Color Invariant Feature Descriptor for Object Recognition

Wong Yuk-Man

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

Supervised by

Prof. Michael R. Lyu

April 2006

Abstract

Object recognition is an important part of computer vision because it is closely related to the success of many computer vision applications. A number of object recognition algorithms and systems have been proposed for a long time in order to address this problem. Yet, there still lacks a general and comprehensive solution. Recently, view-based object recognition by invariant local features have demonstrated good performance on a variety of object recognition problems. However, they tend to ignore color information because of some reasons. In this paper, the most representative invariant local features are first reviewed in the first chapter. We evaluated some of their performances and presented the results in the "experimental results" section. After that, techniques that make color robust enough to be used as a recognition cue are presented. These techniques are keys toward successfully incorporating color into local features. At last, some possible extensions to the recently proposed approaches are proposed and discussed in the paper.

Contents

Chapter 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 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. [23], 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 neglected. There are two main usage 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 properties. 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!

□ **End of chapter.**

Chapter 2

Object Recognition

2.1 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.

2.1.1 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.

2.1.2 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 [13, 14].

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.

□ End of chapter.

2.2 Model-Based Object Recognition

Generative models are the commonly used models in modeling object for matching. They are models described by mathematical functions and operators and is sufficiently complete to generate images of target objects. One of the common usage of model-based object recognition is face modeling and recognition. Generative face models can generate realistic images of a human face, with changeable facial expression and pose. The general steps to recognize an object using model is:

1. Locate the object,
2. locate and label its structure,
3. adjust the model's parameters until the model generates an image similar enough to the real object.

Generative models that are of particular interest are deformable models because objects in a class are often not identical. To be able to identify all objects within the class, we have to deal with variability. Deformable models are the models that capture the principle components of the class of objects and they can deform to fit any of the object in a class.

A number of models have been proposed to address the problem, they include Active Contour Models, Morphable Models, Active Blobs, Active Shape Models and the recently proposed Active Appearance Models etc. Active Appearance Models have been proved to be highly useful models for face recognition and several major extensions to these models have recently emerged. They are Direct Appearance Models and Inverse Compositional Active Appearance Models [15] etc. We will give a brief description on Active Appearance Models and one of its extensions in the following sections.

2.2.1 Active Appearance Models (AAMs)

AAMs are non-linear parametric models that are commonly used to model faces. They model shape and appearance of objects separately. Shape of an AAM is defined by the vertex locations of a mesh. In advance, the shape $\mathbf{s}$ can be expressed as the sum of a base shape $\mathbf{s_0}$ and a linear combination of n orthonormal shape vectors s_i. It can be mathematically defined as follow:

$$\mathbf{s} = \mathbf{s_0} + \sum_{i=1}^{n} p_i \mathbf{s_i}$$

The appearance of an AAM is defined based on the base mesh $\mathbf{s_0}$. If we define the set of pixels that lies inside the base mesh as $\mathbf{x} = (x, y)^T$, then

the appearance of AAM $A(\mathbf{x})$ as the sum of a base appearance $A_0(\mathbf{x})$ and a linear combination of m appearance images $A_0(\mathbf{x})$

$$A(\mathbf{x}) = A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x})$$

We can generate the shape of AAM with solely the AAM shape parameter $\mathbf{p} = (p_1, p_2, ..., p_n)^T$ and generate the appearance of AAM with solely the AAM appearance parameter $\lambda = (\lambda_1, \lambda_2, ..., \lambda_m)^T$. To generate a complete AAM model instance, we need both the shape parameter $\mathbf{p}$ and appearance parameter λ. It can be created by warping the AAM appearance A from the base mesh s_0 to the AAM model shape s. This warping is better represented by Matthews et al. [15] as the piecewise affine warp function $\mathbf{W}(\mathbf{x}, \mathbf{p})$, which can be used to warp any pixel x in s_0 to the corresponding pixel location at s. The AAM model instance M can then be mathematically defined as follow:

$$M(\mathbf{W}(\mathbf{x}, \mathbf{p})) = A(\mathbf{x})$$

As fitting an AAM to an image of object, non-linear optimization solution is applied which iteratively solve for incremental additive updates to the shape and appearance coefficients. Given an input image I, the goal of AAM fitting is to minimize the following error image E with respect to the shape parameters $\mathbf{p}$ and the appearance parameters λ:

$$E(\mathbf{x}) = A_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i A_i(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}, \mathbf{p}))$$

2.2.2 Inverse Compositional AAMs

Inverse Compositional AAMs are proposed by Matthews et al. [15]. The major difference of these models with AAMs is the fitting algorithm. The fitting algorithm adopted by AAMs is an additive incremental update approach. The algorithm is run to solve for $\Delta\mathbf{p}$ and update the parameter $\mathbf{p}$ to $\mathbf{p} + \Delta\mathbf{p}$. Matthews et al. argued that there cannot be any efficient algorithm that solves for the incremental parameter $\Delta\mathbf{p}$. They believed that another parameter update scheme can be made more efficient. That is inverse compositional algorithm. The algorithm updates the entire warp by composing the current warp with the computed incremental warp with parameters $\Delta\mathbf{p}$ following the following update rule:

$$\mathbf{W}(\mathbf{x}, \mathbf{p}) = \mathbf{W}(\mathbf{x}, \mathbf{p}) \circ \mathbf{W}(\mathbf{x}, \mathbf{\Delta p})^{-1}$$

Here the update function of the affine warp $\mathbf{W}$ is an inverted version of the incremental warp, $\mathbf{W}(\mathbf{x}, \mathbf{\Delta p})^{-1}$. The reason is that in inverse compositional algorithm, the roles of the input image and the template $A_0(\mathbf{x})$ are reversed. The incremental warp is computed with respect to the template $A_0(\mathbf{x})$. The error image that the AAM fitting algorithm minimizes becomes:

$$E(\mathbf{x}) = \sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x}, \mathbf{p})) - A_0(\mathbf{W}(\mathbf{x}, \mathbf{\Delta p}))]^2$$

Through these changes, most of the computation in computing $\Delta\mathbf{p}$ can be moved to a pre-computation step and thus this results in a efficient image alignment algorithm.

□ End of chapter.

2.3 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 [10] 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 [23] 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 [1], line segments and groupings of edges [19] 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 [22] and Harris corner detector [8]. Based on these features, image descriptor can be created to increase the matching performance. Schmid & Mohr [21] 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 [13, 14] 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 [20] improved SIFT descriptor by PCA analysis so that the computation speed of feature is greatly improved. Fergus et al.

Figure 2.1: The objects used to test the Nelson's recognition system

2.3.1 Recognition Based on 2D Boundary Fragments

Nelson [19] proposed a method of 3D object recognition based on the use of a general purpose associative memory and a principal views representation. He used automatically, robustly extracted 2D boundary fragments as keys. These keys have sufficient information contents to specify the location, scale and orientation of an associated object and sufficient additional parameters to provide efficient indexing and meaningful verification. The keys extracted from an image are fed into an associative memory to generate a set of all objects that could have produced those keys. He called the results generated from the associative memory as hypotheses. These hypothesis are then fed into a second stage associative memory which maintains the probability of each hypothesis based on the statistics of the occurrence of the keys in the associative memory. Since this recognition approach bases on grouping of local feature rather than global features, it is robust to occlusion and clutter, and does not require prior segmentation.

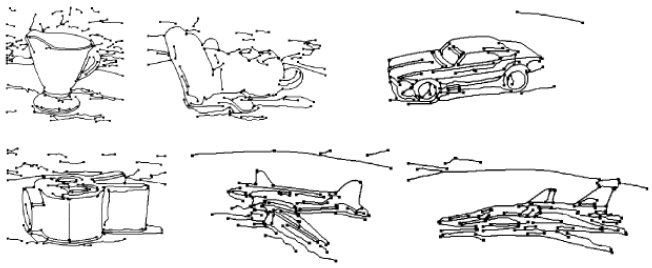

Figure 2.2: The extracted key features of Nelson's testing images

Nelson carried out experiments using keys based on groups of 2-D boundary fragments. He ran tests with databases built for 6, 12, 18 and 24 objects, shown in Figure 2.1, and obtained overall success rates of 99.6%, 98.7%, 97.4% and 97.0% respectively. The total number of training images for the 24 object database was 1802. Training data consisted of 53 clear images per object, spread fairly uniformly, with approximately 20 degrees between neighboring views. Some examples of extracted key features from some testing images are shown in Figure 2.2. The accuracy of the system reported is good. However, the test cases of this experiment is quite ideal because:

1. The number of training images for each object stored in the database is large.
2. The images under test is clean.
3. The objects used were chosen to be different in that they were easy for people to distinguish on the basis of shape.
4. The object viewed from the images has more or less the same scale.
5. There are no occlusion and clutter in every image.

2.3.2 Recognition Based on SIFT

SIFT stands for Scale Invariant Feature Transform. It is a novel method proposed recently by Lowe [13, 14] 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.

Object recognition

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.

2.4 Conclusion

In this chapter, recent techniques of two common object recognition approaches are discussed. Our research will mainly focus on the feature-based recognition techniques used in view-based recognition approach.

□ End of chapter.

Chapter 3

Invariant Local Grayvalue Features

3.1 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.

3.2 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 it by an vector. It is thus necessary to extract only a subset of pixels from an image, which are called interest points, to be described. 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 such 2D window for recognition purpose, they includes Harris corner detector [8], DOG extrema detector [14], Harris-Laplacian detector [16] and affine covariant region detector [18].

3.2.1 Harris Corner Detector

Harris corner detector [8] is widely used in many image matching tasks to select a region that has 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:

$$\mathbf{M} = g(\sigma_I) * \begin{bmatrix} I_x^2(\mathbf{x}) & I_x I_y(\mathbf{x}) \\ I_x I_y(\mathbf{x}) & I_y^2(\mathbf{x}) \end{bmatrix} \tag{3.1}$$

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 kernels 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:

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

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 if they are not the local maxima of $\mathbf{R}$ within its local neighborhood:

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

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

3.2.2 DOG Extrema Detector

DOG Extrema Detector is proposed by Lowe [14, 13] 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 [13] 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.

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

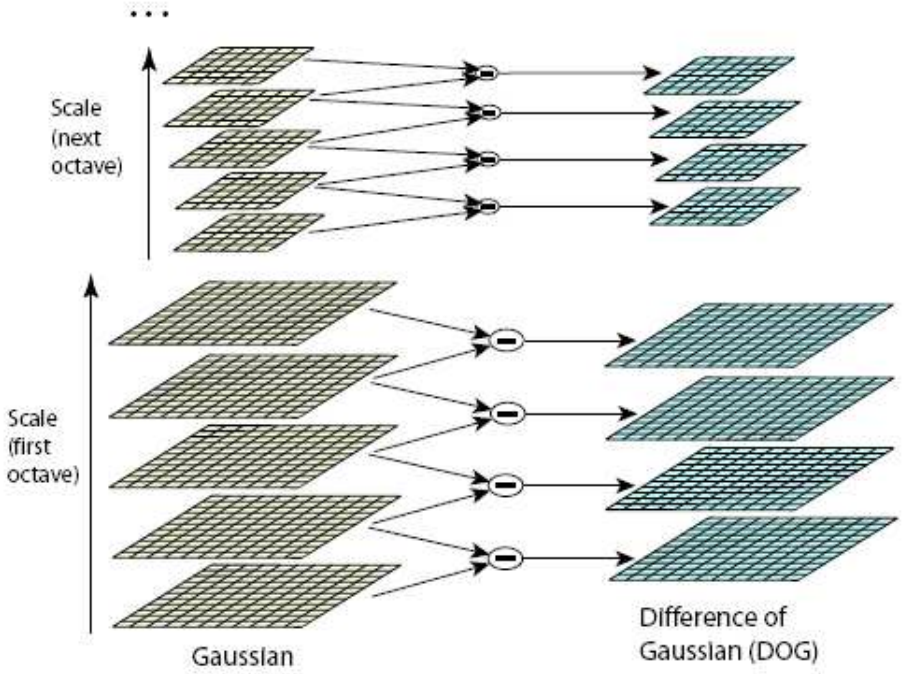

Figure 3.1: A diagram illustrating how differences of gaussian images is prepared from the initial image. The initial image is repeatedly smoothed by Gaussian function, which is shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images, which is shown on the right.

The scale space of the input image is prepared in the way illustrated by Figure 3.1. 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:

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

For the experiments done by Lowe in [14], 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 [14], all extrema having a ratio between the principal curvatures greater than 10 are discarded.

3.2.3 Harris-Laplacian Corner Detector

Mikolajczyk et al. [16] 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:

$$\mathbf{M} = \sigma_D^2 g(\sigma_I) * \begin{bmatrix} I_x^2(\mathbf{x}, \sigma_D) & I_x I_y(\mathbf{x}, \sigma_D) \\ I_x I_y(\mathbf{x}, \sigma_D) & I_y^2(\mathbf{x}, \sigma_D) \end{bmatrix} \tag{3.2}$$

Equation 3.2 differs from equation 3.1 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}.

3.2.4 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 the 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.

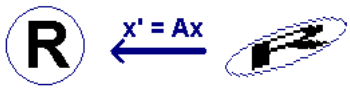

Figure 3.2: This figure shows how the elliptical affine region and the normalized region look like. The transformation matrix $A = M^{1/2}$ projects x to x' such that the eigenvalues of the auto-correlation matrix are equal.

3.3 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 descriptors are shown [9] 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 [13, 14], shape context [1], PCA-SIFT [20], GLOH [17] and GIH descriptors [11]. 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. [17] 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.

3.3.1 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. [17] 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.

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.

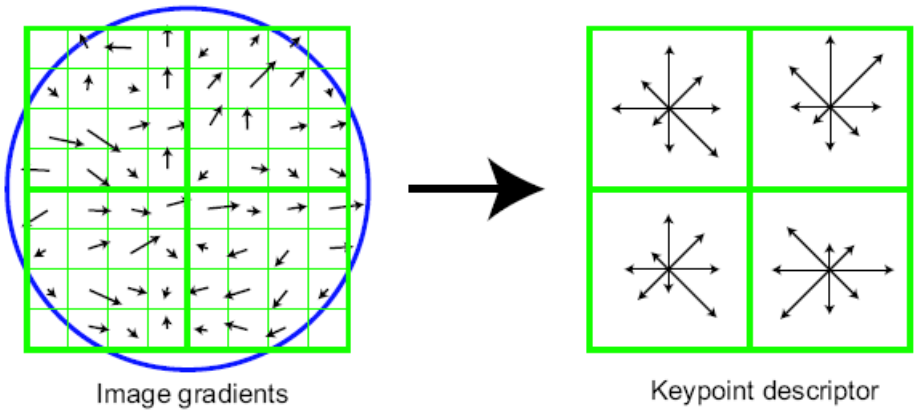

Figure 3.3: Computation of a feature descriptor based on the gradient and orientation of each image sample point in a region around the feature.

Descriptor Representation

The computation of the feature descriptor is illustrated in Figure 3.3. 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 3.3, 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.

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.

$$I(\mathbf{x}) = aI'(\mathbf{x}) + b \tag{3.3}$$

Equation 3.3 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.

$$\widehat{I}(\mathbf{x}) = \frac{aI_x(\mathbf{x})}{a\sum_{x \in W} I_x(\mathbf{x})} = \frac{I_x(\mathbf{x})}{\sum_{x \in W} I_x(\mathbf{x})} \tag{3.4}$$

where W is the set of points within the concerned local region. Equation 3.4

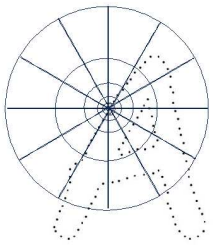

Figure 3.4: Figure illustrating how the bins of shape context is distributed around a given edge point. Belongie et al. [1] use five bins for quantizing distance between the rest of the edge points from the given edge point and 12 bins for quantizing the angle between them.

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.

3.3.2 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 3.4 and 3.5. In the original design of shape context, a histogram h_i is computed by simply counting the number of edge points within a bin:

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

In the modified design by Schmid et al. [9], 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.

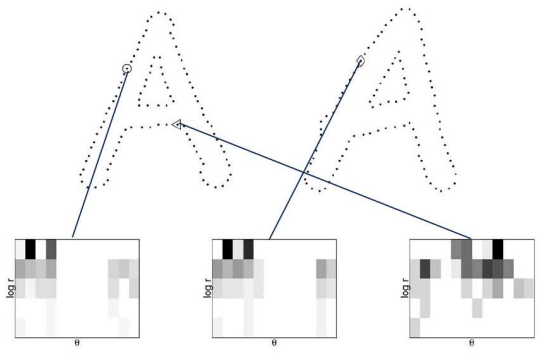

Figure 3.5: Figure showing the shape context histograms of three edge points. Darker bins indicate larger number of edge points are located inside the bins. The first and second histograms are very similar because the edge point they represent are correspondence while the third histogram is very different.

3.3.3 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 [9] and run a lot faster than SIFT because of its 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.

3.3.4 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.

3.3.5 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 [11] 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 3.6. GIH descriptor is created in the following steps:

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.

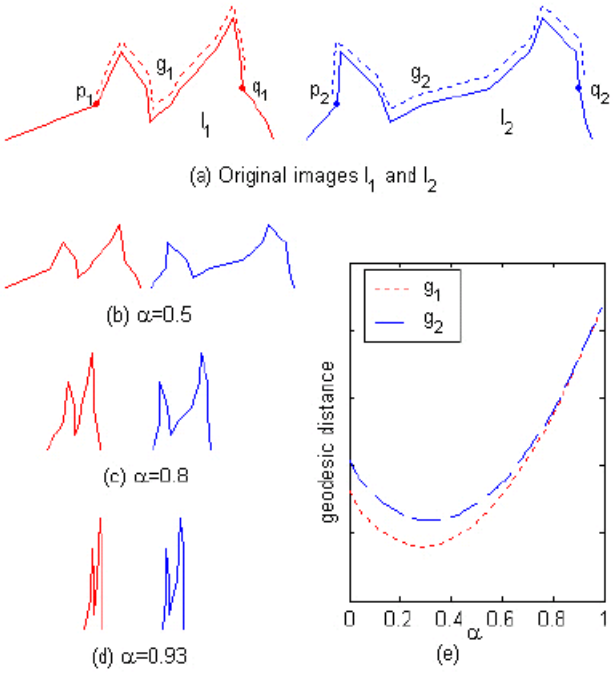

Figure 3.6: Deformation invariance for 1-D images (Figure from [11]).

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 3.7.

3.3.6 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-

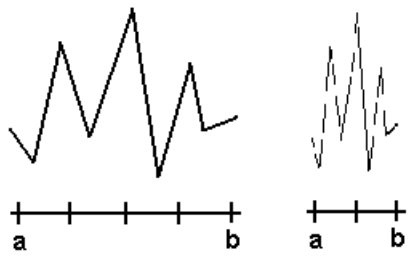

Figure 3.7: Figure illustrating discrete geodesic distance can fail. Due to discrete sampling of image pixels, the geodesic distance between points a and b is large in the left image and small in the deformed image on the right.

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 3.8

Evaluation Criterion

We adopted the evaluation criterion used in [20, 9]. 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 [18, 9] 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

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

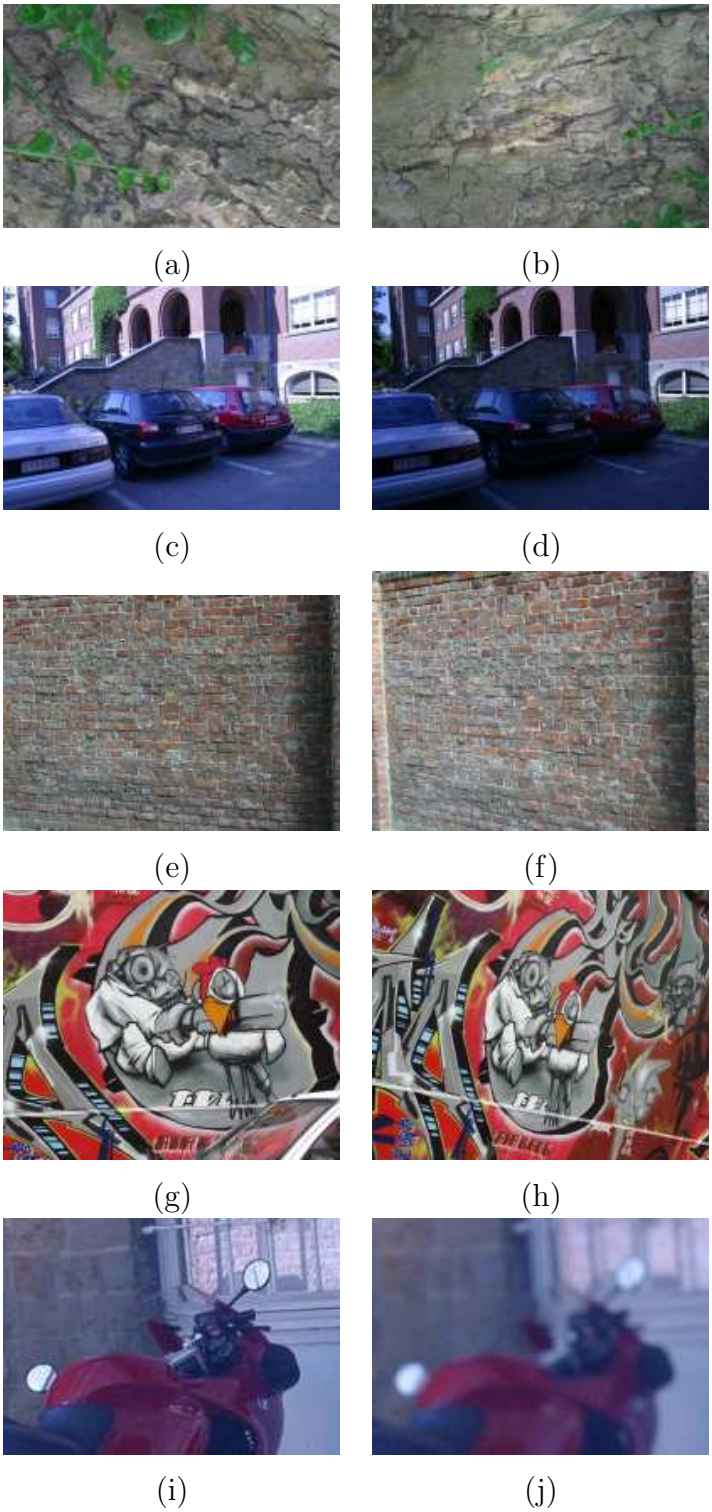

Figure 3.8: (a) & (b) Bark image sequence. (c) & (d) Leuven image sequence. (e) & (f) Wall image sequence. (g) & (h) Graf image sequence. (i) & (j) Bikes image sequence (A small portion).

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 3.8(a),(b) to evaluate the performance for scale change and rotation. Result is shown in figure 3.9. 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 3.8(c),(d) to evaluate the performance for illumination change. Result is shown in figure 3.10. 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 3.8(e),(f) and figure 3.8(g),(h) to evaluate the performance for viewpoint change. Result is shown in figure 3.11 and 3.12. 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 3.8(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 3.13. 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.

3.3.7 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 [3].

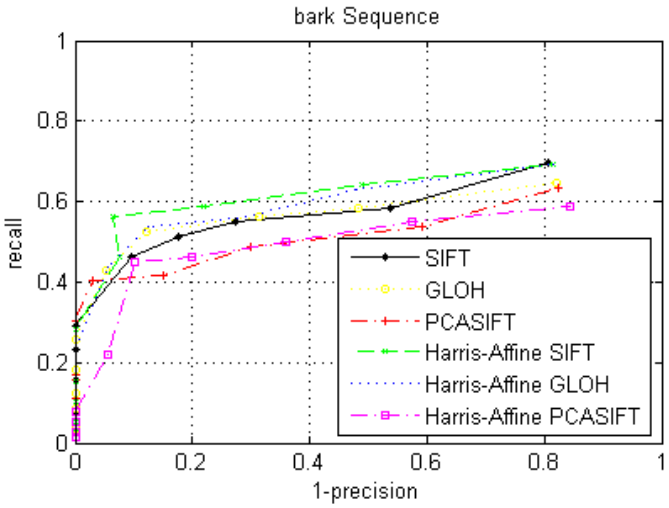

Figure 3.9: Experimental Result for scale change and rotation on bark image sequence.

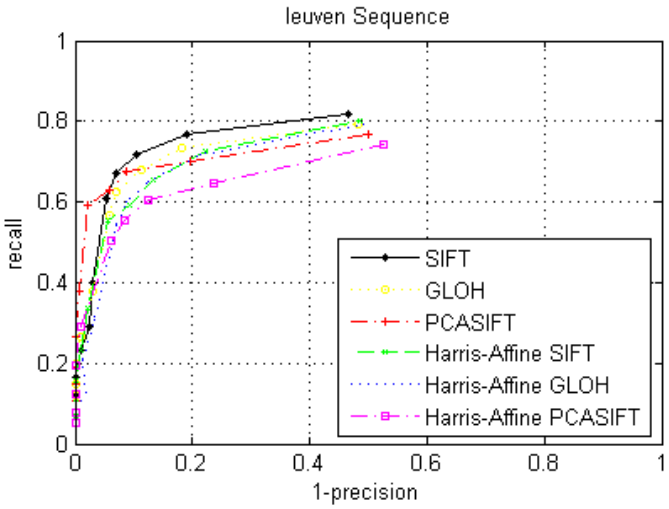

Figure 3.10: Experimental Result for illumination change on leuven image sequence.

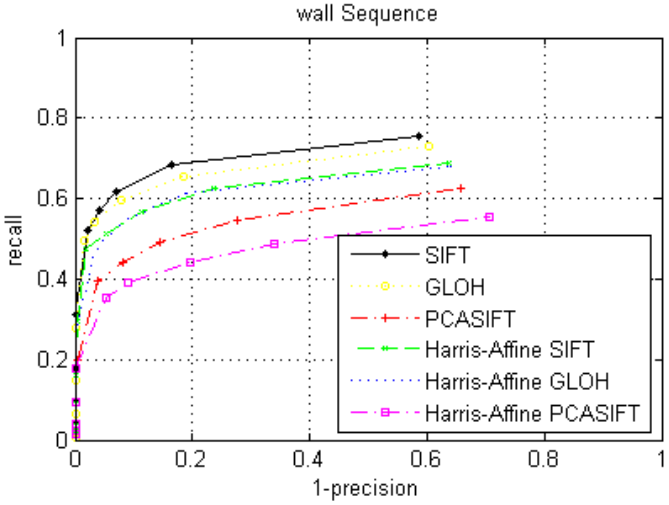

Figure 3.11: Experimental Result for viewpoint change on wall image sequence.

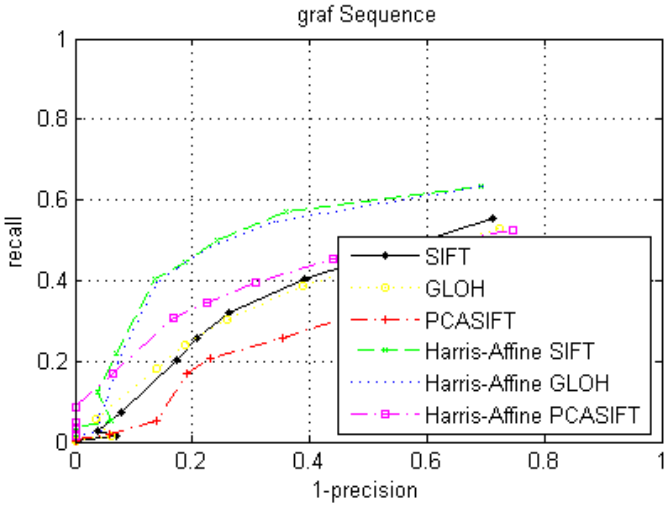

Figure 3.12: Experimental Result for viewpoint change on graf image sequence.

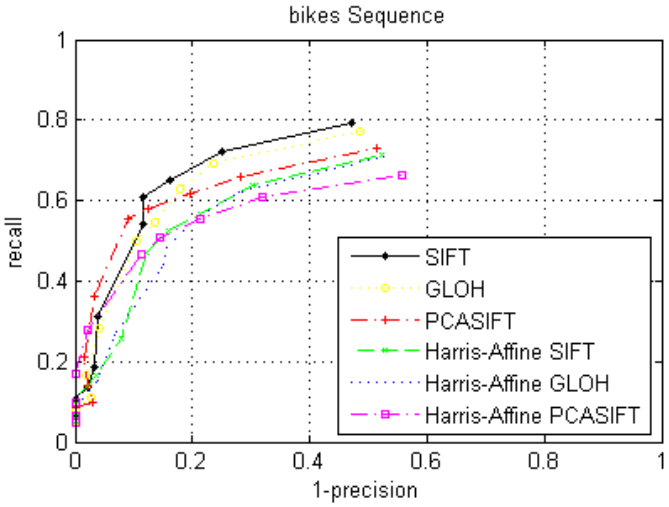

Figure 3.13: Experimental Result for image blur on bikes image sequence.

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 3.14. The hybrid-type descriptor has better performance in general among the three descriptors. Yet, its performance is still far below SIFT's.

3.4 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 [12] 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.

3.4.1 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.

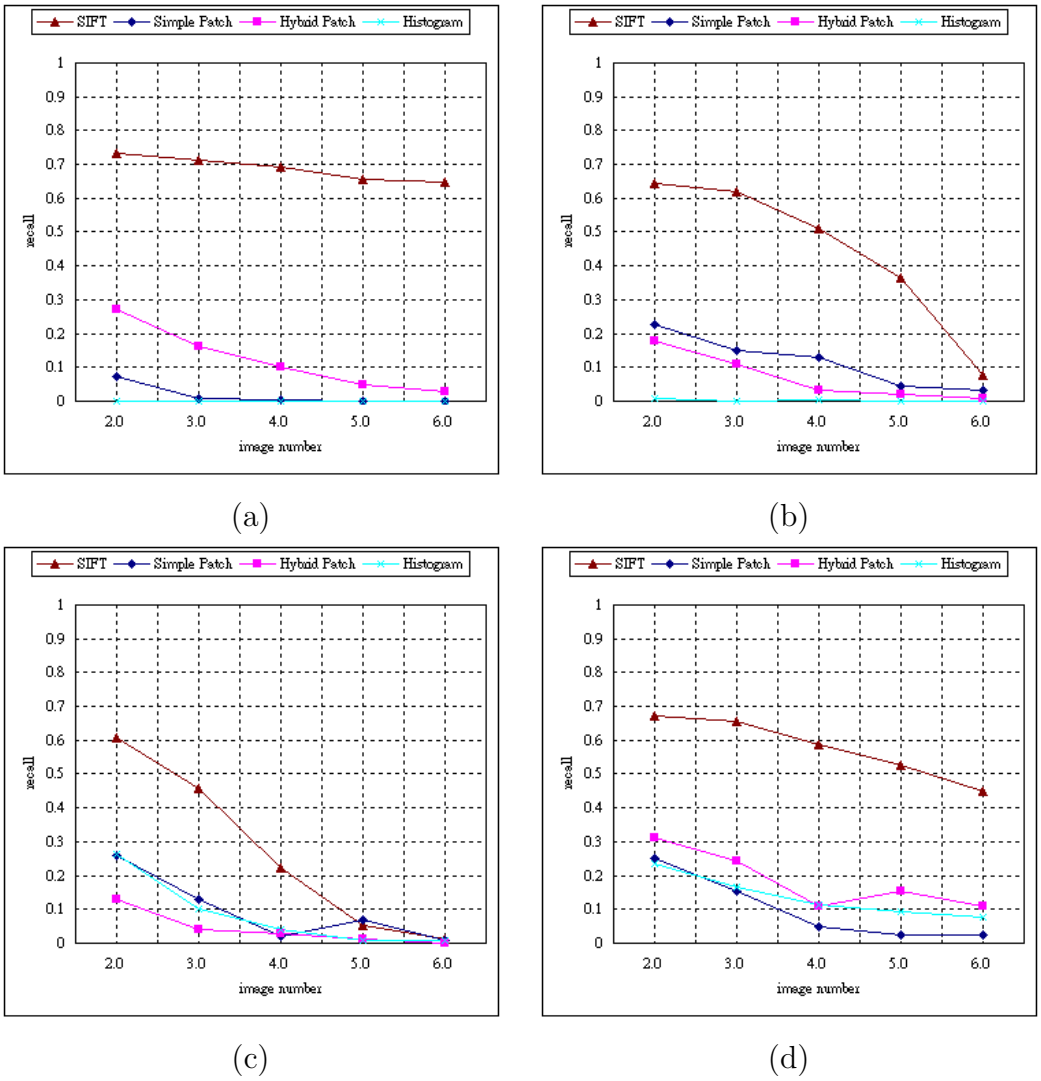

Figure 3.14: (a) Evaluation for illumination change on Leuven image sequence, (b) Evaluation for affine transformation on wall image sequence, (c) Evaluation for affine transformation on graf image sequence, (d) Evaluation for image blur on bikes image sequence.

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 [9].

3.4.2 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. [23].

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.

3.4.3 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.

3.5 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 4

Color Invariant Local Features

4.1 Introduction

Color is always neglected as a recognition cue because many variations can cause significant changes in measured color. Nevertheless, some techniques [6, 25] were proposed to reduce the sensitivity of color information to photometric changes and make color description more robust. Recently, Weijer et al. [25, 24] 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.

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.

4.2.1 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 $B_{average}$.

4.2.2 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}.

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 [5]:

$$C(\mathbf{x}) = g_d(\mathbf{x})\mathbf{d}(\mathbf{x}) + g_s(\mathbf{x})\mathbf{s}(\mathbf{x}) + \mathbf{i}(\mathbf{x}) \tag{4.1}$$

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:

$$C(\mathbf{x}) = g_d(\mathbf{x})\mathbf{b}(\mathbf{x})\mathbf{e} + g_s(\mathbf{x})\mathbf{e} + \mathbf{i}(\mathbf{x}) \tag{4.2}$$

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.

4.3.1 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. [7] proposed to use color ratio to achieve illumination invariance based on similar assumption in order to make the color histogram proposed by [23] 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.

4.3.2 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:

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

4.3.3 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.

4.3.4 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:

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

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.

□ End of chapter.

Chapter 5

Proposed Research

Although model-based object recognition systems have recently demonstrated good performance in face recognition, they generally require a detailed 3D face model to be constructed and trained in supervised learning manner. When the same 3D model is used to model objects other than human face, the systems will fail. And before the system can represent certain object by a 3D model representation, it has to know which 3D model it should apply on that object. In order word, the system have to recognize at least the type of that object before using a generic 3D model to model it. Assume the system know which 3D model it should apply, in order to model every generic object, for each type of objects, the model-based recognition systems will have to obtain their models before they can recognize them. Since the type of objects in the world is tremendous, attempt to model each type of objects manually or train the systems in supervised learning manner is impossible. A possible way to get around with this limit is to teach a system to learn the 3D model structures of objects automatically, there have not been a versatile solution to this problem. Another way to get around with the problem is to adopt another type of approach, the view-based object recognition approach.

View-based object recognition system recently demonstrated good performance on a variety of problems too. Mikolajczyk and Schmid [9] recently evaluated a variety of approaches and identified the SIFT algorithm as being the most resistant to common image deformations. SIFT algorithm is being continuously extended by other researchers and the improved versions PCA-SIFT and GLOH have reported better performance.

5.1 Hybrid of bottom-up approach and top-down approach

The view-based object recognition systems designed by Nelson and Lowe are typically bottom-up approaches. Their recognitions are first carried out at the low level of machine vision, in which low-level features such as edges and local extrema are searched. The recognitions proceed to the mid level of machine vision, in which descriptors of features are created. And finally at the higher level of machine vision, descriptors extracted on the input image are matched with those in the database and matched descriptors are mapped to the possible object types stored in the database by mean of voting algorithm. Sometimes verification process is carried out to refine the vote.

We believe that the process that human recognizes objects is not a purely bottom-up approach. As we recognize an object, we may just observe for a few features on the object and then we will try to guess what the object is. A verification step may follow to pick up the best hypothesis. The major difference in this approach is that the recognition system searches for salient features and try to recognize the object based on its knowledge, memory or say, database, interlacedly. In this way, effort on trying to recognize all features is reduced. Moreover, having recognized an object, its pose and location, it should be easier to segment out that object from the image and make the recognition of other objects more easily and accurately.

Some attempts were made to approach the object recognition problem using a top-down perspective, where recognition is performed by determining if one of many known objects appear in the image. Those approaches would be a good reference to us.

5.2 Hierarchy of features

In the recognition system proposed by Lowe [14], an object is simply a group of features. When features vote for an object pose, all features have the same voting power. However, an object may have most of their features common to other objects and have just a few of them special. If all features have the same voting power, extracted special features may not have enough influence to bias the voting result to the correct object. Thus a weight should be assigned to the features. Instead of assigning a weight to each feature based on its probability of occurrence, we can group a group of features into an small object and assign weight to each small object. An object is composed of many small objects and each small object is composed of many features, as shown in Figure 5.1. Grouping a set of features into a small object is

reasonable because only one feature keypoint usually do not carry much information.

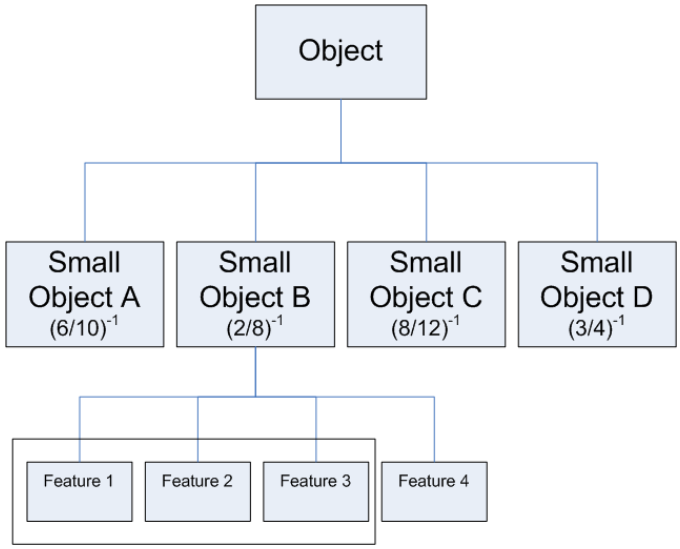

Figure 5.1: Each object is composed of many small objects and each small object is again composed of many features. Each small object is assigned a weight, say value 4 for object B, based on the probability of occurrence of the object B in the database. During features detection, not all features of a small object may be detected. As shown in the figure, there may just be 3 out of 4 features of object B found in the image. This statistic will affect the voting power of the small object.

Grouping of a set of features can be carried out by finding a cluster of features that appears in the image of an object that are geometrically close to each other. If a group of features is similar with any other groups in the database, its weight in voting will be lower. Refractoring process may be carried out when a new image of object is added to the database. This process is mainly to find the cluster of features, group them together and assign a weight to each of them. It also finds out the most distinctive group of features that an object contains. An image containing this distinctive group means that there exists that object in high chance.

5.3 Extension of SIFT Object Recognition

Lowe [14, 13] suggested several extensions to his work. Firstly, the features extracted by the SIFT object recognition system are from monochrome in-

Figure 5.2: Matching using color descriptors over color images done by Fergus *et al.*

tensity images, so further distinctiveness could be derived from including illumination-invariant color descriptors and this has first been explored by Brown and Lowe [2], but detailed description on how to create a color descriptor is absent in that paper. Figure 5.2 shows a matching of invariant features between images using color descriptors. Secondly, local texture measures could be incorporated into feature descriptors. He believed that best results are likely to be obtained by matching many different types of features, which is capable to be implemented in invariant local feature approach. Thirdly, scale-invariant edge groupings can also be incorporated into feature descriptors such that local figure-ground discriminations would be done better at object boundaries. Finally, we could apply SIFT to generic object class recognition. Fergus *et al.* [4] has shown the potential of SIFT in recognizing generic classes of objects by unsupervised scale-invariant learning.

□ End of chapter.

Chapter 6

Conclusion and Future Work

In this paper, a survey of the recent researches on view-based object recognition and invariant local grayvalue features was given. Performance of several popular feature descriptors are evaluated and we found that SIFT feature descriptor performs the best comparing with other descriptors in the experiment. Some feature descriptor prototypes are implemented and presented in this paper. Another experiment done in a slightly different approach is presented. It aim to compare the performance of our local features and the SIFT features and to give another view of performance evaluation on the SIFT features. Some color constancy and invariance techniques are presented in "Color Invariant Local Features" chapter. These techniques may be applied by us to extend SIFT descriptor to color descriptor.

□ End of chapter.

Bibliography

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. 24(4):509–522, April 2002.

[2] M. Brown and D. Lowe. Invariant features from interest point groups. In *BMVC02*, page Poster Session, 2002.

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

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

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

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

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

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

[9] C. S. Krystian Mikolajczyk. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1615–1630, October 2005.

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

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

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

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

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[15] I. Matthews and S. Baker. Active appearance models revisited. *IJCV*, 60(2):135–164, November 2004.

[16] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. pages 525–531.

[17] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, 2005.

[18] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *Int. J. Comput. Vision*, 65(1-2):43–72, 2005.

[19] R. Nelson. 3-d recognition via 2-stage associative memory. In *Univ. of Rochester*, 1995.

[20] Y. K. Rahul. Pca-sift: A more distinctive representation for local image descriptors.

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

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

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

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

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