Invariant Local Feature for Image Matching

Wong Yuk-Man

Term Paper for the Degree of
Master of Philosophy
in
Computer Science and Engineering

Supervised by

Prof. Michael R. Lyu

Abstract

Approaches based on invariant local feature descriptors have been widely employed in many computer vision applications, including image retrieval and object recognition. Since the introduction of the invariant local features for solving image matching problems, many research tasks have been performed to further improve the detection, description and matching process of local features in three interacting aspects: the distinctiveness, the invariance and the speed.

In this paper we propose a new feature descriptor resembling a state-of-the-art descriptor, Scale Invariant Feature Transform (SIFT), which is also invariant to change in background and object color. It commonly happens that images of objects under the same image class differ with background and object color. Thus our feature descriptor is particularly useful for object class recognition purpose. The performance evaluation shows that our descriptor performs much better than others on datasets that capture changes in background and object color.

Another contribution of this paper is in building a system for efficient image retrieval using SIFT-based descriptors to describe image features and Exact Euclidean Locality-Sensitive Hashing (E2LSH) to index the descriptors. To further improve the performance of the system, we propose a new matching strategy and a new verification process that are not adopted by other image near-duplicate detection system. The accuracy is high and the query times are short even for databases of millions of keypoints.

Contents

Chapter 1

Introduction

Invariant local feature descriptors have been widely employed in many computer vision applications, including automatic panorama stitching, wide baseline matching, image retrieval [29], object recognition [22], and object class recognition [23]. They are distinctive, invariant to common image transformations and robust to occlusion in nature. The popularization of invariant local feature descriptors has recognized its potential in tackling the image matching problems.

Since Schmid and Mohr [31] introduce invariant local features for solving image matching problems, many research tasks have been performed to further improve the detection, description and matching process of local features in three interacting aspects: the distinctiveness, the extent of invariance, and the speed of the process. SIFT descriptor [22] is one of the state-of-the-art descriptors that is shown to be very robust to many image transformations. Other SIFT-based descriptors, such as PCA-SIFT [29] and GLOH [15], are also shown to be superior to other types of local feature descriptors, such as spin image [17] and shape context [3] on feature matching tasks. Recently, a new detector-descriptor scheme, called Speeded Up Robust Features (SURF) [2], is proposed. According to our performance evaluation, it has comparable performance to SIFT but is about three times faster in computation than SIFT.

In this paper we propose a new invariant local descriptor, Shape-SIFT (SSIFT), which extends Scale Invariant Feature Transform (SIFT) descriptor to background and object color invariance. Local invariant feature is hereditarily robust to occlusion and clutter background. However, despite the local nature of the described features, background still significantly distorts the features near the contour of an object. his effect appears frequently among corner-like features and tremendously reduces the recognition performance of textureless objects. Since the contour of an object defines its shape and many objects can even only be recognized using the shape only, contour is an important visual cue for object recognition. Thus it is necessary to describe local features in a way that is invariant to change in color of the background. On the other hand, a local descriptor for object class recognition should be invariant to object color because objects in the same object class can share the same shape with different colors. We observe the common causes that make SIFT-based descriptor variant to these transformations and further propose solutions to attack this problem. The performance evaluation shows that our descriptor performs much better than other state-of-the-art descriptors on data sets that capture changes in background and object color.

Another contribution of this paper is in building an image retrieval system for image near-duplicate detection purpose. This system makes use of the state-of-the-art feature detector, descriptor and indexing techniques in building a local feature database for a set of images and in building an index of the feature database. Thus, this system is both accurate and efficient. To further improve the performance of the system, we propose a new matching strategy and a new verification process that are not adopted by other image near-duplicate detection system. The performance evaluation shows that our proposed approaches improve the accuracy of the system significantly.

We will first give some literature reviews on image matching techniques using invariant local features in chapter 2. In that chapter, we will describe the three major components of the approaches using invariant local features, namely feature detector, feature descriptor and feature matching. A number of approaches have been proposed to improve the performance of these three major components. To find out the superior approaches, we compare the performance of different feature descriptors. The experimental result is also presented in chapter 2. In chapter 3, we will present the newly proposed feature descriptor, SSIFT. Detailed explanation on this approach and detailed experimental result will be given. In chapter 4, we will present our image near-duplicate detection system. Again, detailed explanation on the system and detailed experimental result will be given in that chapter. Lastly, we will conclude this paper in chapter 6.

□ **End of chapter.**

Chapter 2

Literature Review

2.1 Introduction

Invariant local features for recognition refer to the representations of image contents, at some particular interest regions on the images of scene or object. These features are local as they are related to small regions on objects instead of the whole object. This property makes feature-based recognition inherently robust to occlusion and clutter. These are the two serious problems in recognition using global features and are usually solved by image segmentation techniques. Since the performance of current image segmentation techniques are still limited, performance of recognition using global features is limited too. On the other hand, recognition using local features solve these problems easily. During recognition, local features for an object can be matched to a database of local features each representing a unique object. By using some voting algorithms, the object in the query image can be obtained and thus "recognized". Since images of the same object can be taken in different environmental and instrumental conditions, they are probably different but related. Differences between these images include image noise level, change in illumination, scaling, rotation and change in viewing angle. In order to match two different images of the same object, the local features should be invariant to these differences. Invariance of a local feature refers to its ability to tolerant these differences. The extend of invariance depends on how its representation is designed. A good local feature should be highly distinctive which means it should allow for correct object identification with high probability. However, the more invariance a feature has, the less distinctive it has. Therefore, there are trade-off between *invariance* and *distinctiveness*.

Three keys processes involved in feature-based recognition are *feature detection*, *description* and *matching*. We will discuss the state-of-the-art techniques used in these three processes in the following sections.

2.2 Feature Detector

Since the resolution of an object's image can be very high, it is not practical in efficiency, storage and accuracy to take every pixel of the image as an feature and describe by an vector. It is necessary to extract only a subset of pixels from an image to be described. We call this subset of pixel as the interest points. There are two main requirements on feature detector. First, corresponding interest points on the object should be repeatedly detected by the feature detector over different images of the same object. Second, interest points detected should be distinctive local features. 2D image windows, where there is some form of 2D texture likes corner, are the most distinctive image patch comparing with other types of image windows. A number of feature detectors have been proposed to use 2D window for recognition purpose, they includes Harris corner detector [13], DOG extrema detector [22], Harris-Laplacian detector [25] and affine covariant region detector [26].

2.2.1 Harris Corner Detector

Harris corner detector [13] is widely used in many image matching tasks to select regions that have significant gradient change in all directions.

The Auto-Correlation Matrix

This detector analyzes the auto-correlation matrix $\mathbf{M}$ of every location in an image that is computed from image derivatives:

$$\mathbf{M} = g(\sigma_I) * \begin{bmatrix} I_x^2(\mathbf{x}) & I_xI_y(\mathbf{x}) \\ I_xI_y(\mathbf{x}) & I_y^2(\mathbf{x}) \end{bmatrix} \tag{2.1}$$

where $\mathbf{x}$ is the pixel location vector, $I_x(\mathbf{x})$ is the x-gradient at location $\mathbf{x}$, $I_y(\mathbf{x})$ is the y-gradient at location $\mathbf{x}$ and $g(\sigma_I)$ is the gaussian kernel of scale σ_I.

Eigenspace Analysis

A point is located at a corner if its corner response is large. The corner response $\mathbf{R}$ can be computed from matrix $\mathbf{M}$ by the following equation:

$$\begin{aligned}\mathbf{R} &= Det(\mathbf{M}) - K \times Trace(\mathbf{M})^2 \\ &= I_x^2 I_y^2 - (I_xI_y)^2 - K \times (I_x + I_y)^2\end{aligned}$$

where K is an empirical constant ranged from 0.04 to 0.06.

Non-Maximal Suppression

To reduce the amount of corners detected, a corner should not be captured by more than one interest point. This objective can be achieved by non-maximal suppression which removes candidate points that are not the local maxima of **R** within its local neighborhood:

$$\mathbf{R}(\mathbf{x}) > \mathbf{R}(\mathbf{x}_w) \forall \mathbf{x}_w \in W \wedge \mathbf{R}(\mathbf{x}) > \mathit{threshold}$$

where W denotes the 8-neighborhood of the pint x.

2.2.2 DOG Extrema Detector

DOG Extrema Detector is proposed by Lowe [22, 21] to detect SIFT features. It extracts interest points in a cascade filtering approach in which the more expensive operations are applied only at locations that pass all prior tests. The major steps of generating interest point from an image are discussed in the following sections.

Scale-Space Extrema Detection

DOG Extrema detection identifies the locations and scales of the interest point that can be repeatedly detected under different views of the same object. As the interest point can be repeatedly detected, we will call it stable features. Detecting stable features that are invariant to locations is achieved by searching for most of the locations over the image. To extend its invariance to scales, all possible scales of the image are searched instead of one scale only.

The scale space of an image which is defined as a function, $L(x, y, \sigma)$, can be prepared by repeatedly convolving the initial image with a variable-scale Gaussian function $G(x, y, \sigma)$:

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

To efficiently detect stable interest point locations in scale space, Lowe proposed [21] using scale-space extrema in the difference-of-Gaussian function, $D(x, y, \sigma)$, which can be computed from the difference of two nearby scales of smoothed images, $L(x, y, \sigma)$, separated by a multiplicative factor k.

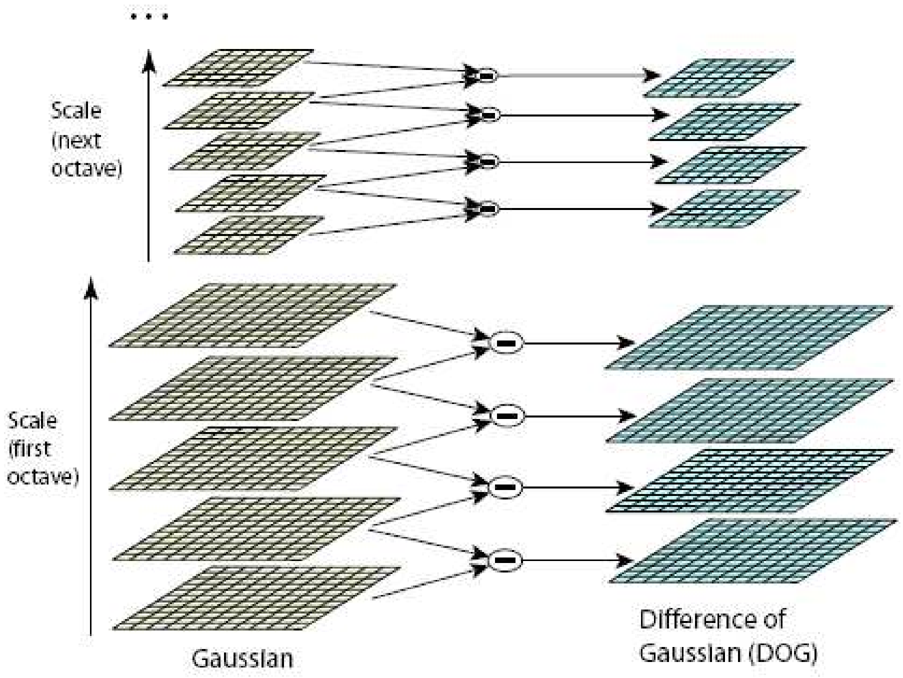


Figure 2.1: A diagram illustrating how differences of gaussian images is prepared from the initial image. The initial image is repeatedly smoothed by Gaussian function, which is shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images, which is shown on the right.

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

The scale space of the input image is prepared in the way illustrated by Figure 2.1. The difference-of-Gaussian function has been proved to be a close approximation to the scale-normalized Laplacian of Gaussian. Therefore, finding extrema in difference-of-Gaussian space is approximately equivalent to finding extrema in Laplacian space. After the scale space has been prepared, each sample point is compared to its eight neighbors in the current image and nine neighbors in the scale above and below in order to detect the extrema of $D(x, y, \sigma)$.

The advantage of searching interest point over a complete range of scales is that both small interest points and large interest points are detected. Small interest points help solving occlusion problem while large interest points contribute to the robustness of the system toward noise and image blur.

Interest Point Localization

The second step is to reject the interest points that have low contrast or are localized along an edge. Low contrast interest points are rejected because they are sensitive to noise. Interest points localized along an edge are also rejected because they in general do not make significant difference with nearby points.

To reject interest points with low contrast, the scale-space function value at each extremum, $D(\hat{x})$, is examined:

$$D(\hat{x}) = D + \frac{1}{2}\frac{\delta D^T}{\delta x}\hat{x}$$

For the experiments done by Lowe in [22], all extrema with a value of $|D(\hat{x})|$ less than 0.03 were discarded.

To reject interest points on edges, Hessian edge detector is applied. The difference-of-Gaussian function, D, will have a large principal curvature across the edge but a small one in the perpendicular direction. Hessian matrix, $\mathbf{H}$, can be computed at the location and scale of the interest point by:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

The derivatives, D_{xx}, D_{xy} and D_{yy}, can be estimated by taking differences of neighboring points around the sampling interest point.

The eigenvalues of $\mathbf{H}$ are proportional to the principal curvatures of D. Thus, the ratio of the two eigenvalues reflects that the interest point is on the edge or not. The solution can be simplified by just checking the following condition:

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(r+1)^2}{r}$$

For the experiments done by Lowe in [22], all extrema having a ratio between the principal curvatures greater than 10 are discarded.

2.2.3 Harris-Laplacian Corner Detector

Mikolajczyk et al. [25] proposed another detector for detecting scale invariant interest points. It is the Harris-Laplacian corner detector. This detector first computes a set of images represented at different levels of resolutions (pyramid) for Harris corner detector. It then select points at which the normalized Laplacian is maximal over scales. Mikolajczyk et al. observed that the amplitude of spatial image derivatives decreases with scale. Thus the derivative function must be normalized according to the scale of observation. They modify the Harris corner detector such that it can be applied over scale-space.

Auto-Correlation Matrix for Scale-Space

The detector analyzes the auto-correlation matrix $\mathbf{M}$ of every location in an image that is computed from normalized image derivatives:

$$\mathbf{M} = \sigma_D^2 g(\sigma_I) * \begin{bmatrix} I_x^2(\mathbf{x}, \sigma_D) & I_x I_y(\mathbf{x}, \sigma_D) \\ I_x I_y(\mathbf{x}, \sigma_D) & I_y^2(\mathbf{x}, \sigma_D) \end{bmatrix} \qquad (2.2)$$

Equation 2.2 differs from equation 2.1 by the differentiation scale σ_D. $I_x(\mathbf{x}, \sigma_D)$ and $I_y(\mathbf{x}, \sigma_D)$ represents the image derivative computed over an image obtained by convolving the full-size image with Gaussian kernels of scale σ_D. The image derivatives are normalized by multiplying with σ_D^2 that is proportional to the scale of the target image.

Scale Selection

After localizing points in 2D space using Harris corner detector, the candidate points are subjected to scale maxima detection. For each level of the scale-space, the detector applies the non-maximal suppression to reduce the amount of candidate points. Then for each of the candidate points found on different levels, it is verified if it is maximum in Laplacian in the scale direction. The Laplacian $\mathbf{F}$ of a point $\mathbf{x}$ is defined by:

$$\mathbf{F}(\mathbf{x}, \sigma_D) = |\sigma_D^2(L_{xx}(\mathbf{x}, \sigma_D) + L_{yy}(\mathbf{x}, \sigma_D))|$$

Candidate point $\mathbf{x}$ at scale σ_{Dn} is maximum in Laplacian in the scale direction if the following condition is satisfied:

$$\mathbf{F}(\mathbf{x}, \sigma_{Dn}) > \mathbf{F}(\mathbf{x}, \sigma_{Dn-1}) \wedge \mathbf{F}(\mathbf{x}, \sigma_{Dn}) > \mathbf{F}(\mathbf{x}, \sigma_{Dn+1})$$

where σ_{Dn-1} is a sampled scale just smaller than σ_{Dn} and σ_{Dn+1} is a sampled scale just larger than σ_{Dn}.

2.2.4 Harris-Affine Covariant Detector

Harris-affine covariant detector is an advance of the Harris-Laplacian detector. This detector can detect the same elliptical regions on images even if the object in the images is taken with significant different viewpoints. this makes feature description later in the recognition process invariant to change of viewpoint. The detected regions are covariant to the affine transformation of object and thus this detector is called affine covariant detector.

Harris-affine covariant detector is based on affine normalization around Harris points. After a set of interest points are detected by Harris-Laplacian detector, iterative estimation of elliptical affine regions around the interest points are carried out. The estimation is done by determining the transformation that transforms the interest region to the one with equal eigenvalues. The transformation can be computed by the square root of the auto-correlation matrix $\mathbf{M}^{1/2}$. Points $\mathbf{x}$ inside the interest region can then be normalized by transformation:

$$\mathbf{x}' = \mathbf{M}^{1/2}\mathbf{x}$$

After projecting every point inside the interest region to a new position, the auto-correlation matrix is computed again and transformation of interest region to the one with equal eigenvalues is carried out again. This process proceeds until the auto-correlation matrix has equal eigenvalues. When all interest regions are normalized, corresponding regions are differed only by a simple rotation. Thus, regions detected from an image are now invariant to the affine transformation.

2.3 Feature Descriptor

Given the interest points detected by the feature detector, the remaining task is to describe them for matching and recognition later. Distribution-based

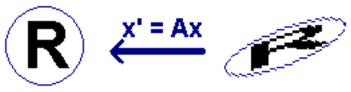


Figure 2.2: This figure shows an example of the elliptical affine region and the normalized region. The transformation matrix $A = M^{1/2}$ projects x to x' such that the eigenvalues of the auto-correlation matrix are equal.

descriptors are shown [15] to be superior to other types of descriptors such as differential descriptors in recognition task. Distribution-based descriptor is a histogram representing in form of a feature vector that captures the distribution of the image context such as pixel intensity, edge point, gradient location and orientation. In this section, five state-of-the-art descriptors are discussed. They are SIFT [21, 22], shape context [3], PCA-SIFT [29], GLOH [24] and GIH descriptors [19]. SIFT descriptor is a 3D histogram of gradient location and orientation direction. Shape context descriptor is a 2D histogram of edge points' locations. Schmid et al. [24] improved shape context to include also the distribution of orientations. PCA-SIFT descriptor is a vector of coefficients of the base image gradient patches obtained by PCA. GLOH descriptor is an extension of SIFT descriptor and is reduced in dimension by PCA. GIH is a geodesic-intensity histogram that is invariant to non-affine deformation.

2.3.1 Scale Invariant Feature Transform (SIFT)

The most important considerations of a feature descriptor are invariance and distinctiveness. SIFT descriptor is a carefully designed representation of image patch that is highly invariant to change in scale, orientation and illumination, and is partially invariant to 3D viewpoint. SIFT descriptor is originally designed to use DOG extrema detector to detect interest points such that the descriptor is invariant to scale change. SIFT descriptor allows feature positions to shift significantly without large changes in the descriptor and thus it can achieve partial invariance to affine distortion and changes in 3D viewpoints. Schmid et al. [24] further enhances its invariance to change in 3D viewpoints by replacing the DOG extrema detector by harris-affine covariant detector. Although the average recall rate is lower, the descriptor showed significant improvement in detecting affine features under large affine distortion.

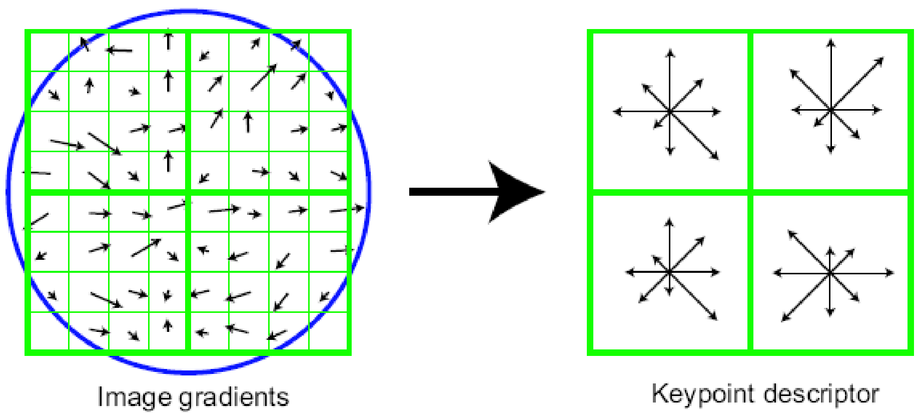


Figure 2.3: Computation of a feature descriptor based on the gradient and orientation of each image sample point in a region around the feature.

Orientation Assignment

For each interest points of each image sample, $L(x, y)$, at a particular scale, the gradient magnitude, $m(x, y)$, and orientation, $\theta(x, y)$, are obtained using pixel differences:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\theta(x,y) = \tan^{-1} \frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}$$

The gradient and orientation information of each interest point can then be used to construct the feature descriptor.

Descriptor Representation

The computation of the feature descriptor is illustrated in Figure 2.3. The approach is to create orientation histograms over 4×4 sample regions around the interest locations. Each histogram contain 8 orientation bins that is the Gaussian-weighted average of the gradient vectors over the corresponding region. For the case illustrated in Figure 2.3, a 32 element feature vector can be obtained for each interest point. Lowe has shown in experiment that a 4×4 array of histogram with 8 orientation bins in each yield the best result. Since the orientation histograms are created over 4×4 regions instead of over every pixel, the descriptor is robust against significant change in gradient position and thus it is partially invariant to change in 3D viewpoint.

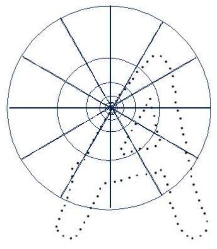


Figure 2.4: Figure illustrating how the bins of shape context is distributed around a given edge point. Belongie et al. [3] use five bins for quantizing distance between the rest of the edge points from the given edge point and 12 bins for quantizing the angle between them.

To make the descriptor further invariant to illumination change, the descriptor is normalized to unit length. This totally cancel the effect of affine change in illumination.

$$I(\mathbf{x}) = aI'(\mathbf{x}) + b \tag{2.3}$$

Equation 2.3 shows how a original pixel's intensity $I'(\mathbf{x})$ at position $\mathbf{x}$ is changed by affine illumination. Assume the constants a and b are the same within a small local region of an image, then the image derivative I_x will not be affected by the inter-reflection light term b.

$$\widehat{I}(\mathbf{x}) = \frac{aI_x(\mathbf{x})}{a\sum_{x\in W} I_x(\mathbf{x})} = \frac{I_x(\mathbf{x})}{\sum_{x\in W} I_x(\mathbf{x})} \tag{2.4}$$

where W is the set of points within the concerned local region. Equation 2.4 showed that the image derivative does not depends on the constant a. Since the SIFT descriptor is created solely using image derivative, it is invariant to affine changes in illumination.

2.3.2 Shape Context

Shape context is a shape descriptor that describes the distribution of the rest of the shape with respect to a given edge points on the shape. It is a histogram of the relative positions of all other edge points in the image. Edge points here refer to a set of points sampled from the shape contours of the target object using edge detector. Shape context uses bins that are uniform in log-polar space to emphasize close-by, local structure as shown in figure 2.4 and 2.5. In the original design of shape context, a histogram h_i is computed by simply counting the number of edge points within a bin:

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

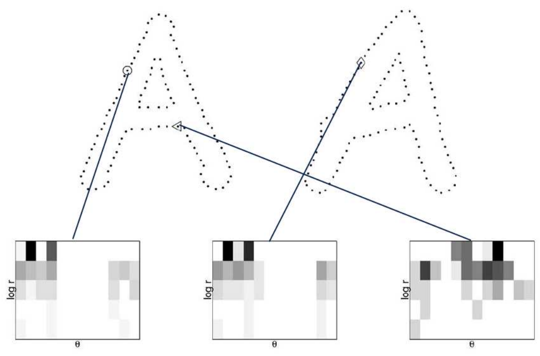


Figure 2.5: Figure showing the shape context histograms of three edge points. Darker bins indicate larger number of edge points are located inside the bins. The first and second histograms are very similar because the edge point they represent are correspondence while the third histogram is very different.

In the modified design by Schmid et al. [15], weight is assigned to the contribution of each point based on its gradient magnitude and orientation of edge points are also captured into the histogram. This makes shape context descriptor very similar to SIFT and GLOH descriptor.

Since shape context is a histogram computed from edge points, it is invariant to change in illumination. To make shape context descriptor invariant to orientation, the feature detector has to help aligning the dominant orientation of the local patch to a canonical direction.

2.3.3 PCA-SIFT

PCA-SIFT descriptor is a vector of coefficients of the base image gradient patches obtained by PCA. It can be created in the following steps:

For each interest point,

1. Locate the 41×41 image patch around the point at the correct scale.
2. Rotate the patch to align its dominant orientation to a canonical direction in the same manner as SIFT.
3. Compute the Image gradients of the patch.
4. Create a vector by concatenating both horizontal and vertical gradient maps.

5. Normalize the vector to unit magnitude to make it invariant to changes in illumination.

6. Project the vector into a pre-computed eigenspace to derive a feature vector. The eigenspace can be pre-computed by applying PCA to the gradient patches in a set of training images.

Although creating PCA-SIFT descriptor is much simpler than SIFT, PCA-SIFT has been shown to have similar accuracy with SIFT in recognition [15] and run a lot faster than SIFT because of its much lower dimension. The success of PCA-SIFT lies in the fact that the patches surrounding the interest points all share some characteristics such as centering at the local extremum in scale-space and orientated to the canonical direction. However, since the dimension of PCA-SIFT is very small (dim = 20), it is worthwhile to evaluate its performance when the database of features increase significantly.

As implied by the steps of creating PCA-SIFT, PCA-SIFT descriptor is invariant to orientation and changes in illumination in the same way as SIFT.

2.3.4 Gradient Location and Orientation Histogram (GLOH)

GLOH is an extension of the SIFT descriptor and is an advance version of PCA-SIFT and shape context. The same as SIFT, GLOH describes the gradient orientations of the image patches. Instead of sampling gradient orientations in a rectangular grid, GLOH samples them in a log-polor location grid like the one used in shape context descriptor. The histogram of each interest point consists of 17 location bins with 16 orientation bins in each. This gives a 272 bin histogram. PCA is then applied to reduce the dimension of GLOH descriptor to 128.

2.3.5 Geodesic-Intensity Histogram (GIH)

GIH is a novel local descriptor that is invariant to deformation based on the fact that the pixel intensity and geodesic distance are invariant to deformation. Geodesic distance is the distance of the shortest path between two points on the embedded surfaces. It is defined by:

$$d = \int_a^b \sqrt{(1-\alpha)^2 x_t^2 + (1-\alpha)^2 y_t^2 + \alpha^2 I_t^2 dt}$$

where a and b represent the coordinates of the two points on the embedded surfaces and the subscripts denote partial derivatives, e.g., $x_t = dx/dt$. Ling proved [19] that the geodesic distance of two points remains unchanged after deformation when $\alpha \longrightarrow \infty$. Geodesic distance for 1-D image is illustrated in figure 2.6. GIH descriptor is created in the following steps:

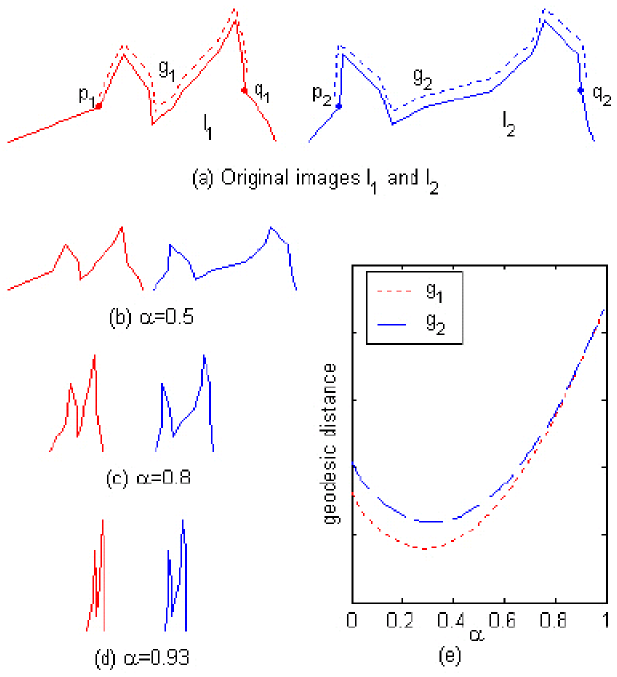


Figure 2.6: Deformation invariance for 1-D images (Figure from [19]).

For each interest point $p_0 = (x_0, y_0)$,

1. Apply fast marching algorithm to compute the points with identical geodesic distances from p_0 at intervals of δ. The aggregate of these points are called level curve.

2. Sample points from each level curve at intervals of δ.

3. Create a 2D intensity-geodesic distance space with intensity and geodesic distance as the two dimensions.

4. Insert all sampled points into the histogram according to its intensity and geodesic distance.

5. Normalize the geodesic distance dimension and then normalize the histogram as a whole.

Ling has made a good attempt to enhance local descriptor to deformation invariance. However, since images are defined on discrete grids, pixels in between two points can merge together to be a few pixels only. In this case, the discrete geodesic distance will vary a lot due to deformation. Refer to figure 2.7.

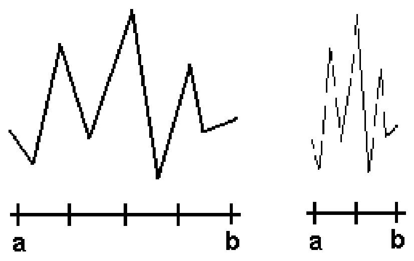


Figure 2.7: Figure illustrating discrete geodesic distance can fail. Due to discrete sampling of image pixels, the geodesic distance between points a and b is large in the left image and small in the deformed image on the right.

2.3.6 Experiment

In this experiment, we aim to compare the performances of the top three local descriptors: SIFT, PCA-SIFT and GLOH. In each experiment, each descriptors will describe both the Harris-affine covariant region and the Harris-Laplacian region. This allows us to compare the performance of Harris-affine covariant detector and Harris-Laplacian detector in matching at the same time. Our experiment evaluates only the accuracy but not the computational time of the local descriptors. At last, we will rank the descriptors based on the experiment we carried out.

Data Set

The data set used in our experiment is obtained from Visual Geometry Group[1]. We have used this data set to evaluate the performance of the four descriptors. Shape context is quite similar to GLOH and thus we evaluated the performance of GLOH only. For each set of images of the same scene, we selected 2 images as the image pair. The images we have used are shown in figure 2.8

[1]http://www.robots.ox.ac.uk/ vgg/research/affine

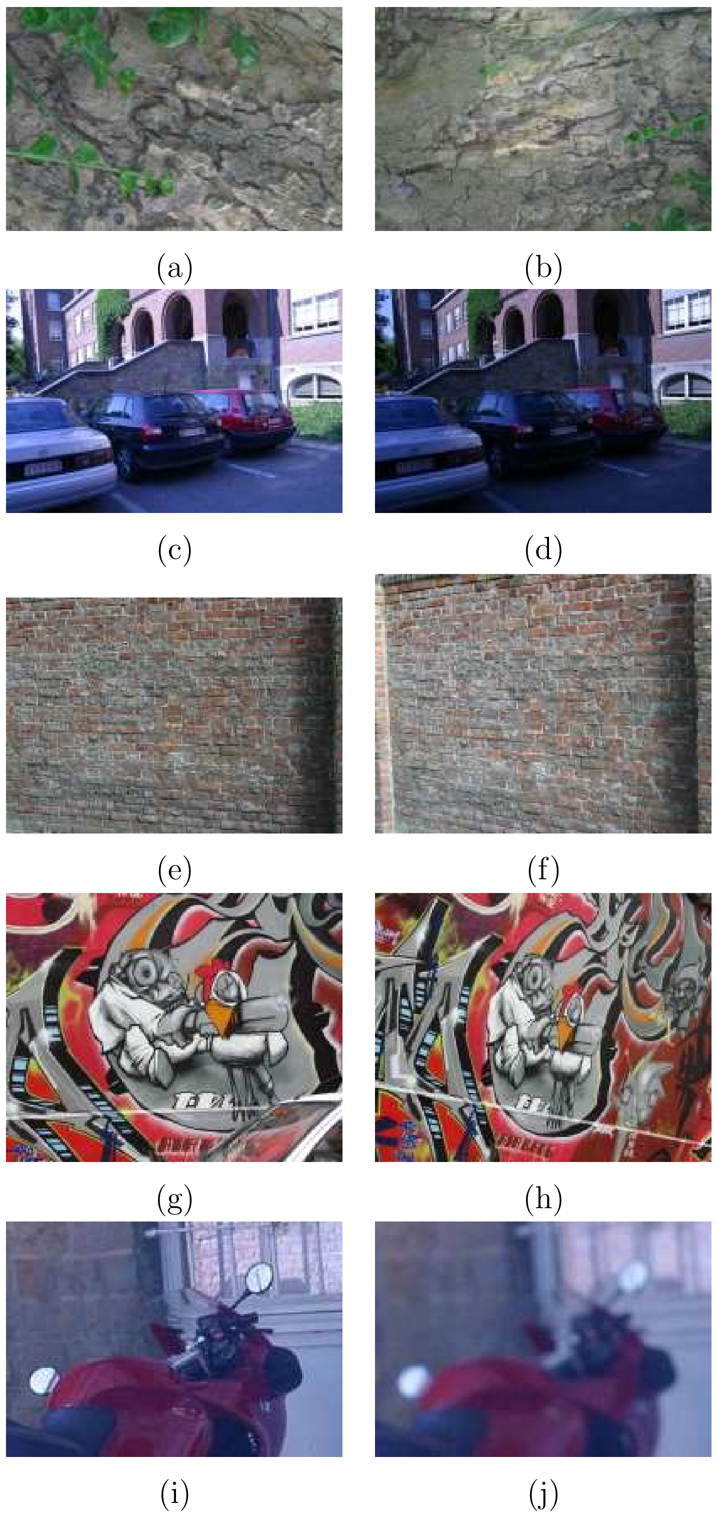


Figure 2.8: (a) & (b) Bark image sequence. (c) & (d) Leuven image sequence. (e) & (f) Wall image sequence. (g) & (h) Graf image sequence. (i) & (j) Bikes image sequence (A small portion).

Evaluation Criterion

We adopted the evaluation criterion used in [29, 15]. It counts the number of correct matches and the number of false matches obtained for an image pair. We want a local descriptor to have large number of correct positives and small number of false positives. Two regions are matched if the Euclidean distance between the two descriptors are below a threshold t. For the elliptical regions detected by Harris-affine covariant detectors, two regions are matched if the overlap error ϵ defined in [26, 15] is less than 0.4. If the match agrees with the ground truth, the match is classified as correct match; otherwise, it is false match. The transformation between each image pair can be described by a homography which can be used as the ground truth. The results are plotted in form of recall versus 1-precision curves. The value of ϵ and t is varied to obtain the curves. Recall is defined as the number of correct matches over the number of possible correct matches in the image pair:

$$recall = \frac{\#correctmatches}{\#correspondences}$$

1-precision is defined as the number of false matches over the sum of the number of matches:

$$1 - precision = \frac{\#falsematches}{\#correctmatches + \#falsematches}$$

Experimental Results

Four common transformations in images are evaluated in this experiment. They are scale change and rotation, illumination change, viewpoint change and image blur. For each transformation, a recall versus 1-precision graph is plotted.

1. **Scale change and rotation.** We used the image pair shown in figure 2.8(a),(b) to evaluate the performance for scale change and rotation. Result is shown in figure 2.9. As observed from the figure, description using Harris-affine covariant regions performs better than using non-affine covariant regions, and SIFT descriptor performs the best.

2. **Illumination change.** We used the image pair shown in figure 2.8(c),(d) to evaluate the performance for illumination change. Result is shown in figure 2.10. These images are obtained by changing the camera setting likes exposure. As observed from the figure, all the descriptors under test are robust to illumination change. The reason is that all

of them uses the same illumination normalization technique. Nevertheless, we observed that SIFT descriptor remains the best among the three descriptors and Harris-Laplacian detector performs better than Harris-affine covariant detector. PCA-SIFT descriptor performs very good at high precision but the recall rate does not increase much when precision is lessen.

3. **Viewpoint change.** We used two image pairs shown in figure 2.8(e),(f) and figure 2.8(g),(h) to evaluate the performance for viewpoint change. Result is shown in figure 2.11 and 2.12. Viewpoint change in the wall image pair is less than that in the graf image pair. As observed from the figures, for the wall image pair, SIFT descriptor based on Harris-Laplacian detector performs the best; for the graf image pair, SIFT descriptor based on Harris-affine covariant detector performs the best. This illustrates Lowe's SIFT descriptor itself is invariant to small amount of viewpoint change and retains the highest distinctiveness. For high amount of viewpoint change, performance of Lowe's SIFT descriptor drops significantly. However, when used with Harris-affine covariant detector, its performance is improved a lot. We observed that Harris-affine covariant detector really help improve the robustness of descriptor. Yet this improvement may be limited only to cases with large viewpoint changes.

4. **Image blur.** We used the image pair shown in figure 2.8(i),(j) to evaluate the performance for image blur. Blur effect is introduced to the image by adjusting the camera focus. Result of the experiment is shown in figure 2.13. Both SIFT and PCA-SIFT descriptor perform well in this image pair. PCA-SIFT, again, performs very good at high precision but SIFT is better at lower precision.

Conclusion

From these experiment, we arrived this conclusion: *SIFT > GLOH > PCA-SIFT*. SIFT descriptor is the best among the three descriptor in most of the cases. SIFT always performs slightly better than GLOH, so GLOH descriptor is only the second best. PCA-SIFT descriptor always performs very good at high precision requirement but not at lower precision so I give it the third rank. This fact does not depend on the types of interest regions.

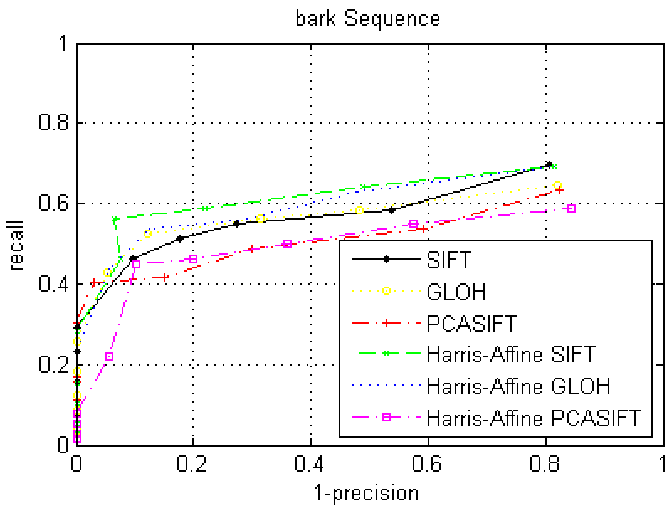


Figure 2.9: Experimental Result for scale change and rotation on bark image sequence.

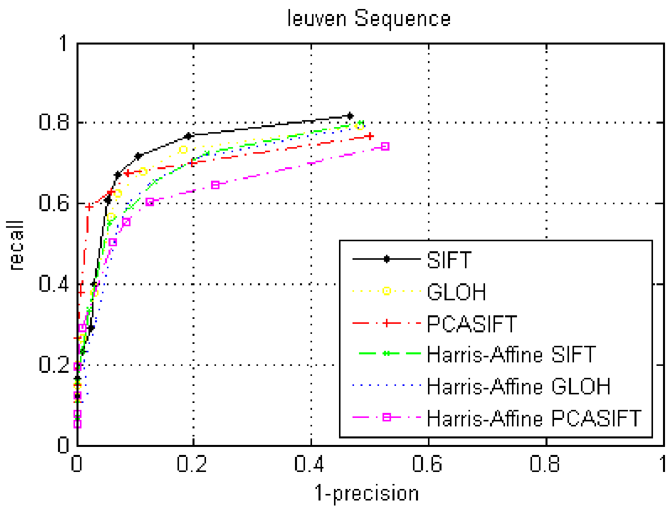


Figure 2.10: Experimental Result for illumination change on leuven image sequence.

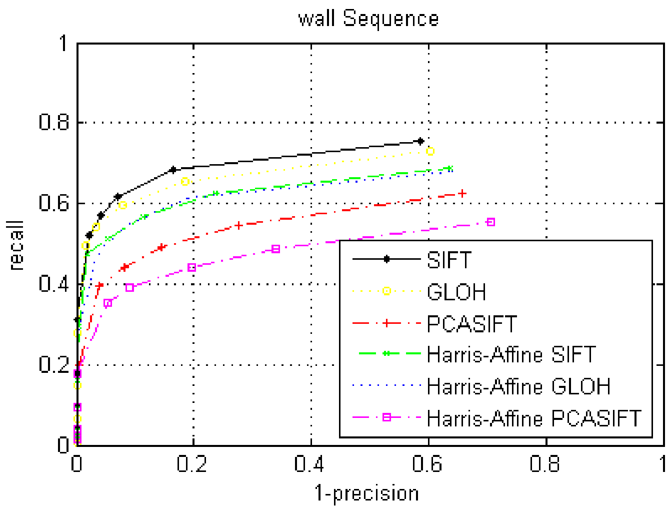


Figure 2.11: Experimental Result for viewpoint change on wall image sequence.

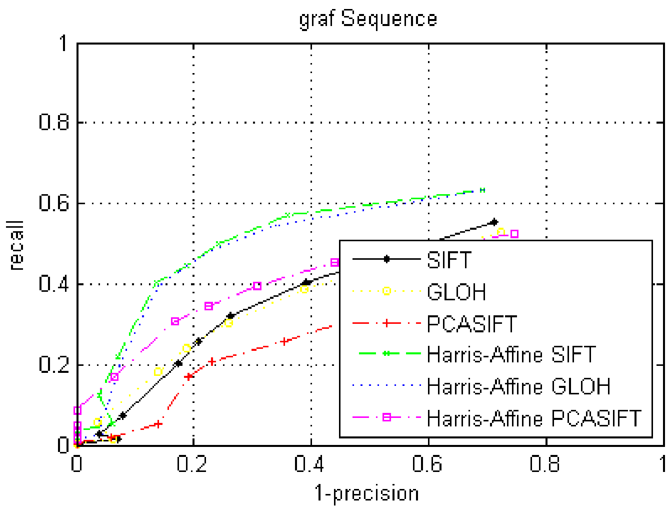


Figure 2.12: Experimental Result for viewpoint change on graf image sequence.

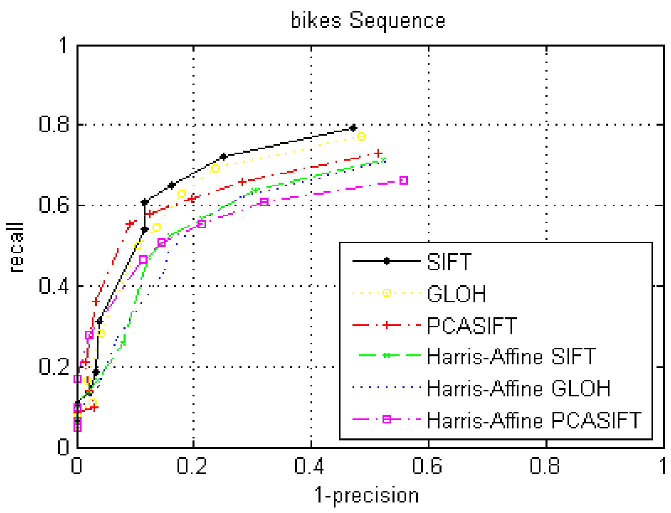


Figure 2.13: Experimental Result for image blur on bikes image sequence.

2.3.7 Descriptor Prototypes

In order to understand the difficulties and considerations of designing a feature descriptor, I have implemented three prototypes of feature descriptors:

Simple Grayscale Patches

I have first implemented a feature descriptor that samples pixels within small 5x5 square window around the detected feature points. This descriptor takes the output of Harris corner detector as feature input.

Hybrid-type Grayscale Patches

This descriptor is designed to be invariant to orientation, illumination change. It is similar to the feature descriptor proposed in [5].

1. **Dominant orientation.** First of all, dominant orientation of the region is found. There are a few different ways to find the dominant orientation. One of the way is to find the average orientation angle within the region, another way is to find the angle that is large in value and at the same time commonly appear within a region. I adopted the method used in SIFT.

2. **Pixel sampling.** Given an oriented interest point, we sample a 7x7 patch of pixels around it, using a spacing of 2 pixels between samples.

3. **Intensity normalization.** After sampling, the descriptor vector is normalized so that the mean is 0 and the standard deviation is 1. This makes the features invariant to affine changes in intensity. The speed of description is fast. The performance is good in bikes, leuven sequences, but the performance is not good in sequences that capture affine transformation.

Histogram-based Descriptor

We create 7 histograms with 10 bins covering the intensity value ranged from 0 to 1. Originated at the detected interest point, we sample pixels around it in a circular manner with radius = 0, 1, ... , 6, in every 30 degree. The sampled pixel may be located at a position that is not an integral value. Then, bilinear interpolation is performed to obtain the target pixel value. The value is added to the corresponding bin of the histogram for that ring according to its intensity value. For each ring, 12 pixels' intensities are added into a histogram. The 7 histograms are then concatenated into a descriptor. Since the directions of pixels around the interest point does not matter, this descriptor is invariant to orientation. The descriptor vector is then convolved with Guassian kernel such that difference between adjacent bins will not be too large and performance can be more stable.

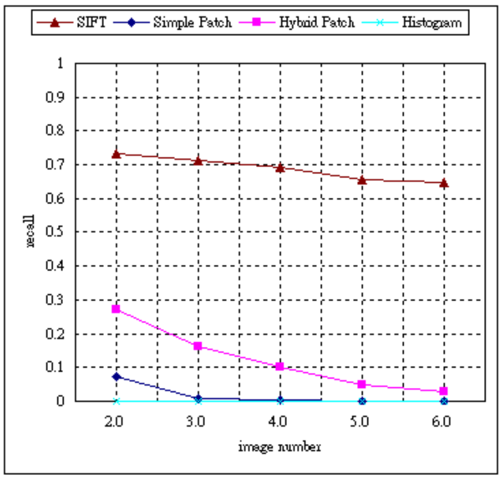
Experimental Results

We compared the performance of the three feature descriptors with SIFT on the same data set from Visual Geometry Group. For each data set, we matched the feature in image 1 to the features in image n (where n = 2, 3, ..., 6) and compute the recall rate. The maximum difference with ground truth is set to be smaller than 3 pixels. The result is shown in figure 2.14. The hybrid-type descriptor has better performance in general among the three descriptors. Yet, its performance is still far below SIFT's.

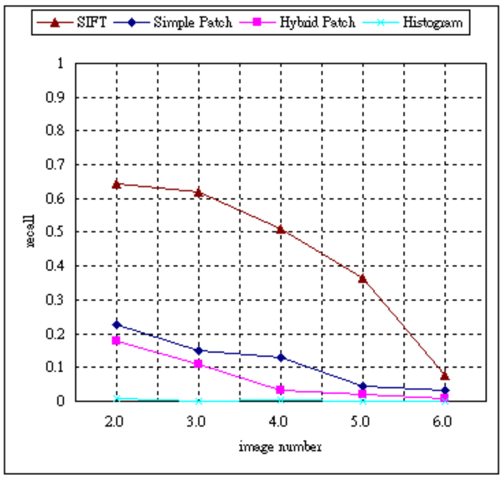
2.4 Color Invariant Local Feature Descriptor

2.4.1 Introduction

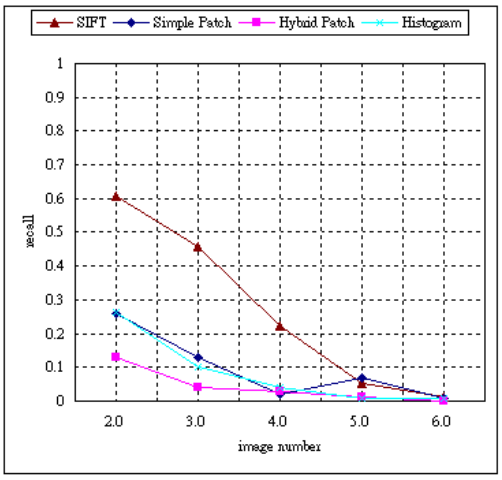
Many computer vision applications tend to ignore color information. From low-level applications like corner detector to high-level applications like object recognition, color information is usually discarded and only luminance information is considered. Reasons of not considering color varies in different applications. While incorporation of color information may not improve



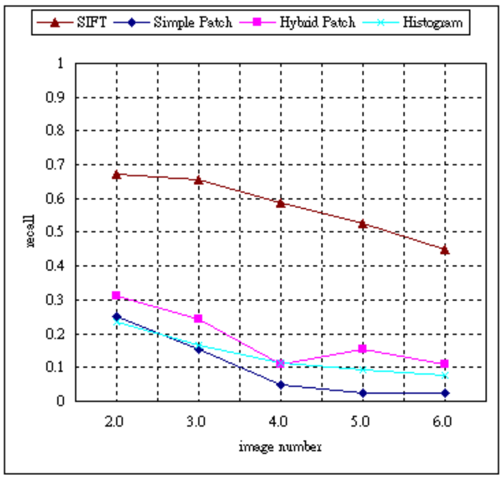
(a)



(b)



(c)



(d)

Figure 2.14: (a) Evaluation for illumination change on Leuven image sequence, (b) Evaluation for affine transformation on wall image sequence, (c) Evaluation for affine transformation on graf image sequence, (d) Evaluation for image blur on bikes image sequence.

the performance of luminance-based techniques much when designed badly, it usually significantly increases the computational complexity. The consequence is that in most applications, color of an image only find its use in obtaining the luminance.

While most of the computer vision tasks can well be accomplished by solely using luminance, color is still a piece of useful information in describing objects. Borrowing the arguments from Swain et al. [33], there are many examples from nature where color is used by animals and plants to entice or warn the others. The manufacturing sector uses color extensively in packaging to market goods. Apart from these examples, there are also a huge support from the advocates of color-based object recognition in using color in recognition.

In our research, we aim to incorporate color into feature-based object recognition. Recently proposed invariant local grayvalue features approach has been proved to be very successful. However, color information is always neglected. There are two main usages of invariant local features: recognizing generic classes of objects and recognizing particular objects. In both cases color information is important. While objects can be classified by shape only, objects can also be classified by colored textures. Colored textures such as wood, marble and metal, are hard to recognize without color. Classes of objects classified by colored textures thus require the incorporation of color into the feature description process. As for recognizing particular objects, color is even more important. Distinctiveness of a feature descriptor is one of its top two requirements. Incorporation of color endues a descriptor the ability to distinguish colored object. It is absolutely a direct upgrade of its distinctiveness. To the extreme, incorporation of color allows the descriptor to distinguish several differently colored objects with the same shape!

Color is always neglected as a recognition cue because many variations can cause significant changes in measured color. Nevertheless, some techniques [11, 35] were proposed to reduce the sensitivity of color information to photometric changes and make color description more robust. Recently, Weijer et al. [35, 34] extended the SIFT local feature descriptor with color information by concatenating a color descriptor to it with a fixed weighting. He has shown by a series of experiments that his combination of color and shape outperforms a pure shape-based approach. It proved color information can enhance the distinctiveness of the local features without losing its robustness.

There are two types of techniques that reduce color's sensitivity to variations: *color constancy* and *color invariant*. We will now discuss the details of these techniques.

2.4.2 Color Constancy

Color constancy refers to the perceptual ability in vision that estimates the chromaticity of an image under canonical illumination based on its chromaticity under unknown illumination. Color correction step is accompanied to maintain a constant perception of color over varying light conditions. Two of the color constancy algorithms are greyworld and white-patch retinex.

Greyworld

Greyworld hypothesis assumes the average reflectance in the world to be grey. The 50% ideal grey under canonical light is given by:

$$RGB_{grey} = \frac{1}{2} RGB_{canonical}$$

where $RGB_{canonical}$ represents the R, G and B channel's color of the canonical illuminant. If the image is taken under canonical light, the average value of R, G and B should be equal to R_{grey}, G_{grey} and B_{grey} correspondingly. For the image taken under unknown illumination, the average of all RGB in the image may not be equal to RGB_{grey}. In this case, we can compare the average with the 50% ideal grey under canonical light and correct the R, G and B color's value of each pixel to the corresponding values under canonical light by:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} \frac{R_{grey}}{R_{average}} & 0 & 0 \\ 0 & \frac{G_{grey}}{G_{average}} & 0 \\ 0 & 0 & \frac{B_{grey}}{B_{average}} \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

where $R_{average}$ is the average of all red values in the image and so as the $G_{average}$ and $B_{average}$.

White-Patch Retinex

Similar to greyworld hypothesis, white-patch retinex assumes the maximum of RGB in an image equals the color of illuminant. Thus the R, G and B color's value of each pixel can be corrected by:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} \frac{R_{canonical}}{R_{max}} & 0 & 0 \\ 0 & \frac{G_{canonical}}{G_{max}} & 0 \\ 0 & 0 & \frac{B_{canonical}}{B_{max}} \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

where R_{max} is the maximum of all red values in the image and so as the G_{max} and B_{max}.

2.4.3 Color Invariant

Color invariant refers to the transformation of RGB that is independent of some variations such as illuminant change, lighting geometry and viewpoint change.

Based on different assumptions on illuminant and surface reflectance, different color invariants are proposed. Color on a surface measured by a camera depends on the diffuse reflection, specular reflection and the ambient light from the environment. We can model the color value at a camera pixel as [10]:

$$C(\mathbf{x}) = g_d(\mathbf{x})\mathbf{d}(\mathbf{x}) + g_s(\mathbf{x})\mathbf{s}(\mathbf{x}) + \mathbf{i}(\mathbf{x}) \tag{2.5}$$

where $\mathbf{C}$ = [R G B]. $\mathbf{d(x)}$ is the image color of an equivalent flat frontal surface viewed under the same light. $g_d(\mathbf{x})$ is a term that varies over space and accounts for the change in brightness due to the orientation of the surface. $\mathbf{s(x)}$ is the image color of the specular reflection from an equivalent flat frontal surface. $g_s(\mathbf{x})$ is a term that varies over space and accounts for the change in the amount of energy specularly reflected. Lastly, $\mathbf{i(x)}$ is a term that accounts for colored inter-reflections, spatial changes in illumination, etc.

Assume we are looking at a single dielectric object but not conductive materials. The inter-reflection term can be ignored. We further assume that light falls evenly on the local feature area. Then color of the source can also be separated from $\mathbf{d(x)}$ such that our model of camera pixel's value becomes:

$$C(\mathbf{x}) = g_d(\mathbf{x})\mathbf{b}(\mathbf{x})\mathbf{e} + g_s(\mathbf{x})\mathbf{e} + \mathbf{i}(\mathbf{x}) \tag{2.6}$$

The body reflection part is now represented by $g_d(\mathbf{x})\mathbf{b}(\mathbf{x})\mathbf{e}$ in which $\mathbf{b}(\mathbf{x})$ is the surface albedo and $g_d(\mathbf{x})$ is the geometrical term that is independent of illumination. The interface reflection is represented by $g_s(\mathbf{x})\mathbf{e}$ in which $\mathbf{e}$ is the image color. The bold face is used to indicate 3-tuple vectors in which the three components in it correspond to R, G and B color channels.

We will introduce several color invariants in the following sections. The title of each section gives the invariance that the color invariant possesses. In the beginning of each section, we will give the assumptions on light source and surface on which the color invariant is based.

Illumination invariant

Assumption:

1. Any surface.

2. No ambient light.

The corresponding pixels, C^1 and C^2, in two image patches of the same scene taken under different illuminants are related by a multiple:

$$\mathbf{C}^1(\mathbf{x}) = a\mathbf{C}^2(\mathbf{x})$$

It is assumed that light falls evenly on the image patches. Thus, the average of pixel values over the two images are also differed by the same multiple. Illumination invariant can then be obtained easily by applying normalization over each color channel:

$$\mathbf{C}'(\mathbf{x}) = \frac{\mathbf{C}(\mathbf{x})}{\overline{\mathbf{C}(\mathbf{x})}}$$

where $\overline{\mathbf{C}(\mathbf{x})}$ is the mean of the color values of pixels within the image patch.

Funt et al. [12] proposed to use color ratio to achieve illumination invariance based on similar assumption in order to make the color histogram proposed by [33] invariant to illumination. Consider two neighbor pixels under the same illumination, the ratio of measured pixel values at $\mathbf{x}_1$ and $\mathbf{x}_2$, yields the ratio of surface albedos:

$$\frac{\mathbf{C}(\mathbf{x}_1)}{\mathbf{C}(\mathbf{x}_2)} = \frac{\mathbf{b}_1}{\mathbf{b}_2}$$

Instead of using the ratio directly, he take logarithms of both sides to turns the ratio into differences:

$$ln(\mathbf{C}(\mathbf{x}_1)) - ln(\mathbf{C}(\mathbf{x}_2)) = ln(\mathbf{b}_1) - ln(\mathbf{b}_2)$$

Applying the Laplacian to the logarithm of the three channels yields a new 3-tuple for every pixel and it is these that are then histogrammed.

Illumination and Inter-reflection invariant

Assumption:

1. Any surface.

If ambient light does exist, the above color invariant can not be used. However, the corresponding derivatives, C_x^1 and C_x^2, in two image patches of the same scene taken under different illuminants are still related by a multiple:

$$\mathbf{C}_x^1(\mathbf{x}) = a\mathbf{C}_x^2(\mathbf{x})$$

Illumination invariant can be obtained by applying normalization over each color channel based on the average image derivative:

$$\mathbf{C}'_x(\mathbf{x}) = \frac{\mathbf{C}_x(\mathbf{x})}{\overline{\mathbf{C}_x(\mathbf{x})}}$$

Lighting Geometry and Viewpoint Invariant

Assumption:

1. Matt surface.
2. No ambient light.

Normalized rgb is invariant to lighting geometry and viewpoint, g_d. Normalized r is computed by:

$$r = \frac{R}{R+G+B} = \frac{g_d b^R e^R}{g_d(b^R e^R + b^G e^G + b^B e^B)}$$

Normalized g is computed by:

$$g = \frac{G}{R+G+B} = \frac{g_d b^G e^G}{g_d(b^R e^R + b^G e^G + b^B e^B)}$$

Normalized b can be computed by $b = 1 - r - g$. If the illuminant must be white illuminant, normalized rgb is invariant to illumination too.

Specularity Invariant

Assumption:

1. Any surface.
2. No ambient light.
3. White illuminant.

Opponent color proposed by Gevers is invariant to specularity:

$$O1 = \frac{1}{\sqrt{2}}(R - G)$$
$$O2 = \frac{1}{\sqrt{6}}(R + G - 2B)$$

Invariant to both lighting geometry, specularity and white illumination can be obtained from the opponent color by:

$$hue = tan^{-1}(\frac{O1}{O2})$$

2.4.4 Conclusion

Color is sensitive to many variations. Color constancy and color invariant are two common approaches in reducing the sensitivity of color to these variations and to make color information a stable and distinctive recognition cue. These techniques may be applied prior to incorporating color information into the latest feature-based recognition approach, it is believed that the performance of feature-based recognition can further be improved through these techniques.

2.5 Feature Matching

Distribution-based descriptor represents local context in form of histogram. Thus, in comparing two descriptors, we can consider the distance measures commonly adopted in comparing two histograms. There are two main types of distance measures: bin-by-bin dissimilarity measures and cross-bin measures. Bin-by-bin dissimilarity measures only compare the contents of corresponding bins of two histogram while cross-bin measures also compare the non-corresponding bins. Recently, a cross-bin measure is proposed by Haibin Ling [20] and it is claimed that it significantly improve the original SIFT feature matching approach. In this sections, we will introduce some of these distance measures, including those adopted in comparing the local descriptors. Then we will introduce some common feature matching techniques. We have assumed there are two histograms: $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_n)$. Histogram Y is one of the histogram stored in a database that histogram X will match with.

2.5.1 Matching Criterions

There are three common criterions in determining whether a feature *matches* with another feature:

1. Similarity Threshold. Two features are matched if the distance between the two features are below a absolute threshold. Each feature may have more than one match under this matching criterion.

2. Nearest Neighbor with threshold. Feature A *matches* with feature B in a database if B is the nearest neighbor of A among other features in the database and the distance between them is lower than a threshold.

3. Nearest Neighbor Distance Ratio. Feature A *matches* with feature B in a database if B is the nearest neighbor of A among other features in

the database and the distance between them is lower than the distance between A and the second nearest neighbor in the database by a multiply constant. This criterion is shown to give higher precision to the above two method in [15].

2.5.2 Distance Measures

Dissimilarity of two features is evaluated by measuring the distance between them.

Minkowski Distance

The Minkowski distance of order p (p-norm distance) is defined as:

$$d_p(X,Y) = (\sum_i |x_i - y_i|^p)^{\frac{1}{p}}$$

2-norm distance is the Euclidean distance. This is the most common distance measures used in comparing local descriptors. SIFT, GLOH, PCA-SIFT and GIH adopt this distance measure.

Histogram Intersection

Histogram intersection is defined as:

$$d(X,Y) = 1 - \frac{\sum_i min(x_i, y_i)}{\sum_i y_i}$$

For 2-D histogram, the distance is related to the area of intersection of two input histograms. The distance is normalized by the area of histogram Y. This distance measure is adopted by the color histogram proposed by Swain et al. [33].

χ^2 Statistic

χ^2 Statistic is defined by:

$$d(X,Y) = \sum_i \frac{(x_i - m_i)^2}{m_i}, m_i = \frac{x_i + y_i}{2}$$

This distance measure is adopted by Shape context descriptor.

Quadratic-form Distance

Quadratic-form distance is a cross-bin distance. Assume X and Y are histograms expressed in form of column vector. It is defined as follow:

$$d(X, Y) = (X - Y)^T A(X - Y)$$

where A is a similarity matrix $A = [a_{ij}]$ where a_{ij} is the similarity between bins i and j which can be defined as:

$$a_{ij} = 1 - \frac{dist(i, j)}{dist_{max}}$$

This distance is commonly used in matching color histograms.

2.5.3 Searching Techniques

Exhaustive Search

Each feature in an image is matched with all features in another image or in the database. This is the most simplest method but it involve a brute-force computation of all distances and its complexity is very high.

k-D Tree

k-D tree is an binary space partition which recursively partitions the feature space at the mean in the dimension with the highest variance. k-D tree is a commonly used data structure for nearest neighbor query and range query. However, the performance of this structure is poor if the dimension of the data entries is high. A modified version called Best-Bin First tree is used by Lowe to match SIFT features.

Locality-Sensitive Hashing

Locality-Sensitive Hashing (LSH) is first proposed by Indyk & Motwani [14] and further improved in [6]. Locality-sensitive hashing is designed to provide a solution for high-dimensional near neighbor problem. It can answer queries in sublinear time with each near neighbor being reported with a fixed probability. This hashing scheme differs with other kinds of hashing in that under this scheme, the probability that two points share the same hash value decrease with the distance between them. This makes it suitable for feature matching purpose in which features similar to the query feature should be returned. Detailed description of this hashing scheme will be given in chapter 4.

2.6 Object Recognition by Image Matching

2.6.1 Introduction

Object recognition is a task of finding 3-dimensional (3D) objects from two-dimensional (2D) images and classifying them into one of the many known object types. It is highly related to image retrieval and image classification. It is an important part of computer vision because it is closely related to the success of many computer vision applications such as robotics, surveillance, registration and manipulation etc. A number of object recognition algorithms and systems have been proposed for a long time toward this problem. Yet, a general and comprehensive solution to this problem has not be made.

Model-Based Object Recognition

In model-based object recognition, a 3D model of the object being recognized is available. The 3D model contains detailed information about the object, including the shape of its structure, the spatial relationship between its parts and its appearance. This 3D model provides prior knowledge to the problem being solved. This knowledge, in principle, can be used to resolve the potential confusion caused by structural complexity and provide tolerance to noisy or missing data. There are two common ways to approach this problem. The first approach involves obtaining 3D information of an object from images and then comparing it with the object models. To obtain 3D information, specialized hardware, such as stereo vision camera, is required to provide the 3D information in some forms. The second approach requires less hardware support but is more difficult. It first obtains the 2D representation of the structure of the object and then compares it with the 2D projections of the generative model.

Using 3D model has both the advantages and the disadvantages. On one side, explicit 3D models provide a framework that allows powerful geometric constraints to be used to achieve good effect. Other model features can be predicted from just a few detected features based on the geometric constraints. On the other side, using models sacrifice its generality. The model schemas severely limit the sort of objects that they can represent and it is quite difficult and time-consuming to obtain the models.

View-Based Object Recognition

In view-based object recognition, 3D model of the object is not available. The only known information is a number of representations of the same object viewed at different angles and distances. The representations of the object can be obtained by taking a series of images of the same object in a panorama fashion. Most of these operate by comparing a 2D, image representation of object appearance against many representations stored in a memory and finding the closest match. Matching of this type of recognition is simpler but the space requirements for representing all the views of the object is large. Again, there are many ways to approach this problem. One of the common way is to extract salient information, such as corner points, edges and region etc, from the image and match to the information obtained from the image database. Another common approach extracts translation, rotation and scale invariant features, such as SIFT, GLOH and RIFT, from each image and compares them to the features in the feature database [21, 22].

View-based object recognition systems have the advantage of greater generality and more easily trainable from visual data. View-based approach is generally a useful technique. However, since matching is done by comparing the entire objects, some methods are more sensitive to background clutter and occlusion. Some methods solve this problem by applying image segmentation on the entire objects so as to divide the image representations into smaller pieces for matching separately. Some other methods avoid using segmentation and solve the problem by employing voting techniques, like Hough transform methods. This technique allows evidence from disconnected parts to be effectively combined.

2.6.2 View-Based Object Recognition

This type of object recognition is also known as appearance-based recognition. There are many object recognition approaches of this type. Correlation-based template matching [18] is one of the approaches that is very commonly used in commercial object recognition system. It is simple to implement and effective for certain engineered environments. However, this type of method is not effective when the illumination of the environment, the object posture and the scale of the object are allowed to change. The result of this method is even more poor when occlusion may occur and image databases is large. An alternative approach is to use color histogram [33] to locate and match image with the model. While this approach is robust to changing of viewpoint and occlusion, it requires good isolation and segmentation of objects from image.

Another approach is to extract features from the image that are salient and match only to those features when searching all locations for matches. Many possible feature types have been proposed, they includes region, shape context [3], line segments and groupings of edges [28] etc. Some of these features are view variant and resolution dependent. They have worked well for certain classes of object, but they are often not detected frequently enough for reliable recognition. Therefore, approach that matches these kinds of features generally has difficulty in dealing with partial visibility and extraneous features. A highly restricted set, such as corners, don't have these problems. They are view invariant, local and highly informative. These feature types can be detected by SUSAN detector [32] and Harris corner detector [13]. Based on these features, image descriptor can be created to increase the matching performance. Schmid & Mohr [31] used the Harris corner detector to automatically detect interest points and create a image descriptor for each point that is invariant to affine transformation and scale. They have also proposed a voting algorithm and semi-local constraints that make retrieval of features from database efficient, and showed that Harris corner detector is highly repeatable. Lowe [21, 22] pointed out that this approach has a major failing at the corner detection part which examines an image at only a single scale. Because of this failing, attempt has to make to match the image descriptors at a large number of scales. Lowe further extended Schmid & Mohr's approach and created a more distinctive image descriptor which is also more stable to changes in affine projection and illumination. Lowe proposed an efficient method to identify stable invariant key points in scale space such that image descriptor for each key point can be calculated only at the same scale as that of the point. Yan Ke [29] improved SIFT descriptor by PCA analysis so that the computation speed of feature is greatly improved. Fergus et al.

Recognition Based on SIFT

SIFT stands for Scale Invariant Feature Transform. It is a novel method proposed recently by Lowe [21, 22] for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object. The extracted features are invariant to image scale and rotation, which means that the same set of features can be detected after the image is scaled and rotated. Image descriptor for each extracted features is carefully designed to provide robust matching across a range of affine distortion, change in 3D viewpoint, addition of noise and change in illumination to the view of an object. The feature descriptors are highly distinctive, which allows a single feature to find its correct match with high probability in a large database of features.

Lowe described an approach to use SIFT features for object recognition. His approach first matches extracted features to a database of features from known objects using a fast nearest-neighbor algorithm. Among all the matches, some matches are mismatch to the wrong objects. Thus the second step is to filter out the wrong matches, this is done by identifying clusters of features belonging to a single object using an efficient hash table implementation of the generalized Hough transform. This is because the probability that several features will match to the same object is by chance much lower than the probability that any individual feature mismatches. Finally each cluster that agree on an object is subjected to a verification process in which the pose of the object is determined. A least-squared estimate is made for an affine approximation to the object pose. In this step, image features consistent with the approximated pose are identified and outliers are discarded. Probability that the cluster of features is belonging to certain object type is then computed.

2.7 Conclusion

In this chapter, the state-of-the-art feature detectors, descriptors and matching techniques are discussed. They are all carefully designed but all consider the grayvalue of an image only. Performance evaluation on feature descriptors in describing features are carried out. The results are presented in the "Experimental Results" section. Some simple feature descriptor prototypes are implemented and their performance are also presented. In the next chapter, techniques related to incorporating color into descriptors will be discussed.

□ **End of chapter.**

Chapter 3

Shape-SIFT

3.1 Introduction

Local invariant feature is widely adopted in object class recognition. R. Fergus et al. [8] proposed to perform Principle Components Analysis (PCA) on the image patch's appearance. K. Mikolajczyk et al. [23] proposed to describe an image patch by SIFT and then perform PCA on the descriptor. Both approaches employ appearance-based descriptors to describe object categories. However, many objects belonging to the same object category are not common in appearance but are common in shape. In other words, the coloring of an object part may vary within the same class. The detachment of coloring information is thus necessary, as it allows us to separate the concern of shape and coloring.

In this paper we propose a new descriptor resembling SIFT that is also invariant to background and object color changes. By object color changes, we mean the colors of different parts of an object may change differently. We observed that if the shape of object remains the same, significant changes in background or object color can cause some SIFT features of an object to change significantly because of the flipping of the image gradient orientations, as illustrated in Fig.3.1. The flipping of gradient orientation affects not only the description process but also the orientation assignment process of SIFT. We propose some methods to handle these problems, and design a new descriptor, Shape-SIFT, for a comprehensive solution. We also present how experiments are carried out and results of the experimental evaluations.

3.2 SHAPE-SIFT Descriptors

SIFT has been the state-of-the-art descriptor with excellent performance [22, 15]. The success of SIFT indicates that image gradient orientations and gradient magnitudes are both distinctive and robust properties of a feature, making them suitable for feature description purpose. SIFT is shown to be very robust to many image transformations. However, it is not robust to changing background and object color. Changing background and object color are commonly encountered in object class recognition tasks. Both of these two transformations occur along the contour of an object or an object's part where the color on either or both sides changes. These are the places where commonly used feature detectors, such as Harris-Laplacian Detector [25] and DOG Extrema Detector [22], frequently sample. According to our preliminary test, SIFT is robust to a small degree of these transformations. This indicates that both the gradient orientations and gradient magnitudes are robust to a small change due to the thresholding and normalization process of SIFT. However, as the change becomes larger, the gradient orientations of the sample points along the contour may flip. This causes significant changes in the features' orientation histograms, leading to a great drop in the matching performance.

3.2.1 Orientation Assignment

Our algorithm takes the output of SIFT as the input. From the keypoint detection and orientation assignment result of SIFT, we recalculate the gradient orientation $\theta(x, y)$ of each image sample point of the following equation:

$$\begin{aligned}\theta(x,y) &= \tan^{-1}\left|\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right| \\ m^2(x,y) &= (L(x+1,y)-L(x-1,y))^2 + \\ &\quad (L(x,y+1)-L(x,y-1))^2\end{aligned} \tag{3.1}$$

Since we take the absolute value on the fraction of pixel intensity difference, $\theta(x, y)$ covers only 180° range of orientations. As flipping of gradient orientations does not affect the values of $\theta(x, y)$, it is insensitive to the change in background and object color. On the other hand, gradient magnitude $m(x, y)$ is calculated in the same way as SIFT.

3.2.2 Canonical Orientation Determination

After each image sample point is assigned to a new gradient orientation, an 18-bin orientation histogram covering 180° is created at each keypoint. Sample points around the keypoint are added to the histogram according to their $\theta(x, y)$. Similar to SIFT, peak orientation bin in the orientation histogram is taken as the canonical orientation of the keypoint. However, due to our new definition of the gradient orientation, the peak orientation covers 180° range of orientations. Thus, we cannot simply base on the peak orientation when determining the canonical orientation of that keypoint. We propose the following approaches to solve this problem.

Approach 1 - Duplication

The simplest method to tackle this problem is to create another keypoint having the same position but with different canonical orientations. The canonical orientations of the two keypoints will be θ_p and $180° + \theta_p$, respectively. Since every keypoint is duplicated, the number of features stored in the database is doubled. This approach solves the problem without lowering the matching performance but it significantly increases the storage requirement, which is not desirable.

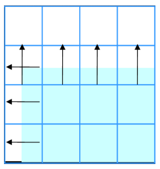
Approach 2 - By the distribution of peak orientation

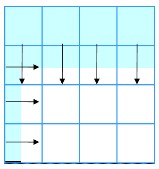
Canonical orientations can be determined using an orientation dependent statistical value over the feature region, as long as the statistical value responds to the two candidate canonical orientations (θ_p and $180° + \theta_p$) differently, and the value is invariant to common image transformations. We propose to use the distribution of peak orientations around the keypoint as the statistical value. The calculation of this value is based on the concept behind the first image moment. First, the coordinates of the descriptor and the gradient orientations of the sample points are rotated by either θ_p or $180° + \theta_p$. Then the feature region is divided into two halves, region A and B. The amount of peak orientations in region A, $Mass_A$ and that in region B, $Mass_B$, are computed by the following equations:

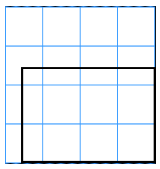
$$Mass_S = \sum_{x,y \in S} [w(x, y) \times m_p(x, y)] \; where \; S \in \{A, B\}$$

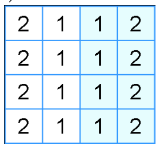
where $m_p(x, y)$ is the gradient magnitudes of the peak orientations and $w(x, y)$ is the weighting function for the gradient magnitudes. The canonical orientation is then determined by the following rules: (1) If $Mass_A > k \times Mass_B$, then the canonical orientation is θ_p, (2) else if $Mass_B > k \times Mass_A$, then it is $180° + \theta_p$. (3) Otherwise, duplicate the keypoint such that each keypoint takes one of the two candidate canonical orientations. If $Mass_A$ and $Mass_B$ are too close to each other, a significant transformation in feature may come up with a totally different canonical orientation, leading to

different feature representations and incorrect matching results. Thus, the difference between $Mass_A$ and $Mass_B$ has to be large enough. This is ensured by an empirical threshold `k`. There are many different ways to divide and weight the region, four of which show good performance in experiment:
(1) *Right-Left Halving Scheme* divides the feature region *vertically* around the center. Each 16×16 feature region is divided into 4 x 4 subregions. The weight of each subregion is represented by the weighting function $w(x, y)$ of the magnitude $m_p(x, y)$ of the peak orientation. The weighting function of this scheme and the two schemes below are shown in Fig.3.1. As pixels near the dividing line may easily fall into the other side and add up to its mass, these pixels get lower weights.
(2) *Diagonal Halving Scheme* divides the region *diagonally.*
(3) *Up-Down Halving Scheme* divides the region *horizontally.*
(4) We further suggest *Hybrid Halving Scheme* which combines the above three schemes. The $Mass_A$ and $Mass_B$ of this scheme is defined as the multiple of the masses of the above three schemes, $Mass_A$ and $Mass_B$ respectively. The hybrid scheme can further boosts the filtering performance.


(a) A Coloring

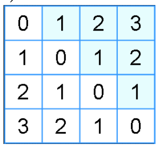

(b) B Coloring


(c) Common contour



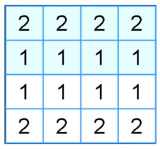
2	1	1	2
2	1	1	2
2	1	1	2
2	1	1	2

(d) R-L Halving



0	1	2	3
1	0	1	2
2	1	0	1
3	2	1	0

(e) Diagonal Halving



2	2	2	2
1	1	1	1
1	1	1	1
2	2	2	2

(f) U-D Halving

Figure 3.1: (a-c) A simple illustration of different coloring to image gradient orientations. (d-e) Different halving and weighting schemes. The shaded halve is region A while the unshaded halve is region B.

To evaluate the performance of these halving schemes and to find the best threshold `k` for each of them, we conduct assessment by two ratios, filtering ratio and matching ratio:

$$FilteringRatio = 1 - \frac{\#\ duplicated\ features}{Original\ \#\ features}$$

$$MatchingRatio = \frac{\#\ correctly\ matched\ features}{Max.\ \#\ correctly\ matched\ features}$$

where the "max. # correctly matched features" is the maximum overall halving schemes in many different choices of the threshold `k`. We prefer both the filtering ratio and the matching ratio to be close to 1. That is, fewer features are to be duplicated and most determined canonical orientations are stable. Fig.3.2 shows the result used to examine the the effect of varying the threshold `k` to filtering and matching ratios. The assessment is performed on data sets from VGG [1] that capture common image transformations. As indicated in Fig.3.2, diagonal halving scheme has a higher filtering ratio at any threshold `k` which achieves nearly the best matching ratio at $k = 1.3$. Thus, we adopt the diagonal halving and weighting scheme in creating SSIFT descriptor.

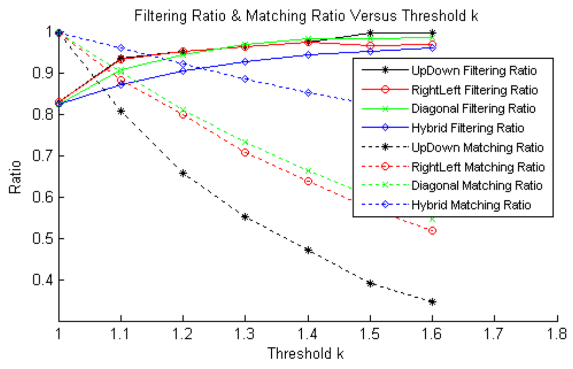


Figure 3.2: Find the best halving scheme and threshold k.

3.2.3 Separation and Reintegration of Shape and Color

After canonical orientation is determined, the coordinates of the descriptor and the gradient orientation of each sample point are rotated according to the canonical orientation. Then, 4-bin orientation histograms [22] covering 180° are built. We call the descriptor built from these histograms SSIFT-64. Since our descriptor is more invariant to change in background and object color than SIFT, it is inevitably a bit less distinctive than SIFT in some cases. To compensate the drop in distinctiveness, another 4-bin orientation histogram that captures the coloring information are built in each subregion and is appended to the descriptor. A matching mechanism is designed to retain the performance of our descriptor on changing background and object color while boosting the performance on other cases. These 4-bin orientation histograms contain north, east, south and west orientation bins, covering 360° range of orientations. The gradient orientation $\theta(x, y)$of each sample point covers the 360° range of orientations and is calculated by Equation (3.1)

[1]http://www.robots.ox.ac.uk/ vgg/research/affine/index.html

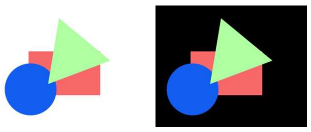
without taking the absolute value before arctan. As the SSIFT descriptor consists of 64 coloring insensitive elements and 64 coloring sensitive elements, we call this descriptor SSIFT-128. Matching SSIFT-64 can be performed by exhaustive searching in the database for the descriptor with the smallest Euclidean distance ratio [22]. This is the feature matching technique we adopted in performance evaluation. The matching mechanism for SSIFT-128 descriptor is as follow: we find the best match $match_{64}$ using the first 64 elements of SSIFT-128 and calculate the distance ratio dr_{64}. Then we refine the match using also the last 64 elements and calculate the distance ratio dr_{128}. We further compare the value of dr_{64} and dr_{128}. If dr_{64} is smaller, then we take the $match_{64}$ as the best match of the descriptor. Otherwise, we take $match_{128}$. This matching mechanism makes the overall performance of SSIFT-128 better than SSIFT-64, as shown in the experimental result section.

3.3 Performance Evaluation

We evaluate the performance of our descriptor on synthetic images and real images with background and object color changes and with different geometric and photometric transformations. Sample images of our data sets are shown in Fig.3.3. We evaluate the performance of our local descriptor with other descriptors on a keypoint matching problem using the same evaluation criterion in [29, 15]. Our evaluation criterion differs from [15] in that: (1) we measure Euclidean distance instead of overlap error when determining the closeness of two descriptors because we do not use affine-convariant detector in the experiments, (2) matching is determined by thresholding distance ratio instead of thresholding distance with the nearest neighbors since distance ratio criterion is shown to yield better results, (3) we do not allow a feature to be repeatedly matched, (4) we also test the original SIFT executable provided by D. Lowe [2], (5) more than ten thousands of different features are added as distracters. Due to the above differences, our experimental results are different from those in [15].

Table 3.1 and Fig.3.4 show the performance of several descriptors on our data sets. We compare the performance of different variants of our descriptor, SIFT implemented by D. Lowe, and SIFT, GLOH and shape context implemented by Mikolajczyk et al. [15]. The last three descriptors use Harris-Laplacian detector while others use DOG extrema detector. This should give advantages to the last three descriptors as Harris-Laplacian detector indicated better repeatability in [25]. However, our results show that they do

[2]http://www.cs.ubc.ca/ lowe/keypoints/

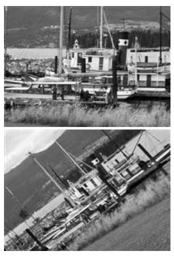

(a) Polygons

(b) PSP

(c) dragon

(e) basmati

(f) bikes


(g) boat

(h) graf

(i) leuven

Figure 3.3: Example images of the data sets that we use. (a) shows two synthetic images that are created by changing the background color of three colored polygons. They are used to evaluate the background color change. (b) shows a sample pair of real images with coloring difference. (c) shows two real sample images that capture background color change. (d) shows one original basmati box [22] and a scene containing another box with coloring difference produced by inverting the intensity of a real image. (f-i) shows sample images with different image transformations from VGG used in [15].

not necessarily perform better than SIFT and our descriptor in many cases.

Fig.3.4(a-c) shows that SSIFT is much more invariant to changes in background color than other descriptors. Fig.3.4(a-d) shows that SSIFT has similar performance with the state-of-the-art descriptor, SIFT, on images with scale, rotation, viewpoint and illumination changes. The performance curves of SSIFT-128 always lie in between that of SIFT and SSIFT. However, the high recall rate of SSIFT-128 on images with background's and part's color changes makes it a promising scheme for the object recognition task.

Table 3.1: Experimental Results

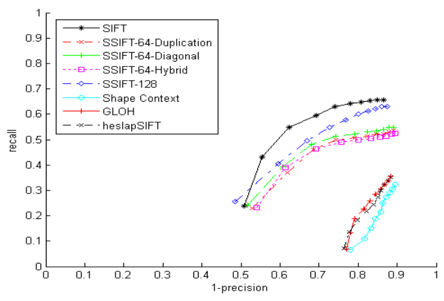
Data Set	Primitives		PSP		Dragon	
Descriptor	recall	1-prec.	recall	1-prec.	recall	1-prec.
SSIFT-128	0.58	0.55	0.29	0.95	0.89	0.35
SIFT [22]	0.16	0.87	0.00	1.00	0.83	0.38
GLOH [15]	0.11	0.92	0.05	0.99	0.53	0.73
Shape Context [15]	0.10	0.93	0.05	0.99	0.52	0.74

Data Set	Basmati		Bikes	
Descriptor	recall	1-prec.	recall	1-prec.
SSIFT-128	0.34	0.88	0.88	0.52
SIFT [22]	0.00	1.00	0.90	0.47
GLOH [15]	0.00	1.00	0.70	0.52
Shape Context [15]	0.00	1.00	0.66	0.55

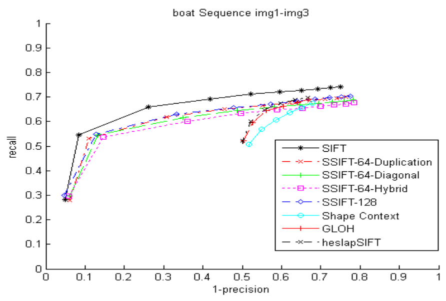
3.4 Conclusion

This paper introduces an alternative to SIFT to build orientation histograms for feature descriptor. Our descriptor, SSIFT, is shown to be much more invariant to background and object color changes than any other descriptors we tested in the experiments. We propose a new canonical orientation determination process to ensure a consistent representation of each feature. The process is shown to be effective in finding a canonical orientation while keeping the adverse effect to feature matching small. Currently, we are trying an alternative way to extend SSIFT-64 to SSIFT-128. Instead of extending SSIFT-64 with more orientation histograms, we employ color histograms which would be a better complement to SSIFT-64.

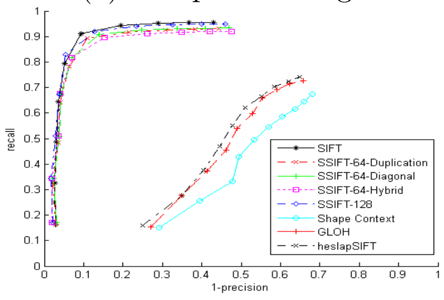
□ **End of chapter.**



(a) viewpoint change



(b) rotation and scale change



(c) illumination change

Figure 3.4: (a-c) Target images have undergone the labeled transformations. They correspond to the images in Fig.3.3(g-i).

Chapter 4

Image Retrieval System for IND Detection

4.1 Introduction

4.1.1 Motivation

We introduce a content-based image retrieval system that is useful in identifying near-duplicate images in a database. Image near-duplicate (IND) detection has been an important application in recent years. Because of the widespread of the Internet, more and more people publish images on the Web for many purposes. People may publish their own images or publish the images bought from photo agencies. However, some people illegally copy the images from the others without acknowledging the owners of the images. This undeniably affects the businesses of photo publishing agencies. Since people always manipulate the pirate images before publishing, the pirate image and the near-duplicate images cannot simply be detected by digital watermarking method. On the contrary, image matching using invariant local feature approaches can easily tackle this problem.

Other than detecting pirate images, our IND detection system can be used to remove duplicated result images returned from a content-based image retrieval system (CBIRS). For each user's query, only a fixed number of images are presented to the users. If many duplicated images exist in the presented results, the user cannot get much information from this fixed number of images. Removing duplicated images using our IND detection system is thus a useful process in improving the quality of CBIRS.

Another possible use of IND detection system is in finding similar web pages on the Internet. Since similar web pages usually have similar images, one can use our IND detection system to detect similar images from two web pages and use it as a cue to evaluate the similarity between the two web pages.

4.1.2 Related Work

Different definitions in near-duplicate image exist in the academia. Yan Ke [16] and Berrani et al. [4] define near-duplicates as images altered with common manual image manipulation process such as changing contrast, saturation, resizing, cropping, framing, jpeg-compression, etc. Images of a scene taken under different environments or camera conditions are not regarded as near-duplicate image at all. On the other hand, Zhang [36] defines near-duplicates as images that are close to exact duplicate but with variations due to content changes, camera parameter changes, and digitization conditions. Both definitions are reasonable but they are suitable for different applications. The former definition is more suitable for copyright protection and CBIR search refinement purposes while the latter definition more is suitable for finding similar pages purpose, for example.

Most of the previous works adopt traditional CBIR approaches in building the IND detection system. Since these approaches generally adopt global features of image as the key to search for near-duplicates, their performances can suffer when significant cropping, resizing or framing is applied on the near-duplicates. Moreover, systems using global features usually lack a self-verification process, likes the geometric verification in [16]. Thus, they tend to have many false positives. Instead of using global features, some IND detection systems build parts-based representation of images using invariant local features were proposed. Berrani [4] proposed a IND detection system employing local differential descriptors and approximate similarity search. Yan Ke [16] proposed to use PCA-SIFT invariant local feature descriptors and Locality-Sensitive Hashing (LSH). The matched features are filtered using geometric verification such that the precision rate is significantly improved. According to the recent performance evaluation done by Mikolajczyk et al. [15], SIFT-based descriptors [22] outperforms the differential descriptor in image matching problem. There also exists some other SIFT-based descriptors, such as SIFT and GLOH, that are better than PCA-SIFT in term of both the recall and precision rate. Therefore, their IND detection systems can be further improved by using more powerful feature descriptors. Locality-sensitive hashing has been proved [16] to be effective in finding near neighbors both in accuracy and speed. However, the LSH algorithm employed by Yan Ke assumes L1 (Manhattan) distance in the analysis of near neighbors which is not as effective as L2 distance, as shown in [16].

4.1.3 Objective

We aim our IND detection system at copyright protection and CBIR search refinement purpose. Therefore, the former definition of near-duplicates is more suitable. That is, images with variations due to camera parameter changes and content changes are not counted as near-duplicates. Our system is a variant of content-based image retrieval system in which the desired images to be retrieved are originated from a source copy which is the same as that of the query image. Although the desired images and the query image share the same source, they are most likely different by some common image transformations due to the manual image manipulation process.

To make our system more practical, we chase for high recall, high precision, and high speed. To achieve this, we employ SIFT feature descriptor which perform the best in our performance evaluation presented in Section 2.3. We would also try the SURF feature descriptor since it is shown to have comparable performance with SIFT but has smaller number of dimensions and thus it is fast to compute and search. To improve the speed of searching, we employ locality-sensitive hashing (LSH) [6] which is a special kind of hashing algorithm first proposed by Indyk & Motwani. It is designed for solving the approximate/exact near neighbor search in high dimensional Euclidean space. The implementation of this algorithm is released to public in [1].

4.1.4 Contribution

The first contribution of this paper to IND detection is the merging of the state-of-the-art SIFT feature descriptor with fast LSH retrieval method that make our IND detection system accurate and practical. The second contribution of this paper is the introduction of a new verification process, called orientation verification, on the matched feature such that the recall and precision can be further improved. The third contribution of this paper is the introduction of a new distance metric, called K-NNRatio, that integrates K Nearest Neighbor algorithm with distance ratio. It improves the average recall and precision rate significantly.

Although our system is targeted at IND detection, it can easily be modified to suit for other CBIR applications. The major component to be modified is the local feature descriptor. Other parts of the system likes the feature searching part, the matched result verification part and the image voting part can be kept unchanged. Thus our contribution is not limited to the IND detection application but to any CBIR application.

4.2 Database Construction

Our system consists of two main phases: the database construction phase and database query phase. In the database construction phase, we process each image in the image collection and extract a set of invariant local features from it. We build the index with all the extracted features using LSH algorithm and prepare it for the database query. In the database query phase, user can issue a query to find the near-duplicates of the submitted query image. From the query image, we again extract a set of invariant local features and issue queries to the pre-built LSH hash table for each extracted feature. The index to the hash table is the extracted feature itself. The returned results of all query features are verified and the most probably images are determined and are returned to the user. Detail description of the database construction phase will be given in this section and the database query phase in the next section, Section 4.3.

4.2.1 Image Representations

Scale-invariant feature transform (SIFT) feature descriptors are very suitable for tackling IND detection problem. The image manipulations commonly applied on near-duplicate images include *changing illumination, contrast, coloring, saturation, resizing, cropping, framing, affine warping, and jpeg-compression.* SIFT descriptors are invariant to all of these transformations but they can, at same time, maintain high distinctiveness. Firstly, the descriptors are normalized such that it is invariant to illumination change and contrast. Secondly, the descriptors are built using the grayvalues of color images and thus it is invariant to coloring and saturation. Thirdly, the descriptors are computed on many different scales of each image such that there are always some features common to two images with different scales only. Thus the descriptors are invariant to resizing. Fourthly, the descriptors are local in nature which means some local changes to the image, like cropping and framing, have little adverse effect to the IND system, as long as the number of features sampled in each image is large enough. Fifthly, the descriptors are orientation histograms over 4×4 sample feature regions. They are less sensitive to significant shift in gradient positions and thus they are invariant to affine warping. Finally, Gaussian filters of different widths are applied on the images before SIFT descriptors operate on them. Thus the effect of the changes in the content of compressed images due to jpeg-compression are reduced. Because of the above reasons, we adopt the powerful SIFT descriptor as the image representation of each image.

Despite of these advantages, there is one strong reason why invariant local descriptor is not desirable to practical system. The reason is the size of local feature database is huge. Since each image can generate hundreds of SIFT features and each feature is a 128 bytes long vector, the size of a feature database of just thousands of images is already huge. Searching for desired features in such database using simple exhaustive search algorithm is not so practical since it takes too long to run. However, a recently proposed feature indexing method, LSH, solve this problem. By building an index of the feature database, the searching process can be performed extremely fast with acceptable accuracy. The detail of index building is to be discussed next.

4.2.2 Index Construction

Among many previously proposed indexing algorithms, hashing is the fastest algorithm to lookup a database. The time complexity of database lookup of a hashing algorithm is $O(1)$ on average, which means a database lookup usually takes constant amount of time independent of the size of the database. Because of this attractive feature, hashing algorithm is commonly employed in indexing large-scale database. However, simple hashing algorithms do not satisfy our requirements. Different from the normal use of hashing, our query key is a high-dimensional vector of real numbers that is usually not an identical copy of any entry in the hash table. Moreover, the desired keys are not limited to an identical copy of itself but all the entries that are close to the query vector in Euclidean space l_2. The hashing algorithms that satisfy our requirements are a recently proposed technique called locality-sensitive hashing (LSH). It is first proposed by Indyk & Motwani [14] and further improved in [6]. Locality-sensitive hashing scheme can answer queries in sublinear time with each near neighbor being reported with a fixed probability. What is special with this type of hashing scheme is that it is locality-sensitive. By locality-sensitive, it means the probability that two points share the same hash value decrease with the distance between them. As for the hashing scheme we employed, the distance is Euclidean distance. Since points close to each other in Euclidean space share the same hash value, we can find the near neighbor of the query point by checking the collided hash buckets.

Advantages

Locality-sensitive hashing compromises speed with accuracy. It only ensure reporting every near neighbor of the query point with a certain probability. However, it is sufficient to our system. It is because an image usually contain at least hundreds of local features. Even if a large proportion of the near neighbors are missed, there are always enough near neighbors reported and contributed to voting the correct image hypothesis.

We have employed the E^2LSH(Exact Euclidean LSH) package [1] to build the LSH index of our feature database. This makes our LSH index conceptually different from that employed by Yan Ke [16]. The LSH algorithm employed by Yan Ke reports c-approximate R-near neighbors problem only while that employed by us reports exact R-near neighbors problem. The definitions of these two types of near neighbor problems are described below:

The c-approximate R-near neighbor problem formulates that if there exist a point p in the set of points P in the database that is at distance at most R from the query point q (i.e., satisfying $||p - q|| \leq R$), any point within the distance of at most cR from the query point (i.e., satisfying $||p - q|| \leq cR$) has to be reported.

The exact R-near neighbor problem formulates that each point p satisfying $||p - q|| \leq R$ has to be reported with a certain probability.

By using the exact R-near neighbor solution, we can ensure that the K nearest neighbors of any query point are found at a certain probability. Of course, if we demand a higher probability, the speed of the LSH algorithm will be slower. As for our system, we can obtain high recall and precision even when a low probability is set.

Scalability Problem

The major problem of E^2LSH is scalability. Although E^2LSH can answer the query in fast speed, it stores all the data points and the R-near neighbor data structures in the main memory. Thus, the size of the database it can index at a time is limited by the amount of free main memory space in the computer. If the IND detection is to be applied on a database once only in a batch mode fashion, the author of E^2LSH - Alexandr Andoni - suggests the following approach to solve the scalability problem:

1. Divide the dataset into several parts.
2. For each of the parts, run E^2LSH to build the index and run all queries on that part.
3. Collect the results from all the parts together to create the total results for the queries.

The hash index of each part will be created once altogether and thus there is no significant difference in speed between creating the index for one huge database and creating the divided databases one by one. However, to achieve this, one would have to obtain **all** the queries beforehand. This is not suitable for online applications in which users can submit query at any time. Nevertheless, this method is suitable for our copyright detection application. As for the CBIR refinement application, we have to adopt other method. Currently, to tackle this problem, we manually set the parameter of E^2LSH such that less memory is required. Moreover, we increase the swap space of the computer so that more memory is available. This is not a complete solution. Therefore, we are going to try a method that make our system scalable.

Solving K-NN

The K-nearest neighbor problem can be solved using E^2LSH. To find the K-nearest neighbor, an effective method is to create several R-near neighbor data structures with $R = \{R_1, R_2, ..., R_t\}$, where R_t is the threshold distance from the query point to its near neighbor. The query can be started with the near neighbor data structure with the smallest radius, say R_1, and continue with the data structure with larger radius until K-nearest neighbors are found. However, this method costs too much memory space to store the data structures and is thus not desirable. Instead, we create one near neighbor data structure with moderate radius only. We identify the K-nearest neighbor according to distance between the query point and the query result points obtained from data structure.

4.2.3 Keypoint and Image Lookup Tables

For the sake of reducing the main memory usage, we do not store the details of the keypoints and their corresponding images in the hash table. We store the details of keypoints and images in separate files on disk. In the keypoint lookup table, the details of each keypoint are stored in one line in plain text according to the following format:

Image ID	x-coordinate	y-coordinate	Scale	Orientation	45 chars

The keypoints are stored such that the indexes of keypoints in the LSH hash table are equal to the line numbers of the corresponding keypoints. Thus, as we retrieve a keypoint represented by a keypoint index from the LSH hash table, we can obtain the detail of that keypoint by looking up the line in the keypoint description file with the line number equals the keypoint index. Since each line is of equal width of 45 characters, the memory address of a specific line can easily be calculated by this formula: $45 * index$.

In the keypoint lookup table, the image associated with each keypoint is represented by "Image ID". The ID is actually an index to the image lookup table. The details of each image are stored in one line in plain text according to the following format:

Image ID	# keypoints	Length of File Name	File Name	
				86 chars

4.3 Database Query

As the index is built in memory, user can issue query to the database. From the query image, we extract out the SIFT features and submit the features to E^2LSH one by one. The keypoints that are sufficiently close to the query keypoint in Euclidean space will be returned. However, not all returned keypoints are regarded as the matches. They are determined by the following matching strategies.

4.3.1 Matching Strategies

Since a query image may have several near-duplicate images in the database, a query keypoint of the query image can have several possible correct matches. Therefore, we should take a group of keypoints instead of a taking a single keypoint as the candidate matches of a query keypoint. There are two commonly used matching strategies that are suitable for our requirements. They are threshold based matching and K-NN matching. We have also proposed a new matching strategies, called K-NNRatio matching, which is a non-trivial integrate of the K-NN matching the and distance ratio matching. E^2LSH supports retrieving keypoint within a threshold radius to the query keypoint. Thus the keypoints returned from E^2LSH are already the candidate matches under the threshold matching strategy. To identify the K nearest neighbor of each query keypoint, we can employ the method discussed in the previous section, Section 4.2.2. The candidate matches under the K-NNRatio matching strategy are actually the same as those under the K-NN matching strategy, but they are additionally assigned a weight specifying its importance in deciding the matching images of the query image. For example, a weight of two assigned to a keypoint A mean there are effectively two keypoint A in the candidate match list. A weight close to zero mean that the keypoint is hardly regarded as a candidate match and we may just remove it from the list of candidate matches.

Threshold Based Matching

Under the threshold based matching strategy, a query keypoint matches with a keypoint in the database if the distance between them is below a threshold R. We have adopted Euclidean distance (L_2) metric for SIFT feature descriptors in our experiment.

It is actually very hard to design a fixed threshold for this strategy. It is because the image transformations applied on different images are variant. However, under different transformations, the average distances between the query points and their matches are usually different. Therefore, a threshold suitable for certain transformation may not be suitable for the others. If we set the threshold too tight, too few matches will be obtained. We may not be able to determine the matching images based on a small amount of candidate matches and thus the recall rate will be low. On the contrary, if we set the threshold too loose, too many matches will be obtained. The matching images will be seem like being chosen by random and thus the precision rate will be low. To overcome these problems, we can set the threshold to a large value and rely on the orientation verification and RANSAC affine transformation verification processes to filter out, hopefully, all of the false matches. However, certainly, this will significantly reduce the speed of the system and not all false matches can be removed by these two processes if the original amount of false matches are numerous.

K-NN Matching

Under the K-NN matching strategy, a query keypoint Q matches with a keypoint K_A in the database if K_A is among one of the K nearest neighbors of Q and if the distance between them is below a threshold. With this approach a query keypoint has up to K matches. The threshold should be set large enough such that keypoints under serious image transformations are still within the threshold radii from the query keypoints such that it is highly probably that no correct matches are missed before choosing the nearest K.

The value K is, again, a fixed value. Since the content of database is variant, we actually cannot tell how many matches exist for each query keypoint. Certainly, we can tell how many matches exist in performance evaluation, but we cannot do so when our system is deployed to public use. As K is set much lower than the actual number of correct matches, not all matches of a query points may be included and this makes the recall rate low. On the contrary, as K is set far higher than the actual number of correct matches, many false matches are included and this makes the precision rate low.

K-NNRatio Matching

Under the above strategies, all keypoints within a threshold radius of a query points or among the K nearest neighbors are counted as the matches regardless of the inter-distances between those keypoints. To the extreme, for instance, there may exist a case in which ten keypoints are located within the threshold radius of a query keypoint. One of them are very close to the query keypoint and all the other are actually very far away from the query keypoint. Even if all the ten keypoints are probably the matches but the nearest one is certainly the probably a correct match while the others are probably false matches.

To solve the above problem and to soften the adverse effect of a fixed value of threshold and K in the above matching strategies, we introduce a new matching strategy called K-NNRatio matching. Under the K-NNRatio matching strategy, a query keypoint Q matches with a keypoint K_A in the database if the distance ratio between K_A and the next nearest neighbor K_B is high enough, and if K_A is among the nearest K and the distance between K_A and Q is below a threshold. Under this definition, we can say the K-NNRatio matching strategy is an extension of the above strategy. We further integrate it with the nearest neighbor distance ratio employed in SIFT [22]. The requirement that the distance ratio between K_A and K_B has to be high enough seems ambiguous. We say so because there is no hard threshold on the distance ratio that the keypoint has to reach to be a candidate match. Instead, a weight is assigned to each keypoint. A keypoint with high weighting means that the keypoint is more important in the image voting process because we have larger confidence that this keypoint is a correct match. The weight assignment follow the following two principles:

1. If a keypoint is nearer to the query point, it is probably the correct match and its weight should be higher. Otherwise, the weight should be lower.

2. If a keypoint has large distance ratio to its next nearest neighbor, no other match seems like the correct match and its weight should be higher. Otherwise, the weight should be lower.

Our motivation to incorporate distance ratio into the weight for determining candidate matches is that, distance ratio matching strategy has been shown [15] to perform better than threshold based and nearest neighbor matching strategy in term of recall and precision. However, distance ratio is designed to be used in determining a single match only. To extend it for determining multiple matches, we process each of the K nearest neighbors and compute the corresponding distance ratio as if there are no nearer neighbors to the query keypoint.

To satisfy the above requirements, the weight of a keypoint is formulated as follow:

$$Weight(K_A) = (\frac{a}{k(K_A)})^b \times (\frac{dist(K_B,Q)}{dist(K_A,Q)})^c \tag{4.1}$$

where a, b, and c are the real numbers to be empirically determined. dist(K,Q) is the Euclidean distance between keypoint K and keypoint Q. $k(K_A)$ is the rank number of K_A among the K nearest neighbors. That is, if keypoint K_A is the nearest neighbor of Q, then $k(K_A)$ is 1. If keypoint K_B is the second nearest neighbor, then $k(K_B)$ is 2, and so on. The weight of a keypoint depends on the rank number of the keypoint in the K nearest neighbor and also the distance ratio. Thus we cannot say how large should be the distance ratio such that it has higher weight. It depends on its rank number in the K nearest neighbor. The term $(\frac{a}{k(K_A)})^b$ is designed to satisfy the first requirement while the term $(\frac{dist(K_B,Q)}{dist(K_A,Q)})^c$ is designed to satisfy the second requirement. To balance the influence of these two terms, we introduce the parameters a, b, and c. We will discuss the choice of these values in the performance evaluation section.

To implement this matching strategy, we first query E^2LSH and obtain the K nearest neighbors of the query point. For each of the nearest neighbors, we calculate the weight taking the first nearest neighbor and the second one into calculation. When calculating the weight of the second one, we take the second nearest neighbor and the third one into calculation.

There are several immediate advantages under this matching strategy. Firstly, the nearest neighbor do not always gain high weight. It will not have high weight if it is far from the query point. Secondly, a keypoint with higher rank number can still gain high weight if it is far away from all the other neighbors with higher rank number. Thirdly, the K nearest neighbors do not get the same weight and thus they have different voting power during image voting. If there are only two possible matches for a query keypoint, ideally only two keypoints will get high weights and all the others will get lower weights. This softens the adverse effect of the fixed value of K and the threshold.

4.3.2 Verification Processes

After the candidate matches for each query keypoint were selected, they are first sorted by their image IDs. Recall that each keypoint owns a keypoint index in the LSH hash table. Using this keypoint index as a key, we lookup the keypoint lookup table for the line of the keypoint's detail. From that line, we can obtain an image ID that uniquely identifying the image from which the keypoint is extracted. By sorting the candidate matches by their image IDs, the keypoints extracted from the same images are brought together. We then group the keypoints extracted from the same image together. The keypoints in each group are then filtered based on their geometric relationship between each other so as to reduce the number of probable false matches. We filter each group through two verification processes: orientation verification and affine geometric verification using RANSAC.

These two processes can only filter certain percentage of false matches. Therefore, we should not flood the inputs of these two processes with a large number of candidate matches of each image and totally rely on these two processes to filter out the large number of false matches. In other words, we should not set the threshold too large when using threshold based matching or set the K too large when using other matching strategies. This is to say, the matching strategy of our system is an important part of the system and is not substitutable.

These two processes are applicable for many applications of our system including the IND detection but not applications like generic object recognition and image class retrieval. This is because in these two applications, the scenes or objects inside a query image and a database image are not the same but belong to the same image category only. The geometrical transformation of the matching keypoint pairs over the image pair is neither affine transformation nor perspective transformation, but a transformation with semantic meaning, for instance, a transformation of a rectangle changing from thin and tall one to a fat and short one. It is possible to model this kind of transformation, but it is not easy. For those applications in which the geometrical transformation of the matching keypoint pairs can be modeled by affine transformation, these two verification processes will work perfectly.

The affine geometric verification process was adopted by Yan Ke [16] in building his IND detection system. However, we observed that the orientation of keypoints are not verified in his system. Thus, we propose a verification process that can work together with the affine geometric verification process to further remove probable false matches.

Orientation Verification

The orientation of a keypoint refers to the canonical orientation of the keypoint. It is determined by the image gradients of pixels in both x and y directions within the feature region. For detail, please see Section 2.3.1. Under most of the image manipulation processes, the orientation of a keypoint will change in similar amplitude as those of any other keypoints in the same image. Therefore, the difference of in orientation between each of the query keypoint and database keypoint pair should be more or less the same.

The orientation verification process of our system makes use of the consistency of this difference to remove the probable false matches and retain only the largest set of candidate matches that have consistent differences for each group.

Here are the steps of the orientation verification process for each group of candidate keypoint matches:

1. *Input* a group of keypoint matches.

2. For each candidate keypoint match, obtain the orientations of the query keypoint and the database keypoint through the keypoint lookup table using their indexes in the hash table as the indexes to the lookup table.

3. For each candidate keypoint match, subtract the orientation of the query keypoint from that of the database keypoint to obtain the difference in orientation. The range of the difference is $[-360°, 360°]$.

4. Fit the difference of each match into the range of $[0°, 360°]$ by adding $360°$ to it if it is negative.

5. Divide the range of difference into 36 bins, each with $10°$ width. Map the difference of each keypoint to one of the 36 bins and add the keypoint match pair into that bin.

6. Slide a moving window of the width 3 bins over the 36 bins. Slide for 1 bin each time for 36 times and wrap the window around at the end.

7. Find the maximum window which is the moving window having the maximum number of match pairs inside its 3 bins.

8. *Replace* the list of candidate keypoint matches of the current group with the list of matches existing in the 3 bins of the maximum window.

With this verification process, the number of false matches are significantly reduced. This can be reflected by the recall and precision rate of the system.

Affine Geometric Verification using RANSAC

The affine transformation between two images can be modeled by the following equation:

$$\mathbf{Ax} = \mathbf{b}$$

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} = \begin{bmatrix} u_0 \\ v_0 \\ 1 \end{bmatrix}$$

where $\mathbf{x}$ are the homogenous coordinates of a keypoint in the query image, $\mathbf{b}$ are the homogenous coordinates of the matched keypoint in the database image, and $\mathbf{A}$ is the transformation matrix with six unknowns. To compute the transformation matrix, we need 3 keypoint match pairs. With the 3 keypoint match pairs $(\mathbf{x}_0, \mathbf{b}_0)$, $(\mathbf{x}_1, \mathbf{b}_1)$ and $(\mathbf{x}_2, \mathbf{b}_2)$, we can compute the matrix $\mathbf{A}$ by solving the following linear equation:

$$\begin{bmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 \\ x_1 & y_1 & 1 & 0 & 0 & 0 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_0 & y_0 & 1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{01} \\ a_{02} \\ a_{10} \\ a_{11} \\ a_{12} \end{bmatrix} = \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ v_0 \\ v_1 \\ v_2 \end{bmatrix}$$

Since the large matrix on the left is a square matrix, we can find the least square solution of the above linear equation by multiplying the inverse of that matrix with the vector on the other side:

$$\begin{bmatrix} a_{00} \\ a_{01} \\ a_{02} \\ a_{10} \\ a_{11} \\ a_{12} \end{bmatrix} = \begin{bmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 \\ x_1 & y_1 & 1 & 0 & 0 & 0 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_0 & y_0 & 1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ v_0 \\ v_1 \\ v_2 \end{bmatrix}$$

In our implementation, we have employed the LU decomposition function and backward substitution function in the Numerical Recipes in C++ package [30] to compute inverse of matrix.

With the matrix $\mathbf{A}$, we can affine warp every query keypoint with homogenous coordinates $\mathbf{x}$ from the query image to the database image by multiplying the the 3×3 matrix $\mathbf{A}$ with vector $\mathbf{x}$.

We adopt RANdom SAmple Consensus (RANSAC) [9] to eliminate probable false matches in the group of candidate matches. Here are the steps of affine geometric verification for each group of candidate keypoint matches:

1. Check if there are at least 3 pairs of keypoint matches, remove the whole group from the list of candidate matches and finish the process if it is false.

2. Randomly pick three keypoint match pairs.

3. Calculate the affine transformation matrix based on these three match pairs only.

4. For all the other keypoint matches, map the query keypoint onto the database image and calculate the Euclidean distance between the mapped coordinates and the coordinates of the database keypoint. Compute the support of the current transformation by counting the number of matches with the distance smaller than a preset threshold, say 10.

5. Loop the above steps for a number of times, say ten times. Find the transformation that receives the greatest support.

6. Replace the list of candidate keypoint matches of the current group with the list of matches that support the transformation with greatest support.

This verification process further improves the recall and precision rate of our system.

4.3.3 Image Voting

After the verification processes, a large percentage of false matches should have been removed. However, there usually still remain a number of groups of candidate matches. Each group represents a different database image that may be the match of the query image. To determine which is more likely the correct match, we compare the support of that group which is defined as the number of orientation and affine transformation verified candidate matches inside that group. The larger the support, the greater the probability that the corresponding image is a near-duplicate of the query image. Under the threshold based and K-NN matching strategy, we sort the groups by their supports in descending order and remove those that have supports fewer than the minimum support which is 5. The top N(=10) groups are returned to the user and counted as a match during performance evaluation. Under the K-NNRatio matching strategy, not only the "quantity" of a group but also the "quality" is used to rank the groups. We first calculate the weight of a group which is defined by the summation of all the weights of the keypoint matches in that group. We then sort the groups by their weights in descending order instead of simply by their supports. This makes the more probable keypoint matches contribute more to the image voting process than those less probable matches. Similarly, those groups that have weights smaller than the 5 are discarded and the top 10 groups are returned to the user.

4.4 Performance Evaluation

We have done a number of experiments to show that our proposed approaches do improve the performance of the whole system and that our system is effective. We followed [16] to use 150 images as the query images and the transformed versions of the query images as the database images in the image database. The images we used are downloaded from [7]. They are photography in many different themes. For each image, 8 different transformations are applied to produce 8 different database images. The transformations include the followings:

1. Three cropping transformations done by cropping the query image by 50%, 70%, and 90% respectively. All cropped images are resized back to original size.

2. Three shearing transformations done by applying an affine warp along the x axis by 5°, 10°, and 15° respectively.

3. Two contrast changing transformations done by increasing contrast by 3× and decreasing it by 3× respectively.

Since each query image produce 8 transformed versions, there are all together 1200 database images. Before building an index, we extract keypoints from each image. Each image contains hundreds of keypoints. Thus, the keypoint database contains 1 million of keypoints.

The k, m and L parameter in E^2LSH is set to be 26, 28, and 50 respectively. All of our experiments use a Intel P4 3.2GHz machine with 2GB of memory running on Fedora Core 3 (Linux Kernel 2.6).

4.4.1 Evaluation Metrics

The performance of our system is evaluated using Receiver Operating Characteristic (ROC). We define a correct match as a match between a query image and one of its transformed versions in the database. Any other matches are false matches. The recall and precision rate are defined as follows:

$$recall = \frac{number\ of\ correct\ matches}{total\ number\ of\ correct\ matches}$$
$$precision = \frac{number\ of\ correct\ matches}{total\ number\ of\ matches}$$

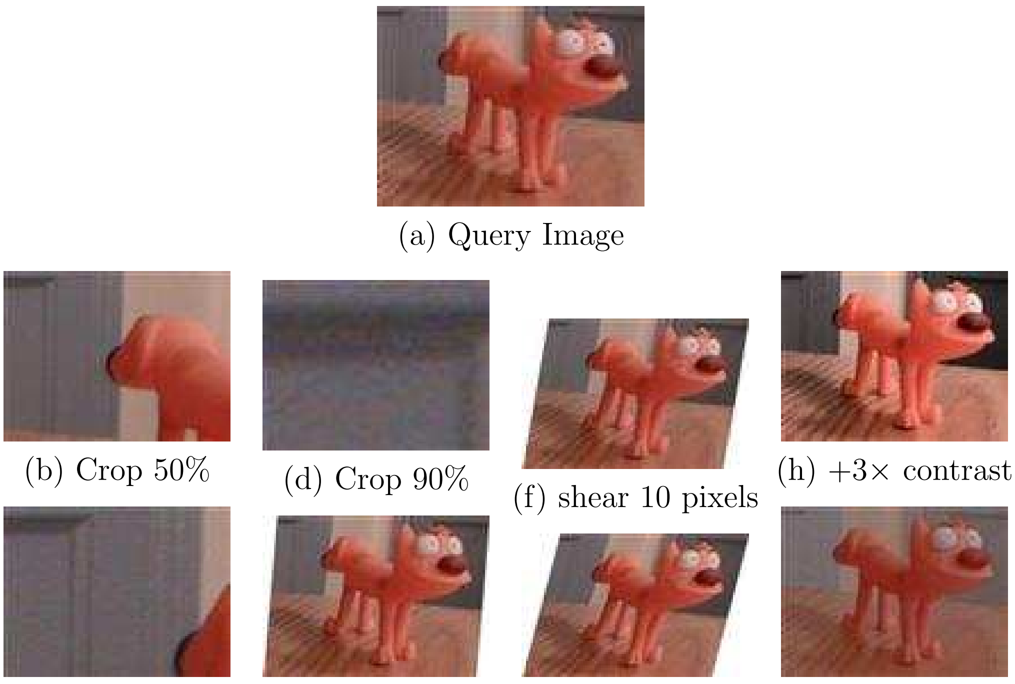
Intuitively, we want both recall and precision rate to be high.

4.4.2 Results

Preliminary Comparison on the Three Matching Strategies

In the this experiment, we compare the performance of our system with different matching strategies by make one query using the query image shown in 4.1. The eight transformed versions are also shown in 4.1.

The setting of each matching strategy is summarized in the table 4.1. The performance comparison is shown in the table 4.2. From table 4.2, we can see that query to system using threshold based matching give zero correct match. It is because there are numerous candidate keypoint matches lie within the threshold R. The image voting seems like a randomized result and thus no correct match result. As for the proposed K-NNRatio matching strategy, it gives one more correct match than K-NN matching strategy. That correctly matched image is the 50%-cropped version of the query image which is difficult to match correctly. There should have a few keypoints voting this image. However, under the proposed K-NNRatio matching strategy, the influence of a few keypoints can be large in image voting. This contributes to the higher recall rate of the K-NNRatio matching.


(a) Query Image

(b) Crop 50%

(d) Crop 90%

(f) shear 10 pixels

(h) +3× contrast

(c) Crop 70%

(e) shear 5 pixels

(g) shear 15 pixels

(i) −3× contrast

Figure 4.1: The Query Image and its eight transformed versions

Matching strategies	Settings
Threshold based	R = 350
K-NN	R = 350, K = 10
K-NNRatio	R = 350, K = 10, a = b = c = 1

Table 4.1: Table summarizes the experiment's settings.

Matching strategies	b	c	d	e	f	g	h	i
Threshold based	0	0	0	0	0	0	0	0
K-NN	0	0	0	67	57	75	17	68
K-NNRatio	10	0	0	41	7	164	10	65

Table 4.2: Table summarizes the results of query under each matching strategy without using any verification process. If any of the images (b) - (i) are among the top 10 during the image voting step, this table will show the support / weight of that image in the column that represents that image and in the row that represents the matching strategy in use.

Result of Orientation Verification

In the this experiment, we evaluate the performance of the orientation verification process. We query the database using the 150 query images (a) with orientation verification only, (b) with affine geometric verification only, and (c) with both verifications. The total number of possible matches is $150 \times 8 = 1200$. The experiment setting is presented in table 4.3.

The results are summarized in the table 4.4. As seen from the table, the orientation verification contributes to further improve the recall and precision rate.

Parameter Name	Value
Matching strategy	K-NN matching strategy
R	350
K	10

Table 4.3: Experiment setting.

Determine a, b, and c parameter of K-NNRatio

To determine the a, b, and c parameter of K-NNRatio matching strategy, we compare the performance of our system under different choice of a, b, c, and

Verification	# correct matches	# false matches	recall	precision
(a)	975	471	81%	67%
(b)	1001	430	83%	70%
(c)	1011	301	84%	77%

Table 4.4: Table summarizing the # correct matches, # false matches, recall, and precision rate.

other system parameters listed in the table 4.5.

Parameter Name	Value
Matching strategy	K-NNRatio matching strategy
Verification	Both
R	350
K	10 - 40
N	10 - 40
a	2 - 10
b	0.11 - 1.00
c	1.00 - 8.30

Table 4.5: Experiment setting. Note that the value N limits the maximum number of images being voted and returned to the user.

Part of the experimental result is shown in table 4.6. This is the part that shows the best setting of the system. The best *a, b, and c* parameter as determined by our experiment are **4**, **0.2**, and **4** respectively. The best *recall* and *precision* rate are **87**% and **85**% respectively. Comparing these results with that performed using K-NN matching strategy, we can see that the recall and precision rate is increased by **3**% and **8**% respectively. As seen from table 4.6, by using K-NNRatio matching, our system can perform 99% precision rate with just a bit lower recall rate, 84%, which is still higher than that of K-NN matching.

Running Time

The speed of E^2LSH is fast. To query 100 keypoints in a 1 millions keypoint database, the query takes only 10 seconds to finish. The only problem is that its memory requirement is large and thus it causes the scalability problem discussed before.

N	a	b	c	#correct	#false	recall	precision
10	4	1	8.3	933	570	0.78	0.62
10	4	1	4	1041	459	0.87	0.69
10	4	1	2.6	1037	456	0.86	0.69
10	4	1	2	1031	454	0.86	0.69
10	4	1	1.6	1029	449	0.86	0.7
10	4	1	1.3	1029	445	0.86	0.7
10	4	1	1.1	1029	440	0.86	0.7
10	4	1	1	1024	443	0.85	0.7
10	4	0.33	8.3	944	539	0.79	0.64
10	4	0.33	4	1041	205	0.87	0.84
10	4	0.33	2.6	1031	71	0.86	0.94
10	4	0.33	2	1024	33	0.85	0.97
10	4	0.33	1.6	1021	26	0.85	0.98
10	4	0.33	1.3	1017	15	0.85	0.99
10	4	0.33	1.1	1017	14	0.85	0.99
10	4	0.33	1	1015	10	0.85	0.99
10	4	0.2	8.3	950	528	0.79	0.64
10	4	0.2	4	1040	177	0.87	0.85
10	4	0.2	2.6	1027	66	0.86	0.94
10	4	0.2	2	1022	31	0.85	0.97
10	4	0.2	1.6	1017	28	0.85	0.97
10	4	0.2	1.3	1016	19	0.85	0.98
10	4	0.2	1.1	1018	16	0.85	0.98
10	4	0.2	1	1015	18	0.85	0.98
10	4	0.14	8.3	949	527	0.79	0.64
10	4	0.14	4	1039	176	0.87	0.86
10	4	0.14	2.6	1027	67	0.86	0.94
10	4	0.14	2	1022	40	0.85	0.96
10	4	0.14	1.6	1018	27	0.85	0.97
10	4	0.14	1.3	1016	21	0.85	0.98
10	4	0.14	1.1	1013	21	0.84	0.98
10	4	0.14	1	1015	13	0.85	0.99
10	4	0.11	8.3	949	528	0.79	0.64
10	4	0.11	4	1037	192	0.86	0.84
10	4	0.11	2.6	1027	76	0.86	0.93

Table 4.6: A portion of the performance evaluation result using different setting of value.

4.4.3 Conclusion

We have demonstrated our IND detection system is effective in detecting near-duplicate images in a large database with high recall and precision rate. The proposed K-NNRatio matching strategy has been shown to be better than K-NN matching strategy in terms of system's recall rate and precision rate. The proposed orientation verification scheme is also shown to be effective in removing probable false matches and this is also reflected in the system's recall rate and precision rate.

□ **End of chapter.**

Chapter 5

Future Work

5.1 Dataset for IND Detection

Currently we evaluate the performance of our IND detection system using a dataset different from that adopted by Yan Ke [16]. To facilitate the performance comparison between our systems, we intend to follow Yan Ke to run our system on the MM270K dataset released in [27] in the near future.

Moreover, we have adopted only 8 out of the 50 transformations used by Yan Ke. Although the eight transformations we are using are the most challenging eight with respect to the degree of distortion applied on the transformed images, it is still meaningful to apply the remaining transformations to the query image so that we can make direct comparison with the others.

5.2 Remove Duplicates in Retrieved Images

IND detection is useful in removing duplicated images from the result of a query to a content-based image retrieval system. Since repeated results are meaningless to the user, IND detection can help increasing the amount of information delivered to the user in a limited number of displayed images. We will try making such application in the near future.

5.3 Incorporation of Global Features

With the proposed K-NNRatio matching strategy, our IND detection system can perform very high precision rate. This means that our system can accurately remove the false matches and return only the correct matches to user. With this advantage, we can consider returning the result from matching global features when there are fewer than 10 images being voted to be the correct images. This help improving the recall rate of our system.

☐ **End of chapter.**

Chapter 6

Conclusion

In this paper, we have discussed several recent research work on invariant local grayvalue features. We have evaluated the performance of several popular feature descriptors and found that SIFT feature descriptor remains the best comparing with other descriptors in the experiment. We have introduced our newly proposed feature descriptor, SSIFT, which extends SIFT feature descriptor to invariant to the change in background and object color. We have evaluated the performance of our descriptor with SIFT and shown that our descriptor does better in the cases that changes in background and object color occur. We have introduced the implementation of our image retrieval system which is designed to remove near-duplicate images from a set of images. The system is efficient due to the integration of powerful feature detector, descriptor, matching scheme, the new matching strategy and the new verification process. In the last chapter, we have discussed some ways to further improve our system and extend it to other applications.

□ **End of chapter.**

Bibliography

[1] A. Andoni. http://web.mit.edu/andoni/www/LSH/index.html.

[2] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *Proceedings of the ninth European Conference on Computer Vision*, May 2006.

[3] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, page 509, 2002.

[4] S.-A. Berrani, L. Amsaleg, and P. Gros. Robust content-based image searches for copyright protection. In *MMDB '03: Proceedings of the 1st ACM international workshop on Multimedia databases*, pages 70–77, New York, NY, USA, 2003. ACM Press.

[5] M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. pages I: 510–517, 2005.

[6] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, New York, NY, USA, 2004. ACM Press.

[7] DPChallenge. http://www.dpchallenge.com/.

[8] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. CVPR, 2003.

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

[10] D. A. Forsyth and J. Ponce. *Computer Vision A Modern Approach.* Pearson Education International, 2003.

[11] B. Funt, K. Barnard, and L. Martin. Is machine colour constancy good enough? page I:445, 1998.

[12] B. Funt and G. Finlayson. Color constant color indexing. 17(5):522–529, May 1995.

[13] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey88*, pages 147–152, 1988.

[14] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, New York, NY, USA, 1998. ACM Press.

[15] C. S. K. Mikolajczyk. A performance evaluation of local descriptors. *PAMI*, 27:1615–1630, 2005.

[16] Y. Ke, R. Sukthankar, and L. Huston. An efficient parts-based near-duplicate and sub-image retrieval system. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 869–876, New York, NY, USA, 2004. ACM Press.

[17] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using affine-invariant regions. In *CVPR*, pages II:319–324, 2003.

[18] W. Li and E. Salari. Successive elimination algorithm for motion estimation. ieee transactions on image processing, 4(1):105 – 107, january 1995.

[19] H. Ling and D. Jacobs. Deformation invariant image matching. In *ICCV*, pages II: 1466–1473, 2005.

[20] H. Ling and K. Okada. Diffusion distance for histogram comparison. In *CVPR06*, 2006.

[21] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999.

[22] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[23] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. In *CVPR*, pages I:26–36, 2006.

[24] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27.

[25] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *ICCV*, pages 525–531, 2001.

[26] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *IJCV*, 65(1-2), 2005.

[27] MM270K. http://www.cs.cmu.edu/∼yke/retrieval/.

[28] R. Nelson, editor. *3-D Recognition Via 2-Stage Associative Memory.* Univ. of Rochester, 1995.

[29] Y. K. Rahul. Pca-sift: A more distinctive representation for local image descriptors. In *CVPR*, pages 511–517, 2004.

[30] N. Recipes. http://www.numerical-recipes.com/.

[31] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *PAMI*, 19(5):530–535, May 1997.

[32] S. Smith and J. Brady. Susan: A new approach to low-level image-processing. *IJCV*, 23(1):45–78, May 1997.

[33] M. Swain and D. Ballard. Indexing via color histograms. In *DARPA90*, pages 623–630, 1990.

[34] J. van de Weijer, T. Gevers, and A. Bagdanov. Boosting color saliency in image feature detection. 28(1):150–156, January 2006.

[35] J. van de Weijer and C. Schmid. Coloring local feature extraction. In *ECCV2006*, 2006.

[36] D.-Q. Zhang and S.-F. Chang. Detecting image near-duplicate by stochastic attributed relational graph matching with learning. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 877–884, New York, NY, USA, 2004. ACM Press.