparent that an algorithm that matches graffiti tags should strive to be invariant to shape distortion, perspective projection, and thickness of writing implement as well.

2.1.1 Locating key points using SIFT

The SIFT algorithm [13, 14] is capable of efficiently identifying stable key locations in the scale-space of a grey-scale image. It uses the following steps to extract a set of descriptors from an image:

1. Scale-space extrema detection
2. Key point localisation
3. Orientation assignment
4. Key point description

Scale-space extrema detection involves searching over all scales of the image to detect key points of all sizes. This is done using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation.

A difference-of-Gaussian pyramid is used to find local extrema at various scales. The pyramid is constructed using the method from Burt and Adelson [4]. For each octave of scale-space, the initial image is repeatedly convolved with Gaussian functions of $\sigma = \sqrt{2}$ to produce a set of scale-space images. Adjacent Gaussian images are subtracted to produce difference-of-Gaussian images. After each octave, the Gaussian image is halved in size, and the process repeated (see Figure 2.2).

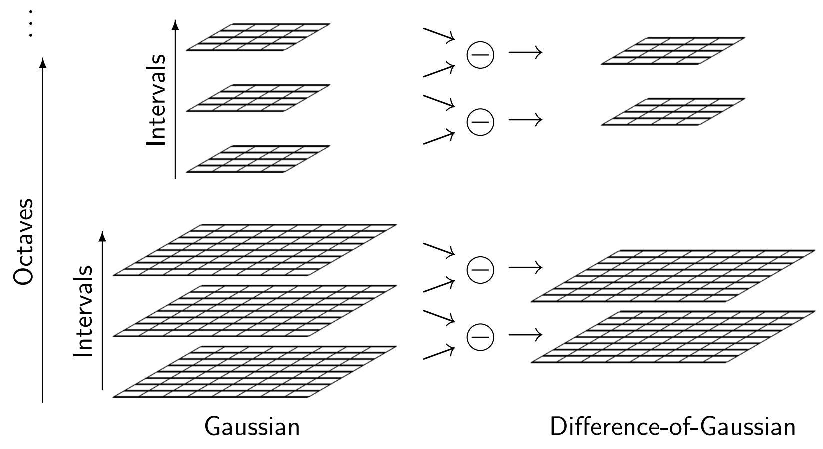


Figure 2.2: Difference-of-Gaussian image pyramid. The Gaussian pyramid is the result of repeated blurring between intervals and down-sampling between octaves. The difference-of-Gaussian pyramid is the subtraction of adjacent Gaussian layers.

The scale-space of an image is defined as the function $L(x, y, \sigma)$. It is the convolution of a variable-scale Gaussian, $G(x, y, \sigma)$, with an input image, $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y),$$

where $*$ represents a convolution in the x and y directions, and

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}.$$

The difference of two adjacent scales, $D(x, y, \sigma)$, separated by factor k, is given by:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned}$$

Maxima and minima in the pyramid of $D(x, y, \sigma)$ images are detected by comparing each sample point to its eight surrounding pixels in the current image and nine surrounding pixels in the scale above and scale below. The sample point is considered to be a local extrema if it is either a maxima or minima amongst these neighbours.

At this point, a collection of key points and corresponding scales have been accurately located. The explicit scale of each key point allows its descriptor to be sampled at an equivalent scale in other images, thus achieving invariance to scaling.

2.1.2 Describing key points using SIFT

In Section 2.1.1, a set of key locations and their scales were established. The next step is to compute orientations for each key point in order to achieve rotational invariance [14].

For each key point, its scale is used to select the Gaussian blurred image L that has the corresponding scale so that further computations are scale-invariant. At this scale, gradient magnitudes and orientations are calculated by looking at pixel differences. An orientation histogram is built with 36 bins covering the 360 degree range. Each sample is weighted by its gradient magnitude and weighted by a Gaussian window, then added to the histogram. Peaks in the histogram correspond to dominant directions. It is possible that more than one descriptor is formed at each key point if there is more than one clear peak in the orientation histogram.

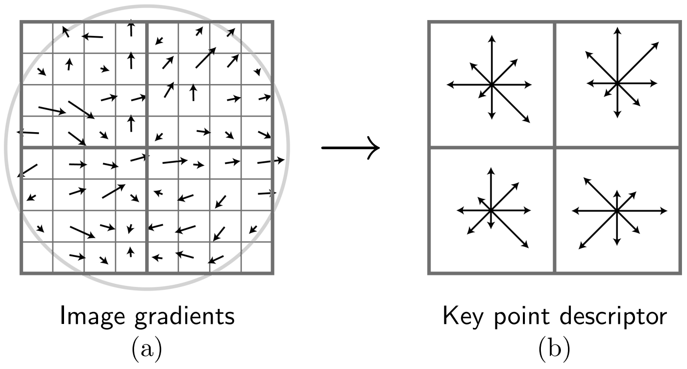


Figure 2.3: Extracting a SIFT descriptor (diagram from Lowe [14]). (a) shows the sample region around a key point. Arrows represent gradient magnitudes and orientations. The circle represents the Gaussian 'window'. (b) shows the resulting descriptor. Each set of eight arrows is the weighted accumulation of gradients from (a). This 8×8 sample produces a $2 \times 2 \times 8$ descriptor. The actual SIFT implementation uses a 16×16 sample and produces a $4 \times 4 \times 8$ descriptor (128 vector).

Figure 2.3 illustrates the computation of a SIFT descriptor. In order to keep the diagram simple, it shows a sample of only 8×8 pixels around a key location that produces a descriptor of size 2×2; the algorithm actually samples 16×16 pixels and produces a descriptor of size 4×4.

Figure 2.3a shows the gradient magnitudes and orientations for the sample space surrounding the key location. The orientations are adjusted relative to the orientations of the key points so that rotational invariance is preserved. The circle represents the Gaussian function that is applied to give more weight to the gradients closer to the key point.

The weighted samples from each of the 4×4 subregions are accumulated into histograms with eight orientation bins as shown in Figure 2.3b. The lengths of the arrows represent the sums of the gradient magnitudes in each of the directions within each subregion. The actual SIFT descriptor has 4×4 histograms, and each contains eight orientations bins, therefore it is a $4 \times 4 \times 8 = 128$ vector.

2.1.3 Locating key points using Shape Context

Shape Context (SC) [2, 3] is a rich descriptor that is simple to compute but effective at matching shapes. Invariance to translation is built in for free and it is not expensive to provide invariance to scaling. The descriptor is robust with regard to outliers, some shape distortion, and occlusion. It is possible to include invariance to orientation, although for this project it will be assumed that the photographer will supply tags that are oriented normally.

SC looks at the contours of an object to determine key locations. In the case of a graffiti tag image, an edge detection method must be used to extract the outline of the tag. The algorithm then sub-samples this outline at regular intervals which gives a set of N points that lie on the outline of the tag (see Figure 2.4).

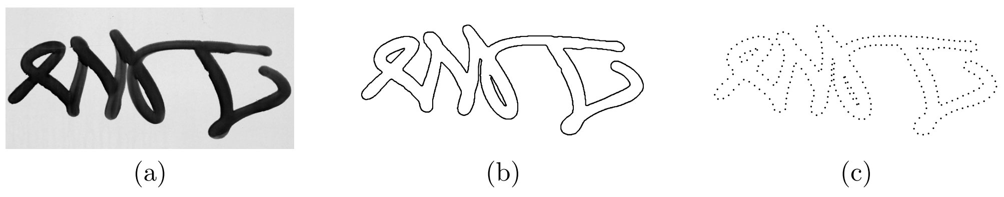


Figure 2.4: Locating key points using Shape Context. (a) is the original image. (b) is the outline of the graffiti tag after a Canny edge detection. (c) is the sub-sampled outline contour. A sub-sampling interval of 10 pixels was used, giving $N = 258$.

2.1.4 Describing key points using Shape Context

An SC feature descriptor expresses the configuration of the entire shape relative to each key location [2, 3]. The previous step of the SC algorithm (see Section 2.1.3) found N key locations. Each of these locations needs to be compared with the $N - 1$ other key locations to create an SC descriptor. This is done by binning sample points into a histogram; the histogram bins are broad enough to allow for small shape distortions and some variation in orientation.

A kernel is defined with n concentric circular contours representing log radial distance ($\log r$) bins and m equally spaced sector contours representing angle (θ) bins. Figure 2.5a shows an instance of the kernel with $n = 5$ and $m = 6$. The choice of log radial distance bins makes the descriptor more sensitive to nearby sample points than those further away [3].

The median distance λ between all N^2 point pairs is used to set the scale of the kernel [1, 2]. This step achieves scale invariance. The median was chosen to provide robustness in the presence of outliers.

The kernel is placed over a key point (see Figure 2.5b) and then a feature histogram for that key point (see Figure 2.5c) is populated by counting the number of other sample points that lie in each of the areas of the kernel. This is repeated for each of the N key points, therefore giving a set of N feature descriptors.

Each feature descriptor consists of a normalised $n \times m$ histogram with n rows, representing $\log r$ bins, and m columns, representing θ bins (see Figure 2.5c).

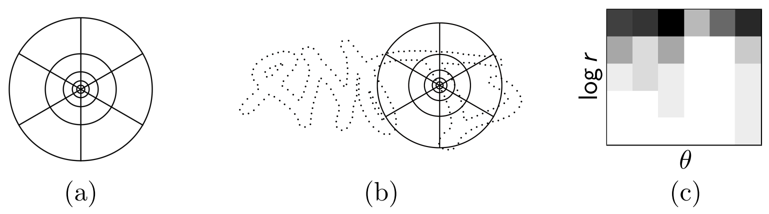


Figure 2.5: Extracting a single feature descriptor. (a) is the Shape Context kernel with five $\log r$ bins and six θ bins. (b) shows the kernel overlaid on one of the sample points. (c) is the resulting feature descriptor after points enclosed by the kernel have been binned into the histogram (darker = larger value).

2.2 Correspondence matching

In describing an image, whether using the SIFT or SC descriptor, a set of feature descriptors is obtained that represents the image. Both types of descriptors are histogram-based, therefore the method for bipartite matching used by Belongie, Malik, and Puzicha [2] can be used to compute matching descriptors.

This section describes the aforementioned method that looks for correspondences between two sets of homogenous descriptors.[1] Two important criteria when determining shape correspondences are that corresponding points should have very similar descriptors, and the correspondences should be unique. The following paragraph describes the algorithm [2] in detail.

Consider a point i on the first shape and a point j on the second shape. Compare descriptors at i and j to get the matching cost $C_{i,j}$ for these two points. Let the histogram at i be $g(k)$ and at j be $h(k)$. Then the cost $C_{i,j}$ is given by the χ^2 statistic

$$C_{i,j} = \frac{1}{2} \sum_{k=1}^{K} \frac{[g(k) - h(k)]^2}{g(k) + h(k)}. \tag{2.1}$$

The set of costs $C_{i,j}$ are inserted into a cost matrix at positions i, j. The implementation in this project then simply searches for the best two-way matches (smallest values) in the matrix, thereby finding the closest matching key points between the two shapes.

[1]Note that SIFT descriptors are incompatible with SC descriptors, therefore cannot be compared with each other.

2.3 Fitting a transformation using RANSAC

Once pairs of corresponding points $\{a_i \leftrightarrow b_i\}$ are established, a transformation is estimated that maps the graffiti tag from one image onto the graffiti tag in the other image. The homographic transformation was chosen to describe this mapping because it models affine and perspective transformations which could arise from variations in the way a tag is written, and the perspective effects when tags are photographed. The homographic transformation is given by

$$\begin{bmatrix} sx_b \\ sy_b \\ s \end{bmatrix} = \mathbf{H}_{ab} \begin{bmatrix} x_a \\ y_a \\ 1 \end{bmatrix}, \text{ where } \mathbf{H}_{ab} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix},$$

s is a scaling factor, and x, y are the 2D coordinats of the data points.

The homographic transformation $\mathbf{H}_{ab}$ is the mapping of data from sample set a onto b. Using the putative correspondences found in Section 2.2 as the sample sets, RANSAC provides an efficient and robust method for estimating the parameters for the homography [8].

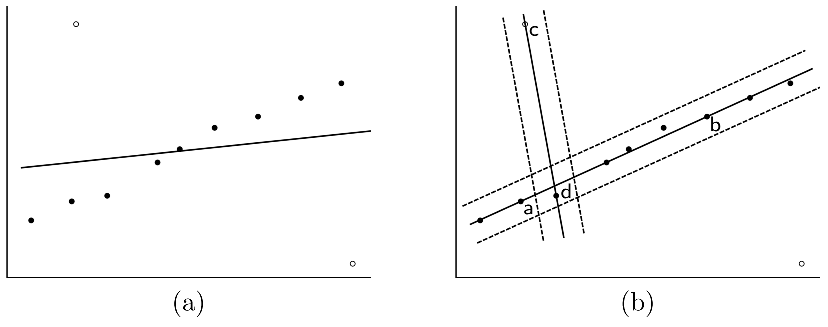


Figure 2.6: RANSAC compared with least squares estimation. Inliers •, outliers ○. (a) shows a least squares estimation of the data. All data points are used and outliers corrupt the estimation. (b) shows a RANSAC estimation of the data. Minimal data is used, outliers have no influence on the estimation. The dotted lines represent the perpendicular distance threshold *t*. In this example, the line *ab* has more support than *cd*.

RANSAC, or random sample consensus, introduced by Fischler and Bolles [7], is an algorithm that estimates the parameters of a mathematical model from a set of data that contains outliers. Outliers are points that do not fit the model; these points would corrupt a least-squares approach. RANSAC consists of two stages: fitting a mathematical model to the data, and classifying the data into inliers and outliers. An example of fitting data to a simple model that has the form $y = mx + c$ is shown in Figure 2.6. Extending this idea, since the homographic transform has eight parameters, four correspondence pairs are necessary to form the homography model.

The algorithm randomly selects the minimum number of points needed to create the model and then checks how much support the model gets from rest of the data. The support is measured by the number of data points that lie within a distance threshold t (illustrated in Figure 2.6 by dotted lines). The process is repeated by choosing a new set of random points, and calculating the support for the new model. The model that gained the strongest support is the final solution.

Testing every combination of sample points could be computationally intensive, and unnecessary. Instead, N trials are run, with N sufficiently high such that there is a probability P of 99% that at least one of the trials yielded no outliers. The calculation of N is explained by Hartley and Zisserman [8] as follows: let w be the probability that any one data point is an inlier, and so $\epsilon = 1-w$ is the probability that it is an outlier. The number of points needed to form the model is s. At least N selections of s points are required, where $(1 - w^s)^N = 1 - P$, therefore

$$N = \frac{\log(1-P)}{\log(1-(1-\epsilon)^s)}.$$

2.4 Word image matching

In a paper by Manmatha *et al.* [16], a solution is proposed to the problem of indexing handwritten historical documents. Rather than using Optical Character Recognition—which breaks up a scanned page into characters, and then attempts to recognise each character—the page is broken up into images of whole words. These images of whole words are clustered by similarity and then the clusters are presented to a human user to provide ASCII equivalents. Once a sufficient vocabulary is provided the document can be at least partially indexed.

Of interest is the technique used to 'spot' these words. However, an obvious disadvantage when applying this technique to the recognition of graffiti tags is that the technique assumes that images of words are consistent in viewpoint which will seldom be the case with photographed tags. Image pre-processing would be necessary to eliminates viewpoint variation, unless special care was exercised when taking photographs.

This section reviews three papers that deal with the matching of images of words.

2.4.1 Word image matching using Dynamic Time Warping

Manmatha *et al.* [16] presented a method for matching words based on Euclidean distance mapping. In a later paper by Rath and Manmatha [19], the method for word spotting was refined by using Dynamic Time Warping (DTW).

Although the images of words are 2D, they are recast as 1D using the idea that words, loosely speaking, are written from left to right as a function of time. This results in a plot of a profile of the word image against time.

Since the document is segmented into words by drawing bounding boxes around each word, an initial pruning can be performed by comparing widths, heights, areas and aspect ratios.

The candidate images are normalised for slant and skew and then features are extracted and combined into a time series. DTW is used to compute a matching cost between two time series. Due to local variation in an individual's handwriting, the profiles of two instances of the same word may look similar but could be skewed on the time axis. DTW works to warp the time axis such that it can find the column samples in each profile which correspond. Constraints are added to ensure that time warping is permitted only to a certain degree.

Results of experimentation [19] show that DTW performed better than five other matching techniques including: matching using difference images, sum of squared differences technique, the technique by Scott and Longuet-Higgins described in the paper *An algorithm for associating the features of two patterns* [21], Shape Context matching [3], and Euclidean distance mapping. On a particular set of ten pages this algorithm showed a precision of 65% and performed faster than the other methods that were examined.

2.4.2 Features for word spotting in historical manuscripts

Rath and Manmatha [18] discussed in more detail the features used in conjunction with DTW.

Before feature generation, the image is normalised with respect to slant and skew, and then the image is padded at the bottom such that the ratio of the area below the lower baseline to that above is 2/1. Comparability of the feature values across multiple words is ensured by normalising the feature range to $[0, 1]$.

Although the experiments [18] compared the performances of many types of features, the best average precision was achieved by an equal weighting of the features discussed in the following paragraph.

The following single-valued features (i.e. one scalar value per image column) are calculated:

- Projection profile—The normalised plot of the column-wise sum of pixel values in the word.
- Upper/lower word profile—For each image column, this is the plot of the distance from the top of the image to the top-most pixel in that column, and from the bottom of the image to the bottom-most pixel in that column. If no pixel is encountered, linear interpolation is used to fill the gap.
- Background-to-ink transitions—The column-wise count of the number of times a transition between the background colour and the ink colour is encountered in a binary threshold version of the image.

The experiment also tested:

- Partial projection profile—Equivalent to three projection profiles, one above the upper baseline, one below the lower baseline, and one between the baselines.
- Gaussian smoothing—The original image is smoothed and resized to a generic height and then each row of the image is used as a feature.
- Gaussian derivatives—Obtained similarly to the previously mentioned feature except are produced by a convolution in the horizontal and vertical directions of a Gaussian kernel partial derivative. Gaussian smoothing and Gaussian derivatives obtain feature sets rather than single-valued features.

Using this technique, an average precision of 72% was achieved. It outperformed competing techniques in speed and precision [18].

2.4.3 Holistic word recognition for handwritten historical documents

A paper by Lavrenko, Rath, and Manmatha [10] builds upon related previous work by Manmatha *et al.* [16, 18, 19].

For a given word image, its features are drawn from a state-conditional distribution. To achieve this, each word image is represented by a feature of fixed length. In order to turn features into a fixed length, low order coefficients from a

Discrete Fourier Transform (DFT) are used. The scalar and profile-based features form a vector of fixed length for word images of all sizes.

Images are first slant and skew normalised and cleaned up. The amount of padding below is corrected and the background of the image is set to a constant colour. The following scalar attributes are calculated for the word: height, width, aspect ratio (width/height), area, an estimate of the number of descenders in the word (strokes below the baseline caused by letters like 'p'), and an estimate of the number of ascenders. The following profile-based features are calculated: projection profile, upper word profile, and lower word profile. The profile time series are estimated by the lower-order coefficients of the DFT and the first four real (cosine) components and three imaginary (sine) components are extracted for use as scalar features.

These words are then modelled as a Markov Process with hidden states corresponding to words in the vocabulary, and observable states representing a (handwritten) rendition of those words.

Experiments in [10] show recognition accuracy of 65%, which exceeds performance of other systems which operate on non-degraded (non-historical documents).

CHAPTER 3

Extracting features from graffiti tags using SIFT

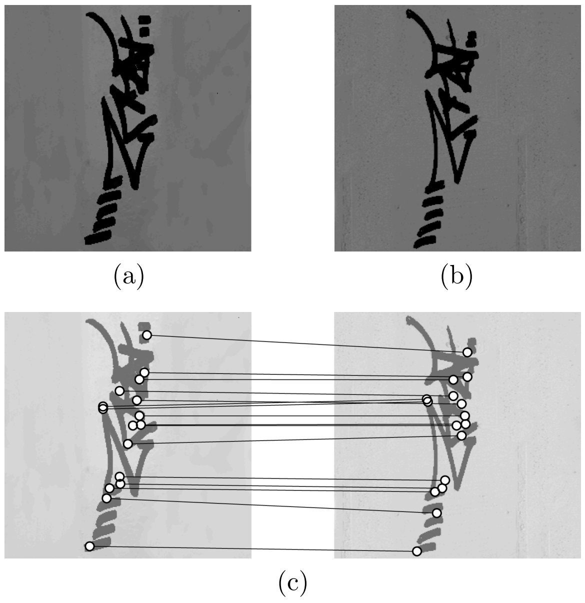


Figure 3.1: Matching graffiti tags using SIFT. Images (a) and (b) are the tags to be compared, (c) shows the matching between the two images using the SIFT algorithm.

The implementation of SIFT used here was written for MATLAB by El-Maraghi [6]. This chapter discusses its adaption, based on literature reviewed in Section 2.1, for use in the experiments.

The SIFT algorithm requires input images to be grey-scale. This implementation allows a binary image mask to be input as well. The mask indicates the areas of the image in which to look for key locations. Using an image mask reduces computation of image information that is irrelevant to the graffiti tag, thus saving time and producing a less cluttered representation of the tag.

3.1 Locating key points

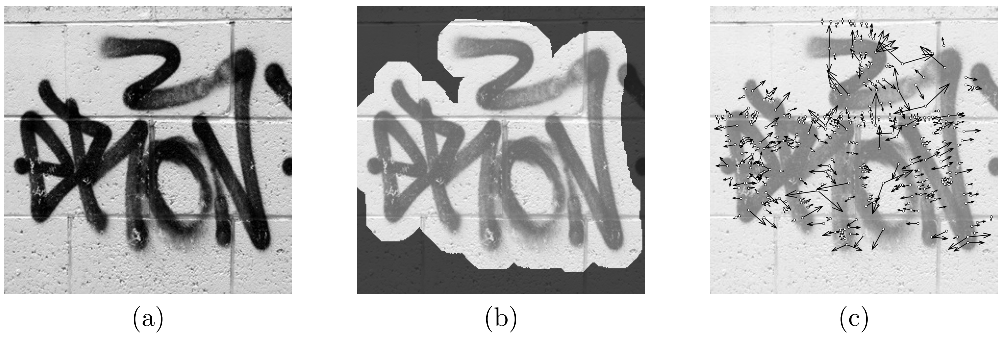


Figure 3.2: Locating key points using SIFT. The original 300×300 pixel image (a). Key points may only be located in the lighter area designated by the image mask (b). The image mask is the result of a generous dilation of a binary representation of the tag. The result of key point localisation (c). The length of each arrow is proportional to they scale of its key point, and direction of each arrow represents the orientation of its key point.

The initial stage in detecting key locations using SIFT is to build an image pyramid similar to the one shown in Figure 2.2. The multiple levels of the pyramid allow key points of various scales to be detected. Local extrema found by traversing the scale-space pyramid give rise to key points, provided the points lie within the bounds of the image mask.

The scale of key points that are detected doubles with each octave step. Intervals provide intermediate steps that allow the detection of key locations at scales in between the octave levels. The algorithm allows the number of octaves and intervals to be specified. In experiments conducted by Lowe [14], four octaves and two intervals were used.

At this stage the 2D coordinates of key points have been established by choosing local extrema in the difference-of-Gaussian pyramid, and their corresponding scales have been stored. The next stage is to remove key points with low contrast. This limits the effect of noise by ensuring that only the key points that have substantial pixel gradients are picked. A measure of key point contrast is explained by Lowe [14]. Given that image information lies in the range $[0, 1]$, a value for the minimum contrast threshold of 0.03 is recommended by Lowe.

Lowe suggests the removal of responses along edges. This is achieved by setting a threshold value for the maximum allowable ratio of principal curvature across an edge to that in the perpendicular direction. In experiments by Lowe [14], the value for the maximum allowable ratio was chosen to be 10.

It was considered, when applying SIFT to the matching of graffiti tags, that it may be useful to preserve edge responses as they tend to circumscribe the shape of the tag. An experiment was performed in which two instances of the similarity database (explained in Chapter 6) were constructed using 32 graffiti images, once with a curvature threshold of 10 (to eliminate edge responses as recommended by Lowe) and once with no curvature threshold (to preserve edge responses). By preserving edge responses, roughly 50% more key points were generated, which increased computation time when describing key points. However, both databases were found to have identical success rates of 12.5% in retrieving correct results from the database of 32 images (Section 6.3 explains the method for computing the success rate of a similarity database). The extra edge responses did not yield any advantage for matching tags; the reason is likely to be that key points that lie on straight edges tend to be small and therefore lack the discriminative power of feature vectors that have larger scales. It was therefore decided, for this project, that a curvature threshold of 10 would be used to eliminate edge responses.

A set of N stable key points has now been established. The (x, y) coordinates of the locations of these key points are stored in an $N \times 2$ matrix. Their scales are stored in an $N \times 3$ matrix; the first column specifies the octave, the second specifies the interval, and the third specifies the value of σ. Orientations of the key points are computed and stored in an $N \times 1$ matrix of angles in the range $[-\pi, \pi]$.

3.2 Describing key points

The orientation and scale matrices found in Section 3.1 are used to establish the pixel neighbourhood from which each feature descriptors will be formed. A feature descriptor is then computed at each key location as described in Section 2.1.2. A single feature is a 128 vector, therefore each graffiti tag image is represented by an $N \times 128$ matrix.

For efficiency, a database for images and their SIFT features was implemented so that each of the steps described in Sections 3.1 and 3.2 only need to be performed the first time an image is introduced to the system. Upon subsequent running of the comparison algorithm, the features are simply retrieved from the database, rather than extracted again from the image.

CHAPTER 4

Extracting features from graffiti tags using Shape Context

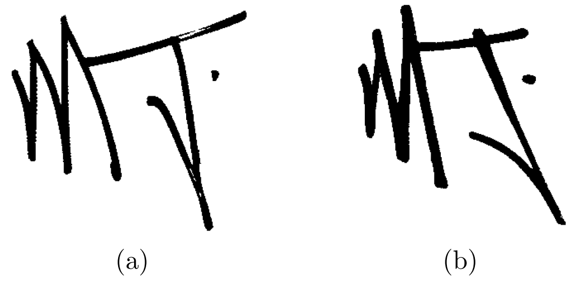
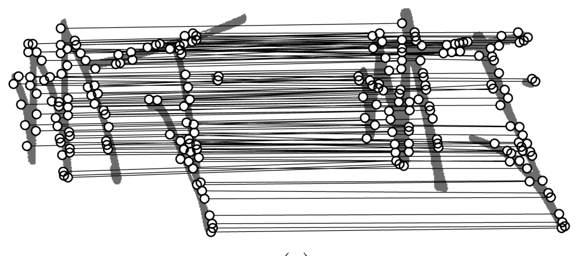


(a) (b) (c)

Figure 4.1: Matching graffiti tags using Shape Context. (a) and (b) are the original silhouette images to be compared. (c) shows the correspondences between the two tags after running the SC algorithm.

This chapter discusses the implementing of the SC algorithm, based on literature reviewed in Section 2.1, for use in the experiments.

SC extracts features from binary silhouette representations of shapes. A dimension of 300×300 pixels was chosen as the input image size for this implementation. At this size, computation time is kept low whilst maintaining adequate resolution to represent a graffiti tag.

4.1 Locating key points

The first step in locating key points for SC is to find the outline of the graffiti tag shape using an edge detection method. The Canny method was used in this implementation to produce the image in Figure 4.2a.

The next step is to sub-sample the outline to produce the dotted representation in Figure 4.2b. This is achieved by first converting the edge contours[1] into lists of contiguous pixel coordinates, then concatenating these lists to form one list; finally this list is sampled at every ith entry to produce a set of N points (where i is the sub-sampling interval).

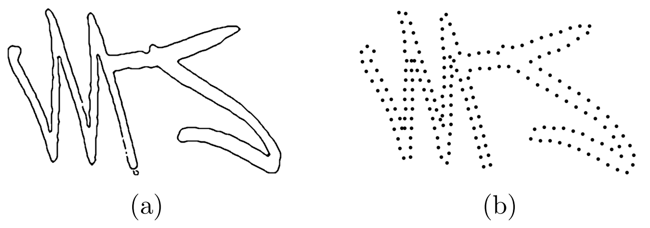


Figure 4.2: Locating key points by sub-sampling the outline of a shape. The outline (a) is found using the Canny method. The key points are located by sub-sampling the outline at an interval $i = 10$, resulting in a set of $N = 193$ points (b).

If i is chosen to be 1, then N becomes the size of the entire contour. This complete representation of the outline is unnecessarily detailed and significantly increases computation time in future steps. It was decided, for graffiti images of this size, that $i = 10$ yields a sufficient set of key locations for images of this level of detail. Figure 4.2b shows the even distribution of key locations produced by sub-sampling.

4.2 Describing key points

The kernel was constructed with five log radial distance bins and six equally spaced θ bins as recommended by Belongie and Malik [1]. Using broad θ bins achieves invariance to small changes in orientation, but the descriptor remains highly discriminative.

For scale-invariant matching the scale of the kernel must be normalised relative to the size of the tag. This is done by calculating the median distance

[1]There can be more than one edge contour if a graffiti tag is composed of unconnected parts.

λ between all N^2 key points pairs, as discussed in Section 2.1.4. The scale of the kernel is then set as $C\lambda$, where C is the scaling factor for λ. In the papers that introduce SC [1, 2, 3], the median distance λ is used to scale the kernel with no further scaling by C, (that is $C = 1$). Due to time constraints for this project, experimentation was not done to test the effects of changing the value of C. However, the value $C = 1$ was chosen for this implementation, based on the results achieved by Belongie and Malik [1].

CHAPTER 5

Matching feature descriptors

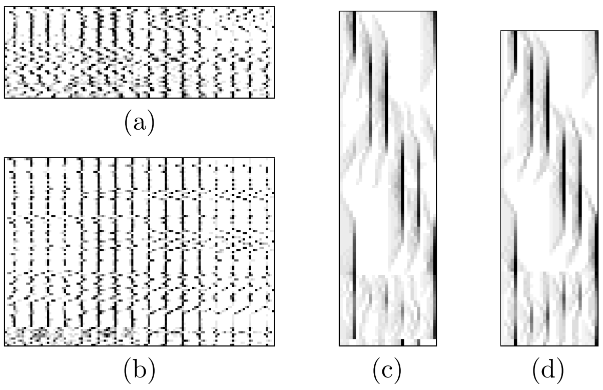


Figure 5.1: Feature descriptor matrices. Images (a) and (b) are SIFT descriptor sets generated from two similar input images. The images (c) and (d) are descriptor sets created from the same two input images using SC. (a) is an $N \times 128$ matrix, where $N = 43$. In (b), $N = 89$. (c) is an $N \times 30$ matrix, where $N = 104$. In (d), $N = 98$.

At this point, each graffiti tag is represented by a set of N feature descriptors (either SIFT descriptors or SC descriptors). The value of N differs between images as it depends on the number of key points that were located in the image.

A SIFT descriptor is a row vector of size 128. For an image, the N SIFT descriptors are concatenated to form an $N \times 128$ matrix (see Figure 5.1a).

An image that is described using SC results in N SC descriptors, each consisting of an $n \times m$ matrix (see Figure 2.5c). These $n \times m$ matrices are reshaped into row vectors of size $1 \times nm$. Experiments in this paper have $n = 5$ and $m = 6$, thus the reshaped row vectors are of size 30. The descriptor vectors are concatenated to form an $N \times 30$ matrix (see Figure 5.1c).

These feature set matrices are stored in the database, with the corresponding image coordinates, so that the descriptors need not be computed each time a comparison is done with an already processed image.

5.1 Putative matching

This is the first step towards the actual matching of two images of graffiti tags. The following computation is performed with homogenous descriptors (that is, SIFT descriptors can only be compared with SIFT descriptors, and likewise for SC descriptors).

In order to find the putative matching between two images, $image_A$ and $image_B$, their descriptor matrices are compared. The aim is to find pairs of descriptors, one from each of $matrix_A$ and $matrix_B$, that minimise the matching cost χ^2 (see Equation 2.1).

An $N_A \times N_B$ matrix is constructed to store matching costs, where N_A and N_B are the number of feature descriptors extracted from $image_A$ and $image_B$ respectively. Each feature vector in $matrix_A$ is compared with each feature vector in $matrix_B$ using the χ^2 measure. The results are stored in their corresponding cells in the matching matrix.

The matching matrix is traversed column-wise to obtain minima, which represent the strongest match for each of the N_A feature vectors from $matrix_A$ against each of the features vectors in $matrix_B$. This step is then performed row-wise to obtain the strongest matches in the reverse direction. These lists are then compared to find matches that are consistent in both directions; those that are not are discarded. The points that remain in the two lists represent the putative matching between $matrix_A$ and $matrix_B$.

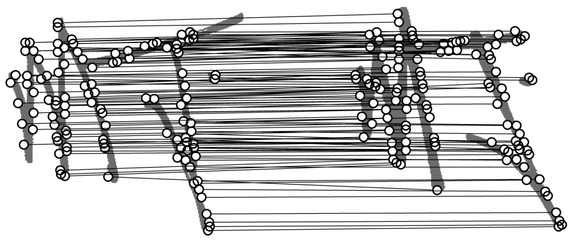


Figure 5.2: Putative matching using the χ^2 statistic. Feature descriptors were produced using SC. 467 features were extracted from the image on the left, 394 from the image on the right. After performing the putative matching, 108 correspondences remain.

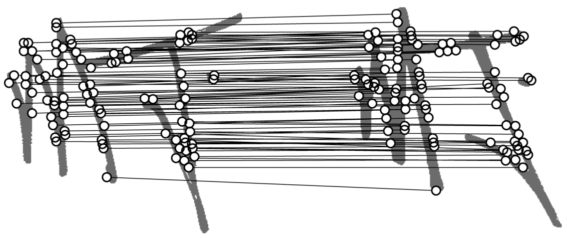
Figure 5.2 shows the putative matching results of the comparison between graffiti images shown in Figures 4.1a and 4.1b. In this example, SC was used to produce the feature descriptors. The next step is to establish a mathematical transformation that describes the mapping between the two tags.

5.2 Fitting a transformation

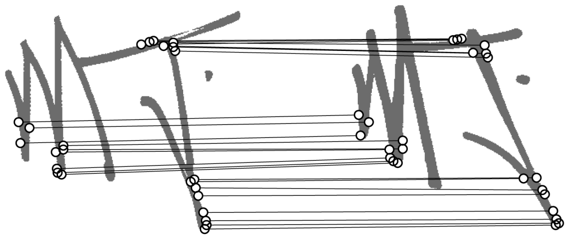
It was decided that it is necessary to use a homography to model the mapping between two instances of the same tag. This is because the homography deals with the affine transformations that model scaling and rotation of tags, as well as the perspective distortion that is introduced by the camera.

The putative matching in Section 5.1 provides a set of pairs of corresponding points. Using this data, the RANSAC algorithm fits a homography, and data points that support the model are classified as inliers. Point pairs that do not obey the model arise if the shape distortion between the two tags is too large. When comparing two completely different tags, the number of inliers is small.

An implementation of the RANSAC algorithm was provided by Kovesi [9]. The implementation takes in the two sets of points a and b, such that $\{a_i \leftrightarrow b_i\}$, and a value for the distance threshold t. The spread of data points is normalised such that the data points have a mean position of $(0, 0)$ and a mean distance from the origin of $\sqrt{2}$ (as recommended by Hartley and Zisserman [8]). The threshold size t is defined with respect to the normalised coordinates. The 3×3 homography matrix $\mathbf{H}$ is returned, with the list of points that are inliers.



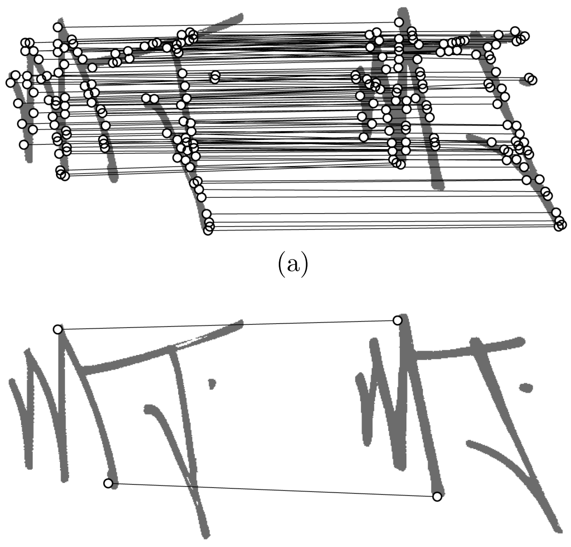
(a)



(b)

Figure 5.3: Inliers and outliers after RANSAC. (a) shows the 84 point correspondences found to be inliers to the homography that was established by the RANSAC algorithm. (b) shows 24 outliers. In this example, $t = 0.03$. A similarity score of 21.3% was obtained using the scoring formula given by Equation 5.1 in Section 5.3.

In the example shown in Figure 5.3b, there are point correspondences that appear to be correct to the human eye, but were classified as outliers because they did not obey the established transformation. This is due to subtle shape distortion between the two tags which arises because of the imperfections in reproducing a graffiti tag by hand. Tag recognition is more prone to shape distortions than generic object recognition problems, therefore, in order to improve the similarity score, a second pass of the RANSAC algorithm is performed on the set of outliers. The improvement in the score of genuinely matching tags outweighs the increase in false positive that arise in the second pass of the algorithm.


(a)

(b)

Figure 5.4: Improving the score with two passes of RANSAC. The number of inliers, shown in (a) has risen to 106. (b) shows only 2 outliers remain. The new similarity score is 26.9%.

The second pass of the RANSAC algorithm is relevant, in the case of comparing tags, when a tag is composed of several elements where each corresponding element may be similar, but their spatial arrangement is not. In Figure 4.1, the spatial arrangement of the 'M' and the 'J' differs slightly between the two tags. With the first pass of the RANSAC algorithm, the outliers are caused largely by the arrangement of the letter 'J' and the lower part of the letter 'M' (see Figure 5.3b). With the second pass, only two outliers remain as shown in Figure 5.4b.

5.3 The similarity score

When comparing tags, a score is needed to rate the similarity between them. Two images, $image_A$ and $image_B$, are described by N_A and N_B features respectively. The similarity score is calculated as the ratio

$$score = \frac{N_{inliers}}{min\{N_A, N_B\}}, \tag{5.1}$$

and is given as a percentage, where $N_{inliers}$ is the total number of inliers that were found, having run two passes of the RANSAC algorithm.

The lower of the values N_A and N_B is used since $N_{inliers} \leq min\{N_A, N_B\}$, therefore $score \leq 1$. Thus the maximum attainable similarity score is 100%.

CHAPTER 6

Searching the database of graffiti tags

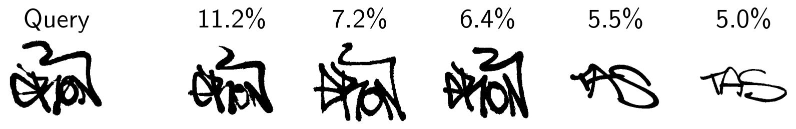


Figure 6.1: Searching the database with a query image.

This chapter describes the method used for converting a collection of photographs into a searchable database, and the method for retrieving tags from the database given a query image.

A matrix that holds similarity scores is used to facilitate quick retrieval of tags from the photograph database. This matrix is generated for a particular photographic collection by comparing every tag in the collection to each other, and then the similarity scores are arranged as a distance matrix (see Figure 6.2).

6.1 Creating the database

The database contains a matrix of similarity scores $\mathbf{S}$, and a list of tags $\mathbf{L}$. Each element in the list refers to a graffiti tag image in the collection of photographs such that $\{\mathbf{L}_i \mapsto image_i\}$. The matrix $\mathbf{S}$ is a distance matrix (and is therefore symmetric), with rows and columns referring to the tags in $\mathbf{L}$, such that the value at $\mathbf{S}_{i,j}$ is the similarity score between the tag $\mathbf{L}_i$ and tag $\mathbf{L}_j$. In a fully implemented system, the list $\mathbf{L}$ could be extended to store additional information about each tag, such as location, date found, offender (if known), etc.

Inserting a new graffiti tag image into the database involves adding its reference to the end of the list $\mathbf{L}$, and adding the similarity scores—into their respective positions—between itself and every other tag in the database.

	Tag A	*Tag B*	*Tag C*	*Tag D*	*Tag E*	*Tag F*	*Tag G*	*Tag H*
Tag A	–	31.63	23.02	24.42	8.14	8.12	8.84	7.35
Tag B	31.63	–	18.71	17.91	6.00	9.90	8.23	8.40
Tag C	23.02	18.71	–	20.59	8.57	8.88	6.75	5.77
Tag D	24.42	17.91	20.59	–	9.64	11.17	10.76	8.66
Tag E	8.14	6.00	8.57	9.64	–	17.77	12.42	9.45
Tag F	8.12	9.90	8.88	11.17	17.77	–	14.47	6.82
Tag G	8.84	8.23	6.75	10.76	12.42	14.47	–	10.76
Tag H	7.35	8.40	5.77	8.66	9.45	6.82	10.76	–

Figure 6.2: An example of a matrix of similarity scores (as percentages). This matrix was generated from a collection of eight photographs of tags using the SC descriptor, and a RANSAC threshold of 0.02. A query for the three closest matches of *Tag A* would return *Tag B* 31.63%, *Tag D* 24.42%, and *Tag C* 23.02%, in that order.

6.2 Querying the database

In this implementation, a query consists of the file name of the graffiti tag image. The file name must correspond to one of the tags listed in $\mathbf{L}$.

The column in the matrix $\mathbf{S}$ that belongs to a query tag Q contains the similarity scores between itself and all other tags in the collection of photographs. The database query function retrieves the n best matches for the tag Q by sorting the column in the descending order of similarity scores and then choosing the top n results. The original row positions of each of the results are used as indices into the the list $\mathbf{L}$ in order to access information pertaining to each of the query result tags.

6.3 Comparing the databases: SIFT vs. Shape Context

A collection of 32 graffiti tag photographs was used for experiments carried out in this section. The collection represents eight different graffiti writers; it contains four instances of tags from each writer. The collection displays a high degree of external variation (the variation between the tags of different writers), and a low degree of internal variation.

For a particular tag in this collection, the perfect result would be the retrieval of the remaining three tags belonging to the same writer—as the three closest matches. The correctness of the retrieval results can therefore be scored as a ratio of the number of correct matches that appear in the top three results. To measure the accuracy of the entire database, each of the 32 images are used to query the database, and the correctness scores are averaged to give a final success rate for the retrieval of correct results.

Since the similarity scores in the matrix **S** (see Section 6.1) depend upon the RANSAC distance threshold t, an experiment was necessary to decide an optimal value for t that yields the best matching results. In effect, the threshold controls the amount of shape distortion that is permissible. For example, a value for t of 0.02 will allow a variation in shape of up to 2%. If the threshold is too low, then very few points are classified as inliers, resulting in false negatives. However, if the threshold is too high, then point correspondences that should be considered outliers are incorrectly classified as inliers, giving rise to false positives.

The experiment involved computing ten **S** matrices, each using a different value of t. Having done some preliminary experiments, the range for t was chosen as $0.005 \leq t \leq 0.05$, at 0.005 intervals. The correctness rates of each of the **S** matrices were then computed. The entire experiment was performed once using the SIFT descriptor as the matching criteria, and once using SC. The correctness rates are shown on the graph in Figure 6.3.

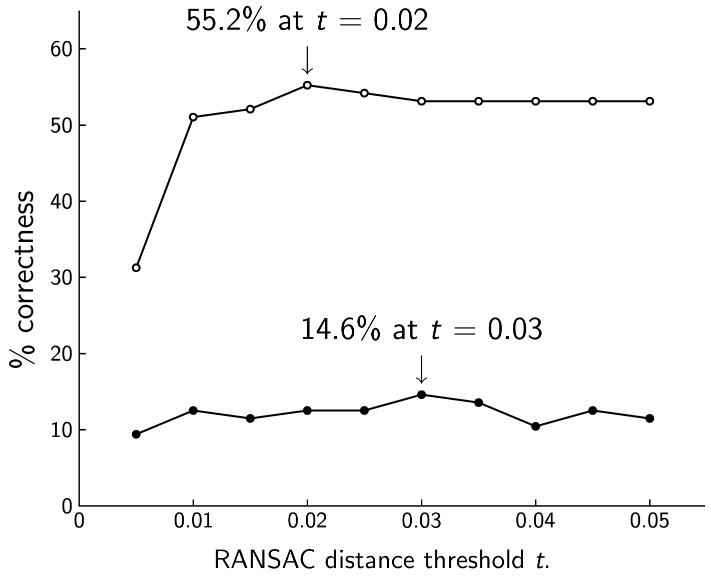


Figure 6.3: Finding an ideal value for the RANSAC threshold for both SIFT • and SC ∘. The graph shows the effect of the value of t on the correctness of the matching matrix.

Referring to Figure 6.3, the graph shows that the value of t is not critical for either SIFT or SC. Even so, the peak in the graph of 14.6% implies that 0.03 is the better value for t when using SIFT, and the peak in the graph of 55.2% implies that 0.02 is the better value for t when using SC. Retrieval results for the best performing SIFT database and the best performing SC database are included as Appendices A and B respectively.

The **S** matrices that were created using SIFT and SC using values for t of 0.03 and 0.02 respectively were used to produce the histograms in Figures 6.4 and 6.5.

For a particular tag in the collection of 32 photographs there are three other tags in the collection that are considered to be positive matches. A match with any of the remaining 28 tags is considered to be a negative match.

Using its **S** matrix, the distributions of similarity scores for positive matches and for negative matches are shown by the two histograms (black and grey) in Figure 6.4 for SIFT. The equivalent histograms are plotted for SC, in Figure 6.5, using its **S** matrix.

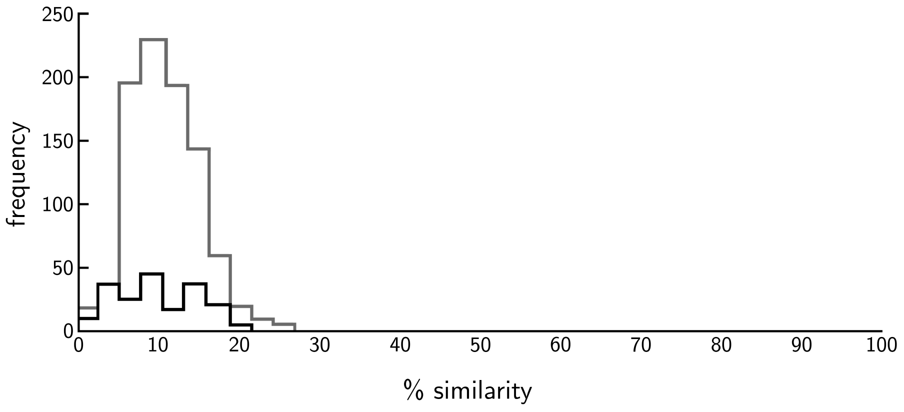


Figure 6.4: Distribution of similarities for the SIFT database. The histogram outlined in black represents the positive matches. The histogram outlined in grey represents the negative matches. Similarity scores are taken from the respective **S** matrix.

The fact that the distribution of positive matches lies with the distribution of negative matches, as shown in Figure 6.4, suggests that true positive results do not exist in the SIFT **S** matrix. Therefore a query using any tag will not return a reliable matching result.

Figure 6.5 shows a better performance than the previous figure because there is less overlap between the two histograms. The distribution of positive matches at 20% and above represents true positive results. The fact that the positive distribution spans the range almost down to zero means that there is no significant indication of true negatives. This means that results can become cluttered with negative matches, but not as badly as when using SIFT.

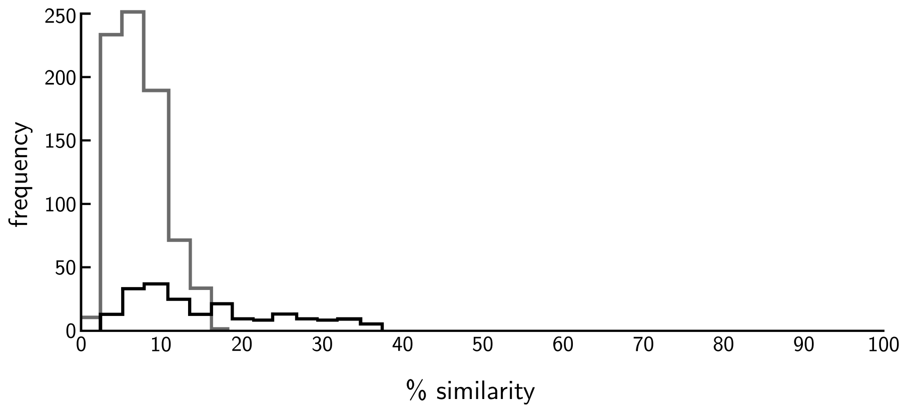


Figure 6.5: Distribution of similarities for the SC database. The same legend applies as in Figure 6.4.

The lack of clear separation between the positive and negative distributions, in both figures, is disappointing. Nevertheless, for future work, these histograms provide a useful way to visualise and evaluate the effects of changing attributes of the overall system, such as parameters, choice of descriptor, or ways of scoring similarities.

CHAPTER 7

Conclusion

The outcome of this thesis demonstrates that the automatic recognition of graffiti tags is possible and that SC is a suitable algorithm for the task. A system for matching similar graffiti tags was developed based on the following structure: feature descriptors are extracted from images, images are matched by comparing the feature descriptors, the results are used to construct a database that facilitates quick retrieval of tags based on similarity.

For the initial step of extracting feature descriptors from images, the SIFT and SC descriptors were used and the performance of each was compared empirically in Section 6.3. Using this measure, the database created using SC as the descriptor was found to be 55.2% correct, while the database created using SIFT as the descriptor was only 14.6% correct.

In terms of the graph shown in Figure 6.5, SC demonstrated better matching results than SIFT (Figure 6.4) since, with SC, there was less overlap between the histograms that represent positive and negative matches. Future work should strive to achieve greater separation between the two distributions, thus a more accurate grouping of tags.

Notably, the SC descriptor has only 30 elements and yet, in the context of recognising graffiti tags, it is clear from the observations that it greatly outperforms the SIFT descriptor that has 128 elements.

Although SIFT is highly regarded for its abilities in locating a particular object in a scene, it does not perform well when it is used to locate an arbitrary instance of an object. While tags that belong to the same writer may appear similar on the global level, there are often variations caused by small changes such as writing implement and surface texture that have a large negative impact on the effectiveness of the SIFT descriptor. Referring to Figure A.1, it is unclear, without further investigation, why the closest matching result is dominated by one tag for almost all queries. If the problem is occurring in the fitting of a transformation, it is suggested that, in future work, constraints should be placed on the homography, such as not to allow radical transformations to be computed.

It is also possible that a homography is not a suitable model for the mapping between two tags matched using SIFT. Lowe [13] uses an affine transformation, which was initially decided against, for this project, because it does not allow perspective projection.

SC is not as badly affected by the characteristics that vary between instances of the same tag because, unlike SIFT, its descriptor naturally operates on the global level. Thus the overall shape of the tag is important to SC while it ignores the details on the pixel level.

Further experimentation using SC should involve examining the effects of scaling the SC kernel. Also, a sophisticated method for achieving rotation invariance and robust transformation fitting in the presence of shape distortion is discussed in a paper by Belongie, Malik, and Puzicha [3], and could be incorporated into the current implementation in order to improve matching results.

Other techniques for shape matching exist—such as the techniques for word image matching discussed in Section 2.4—that could not be tested in this thesis due to time constraints.

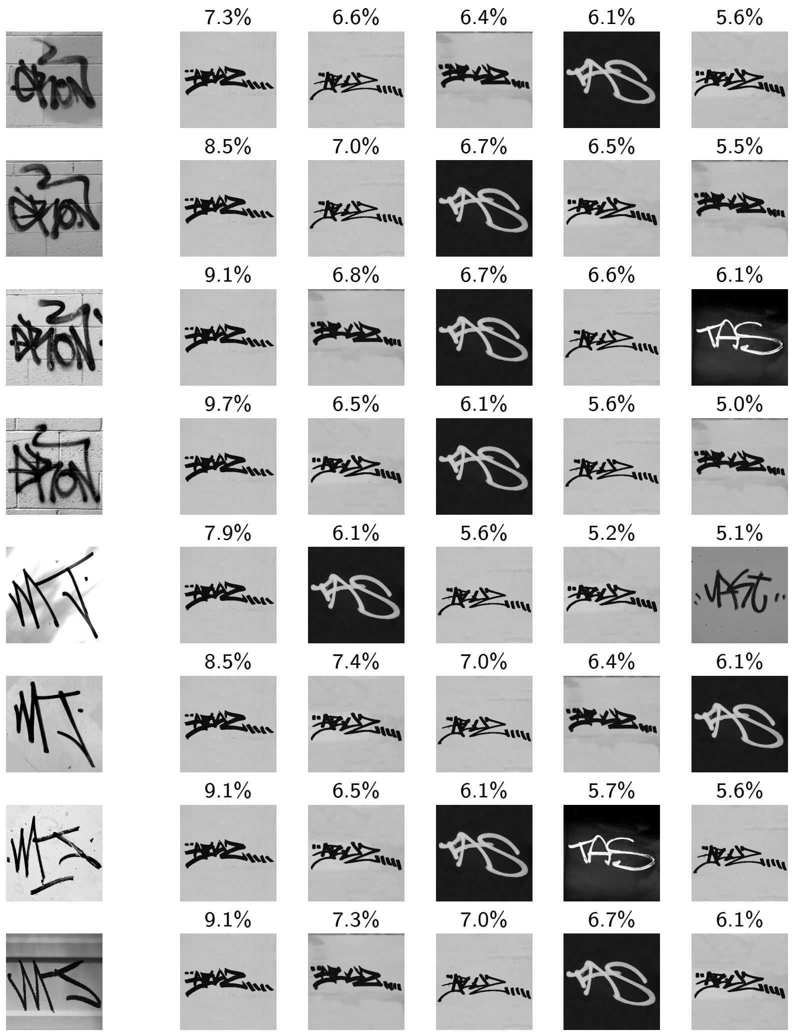
It is pleasing to observe, in Figure B.1, that even when a perfect retrieval is not obtained, correct results are often highly ranked. A human user would be able to quickly eliminate incorrect results.

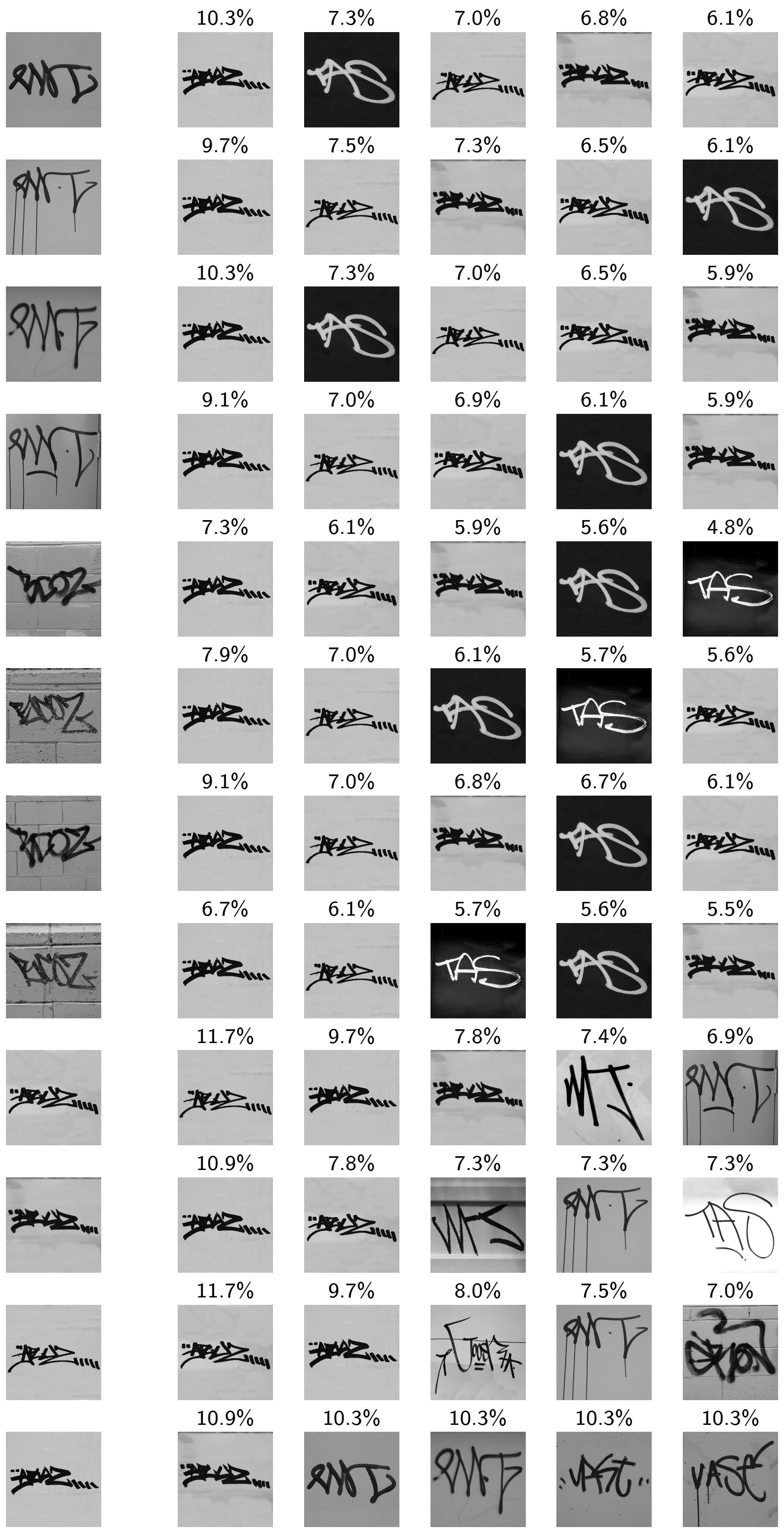
This thesis provides a foundation for future work in the area of recognising graffiti tags using shape matching. Future research on this subject should aim to improve the results obtained using SC, and introduce other shape matching techniques to the problem. The recognition of graffiti tags is a non-trivial application for shape matching, therefore the matching results using SC are promising, but the correctness score of 55.2% would need to be improved for a real-life system.

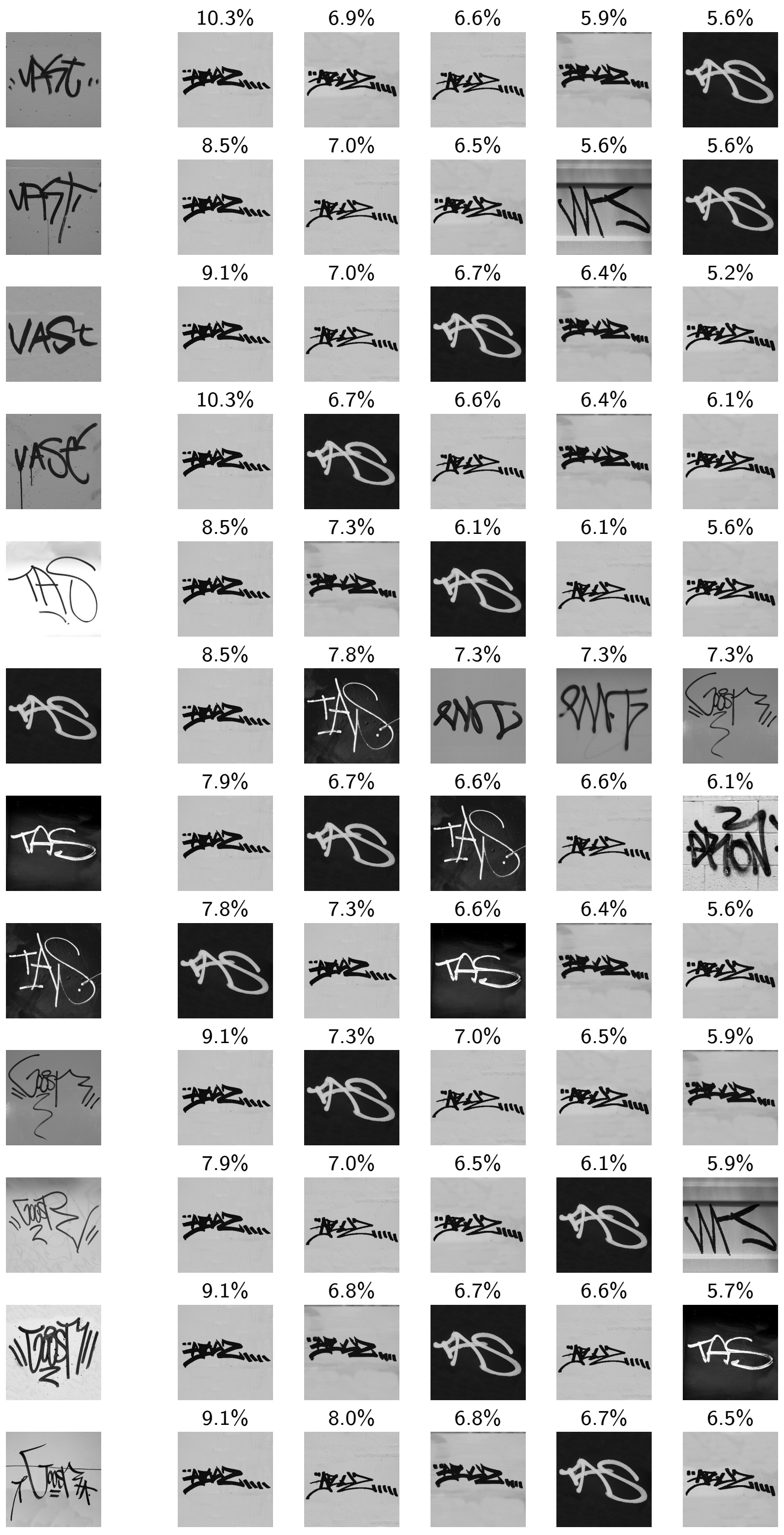
APPENDIX A

Database query results: SIFT

Figure A.1: The retrieval results, using the SIFT descriptor and a RANSAC distance threshold of 0.03 for a collection of 32 tag images (four instances of tags from eight different writers). The success rate for correct retrieval is 14.6%. Query images shown on the left.



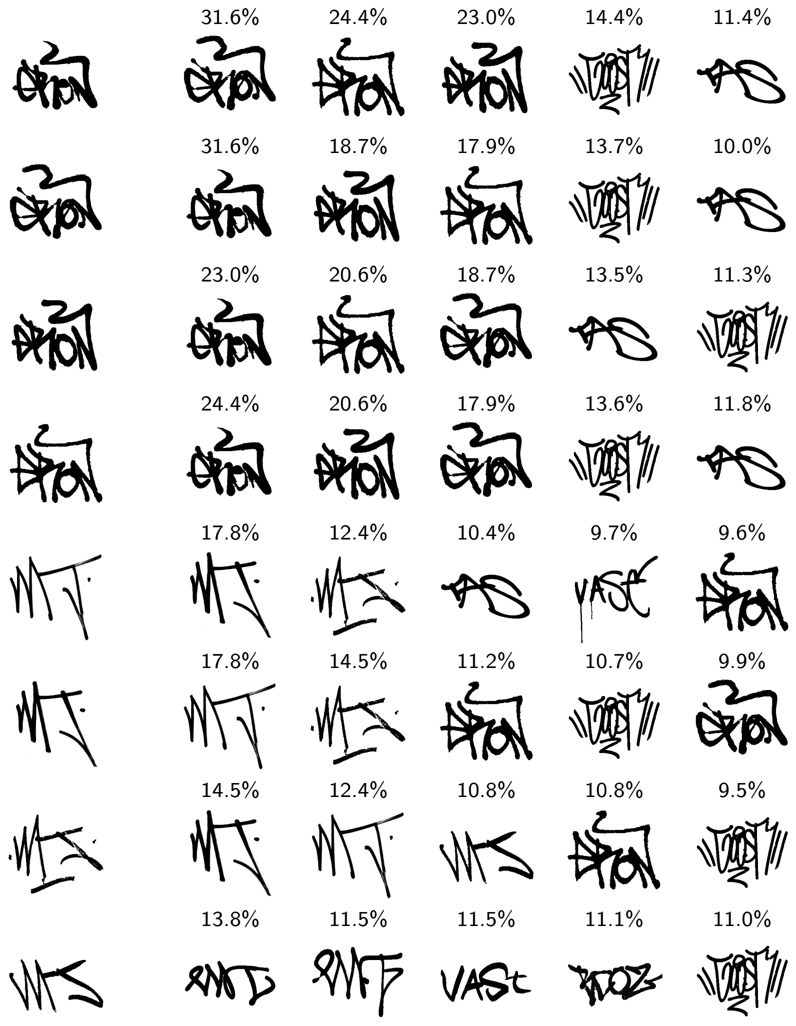

10.3% 7.3% 7.0% 6.8% 6.1%
9.7% 7.5% 7.3% 6.5% 6.1%
10.3% 7.3% 7.0% 6.5% 5.9%
9.1% 7.0% 6.9% 6.1% 5.9%
7.3% 6.1% 5.9% 5.6% 4.8%
7.9% 7.0% 6.1% 5.7% 5.6%
9.1% 7.0% 6.8% 6.7% 6.1%
6.7% 6.1% 5.7% 5.6% 5.5%
11.7% 9.7% 7.8% 7.4% 6.9%
10.9% 7.8% 7.3% 7.3% 7.3%
11.7% 9.7% 8.0% 7.5% 7.0%
10.9% 10.3% 10.3% 10.3% 10.3%

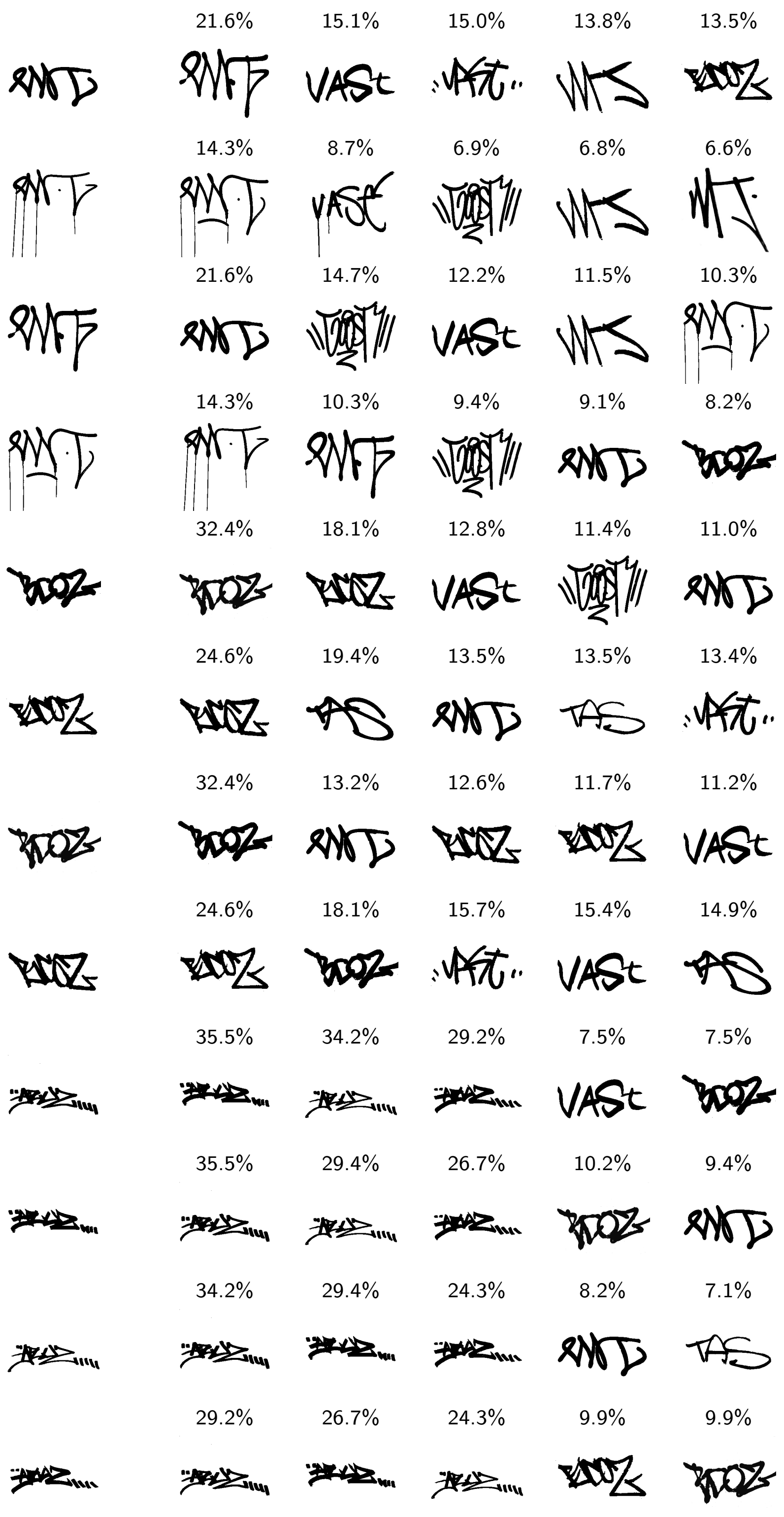

10.3% 6.9% 6.6% 5.9% 5.6%
8.5% 7.0% 6.5% 5.6% 5.6%
9.1% 7.0% 6.7% 6.4% 5.2%
10.3% 6.7% 6.6% 6.4% 6.1%
8.5% 7.3% 6.1% 6.1% 5.6%
8.5% 7.8% 7.3% 7.3% 7.3%
7.9% 6.7% 6.6% 6.6% 6.1%
7.8% 7.3% 6.6% 6.4% 5.6%
9.1% 7.3% 7.0% 6.5% 5.9%
7.9% 7.0% 6.5% 6.1% 5.9%
9.1% 6.8% 6.7% 6.6% 5.7%
9.1% 8.0% 6.8% 6.7% 6.5%

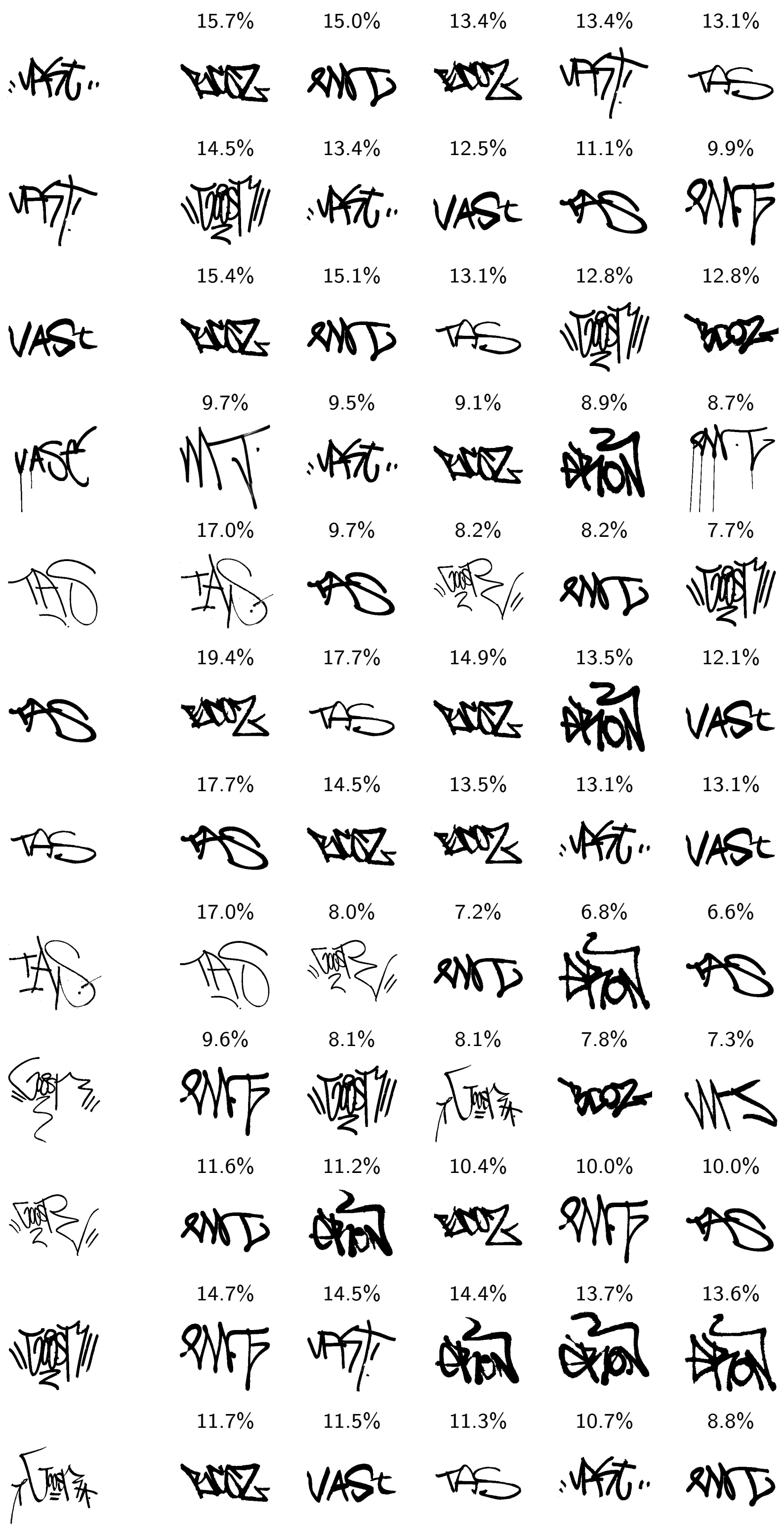
APPENDIX B

Database query results: Shape Context

Figure B.1: The retrieval results, using the SC descriptor and a RANSAC distance threshold of 0.02 for a collection of 32 tag images (four instances of tags from eight different writers). The success rate for correct retrieval is 55.2%. Query images shown on the left.




21.6% 15.1% 15.0% 13.8% 13.5%
14.3% 8.7% 6.9% 6.8% 6.6%
21.6% 14.7% 12.2% 11.5% 10.3%
14.3% 10.3% 9.4% 9.1% 8.2%
32.4% 18.1% 12.8% 11.4% 11.0%
24.6% 19.4% 13.5% 13.5% 13.4%
32.4% 13.2% 12.6% 11.7% 11.2%
24.6% 18.1% 15.7% 15.4% 14.9%
35.5% 34.2% 29.2% 7.5% 7.5%
35.5% 29.4% 26.7% 10.2% 9.4%
34.2% 29.4% 24.3% 8.2% 7.1%
29.2% 26.7% 24.3% 9.9% 9.9%


15.7% 15.0% 13.4% 13.4% 13.1%
14.5% 13.4% 12.5% 11.1% 9.9%
15.4% 15.1% 13.1% 12.8% 12.8%
9.7% 9.5% 9.1% 8.9% 8.7%
17.0% 9.7% 8.2% 8.2% 7.7%
19.4% 17.7% 14.9% 13.5% 12.1%
17.7% 14.5% 13.5% 13.1% 13.1%
17.0% 8.0% 7.2% 6.8% 6.6%
9.6% 8.1% 8.1% 7.8% 7.3%
11.6% 11.2% 10.4% 10.0% 10.0%
14.7% 14.5% 14.4% 13.7% 13.6%
11.7% 11.5% 11.3% 10.7% 8.8%

APPENDIX C

Original Honours Research Proposal

Title: Graffiti Tag Recognition

Author: Paul Schwarz

Supervisor: Dr David Glance & Dr Peter Kovesi

Background

A tag is an individual graffiti writer's signature that either accompanies a work of graffiti art or is inscribed in isolation as a personal identifier. Investigations into the graffiti subculture have established that writers operate either alone or in gangs to mark territory [12]; the aim being to 'claim' public space and to command respect from rivals. Artists desire to be recognised based on their tags and therefore they develop unique styles.

Developing a tagging style requires practice, much like how an individual learns to write. Handwriting is a complex behaviour which is developed through repetition. The brain is trained to control the muscles which hold the writing implement. Once writing becomes routine the style tends to remain constant [17]. This project assumes that tagging styles differ significantly enough, from person to person, and remain constant enough on an individual basis, to be detectable by an algorithm.

The proposed approach to the graffiti tag recognition problem is to develop a system that can take an input image and search a database of tag images and returning all similar tags. It will be a used to establish whether a tag is a one-off instance or one of multiple offences; the latter carrying a significantly higher penalty [11].

The known presence of a fully implemented tag recognition system could be used as a tool to deter vandals from tagging public spaces, thus saving money for city councils that employ clean-up operations. The City of Subiaco, Perth, Western Australia, allocated $161 500 for safety and security, including graffiti management in their 2005 – 2006 budget [5]. The system will also play a role in forensic investigations and may be used to collect evidence for trials.

Although algorithms for pattern-matching and recognition have not previously been used in comparing graffiti tags, much research has been done in the fields of object recognition, and signature and handwriting recognition. These techniques are of interest to the project since they address similar problems to those of tag recognition.

In his work on object recognition from local scale-invariant features [13], Lowe presents an object recognition system based on local feature comparison. This technique involves transforming an image into a large collection of local feature vectors which are invariant to image translation, scaling and rotation, and partially invariant to illumination changes and affine or 3D projection. The Video Google application by Sivic and Zisserman [22] is an example of how this object recognition technique can be used to index a database of images and facilitate efficient retrieval based on an image search.

A technique of off-line signature verification, proposed in a paper by Sabourin, Genest and Prêteux [20], is based on visual perception, focusing attention on fixed regions by restricting the field of view of 'retinas'. The algorithm represents the signature based on a local analysis of the scene, using morphological operators to evaluate the amount of signal activity in each scene. The paper claims a low error rate in the comparison of genuine signatures with forgeries.

Manmatha, Han, Riseman, and Croft devised a system for indexing historical, handwritten documents [16] using a technique of firstly delimiting the images of each of the words, then matching and grouping like words, and finally requiring a human user to enter ASCII equivalents for the top couple of thousand equivalence classes. Of interest is the technique used to match the words. A bounding box is drawn around each word; then aspect ratios of these boxes are compared for similarity. If similar, the images are aligned and an XOR function is applied followed by a Euclidean distance mapping. The images are then ranked by Euclidean distance error. This correlates with the degree of similarity between words.

These techniques offer possible approaches to the graffiti matching problem.

Aim

A number of algorithms exist that are capable of pattern-matching regions of images. The aim of this research is to find and test algorithms capable of comparing images of graffiti tags and then to implement and refine the algorithm which performs the best with regard to accuracy and speed.

Method

- A meeting with the Centre for Forensic Science at UWA will take place to gather background information on the study of graffiti tagging and to understand the already existing methods of identification.
- Requirements must be elicited for the design of a database which will store records of photographed tags alongside information which may be useful to a city council or a forensic detective. The database will be implemented.
- A user interface will be designed and implemented to form the basic working environment for the end user, and for the researcher during the development of the system.
- Techniques for image processing and comparing of image regions will be researched and compared. Prospective methods will be adapted or implemented and tested for performance with regards to accuracy of results and speed.
- Further implementation may involve indexing the data and structuring the database to provide efficient searches for matching tags.

Plan

Task	Completion
Meeting with Centre for Forensics	March
Reading and researching	April
Database requirements elicitation	April
Implementation of database and UI	April/May
Implementation of techniques for image comparison	June/July
Extending functionality to search database	August
Testing and concluding research	September
Prepare presentation	October
Finish dissertation	October

Software and Hardware Requirements

There are no special software or hardware requirements for this project. Software packages including Matlab, MySQL, and Visual Studio – to name a few which may been necessary for this project – are provided on the UWA CSSE lab computers.

Bibliography

[1] Belongie, S., and Malik, J. Matching with shape contexts. In *CBAIVL '00: Proceedings of the IEEE Workshop on Content-based Access of Image and Video Libraries (CBAIVL'00)* (Washington, DC, USA, 2000), IEEE Computer Society, p. 20.

[2] Belongie, S., Malik, J., and Puzicha, J. Shape context: A new descriptor for shape matching and object recognition. In *NIPS* (2000), pp. 831–837.

[3] Belongie, S., Malik, J., and Puzicha, J. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell. 24*, 4 (2002), 509–522.

[4] Burt, P. J., and Adelson, E. H. The Laplacian Pyramid as a compact image code. *IEEE Transactions on Communications COM-31, 4* (1983), 532–540.

[5] City of Subiaco. City of Subiaco Budget 2005-2006, Mar. 14 2006. http://www.subiaco.wa.gov.au/cos/reports/pubdoc/2005-2006budget.pdf.

[6] El-Maraghi, T. F. MATLAB SIFT tutorial. Computational Vision Group, The University of Toronto. Available from: ftp://ftp.cs.utoronto.ca/pub/jepson/teaching/vision/2503/SIFTtutorial.zip.

[7] Fischler, M. A., and Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24*, 6 (1981), 381–395.

[8] Hartley, R. I., and Zisserman, A. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521623049, 2000.

[9] Kovesi, P. D. MATLAB and Octave functions for computer vision and image processing. School of Computer Science & Software Engineering, The University of Western Australia. Available from: http://www.csse.uwa.edu.au/∼pk/research/matlabfns/.

[10] Lavrenko, V., Rath, T. M., and Manmatha, R. Holistic word recognition for handwritten historical documents. In *DIAL '04: Proceedings of*

the First International Workshop on Document Image Analysis for Libraries (DIAL'04) (Washington, DC, USA, 2004), IEEE Computer Society, p. 278.

[11] Legislative Assembly. Police Amendment Bill – Committee, Sept. 9 1998. http://www.parliament.wa.gov.au/hansard/hans35.nsf/(ATT)/BBEEBAB4FCC464E94825669500126AAE/$file/A0909012.PDF.

[12] Ley, D., and Cybriwsky, R. Urban graffiti as territorial markers. In *Annals of the Association of American Geographers* (Dec. 1974), vol. 64, pp. 491–505.

[13] Lowe, D. G. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on* (Sept. 20–27 1999), vol. 2, pp. 1150–1157.

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

[15] Manmatha, R., Chengfeng Han, and Riseman, E. M. Word spotting: A new approach to indexing handwriting. Tech. rep., Computer Science Department, University of Massachusetts at Amherst, MA, 1995.

[16] Manmatha, R., Chengfeng Han, Riseman, E. M., and Croft, W. B. Indexing handwriting using word matching. In *DL '96: Proceedings of the first ACM international conference on Digital libraries* (New York, NY, USA, 1996), ACM Press, pp. 151–159.

[17] Perl, W. R. On the psychodiagnostic value of handwriting analysis. *American Journal of Psychiatry*, 111 (1955), 595–602.

[18] Rath, T. M., and Manmatha, R. Features for word spotting in historical manuscripts. In *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition* (Washington, DC, USA, 2003), IEEE Computer Society, p. 218.

[19] Rath, T. M., and Manmatha, R. Word image matching using dynamic time warping. vol. 2, pp. 521–527.

[20] Sabourin, R., Genest, G., and Preteux, F. J. Off-line signature verification by local granulometric size distributions. In *Transactions on Pattern Analysis and Machine Intelligence* (Sept. 1997), vol. 19, pp. 976–988.

[21] Scott, G. L., and Longuet-Higgins, H. C. An algorithm for associating the features of two patterns. In *Proceedings of the Royal Society of London B224* (1991), pp. 21–26.

[22] Sivic, J., and Zisserman, A. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision* (Oct. 2003), vol. 2, pp. 1470–1477.