2016

An Autoencoder-Based Image Descriptor for Image Matching and Retrieval

Chenyang Zhao
Wright State University

An Autoencoder-Based Image Descriptor for Image Matching and Retrieval

A thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

by

Chenyang Zhao
Ph.D., Northwest University, 2012
B.S., Northwest University, 2006

2016
Wright State University

Wright State University
GRADUATE SCHOOL

April 13, 2016

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Chenyang Zhao ENTITLED An Autoencoder-Based Image Descriptor for Image Matching and Retrieval BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Arthur A. Goshtasby, Ph.D.
Dissertation Director

Michael Raymer, Ph.D.
Director, Department of Computer Science and Engineering

Robert E. W. Fyffe, Ph.D.
Vice President for Research and Dean of the Graduate School

Committee on Final Examination

Arthur A. Goshtasby, Ph.D.

Jack Jean, Ph.D.

Thomas Wischgoll, Ph.D.

Caroline GL Cao, Ph.D.

ABSTRACT

Zhao, Chenyang. Ph.D., Department of Computer Science and Engineering, Wright State University, 2016. *An Autoencoder-Based Image Descriptor for Image Matching and Retrieval.*

Local image features are needed in many computer vision applications. For this purpose, a large number of point detectors and descriptors have been developed throughout the years. However, creation of effective descriptors is still a topic of research. The Scale Invariant Feature Transform (SIFT) proposed by David Lowe is an example of a widely used image descriptor in image analysis and image retrieval. SIFT first detects interest points in an image based on a scale-space analysis. It then creates a descriptor for each detected point, encoding gradient information in a patch centered at the point. SIFT descriptors are scale and rotation invariant and provide a high matching rate; however, they are computationally too slow to be useful in many image analysis and retrieval applications.

Autoencoder is an effective computational method for representation learning. In this dissertation it is used to construct a low-dimensional representation for a high-dimensional data while preserving structural information within the data. In many computer vision applications, a high dimensional image data implies a high computational cost. The main motivation in this work is to improve the speed of image descriptors significantly without reducing their match ratings noticeably. A new descriptor is designed that is based on the autoencoder concept. The proposed descriptor can reduce the size and complexity of a descriptor significantly, considerably reducing the time required to find an object of interest in an image or retrieve a desired image from a database.

Contents

List of Figures

List of Tables

Acknowledgment

First, I would like to thank my advisor Dr. Arthur Goshtasby, for his patience, guidance, encouragement and continuous support throughout my Ph. D. studies.

I am pleased to thank my Ph.D. committee members: Dr. Caroline Cao, Dr. Jack Jean and Dr. Thomas Wischgoll. Their kindly guidance and help on my Ph. D. program.

Finally, I would like to thank my parents. All my achievements would not be possible without their love and support.

Dedicated to

This dissertation is dedicated to my husband and my son.

Introduction

Image features are needed in many computer vision applications, such as image matching (Tuytelaars and Van Gool, 2004), Image alignment (Szeliski, 2006), object recognition (Lowe, 1999), and image retrieval (Mikolajczyk and Schmid, 2001). Many image detectors and descriptors have been proposed throughout the years. Different image analysis problems require different image features. In this chapter, I will describe the motivation and contributions of this work. Then a brief overview of challenges in extracting image features is provided. Finally, the organization of this dissertation is provided.

1.1 Motivation

Image matching is a major area of research in computer vision and image analysis. It involves finding two similar images or image patches using various features. Image features are of two kind: global features and local features. Global features are such as statistical properties of an image, like standard deviation of the intensity distribution. Local image features can overcome limitations of global features, as they can distinguish image patterns that differ from their immediate neighborhoods (Tuytelaars and Mikolajczyk, 2008). Local features require that first interest points be detected in an image. They then provide properties of regions centered at the interest points.

Many methods have been proposed for detecting and describing local image features. Mikolajczyk and Schimid evaluated the performances of several feature detectors and de-

scriptors including steerable filter (Freeman and Adelson, 1991), moment invariants (Hu, 1962), scale invariant feature transform (Lowe, 1999) and cross-correlation (Mikolajczyk and Schmid, 2005). Based on the obtained experimental results, Scale Invariant Feature Transform (SIFT) has been found to provide consistently high performance measures while remaining robust under various image transformations (Mikolajczyk and Schmid, 2005).

The Scale Invariant Feature Transform (SIFT) proposed by Lowe (1999) is invariant to translation, rotation, and scale of an image, and so it has been widely used in object recognition, image matching, image classification and image retrieval. The SIFT detector finds interest points based on scale-space analysis, and is invariant to changes in an image's scale/resolution. SIFT descriptor encodes gradient information in an image patch centered at a point of interest. A 128-dimensional feature vector is generated using the histogram of the gradient directions within a patch. Image matching can then use it to find corresponding points in images. In addition to translation, scale and rotation invariance, SIFT is invariant to significant intensity changes. However, a SIFT descriptor is too large and is computationally very expensive.

Autoencoders have been used in representation learning and constructing low-dimensional representations of high-dimensional data (Vincent et al., 2008). This is done while preserving the structural information in data. In many computer vision tasks, the high dimensionality of input data implies a high computation cost. Autoencoders are a representation learning method that can learn data and transform data into lower dimensions.

As image databases grow in size, modern solutions to local feature-based image indexing and matching must not only be accurate but also highly efficient to remain viable (Trzcinski et al., 2015). This dissertation focuses on the design of a new image feature descriptor using the autoencoder concept and evaluates its application in image matching and retrieval. The new descriptor can reduce the size and complexity of an existing descriptor to improve its performance in image matching and image retrieval. Examples of the type of images considered for comparison and retrieval are shown in Figs. 1.1–1.4

1.2 Challenges to overcome

In this dissertation, the focus is on image matching and image retrieval using local image features. Different kinds of point detectors and descriptors have been proposed, but this is by no means a solved problem. Properties desired of descriptors are:

- Efficient

  The descriptors should be sufficiently fast to be practical. This is especially true when searching images in a large database.

- Distinct

  Different descriptors in an image should identify different objects.

- Positionally accurate

  A descriptor should accurately locate an object in an image.

- Insensitive to illumination changes - Figure 1.1

  Illumination changes across an image changes image intensities. Such changes influence object detection and recognition. For example, in Figure 1.1, it should be possible to detect a pair of shoes under different lighting conditions.

- Invariant to changes in scale/resolution of an image - Figure 1.2

  An image may show the desired object nearby or far away. It should be possible to identify an object irrespective of its size.

- Rotation invariant - Figure 1.3

  The object in an image may be rotated by an arbitrary angle, and the descriptors employed should be able to locate and recognize an object independent of its orientation.

Figure 1.1: Illumination changes.

Figure 1.2: Scale changes.

- Invariant to viewpoint changes - Figure 1.4

  Changing the position of the camera could change the appearance of an object in the image. A descriptor should be insensitive to viewpoint changes also.

Figure 1.3: Changes in orientation.

Figure 1.4: Viewpoint changes.

1.3 Contributions

The main contribution of this dissertation is to improve image matching speed without sacrificing its accuracy by designing an image descriptor based on autoencoders. Overall, the contributions of this work can be summarized as follows:

- To carry out a survey of current image descriptors for image matching and image retrieval.
- To develop a means to convert an existing descriptor to one that can be used in search more efficiently by using Autoencoder to reduce the dimension of the descriptor.
- To evaluate the performance of the new descriptor in image matching.
- To evaluate the performance of the proposed descriptor in image retrieval.
- To apply the codebook method to further improve the image retrieval speed.
- To summarize the obtained results and give directions to future research in the area.

1.4 Organization of the dissertation

In brief, Chapter 2 describes background summarizes existing literature in image matching and image retrieval. Chapters 3 – 5 detail the proposed descriptor and evaluate its performance experimentally. In Chapter 6, the results are summarized, concluding remarks are

made, and items for future research are introduced. Further details about the chapters are provided below.

Chapter 2: Background and related work

In this chapter, the basic concepts are reviewed, and related work on image matching and image retrieval are described. First, the SIFT descriptor and its extension which includes PCA-SIFT and SURF, is described. Then the concept of Content Based Image Retrieval(CBIR), using color, texture and shape features are reviewed.

Chapter 3: AED – Autoencoder based image descriptor

In this chapter, the novel descriptor proposed in this dissertation is detailed. First, an overview of the antoencoder is provided. Then, the details of the new descriptor are provided, including: (1) implementation of the SIFT descriptor, and (2) implementation of the proposed descriptor using the concept of autoencoder.

Chapter 4: Image matching using AED

In this chapter an overview of the datasets used in image matching is provided. Also described will be the procedures to test and evaluate the proposed autoencoder based descriptor in image matching. Detailed experimental results will be provided.

Chapter 5: Image retrieval using AED

This chapter presents the results obtained by the new descriptor in image retrieval. Also, use of the codebook method to improve image retrieval performance is described.

Chapter 6: Conclusions

After discussing the strengths and weaknesses for the proposed methods, ideas about future research is provided.

Background and related work

In this chapter we will review the background and work related to this dissertation. We first discuss the SIFT detector and descriptor and it's extensions. Then, the problem of image representation using color feature, texture feature, and shape features for image retrieval is reviewed.

2.1 Local image feature

Image matching is a fundamental problem in computer vision. A lot of researches have studied it for many decades, but it is still an unsolved problem. Thousands of images are generated daily, requiring an easy and fast method organize the images and access them later. An image can be represented by a set of features, so the main idea of image matching is to find correspondence between features in the images. In order to improve the image matching performance, various approaches have been proposed. The approaches usually have three components: feature detection, feature description, and feature matching. Local image feature detectors and descriptors can extract the unique points in an image and generated feature vectors for identifying similar points in two images.

The first step in the image matching process involves detecting interest points in an image. Feature detectors are used to find interest points, which are unique. Different feature point detectors have been proposed, including SUSAN point detector (Smith and Brady, 1997), Harris corner detector (Harris and Stephens, 1988), and SIFT point detector.

Different detectors can be used to detect different types of features such as corners, blobs, and points. The most important property of a point detector is its repeatability (Mikolajczyk and Schmid, 2001).

After interest points are extracted from an image, a feature descriptor is generated for each point describing the properties of its neighborhood. The easiest way to generate an image descriptor is to use the pixel values around the point. But, this will not produce a robust descriptor that will be insensitive to rotation and changes in scale of the image, as well as insensitive to changes in viewpoint and lighting. So the most import property of an image descriptor is for it to be invariant to different transformations.

Points are matched by minimizing the distance between their descriptors. The distance measure can be Euclidean distance or Mahalanobis distance. Representing images by a set of descriptors is an effective means of matching them. The performance of image matching mainly depends on its descriptors.

2.1.1 Scale-Invariant Feature Transform

The most successful image descriptor is known to be the Scale Invariant Feature Transform (SIFT) (Lowe, 1999). It extracts distinct and invariant features from an image for object recognition. SIFT is one of the most widely used image descriptors. Several SIFT based descriptors have been introduced including the PCA-SIFT descriptor by Ke and Sukthankar (2004), Color-SIFT Verma et al. (2010), GLOH Mikolajczyk and Schmid (2005), and ASIFT Morel and Yu (2009). These SIFT based image descriptors have been used in various image matching and retrieval problems.

The SIFT detector and descriptor consists of four steps:

- **Scale-space extrema detection:** SIFT first searches in scale-space using a Difference of Gaussian operator (DOG) to identify interest points. Given an image $I(x, y)$, a smoothed version of it is obtained by convolution with a Gaussian filter is obtained:

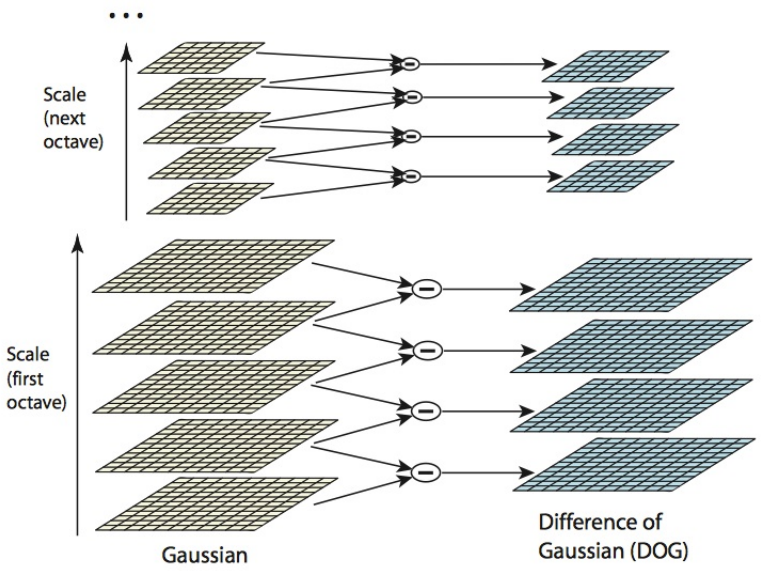


Figure 2.1: Difference-of Gaussians at different scales (Lowe 2004).

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

$G(x, y, \sigma)$ is a 2-D Gaussian of standard deviation σ:

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

The Laplacian of Gaussian is then approximated by a difference of two Gaussians:

$$D(x, y, \sigma) = J(x, y, k\sigma) - J(x, y, \sigma)$$

The constant $k = 1.61$ is a constant and represents the scale factor of the Gaussians in scale-space. Figure 2.1 shows construction of $D(x, y)$ from $J(x, y)$. In SIFT $k = 1.4$ is used. In order to detect extrema points from a stack of DOG images a point in the scale-space that is locally maximum within a $3 \times 3 \times 3$ neighborhood is taken to represent a key point, as shown in Figure 2.2.

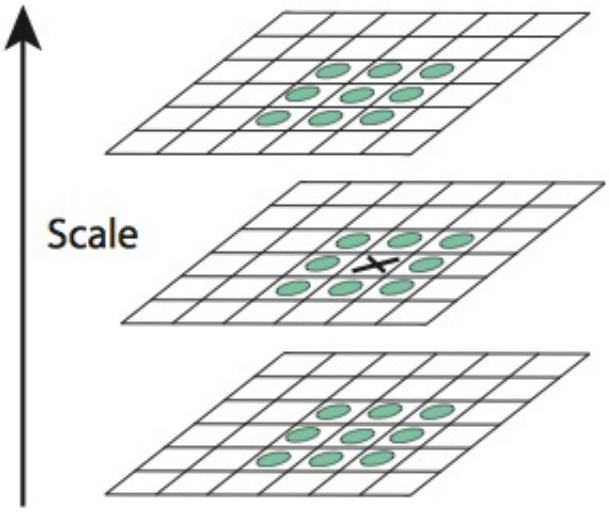


Figure 2.2: Comparison of DOG computed at a $3 \times 3 \times 3$ neighborhood. (Lowe 2004).

- **Keypoint localization:** The location and scale of each candidate point is determined and keypoints are selected based on measures of stability. Then, these locations are refined by discarding points of low contrast. Not all detected local extrema are keypoints, points that have low contrast or localized along an edge are rejected. A keypoint is determined by fitting a 3-D quadratic function using a second order Taylor expansion with the origin at the point of interest. Then, local extrema with low contrast that correspond to edges are discarded.

- **Orientation assignment:** A dominant orientation is assigned to each keypoint based on local image gradients. After the location of the keypoint is determined, a dominant orientation is assigned to each keypoint based on local image gradients. For each pixel of the region around the detected location the gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$ are computed using pixel differences :

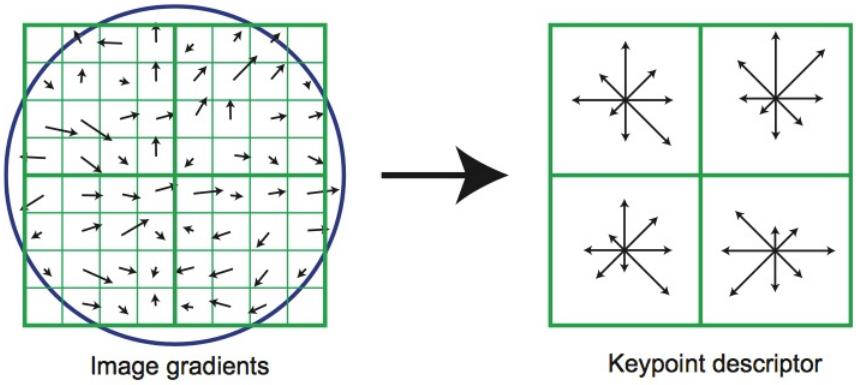


Figure 2.3: The creation of a SIFT descriptor (Lowe, 2004).

$$\begin{aligned} m(x,y) &= ((J(x+1,y) - J(x-1,y))^2 \\ &\quad +(J(x,y+1) - J(x,y-1))^2)^{\frac{1}{2}} \end{aligned}$$

- **Keypoint descriptor:** A descriptor is generated for each keypoint using local image gradients at the obtained scale. The window centered at a keypoint is subdivided into a grid of 4x4 subwindows and gradient directions within each subwindow are quantized into 8 orientations, creating a histogram with 8 bins for each subwindow, as shown in Figure 2.3. Overall, a feature vector with 128 values is generated, representing the descriptor for the keypoint. This descriptor is orientation invariant, because it is calculated relative to the dominant orientation. Finally, to achieve invariance against change in illumination, the descriptor is normalized to have unit magnitude.

The original SIFT algorithm was proposed in 1999. The descriptor is efficient to compute, and it is robust under image variations. Small improvements to it have been made over time. First, the problem of high dimensionality of the original SIFT descriptor

has been addressed by PCA-SIFT.

2.1.2 PCA-SIFT

PCA-SIFT is invented by Ke and Sukthankar (2004). It maps the SIFT descriptor to a smaller vector through Principal Components Analysis (PCA). PCA is a standard dimensionality reduction technique, which linearly-projects high-dimensional data into a low-dimensional feature space (Jolliffe, 2002). The basic idea is that data along lower dimensions that are most distinctive are kept, and the rest are considered noise and discarded.

PCA-SIFT uses the same interest points as SIFT. The PCA-SIFT descriptors are created by sampling local gradient content with a 41×41 image patch. The size of local patch is 41×41, the gradients of pixel are extracted along the x- and y-components. The extracted values are a gradient vector with $39 \times 39 \times 2 = 3042$ elements. This is the raw descriptor, which is then reduced with PCA.

PCA-SIFT builds the eigenspace by extracting gradients within patches from a large number of training images. Each image patch is treated as a point, then a covariance matrix is built from the points. After we created the covariance matrix, the PCA method can be used to the eigenvectors. The top 20 eigenvectors in the matrix are kept.

At the online stage, the 3042-element feature vector is reduced by projection to precomputed eigenspace. The descriptor is matched by using the Nearest-Neighbor method from among the training images.

2.1.3 SURF

Speed Up Robust Feature (SURF) is similar to SIFT and scale and rotation-invariant which has been proposed by Bay et al. (2006).

- SURF interest point detector

SURF detects keypoints at different image scales based on Hessian-matrix. First, it computes the integral image of the given image, which speeds up further calculations. For the integral image I, the value at point (x, y) is computed from:

$$I_{\sum}(x, y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j)$$

Given a point (x, y) in an image, the Hessian matrix defined by $H(x, \sigma)$ is computed from:

$$H(X, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix}$$

where $L_{xx}(x, \sigma)$, $L_{xy}(x, \sigma)$ and $L_{yy}(x, \sigma)$ represent convolution of various Gaussian second derivatives with the image (Bay et al., 2006). The Gaussian second derivates are approximated by box filters to obtain D_{xx}, D_{xy}, and D_{yy}, so the Hessian determinant can be defined by:

$$det(H_{approx}) = D_{xx}D_{yy} - 0.9D_{xy}^2$$

In order to localize the interest points, a non-maximum suppression in a $3 \times 3 \times 3$ neighborhood is applied (Bay et al., 2006).

- SURF descriptor

The SURF descriptor is similar to SIFT, which is based on the square region around the keypoint. First, an orientation is assigned to every keypoint, which is calculated by two summed wavelet responses. Then a square region is generated around the keypoint. In the second step, the square region can be split up into 4×4 sub-regions. The horizontal and vertical wavelet responses are computed for 5×5 sample grids. The responses of the Haar-wavelets in horizontal d_x and vertical d_y directions are summed over for each sub-region.

The sum of the absolute values $|d_x|$ and $|d_y|$ are also be computed. So, a 4-dimensional descriptor vector for each sub-region is:

$$V = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$$

The length of the descriptor vector is 64.

2.2 Content-based image retrieval

Content based image retrieval (CBIR) is one of the most important research fields in computer vision. Image retrieval is a challenging problem that is based on finding the image in a database that is most similar to an image of interest. Content-based image retrieval (CBIR) is the problem of retrieving a given query image from a image database by using image features. An example is given in Figure 2.4[1] (Smeulders et al., 2000).

2.2.1 Concept

With the development of digital image technology, image retrieval has become a very active research area in recent years. Image retrieval is to find relevant images from a database that best matches the query image. The image retrieval technology involves: Text-based image retrieval(TBIR) and Content-based image retrieval (CBIR).

The Text-based image retrieval can be traced back to late 1970s (Rui et al., 1999). Images are manually annotated and searched by text. If images are annotated correctly, search results can be very good, but the text based approach has some limitations. The first limitation is the amount labor needed to manually annotate all image, which is not always possible. The second limitation is the inaccuracy caused by the subjectivity of human perception. Even for the same image, different people can have different interpretations.

[1]The WANG dataset is available at http://wang.ist.psu.edu/docs/related/

Figure 2.4: Content-base image retrieval. These images are from the WANG dataset.

To overcome the limitations and drawbacks of Text-based image retrieval, Content-based image retrieval was introduced. CBIR uses the images content instead of text based keywords, which search similar image content in the database (Smeulders et al., 2000). An image form the database can be represented by a feature vector. The size of the feature vector is much smaller than the image it is representing. We can use the feature vector to compare similar images. The content based image retrieval usually has three steps:

- **Feature extration:** The features of query image and the images in the database are extracted. In this step, feature descriptors can generate a set of feature vector from both query and images in the database.

- **Similiarity search:** This step will compare the feature vectors between query image and images in database. The similarity can be decided by distance between feature vectors. The smaller the distance between two feature vectors, the more similar the two image are.

- **Index search:** The third step is to index and display the results.

2.2.2 Features for image retrieval

A feature is defined to capture a certain visual property of an image, either globally for the entire image or locally for a small group of pixels (Datta et al., 2008). Image features can affect image retrieval performance, so successful extraction of image features is important for CBIR. The image features such as color, shape, and texture can be used for matching the query image and other images. Color features include color histograms, color moments, and correlograms. Texture features include Tamura texture, co-occurrence matrices and Gabor filters. Shape features consist of Fourier descriptors and Hu moments. The features are extracted automatically using computer vision techniques, and using a similarity to match images (Rui et al., 1999). In order to improve the image retrieval performance, various combinations of features have been proposed.

Color features

Color features are simple and easy to compute. They can be represented by color histograms, which are widely used in Content-Based Image Retrieval. Color features are relatively robust to background changes and are independent of image size and orientation (Rui et al., 1999). The common representations for color feature are color histograms (Swain and Ballard, 1991), color moments (Stricker and Orengo, 1995), and color sets (Smith and Chang, 1995).

- Color histograms

  Color histogram was introduced by Swain and Ballard (Swain and Ballard, 1991). It gives an estimation of the distribution of colors in an image. Color histograms have been widely used in image retrieval, because they are trivial to compute, and tend to be robust against small changes in camera viewpoint

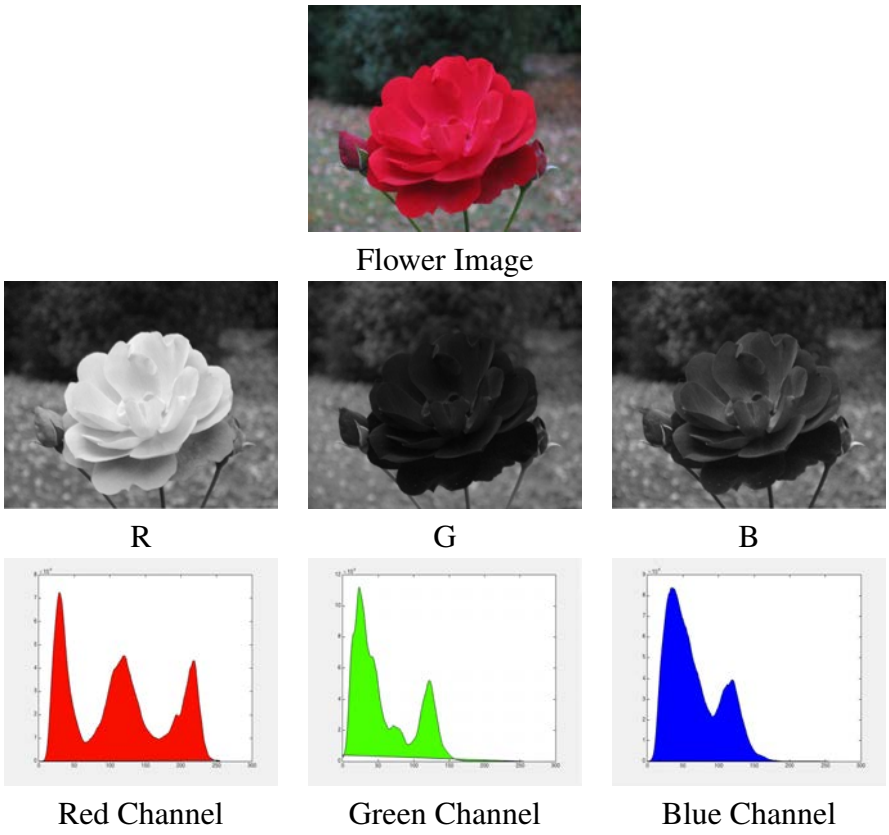


Figure 2.5: A flower image and its color histograms.

(Pass and Zabih, 1996). The color histogram can be computed as three independent color distributions. Given a Color space defined by some color axes, the color histogram is obtained by discrtizing the image colors and counting the number of times each discrete color occurs in the image array (Swain and Ballard, 1991). The color histogram method has three steps. First, the color space is partitioned into cells. The color space can be RGB or HSV. Then, each cell associated with a histogram bin, and the number of image pixels within each cell is counted and stored in the corresponding histogram bin. An example is given in Figure 2.5. The drawback of the color histogram method is that different images could produce the same color histogram(s).

- Color moments

Stricker and Orengo (Stricker and Orengo, 1995) proposed color moments for measuring the color similarity between different images. Color moments are used to represent color distribution in images. Stricker and Orengo used the first order (mean), the second order (variance) and the third order (skewness) moments of color components. The first three central moments are defined by:

Mean: Mean is the average color of the image.

$$\mu_i = \frac{1}{N}\sum_{j=1}^{N} f_{ij}$$

Standard Deviation: It is the square root of the variance of the distribution.

$$\sigma_i = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(f_{ij} - \mu_i)^2}$$

Skewness: It is a measure of the degree of asymmetry in the distribution.

$$\xi_i = \sqrt[3]{\frac{1}{N}\sum_{j=1}^{N}(f_{ij} - \mu_i)^3}$$

Where f_{ij} is the value of the i-th color component of the image at pixel j and N is the number of pixels in the image.

Color moments have better performances than color histograms, but it is sensitive to illumination changes. Beside the color histogram and color moments, other color feature extraction methods have been introduced, include Color Sets (Smith and Chang, 1995) and Coherence Vector (Pass and Zabih, 1996) to overcome the lack of spatial information in a created descriptor.

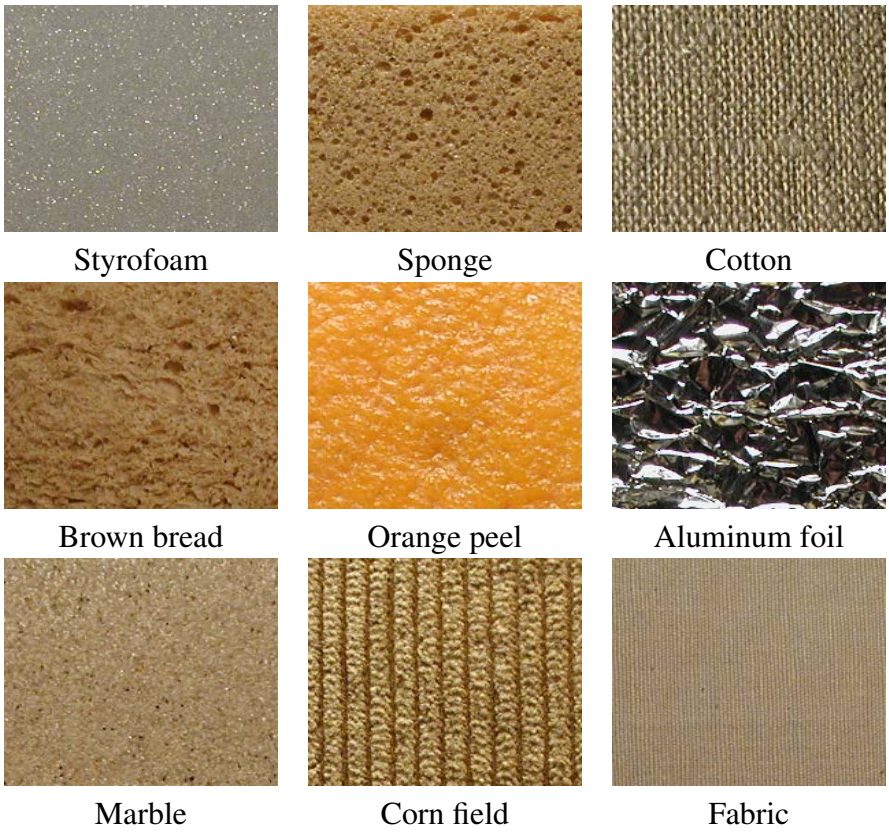


Figure 2.6: Some examples of texture features. These images are from KTH-TIPS image dataset.

Texture features

Texture is a visual patterns that have properties of homogeneity that do not result from the presence of only a single color or intensity (Smith and Chang, 1996a). Texture is an important image feature, it can reflects information of spatial and structure. Texture is a characteristic of an image that can provide a higher-order description of the image and includes information about the satial distribution of tonal variations or gray tones (Haralick, 1979). So texture feature can be widely used in image retrieval, user can search the similar texture image by texture feature. There are a lot of texture feature extraction methods, which included Tamura texture (Tamura et al., 1978), co-occurrence matrix (Haralick et al., 1973) and Gabor filtering (Manjunath et al., 2002) .

- Tamura texture features

Tamura features (Tamura et al., 1978) characterize low-level statistical properties of textures. The six Tamura features are coarseness, directionality, regularity, contrast, line-likeness, contrast and roughness. The first three features have been shown to significant than the last three. The last three features are related to the first three and do not add much to the effectiveness of a texture description (Liu et al., 2007).

Coarseness: Coarseness is the most fundamental texture feature in image retrieval. coarseness can be described by coarse or fine.

$$A_k(x,y) = \sum_{i=x-2^{k-1}}^{x+2^{k-1}-1} \sum_{j=y-2^{k-1}}^{y+2^{k-1}-1} g(i,j)/2^{2k}$$

$$F_{crs} = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} S_{best}(i,j)$$

Contrast: Contrast is used to measure the changing quality of an image. The contrast represents the difference between a pixel and other pixels with its neighborhood.

$$F_{con} = \frac{\sigma}{\alpha_4^{1/4}}$$

Directionality: Directionality is a global property over a region. A directional texture has one or more orientations for a pattern, wheras an isotropic texture has no orientation.

$$F_{dir} = \sum_{p}^{n_p} \sum_{\phi \in w_p} (\phi - \phi_p)^2 H_D(\phi)$$

- Co-occurrence matrix

Haralick et al. (Haralick et al., 1973) proposed the co-occurrence matrix to represent the texture features. It is a traditional method for texture information, and has been used in

image retrieval. The method uses the gray level co-occurrence matrices to extract second-order statistics from an image (Howarth and Rüger, 2004). First, a co-occurrence matrix is calculated by the distance and orientation between image pixels. Secondly, the texture feature can be represented by statistics from the co-occurrence matrix. The features calculated form the co-occurrence matrix are:

Energy: Energy is the sum of squared elements in the co-occurrence matrix.

$$\sum_i \sum_j P^2(i,j)$$

Entropy: Entropy gives measures of informativeness of the image.

$$\sum_i \sum_j P(i,j) log P(i,j)$$

Contrast: Contrast measures intensity between a pixel and pixels in its neighborhood when measures over the entire image.

$$\sum_i \sum_j (i-j)^2 P(i,j)$$

Homogeneity: It describes how slowly intensities in an image change.

$$\sum_i \sum_j \frac{P(i,j)}{1+ \mid i-j \mid}$$

- Gabor features

These are different filter responses, known as Gabor features, are commonly used in image retrieval, and are shown to produce high matching rates (Manjunath et al., 2002). A set of Gabor filters to extract texture features from an image $I(x,y)$, can be obtained from:

$$W_{mn}(x,y) = \int I(x,y) g^*_{mn}(x-x_1, y-y_1) dx_1 dy_1$$

The feature vector can be represented by the mean and standard deviation of the magnitude of $\mid W_{mn} \mid$.

Shape features

Some problems in image retrieval require shape features. Shape features provide important characteristics of some images. An object may have a variety of colors, but its shape is always the same. Shape features have proven to be useful in image retrieval.

There are generally two types of shape features: boundary-based features and region-based features. The boundary-based features use only the outer boundary of a shape. Fourier descriptor uses the Fourier transformed boundary as the shape feature. Hu proposed region based invariant moments, the moments are invariant to rotation and scale.

- Fourier shape descriptors

The Fourier descriptor use the Foruier transformed boundary for the shape feature. The discrete Fourier transform is defined by:

$$a_n = \frac{1}{N} \sum_{t=0}^{N-1} u(t) exp(-j2\pi nt/N) \; n \in \mathbb{Z}$$

The coefficients a_n are used to as the Fourier descriptors of the shape.

- Moment invariants

Hu (1962) used the moment invariants of an image to address the pattern recognition problem. Moment invariants are based on an object's shape in an image. The feature vectors can be calculated by the moment invariants. The central moments for an image I are computed from:

$$\mu_{p,q} = \sum_{(x,y) \in R} (x - x_c)^p (y - y_c)^q$$

(x_c, y_x) is the center of gravity of the image. Normalized central moments are then defined by:

$$\eta_{p,q} = \frac{\mu_{p,q}}{\mu_{0,0}^{\gamma}}, \gamma = \frac{p+q+2}{2}$$

Hu uses nonlinear combinations of the lower order normalized central moments to create seven moments (Hu, 1962), described by M1 – M6 as absolute orthogonal invariants, M7 as a skew orthogonal invariant. These invariant moments have been shown to be invariant to translation, rotation, and scaling. The invariant moments are:

$$\phi_1 = \mu_{2,0} + \mu_{0,2}$$

$$\phi_2 = (\mu_{2,0} - \mu_{0,2})^2 + 4\mu_{1,1}^2$$

$$\phi_3 = (\mu_{3,0} - 3\mu_{1,2})^2 + (\mu_{0,3} - 3\mu_{2,1})^2$$

$$\phi_4 = (\mu_{3,0} + \mu_{1,2})^2 + (\mu_{0,3} + \mu_{2,1})^2$$

$$\begin{aligned}\phi_5 = {} & (\mu_{3,0} - 3\mu_{1,2})(\mu_{3,0} + \mu_{1,2})[(\mu_{3,0} + \mu_{1,2})^2 - 3(\mu_{0,3} + \mu_{2,1})^2] \\ & +(\mu_{0,3} - 3\mu_{2,1})(\mu_{0,3} + \mu_{2,1})[(\mu_{0,3} + \mu_{1,2})^2 - 3(\mu_{3,0} + \mu_{2,1})^2]\end{aligned}$$

$$\phi_6 = (\mu_{2,0} + \mu_{0,2})[(\mu_{3,0} + \mu_{1,2})^2 - (\mu_{0,3} + \mu_{2,1})^2] + 4\mu_{1,1}(\mu_{3,0} + \mu_{2,1})(\mu_{0,3} + \mu_{2,1})$$

$$\begin{aligned}\phi_7 = {} & (3\mu_{2,1} - \mu_{0,3})(\mu_{3,0} + \mu_{1,2})[(\mu_{3,0} + \mu_{1,2})^2 - 3(\mu_{0,3} + \mu_{2,1})^2] \\ & +(\mu_{3,0} - 3\mu_{2,1})(\mu_{0,3} + \mu_{2,1})[(\mu_{0,3} + \mu_{2,1})^2 - 3(\mu_{3,0} + \mu_{1,2})^2]\end{aligned}$$

2.2.3 Similarity measure

In image retrieval, we need to compare the similarity between the query image and images in the database. In order to find the matched images, the retrieval result is ranked by the obtained similarity measures. A number of similarity measures have been used in image retrieval. Some of these measures will be mentioned below.

L_1 and L_2 norms

The L_1 and L_2 norms are commonly used in image retrieval applications. L_1 measures the absolute distance between two feature vectors, L_2 can be calculated by the square of the sum of differences between two feature vectors.

$$L_1 = \sum_{i=1}^{N} \mid A_i - B_i \mid$$

$$L_2 = \sum_{i=1}^{N} (A_i - B_i)^2$$

Histogram intersection

Histogram intersection (Swain and Ballard, 1991) is used to measure the similarity between two color histograms.

$$\sum_{j=1}^{N} min(I_j, Q_j) / \sum_{j=1}^{N} Q_j$$

Quadratic form

Quadratic form (Hafner et al., 1995) is a bin-to-bin distance. It depends on the number of bins in the histograms being compared. Denoting two histograms by Q and I and letting A be the bin-similarity matrix, the distance between the histograms is computed from:

$$D = (Q - I)^t A (Q - I)$$

Mahalanobis distance

Mahalanobis distance (Bay et al., 2006) can be used to compare two feature vectors with some dependency between vector components.

$$D_{mahal} = (A - B)^t C^{-1} (A - B)$$

where C is the covariance matrix of the feature vectors.

AED – Autoencoder based image descriptor

In this chapter, first, the autoencoder concept and it's extensions are described. Then, the offline training process for autoencoers of image descriptors is outlined. Finally, a new image descriptor based on autoencoders is detailed.

3.1 What is an autoencoder?

An autoencoder is a specific form of an artificial neural network (Hinton and Salakhutdinov, 2006). The purpose of an autoencoder is to learn another representation of input data, in compressed or sparse representation. More specifically, an autoencoder is an unsupervised learning method that sets the target values to be equal to the input values, i.e., $y_i = x^{(i)}$. Generally, an autoencoder contains one input layer, one or more hidden layers, and one output layer, which has exactly the same number of units as the input layer, as shown in Fig. 3.1. Different layers of the network apply a series of transformations (non-linear in most cases) to input data. The hidden layers work with different representations of the input data.

Functionally, an autoencoder contains two components during the training process, an *encoder* and a *decoder*. The encoder is used to encode the input data to the desired *lossy*

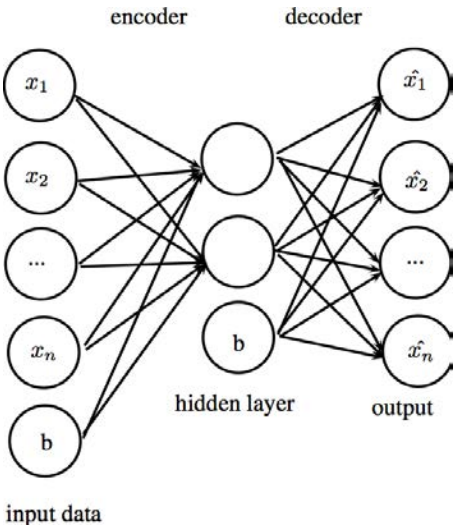


Figure 3.1: The structure of a simple 3-layer autoencoder.

compressed (or sparse) representation by applying transformations h_j (the j layer), while the decoder decodes this compressed representation to an approximation of the inputs $\hat{x}^{(i)}$, with $\hat{x}^{(i)}$ as close to $x^{(i)}$ as possible. Usually, the (non-linear) transformation h is a sigmoid function, i.e., the logistic function $h(z) = \frac{1}{1+\exp(-z)}$, where $z = Wx + b$ and W is a weight matrix, and b is a bias vector. It is typically the weight matrix W' in the decoder, which is the transpose of the weight matrix W, in the encoder, i.e.,

$$W' = W^T$$

In this case, the autoencoder is said to have *tied weights*. In the training phase of an autoencoder, the parameters W and b are optimized such that the average reconstruction error is minimized.

The reconstruction error is used to measure the similarity between $\hat{x}^{(i)}$ and $x^{(i)}$, which can be measured in many ways. A simple but commonly used one is the squared error, that is, for any input $x^{(i)}$,

$$L(x^{(i)}) = \sum_{k=1}^{d} (x_k^{(i)} - \hat{x}_k^{(i)})^2$$

where $x^{(i)}$ is a d-dimensional vector. Another popular measure of the reconstruction error is the *KL-divergence*,

$$KL(x^{(i)}) = -\sum_{k=1}^{d}(x_k^{(i)} \log \hat{x}_k^{(i)}) + (1 - x_k^{(i)}) \log(1 - \hat{x}_k^{(i)})$$

which is often be used to measure the similarity of the probability distributions. In this research, the squared error is used.

Note that if there is a need to build an autoencoder with only one *linear* hidden layer (with k hidden units), to minimize the squared error is equivalent to learn a projection of the span of the first k principal components of the data. Thus, the resulting autoencoder is an alternative implementation of PCA approach. However, if an autoencoder is consisted by multiple *nonlinear* hidden layers, it is able to learn more complicated transformations and behaves much different than PCA.

The backpropagation algorithm is usually applied to train an autoencoder by propagating the error information (gradients) from the output layer back to the input layer (Hagan and Menhaj, 1994) by using a gradient descent approach (or its variants, such as stochastic gradient descent (SGD) and L-BFGS). More specifically, the gradient of the reconstruction error is "back propagated" down the neural network structure from output layer to input layer, adjusting the model parameters (weight matrix and bias vector) along the way. Back propagation is a supervised learning approach that requires the labeled data. For training of an autoencoder, the "label" of an instant $x^{(i)}$ is just itself, i.e., $y^{(i)} = x^{(i)}$.

3.1.1 Stacked autoencoder

For an autoencoder with multiple (more than one) hidden layers, it is difficult to optimize the weights by using the back propagation algorithm. For one, the back propagation algorithm does not work well in a deep neural network structure due to the diffusion of gradients problem. For another, the optimization may get stuck at a poor local minimum when the

neural network is initialized with random weights. To alleviate this problem, Hinton and Salakhutdinov (2006) proposed a stacked autoencoder approach. The motivation behind this is that gradient descent tends to work well when the initial weights are close to a good solution. To find this *good solution* of initial weights, a layer-wise pre-training process is used. Specifically, this process consists of the following steps:

1. Train the bottom-most autoencoder, which consists of the input layer and the bottom-most hidden layer.

2. Remove the decoder layer of the trained autoencoder; and then construct a new autoencoder by taking the hidden layer of the previous autoencoder as input.

3. Train the new autoencoder.

4. Repeat Step 2–3 until all weights are pre-trained.

After the pre-training process, the next stage is to fine tune the weights of the network in a supervised fashion using the back propagation algorithm. In this stage, the weights obtained from pre-training are assigned as the initial weights, and the goal is to minimize the reconstruction error.

3.1.2 Denoising autoencoder

A denoising autoencoder (Vincent et al., 2010) is an autoencoder capable of handling noise corruptions. The main idea here is to allow a denoising autoencoder to force the hidden layer to learn more robust features by involving the reconstruction of a clean input data from a partially corrupted one. In (Vincent et al., 2010), the denoising autoencoder is described as a stochastic process: a denoising autoencoder intends to encode the input data while preserving its information, then undoing the corruption process, stochastically applied to the input of an autoencoder. That is, the stochastic corruptioin process randomly masks

partial elements of x as zeros. The goal is to predict the corrupted (masked) input form the uncorrupted input by randomly selecting subsets of masked patterns.

It is simple to convert a traditional autoencoder to a denoising autoencoder. First, a stochastic corruption is applied to input data. A typical method for doing this is to randomly set partial inputs to zero. Next, compute the reconstruction error measured on the original uncorrupted input data.

In recent years, denoising autoencoder techniques are widely used in a broad range of representation learning problems. There are three major benefits of a denoising autoencoder: 1) the resulting (compressed) representation of the input data is more robust to small irrelevant changes in input; 2) converting a traditional autoencoder to a denoising autoencoder is very simple and is easy to implement, with almost no extra computional costs; 3) the denoising autoencoder technique can be easily employed to training of a stacked autoencoder. Such encoders are called stacked denoising autoencoders.

3.1.3 Other autoencoder variants

A popular autoencoder variant is the Sparse Autoencoder (Goodfellow et al., 2009). Instead of learning a compressed representation of the input data, a sparse autoencoder tries to learn a sparse representation of the input. This is usually a sparse representation, which has a higher dimension than that of the original input. More specifically, the algorithm forces a sparsity constraint on the hidden units, i.e., the hidden units are constrained to be 0 most of the time during training. In order to achieve this goal, there is an additional extra penalty term in the objective that penalizes the hidden units whose value is deviated significantly from a predefined *sparsity parameter*. While sparse autoencoders are well researched, it will not be used in this work because the objective of this research is to learn a *compressed* representation of input data.

The Convolutional Autoencoder (Masci et al., 2011) is a coding technique that extends the idea from the Convolutional Neural Network (CNN) (Lawrence et al., 1997). The

network structure in a convolutional autoencoder consisting of three layers: a convolutional layer, a max-pooling layer, and an output layer. The main advantage of the convolutional autoencoder is that it scales well to high-dimensional inputs since the number of free parameters describing their shared weights does not depend on the input dimensionality (Masci et al., 2011). As the input dimensionality is not an issue in this research (typically 3042 elements), the fully connected neural network is used for simplicity.

Besides the denoising autoencoder is the contractive autoencoder (Rifai et al., 2011), which is a learning method that is robust to irrelevant changes in input. The contractive autoencoder achieves the effect of contraction by adding a penalty term to the objective function. This penalty term is the *Frobenius norm* of the Jacobean matrix of the hidden layer representing the input. The contractive autoencoder is known to be good at learning an over-complete (the latent representation has a higher dimension than the original input), thus it is not suitable for the learning problem in this research.

3.2 Offline autoencoder training

An autoencoder enables us to project a high-dimensional data into a compressed low-dimensional representation. For the applications in this research, an autoencoder will be trained offline and stored. The offline training process can be summarized in the following steps:

1. Choose images for offline autoencoder training.
2. Use SIFT detector to extract interest points in these images.
3. Extract the local gradient patch centered at each interest point, and generate the gradient vector for each interest point.
4. Train the autoencoder with the gradient patch as the input.

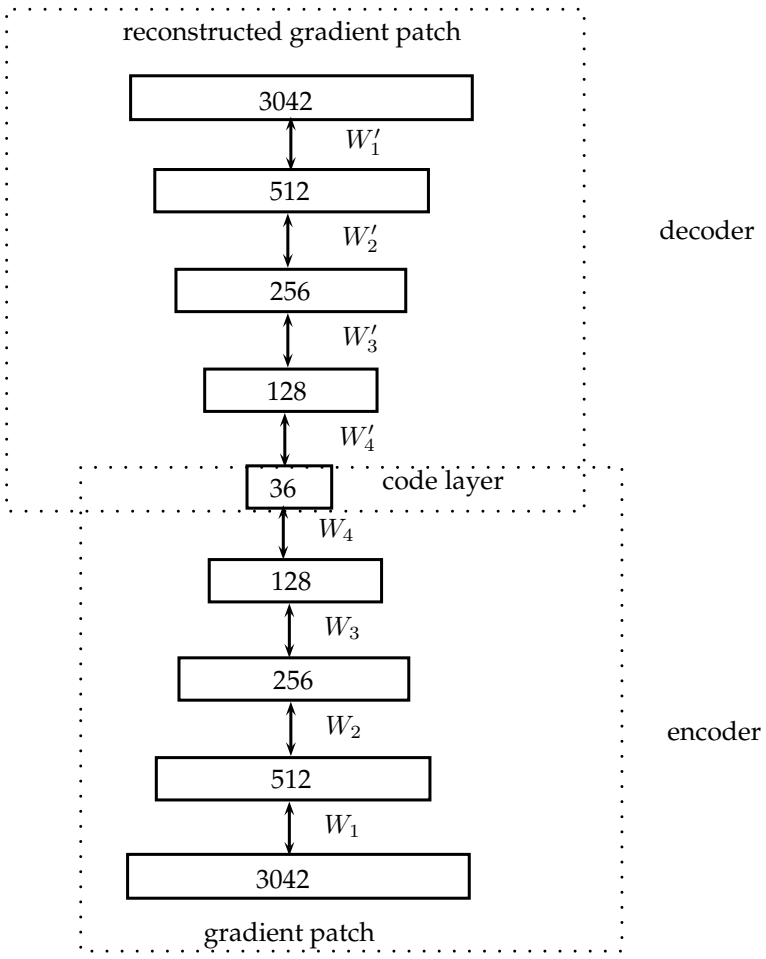


Figure 3.2: The structure of the proposed autoencoder.

To prepare data for autoencoder training, first over 40000 41×41 patches from diverse images were collected. These images or patches are not used later in the evaluation. Then, the horizontal and vertical gradients at patches are calculated and stored in a $2 \times 39 \times 39 = 3042$ vector. Finally, this vector is normalized so each feature value falls in the range $[0, 1]$.

The stacked denoising autoencoder is chosen to train the projection model as the compressed representation that learns by a denoising autoencoder, which is robust to small irrelevant changes in input, a very important property of the proposed autoencoder. The autoencoder consisted of an encoder with layers of 3042-512-256-128-36, as shown in Figure 3.2. The 36-dimensional vector is the code layer of the encoder part of the deep

autoencoder.

In the code layer, the 36 real-valued numbers are the encoded version of the 3042 gradient patch. The decoder part (which is symmetric to the encoder) of a deep autoencoder learns to decode the compressed vector, which becomes the reconstructed gradient patch as it makes its way back. Figure 3.2 illustrates this.

The pre-training process is shown in Figure 3.3. We first train the bottom-most autoencoder, where the corresponding encoder with layers of 3042-512, and the decoder with layers of 512-3042. The goal is to optimize the weight matrix W_1 such that the reconstruction error, which is measured by

$$L(x^{(i)}) = \sum_{k=1}^{d} (x_k^{(i)} - \hat{x}_k^{(i)})^2$$

is minimized. In this step, the gradient patch is encoded to a 512-dimensional vector, and the transpose of W_1 is used to decode this 512-dimensional vector to the reconstructed gradient patch with 3042 units. After training of this bottom-most autoencoder, a new autoencoder is constructed by taking the 512-dimensional vector as input, with layers of 512-256 in its encoder part and 256-512 in its decoder part. The goal here is to try to reconstruct this 512-dimensional vector such that the reconstruction error is as low as possible. The third and the fourth autoencoders are trained in the same way, and finally the weights W_1, W_2, W_3, and W_4 are obtained. These weights are used to initialize the autoencoder in Figure 3.2.

After the pre-training process, a global fine-tuning process is applied to fine-tune the weights such that the overall reconstruction error is minimized, as shown in Figure **?**. From a high level perspective, the fine tuning process treats all layers of the autoencoder as a single model, so that all the weights in the autoencoder are tuned at each iteration. As pre-training, the reconstruction error is measured by squared error too.

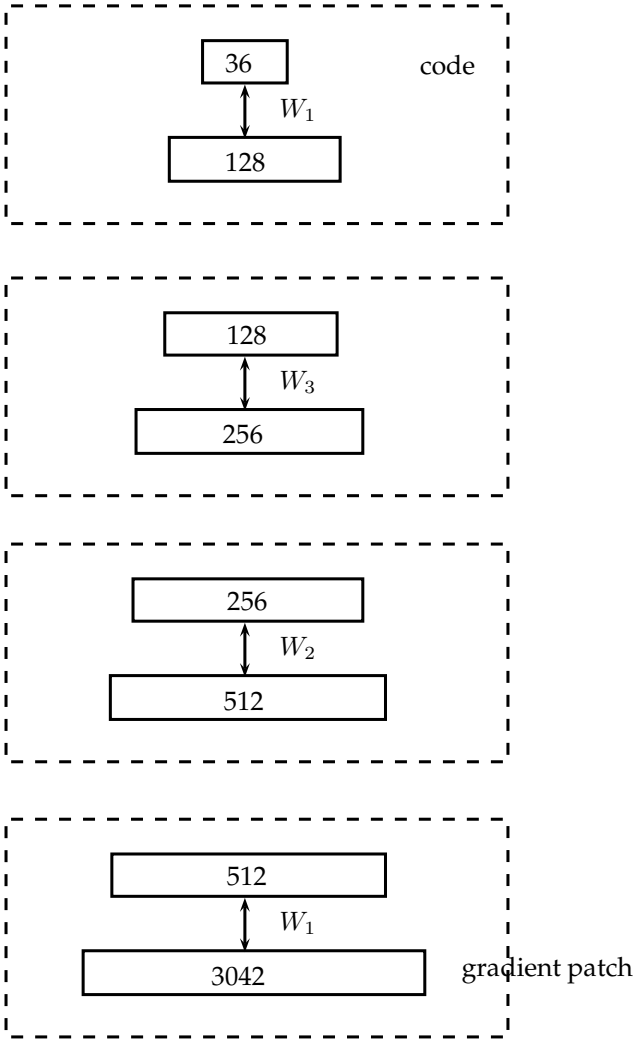


Figure 3.3: The pretraining process.

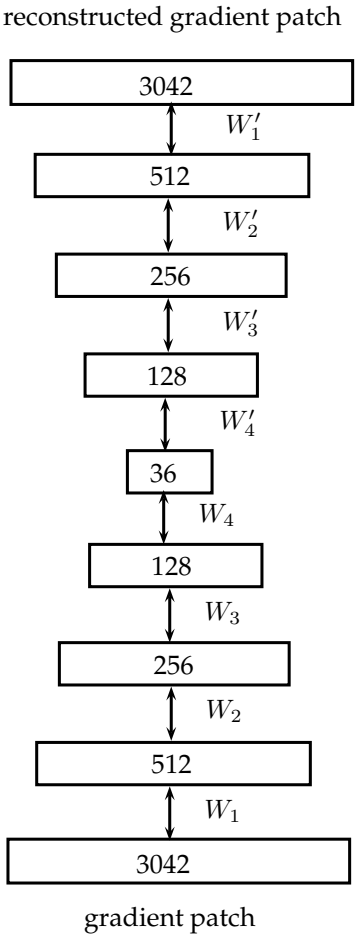


Figure 3.4: The fine tuning process.

The mini-batch stochastic gradient descent (SGD) algorithm is used to perform the optimization, where the batch size is 100. During the pre-training and fine-tuning, the weights are updated after each mini-batch iteration. For pre-training, each hidden layer is trained for 500 epochs through the entire input, weights are initialized with small random numbers subject to Gaussian distribution with 0 mean and 0.1 standard deviation, and the learning rate is set to be 0.05. For fine-tuning, the hidden layers are trained for 700 epochs through the entire training gradient patches, weights are initialized with the resulting weights of pre-training, and the learning is set to be 0.03. For both pre-training and fine-tuning, the logistic function is used as the sigmoid transformation function, and 30% of input are masked as 0 to perform the denoising autoencoder.

3.3 Keypoint description and matching

After offline training of the autoencoder, the encoder part is used to project a gradient patch to a compressed representation, which consists of 36 real values. This 36-dimensional vector can be used with the same matching algorithm as SIFT, but note that this compressed vector is significantly smaller than the feature vector of SIFT, which has 128 dimensions. Thus, the proposed AED autoencoder speeds up feature matching by a factor of 3 compared to the original SIFT method.

More specifically, the first step in AED is to detect the interest points. For that we use the difference-of-Gaussian detector, which is similar to SIFT. For SIFT descriptor, orientation histogram is utilized as the feature vector to represent the features at a keypoint. AED uses autoencode of the histogram to represent the gradient patch. For each detected interest point, the feature extraction window is used to extract the gradient patch centered at the keypoint. Then the patch is rotated so its main orientation aligns with the x-axis, and scaled according to the keypoint scale. For an image patch with size 39×39, the feature vector has 3042-elements ($39 \times 39 \times 2$). A 3042-512-512-256-256-128-36 autoencoder is

trained to extract feature vectors from the image. The trained autoencoder, as described in the last section, is tested on new images. By reducing the dimension of the feature vector to 36, a significantly smaller feature vector compared to standard SIFT feature with 128-element vectors is obtained.

After generation of AED feature vectors in an image, nearest-neighbor with distance ratio (NNDR) and Euclidean distance between normalized feature descriptors is used in matching, with the requirement that the nearest neighbor distance must be different by a certain percentage of the second nearest neighbor distance to consider a match unique.

Image matching using AED

In this section, we present experimental results comparing the performance of SIFT and AED in different conditions: change in scale and rotation, change in image blur, change in illumination and affine transformation. In order to evaluate the performance of AED, we also add PCA-SIFT (Ke and Sukthankar, 2004) to the comparison. PCA-SIFT is a variant of SIFT which tries to handle the high computational cost issues by mapping the gradient field to a vector based on the concept of Principal Components Analysis (PCA).

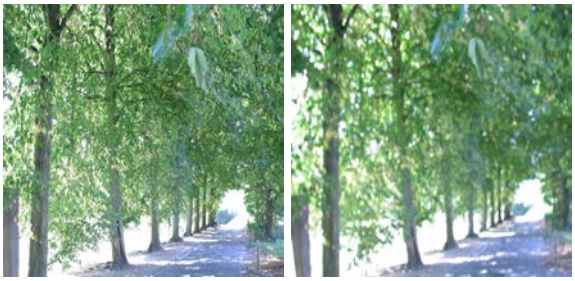
4.1 Dataset

The standard Mikolajczyx database [1] was used to evaluate the proposed descriptor under different transformations, such as image blur, viewpoint change, and degradations caused by jpeg compression. This dataset contains 8 groups of images, some of which are shown in Figure 4.1.

4.2 Evaluation criteria

In recent years, precision and recall have become popular evaluation metrics for image matching (Smith and Chang, 1996b). In the AED-based image matching, the performance

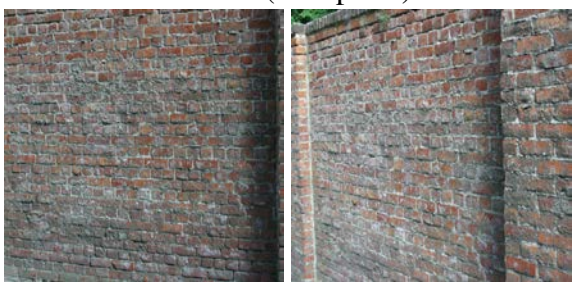
[1] *The dataset is available at http: //www. robots.ox.ac.uk./˜vgg/research/affine/



Trees (blur)

Graffiti (viewpoint)

UBC (JPEG compression)



Wall (viewpoint)

Bark (zoom+rotation)

Leuven (light)

Bike (blur)

Boat (zoom+rotation)

Figure 4.1: Images from the Mikolajczyx dataset

is evaluated in terms of precision and recall of the matching method. Precision and recall are based on the number of correct and false matches between two images. The standard definitions of these two measures are:

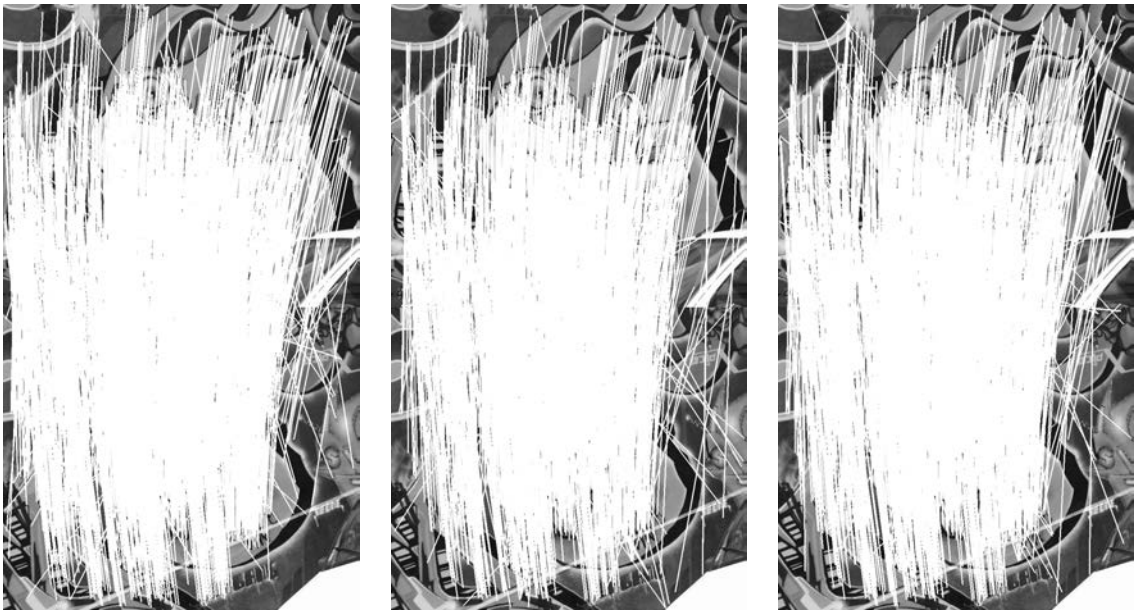
$$Precision = \frac{\text{Number of true-positives}}{\text{Total number of matches}}$$

$$Recall = \frac{\text{Number of true-positives}}{\text{Total number of positives}}$$

4.3 Performance evaluation under different conditions

In our experiments, test images are taken from the Graffiti dataset to evaluate the descriptors. Six different changes in imaging conditions are evaluated: affine transformation; scale and rotation changes; image blur; illumination changes; viewpoint change; and image compression. In the viewpoint change test, the camera is varied from a front-parallel view to one with foreshortening at about 20 degrees to the camera. The scale change and blur images are acquired by varying the camera zoom and focus respectively. The scale changes are about a factor of four. The light changes are introduced by varying the camera aperture (Mikolajczyk and Schmid, 2005).

In order to make a fair comparison, we determined good values for the dimensionality of the feature space n for each algorithm. As suggested in (Lowe, 1999) and (Ke and Sukthankar, 2004), we set n to 128 and 36 for standard SIFT and PCA-SIFT respectively. For our proposed AED algorithm, as same as PCA-SIFT, we set n to 36. Low dimensionality means PCA-SIFT and AED have significantly reduced the computation cost compared to SIFT. In order to show the matching results clearly, we manually set the thresholds to $m = 10$ matching pairs for each algorithm's return. The effect of feature point matching compared with SIFT, PCA-SIFT and AED is shown in Figure 4.2-4.14.

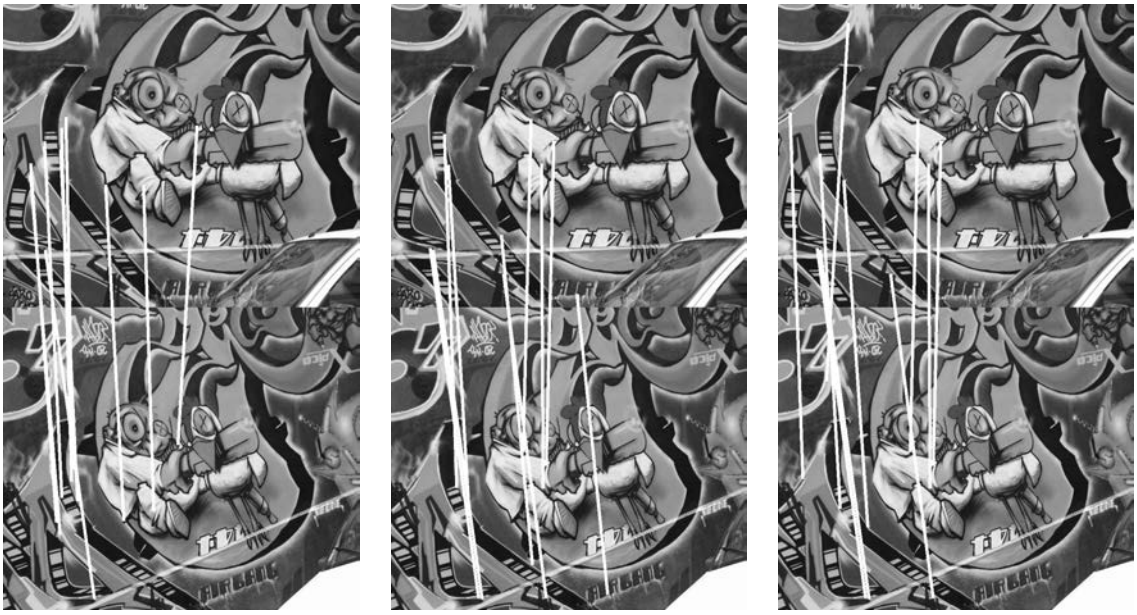


SIFT PCA-SIFT AED

Figure 4.2: Connecting all matching keypoints in best-matching images under affine invariance.

Affine invariance: The first set of experiments is conducted on the set of graffiti image for investigating the performance of SIFT, PCA-SIFT and AED under affine invariance, where each image has a different viewpoint. Figures 4.2, 4.4, and 4.6 show the matching keypoints between the first and the second, the third, and the fourth images in the graffiti image set, for SIFT, PCA-SIFT and AED respectively. For the sake of clarity, we only choose the first 10 matching keypoints, as shown in figure 4.3, 4.5, and 4.7. As we can see from Figures 4.3 and 4.5, all 10 matched keypoints are correct for each algorithm since the second and third images are only slight rotated with respect to the first image. As the fourth image rotates more than the second and the third images, correctly matching this image to the first image is more difficult. As shown in Figure 4.7, the number of matched keypoints extracted by SIFT, PCA-SIFT and AED are 4, 8 and 10, respectively. The results indicate that AED is very suitable for image with affine geometric differences.

Scale and rotation invariance: The second set of experiments is conducted on a set of bark images for investigating the performances of SIFT, PCA-SIFT and AED under



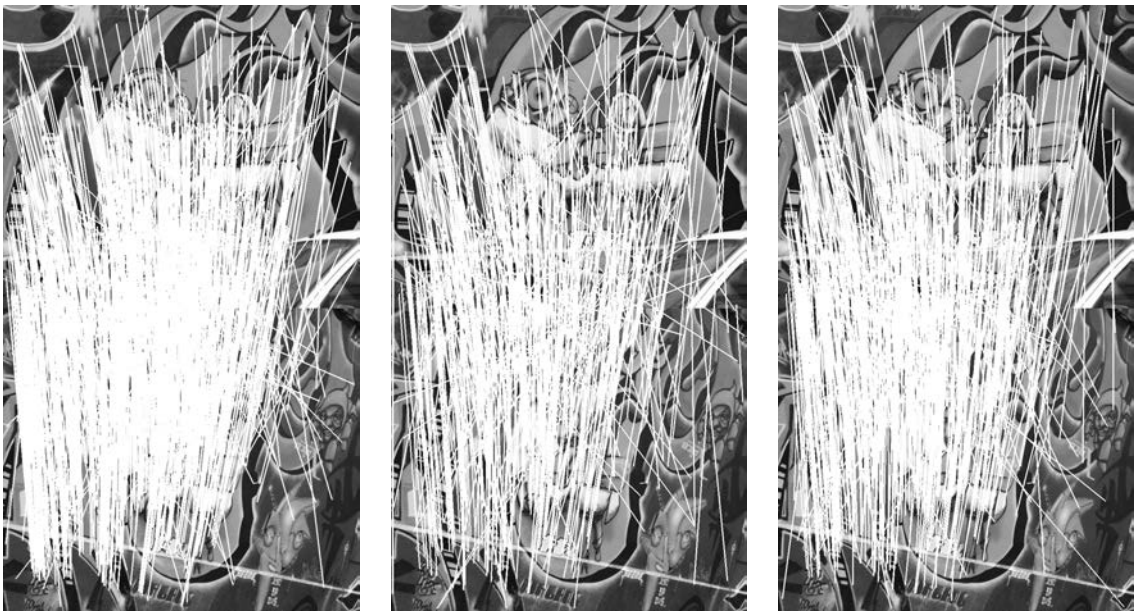
SIFT(10/10) PCA-SIFT(10/10) AED(10/10)

Figure 4.3: Connecting the best 10 matching keypoints in best-matching images under affine invariance.

scale and rotation invariance. Figure 4.8 shows that AED and PCA-SIFT have 10 matches, and SIFT has 9 matches. So both AED and PCA-SIFT are slightly better than SIFT. Boat is another set of images used to evaluate each algorithm under scale and rotation invariance. As shown in Figure 4.14, all matched keypoints of 3 algorithms are correct.

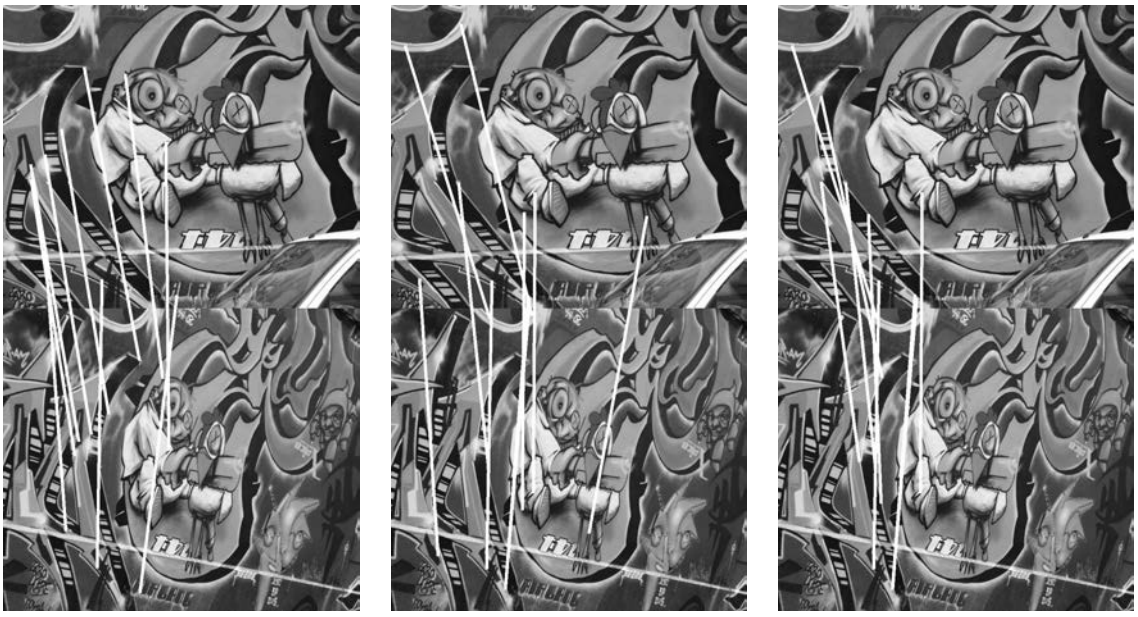
Blur invariance: The third set of experiments is conducted on a set of tree and bike images to examine the impact of image blur on the performance measures. Figures 4.9 and 4.13 show the results of the SIFT, PCA-SIFT and AED, all having 10 matches. Thus, all 3 algorithms are well-suited for matching of images with blurring differences.

Viewpoint change: The fourth set of experiments is carried out on a set of wall images where the images are taken by different viewpoints. The results shows that SIFT and AED have 10 matches which are better than 8 matches by PCA-SIFT. The experimental results are shown in Figure 4.10. As can be seen in images from Figure 4.10, the matched



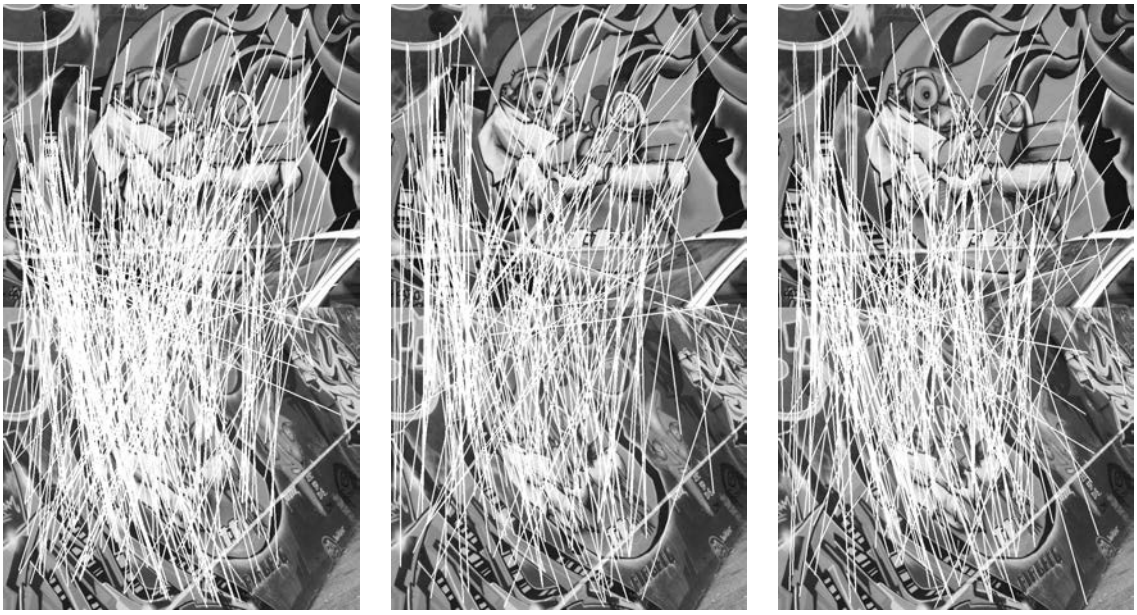
SIFT PCA-SIFT AED

Figure 4.4: Connecting all matching keypoints in second-best-matching images under affine invariance.



SIFT(10/10) PCA-SIFT(10/10) AED(10/10)

Figure 4.5: Connecting the best 10 matching keypoints the second-best-matching images under affine invariance



SIFT PCA-SIFT AED

Figure 4.6: Connecting all matching keypoints in third-best-matching images under affine invariance.

keypoints by SIFT, PCA-SIFT and AED are 10, 8 and 10, respectively.

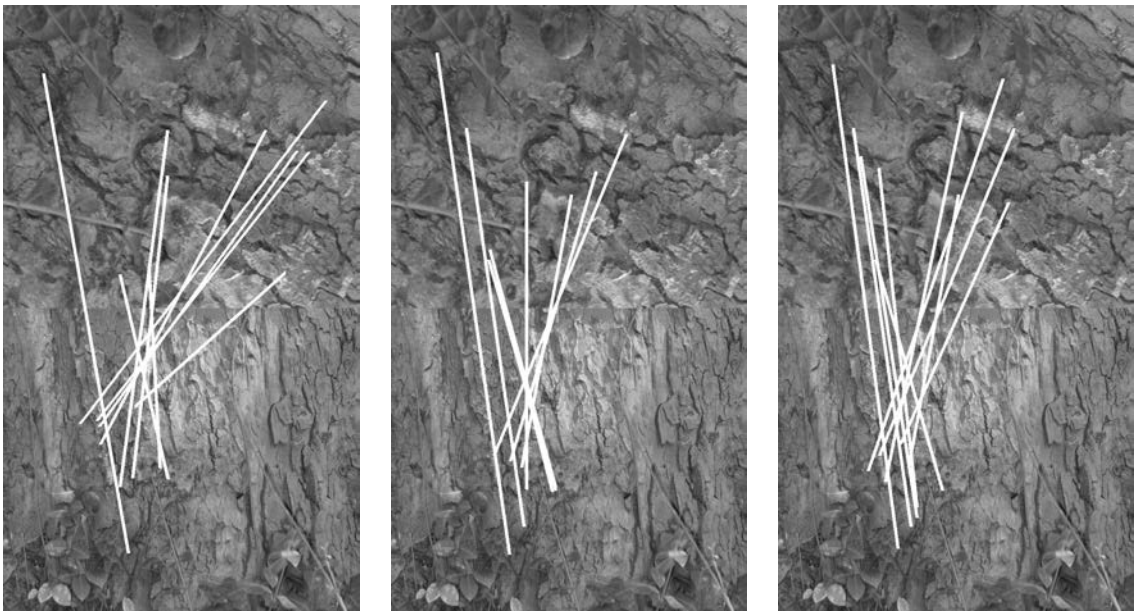
Light change: The fifth set of experiments is conducted on a set of Leuven images. Figure 4.11 shows the results of the SIFT, PCA-SIFT and AED, all of them have produced 10 matches, so they are well-suited for matching images with illumination changes.

Compression: The sixth set of experiments is conducted on a set of UBC images for investigating the performance of SIFT, PCA-SIFT and AED under image degradations caused by image compression. Figure 4.12 shows that all method are works well with image compression. Since they have all produced 10 matched keypoints out of 10.

Considering the results obtained for SIFT, PCA-SIFT and AED under changes in im-

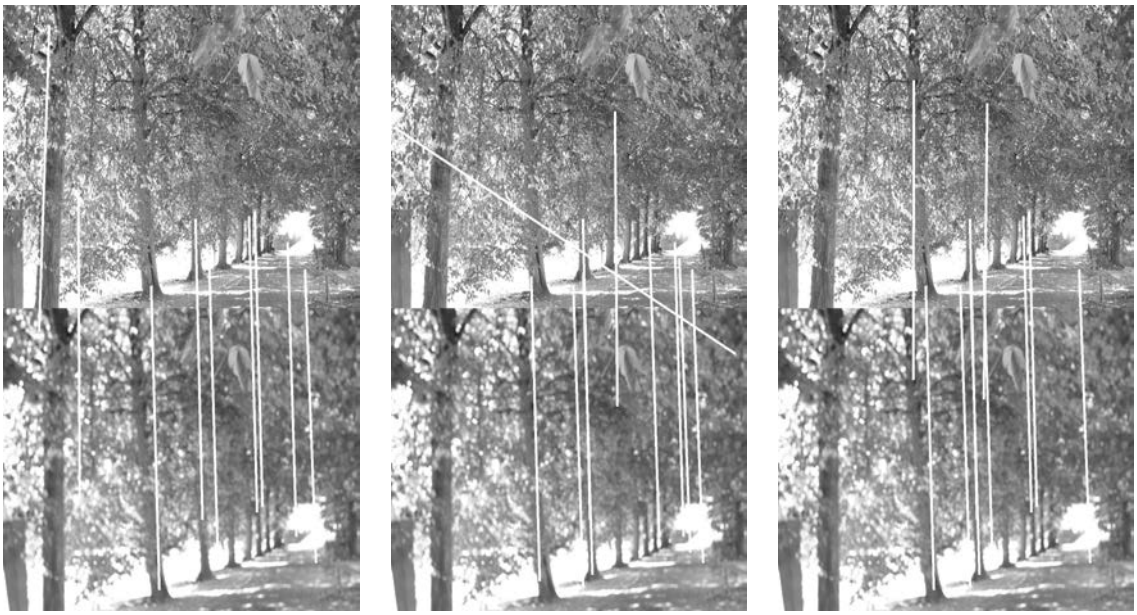
SIFT(4/10) PCA-SIFT(8/10) AED(10/10)

Figure 4.7: Connecting the best 10 matching keypoints in third-best-matching images under affine invariance.



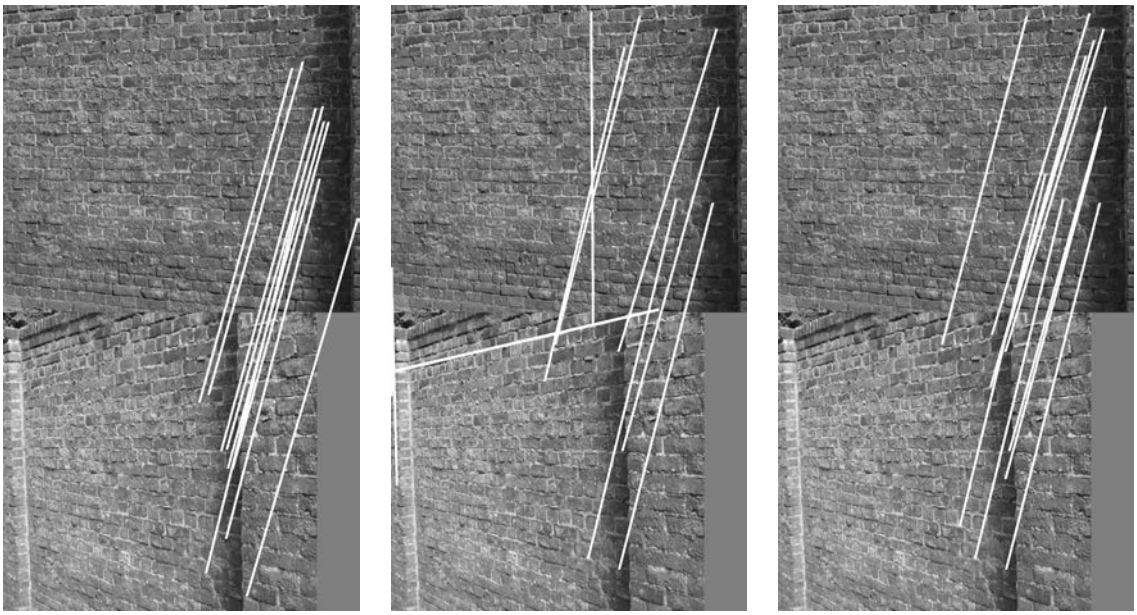
SIFT(9/10) PCA-SIFT(10/10) AED(10/10)

Figure 4.8: The first 10 matching key points between images scale and rotation invariance



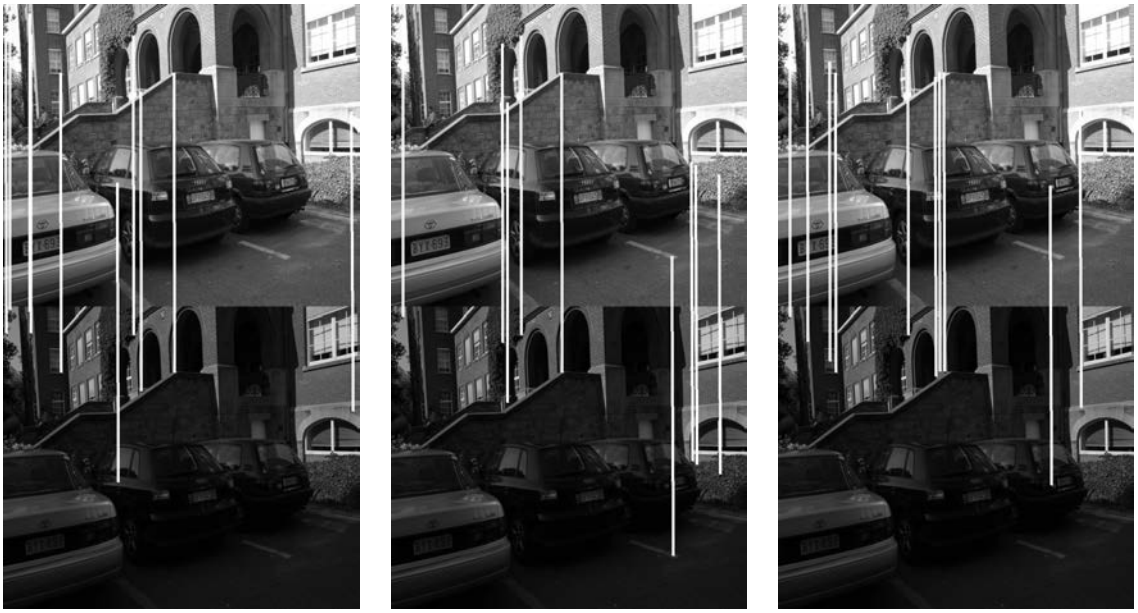
SIFT(10/10) PCA-SIFT(9/10) AED(10/10)

Figure 4.9: The best 10 matching keypoints in images with blurring difference.



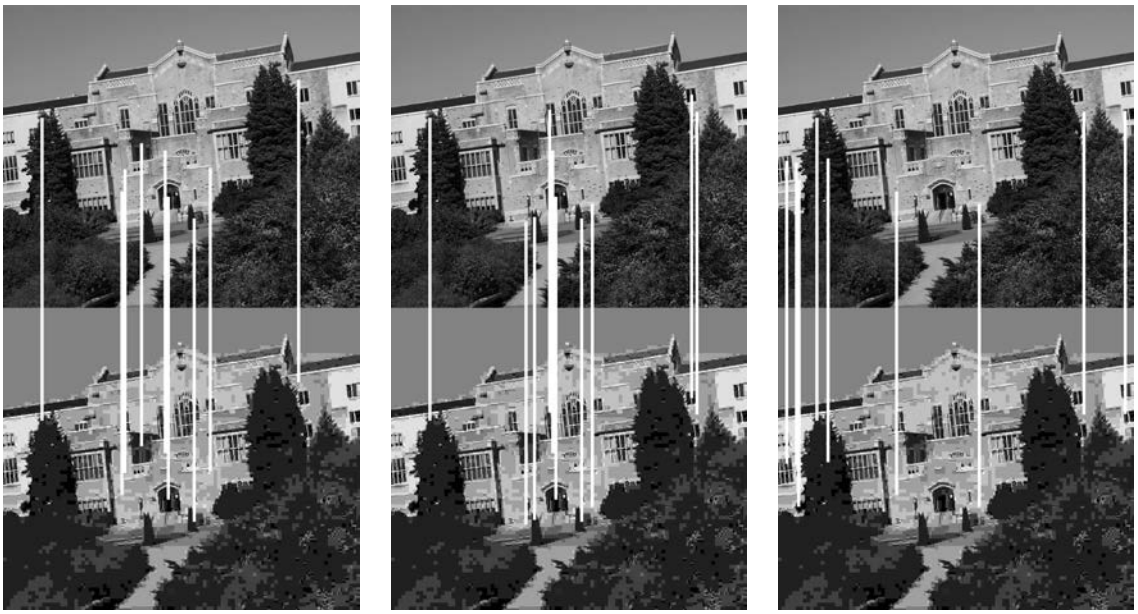
SIFT(10/10) PCA-SIFT(8/10) AED(10/10)

Figure 4.10: The best 10 matching keypoints in images from different views.



SIFT(10/10) PCA-SIFT(10/10) AED(10/10)

Figure 4.11: The best 10 matching keypoints in images with lighting differences.



SIFT(10/10) PCA-SIFT(10/10) AED(10/10)

Figure 4.12: The best 10 matching keypoints in images with and without compression degradation.

SIFT(10/10) PCA-SIFT(10/10) AED(10/10)

Figure 4.13: The best 10 matching keypoints in images with blurring difference.

SIFT(10/10) PCA-SIFT(10/10) AED(10/10)

Figure 4.14: The best 10 matching keypoints in images with scale and rotation invariance.

age bluring, lighting, and compression degradation, we see that AED has consistently produced best results under affine transformation, and changes in scale and orientation. From these results we see that among the three methods, AED is the most distinctive and robust under various image deformations, while being more compact than the SIFT descriptor.

Plotting the Precision vs. Recall curves of SIFT, PCA-SIFT, and AED for matching the Graffiti images 1 and 2 in Figure 4.15, we see that PCA-SIFT and AED are highly competitive in this matching task, both performing better than SIFT. Figure 4.16 shows the Precision vs. Recall curves of SIFT, PCA-SIFT, and AED for matching Graffiti images 1 and 3. This matching is more difficult than the previous one, AED showing its superiority over the other two methods under the affine transformation. When matching Graffiti images 1 and 4, the Precision vs. Recall curves of the algorithms are shown in Figure 4.17. Again, AED performs better than the other two methods.

Based on suggestion from previous studies (Ke and Sukthankar, 2004; Freeman and Adelson, 1991; Mikolajczyk and Schmid, 2005), we extract a 41×41 patch centered at each keypoint to generate the local feature descriptors. The size of this image patch could affect the performance of image matching, especially for images with scaling differences. The optimal size should depend on the task in hand. However, we have experimentally found that the current patch size (41×41) is a suitable size for a wide range of images in matching images in the Mikolajczyx dataset. For example, varying the patch size to 21×21 and 61×61 and comparing the matching results with the current patch size are shown in Figure 4.19. In this experiment, we use Bark 5 image since it is a scaled image of Bark 1, as shown in Figure 4.18. The Precision vs. Recall curves in Figure 4.19 show that the result of patch size 21×21 is clearly worse than the other two settings, while there is no significant difference between the patch size 41×41 and 61×61, use of patch size 41×41 makes the process faster. SIFT includes scale to the feature vector, so it is scale invariant.

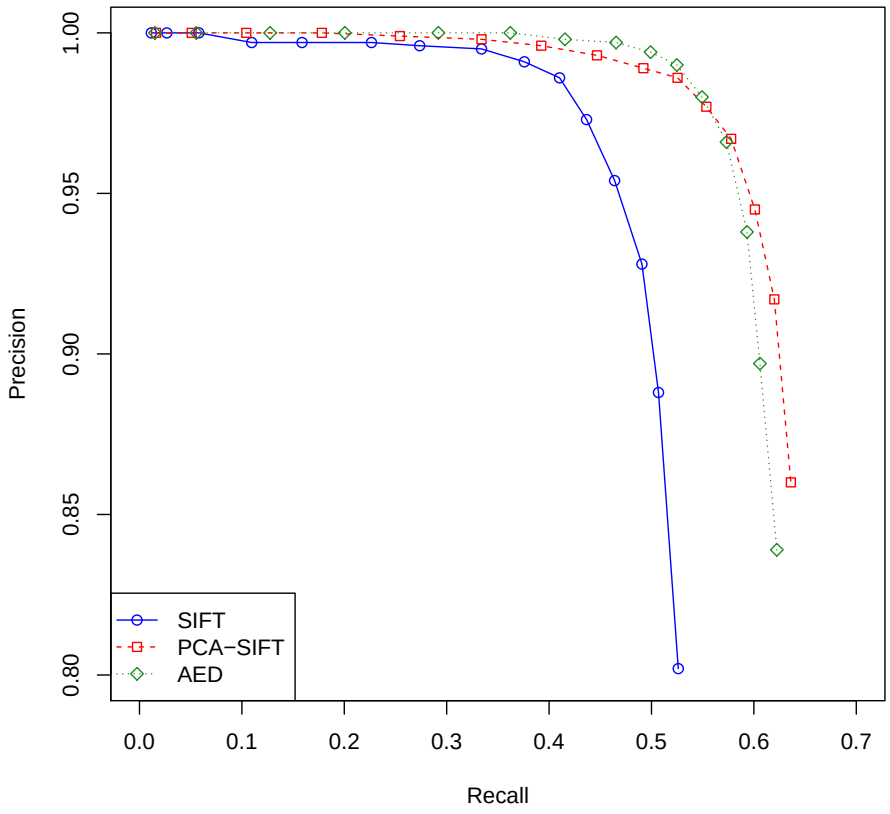


Figure 4.15: Precision vs Recall curves of SIFT, PCA-SIFT, and AED on Graffiti 2.

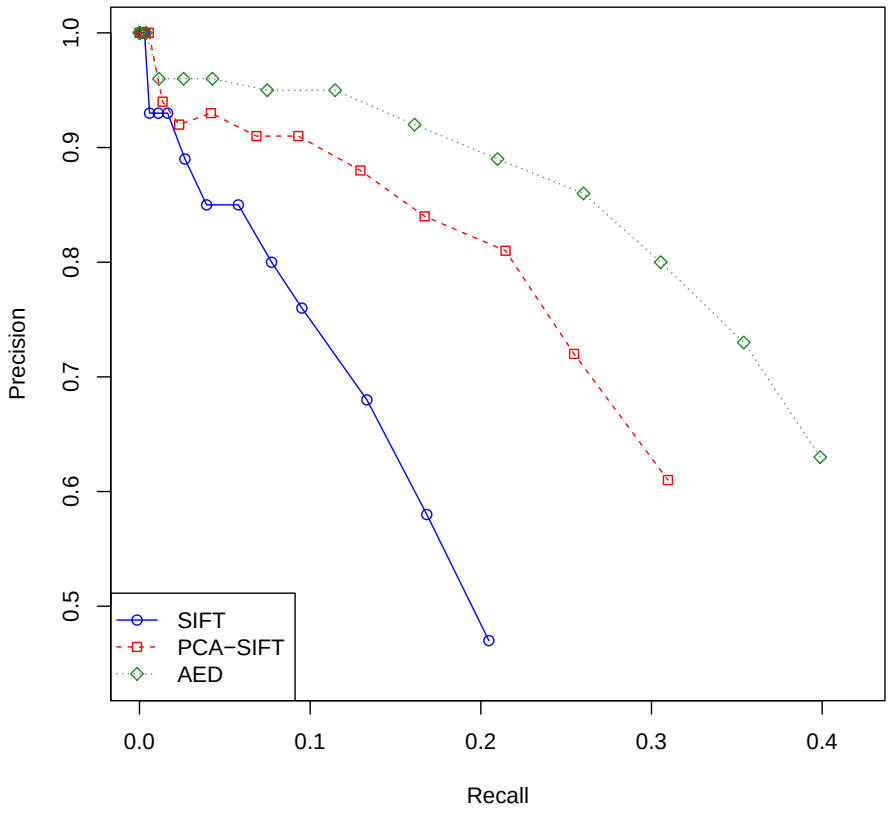


Figure 4.16: Precision vs. Recall curves of SIFT, PCA-SIFT, and AED on Graffiti 3.

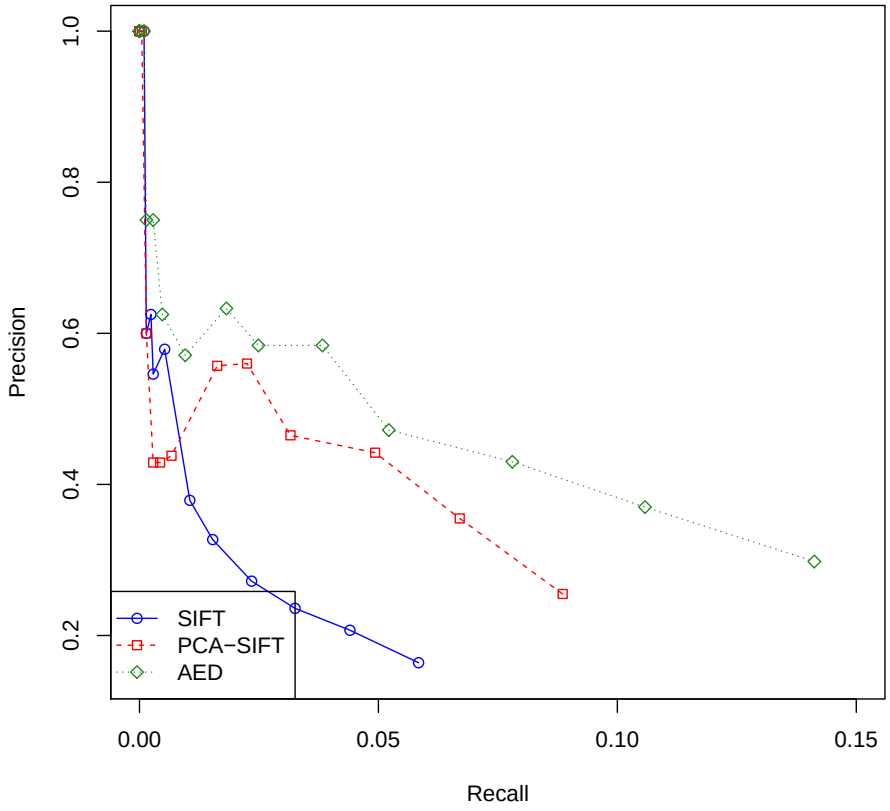


Figure 4.17: Precision vs. Recall curves of SIFT, PCA-SIFT, and AED on Graffiti 4.

Figure 4.18: Images of Bark 1 and Bark 5.

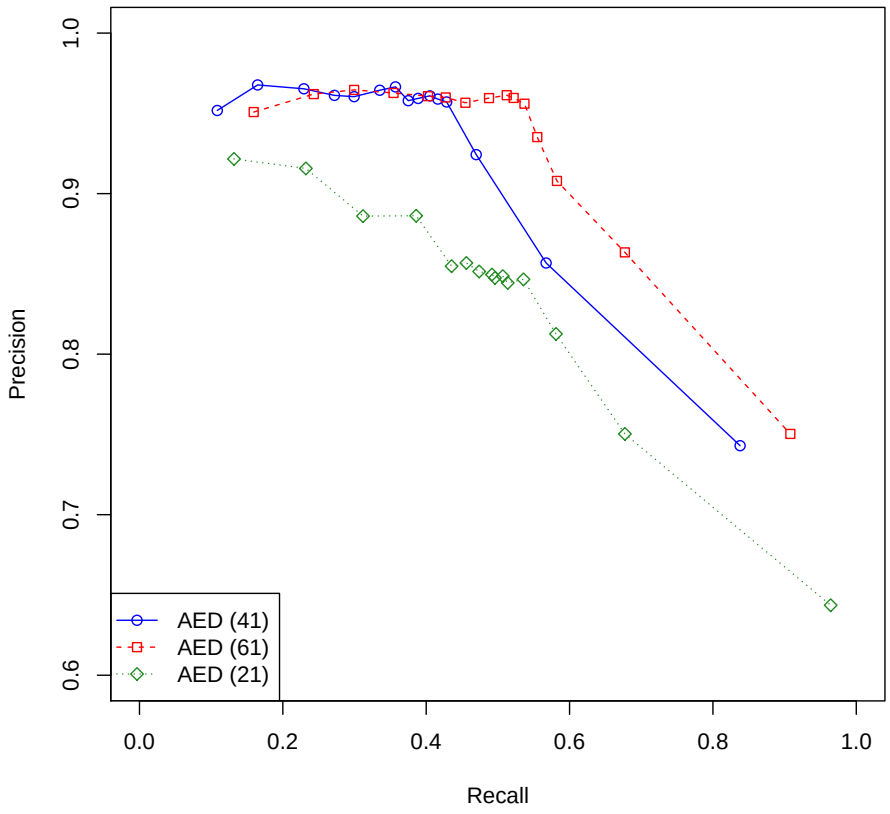


Figure 4.19: Precision vs. Recall curves of AED with 41×41, 61×61, and 21×21 patch sizes on Bark 5.

In the reported experiments, the proposed descriptor chooses a fixed image patch size of 41×41 pixels. This 41×41 window size may bot be optimal in some images. Therefore, knowing the type of images to be used in a matching task, the optimal window size can be estimated using the coarseness measure (Tamura et al., 1978) of the image. Coarser images require larger windows to contain sufficient information about objects that are larger than objects appearing in finer images.

Similar to SIFT and PCA-SIFT, AED describes an image patch using its intensity gradients. What if instead of intensity gradients, the raw intensities were used. How would that affect precision and recall rates? To answer that question, we conducted an experiment

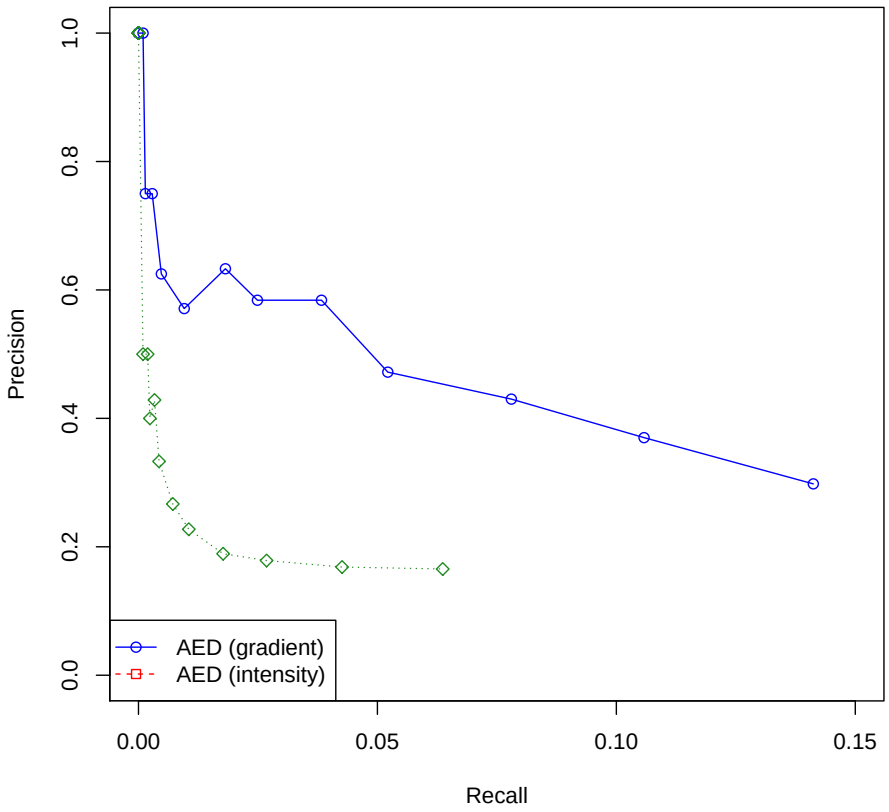


Figure 4.20: Precision vs. Recall curves of AED when using image gradients and intensities of Graffiti 4.

using intensities rather than intensity gradients. The experimental result is shown in Figure 4.20. It is clear that image gradients produces a superior result than the result obtained when using image intensities. The same result was observed by PCA-SIFT (Ke and Sukthankar, 2004).

Image retrieval using AED

When a query image is provided, Content-Based Image Retrieval requires searching the relevant images in a large database by using the contents of the images. Using feature detection and image matching in content-based image retrieval is an important application. The performance of an image retrieval method depends on two factors: a suitable feature descriptor and a powerful image matching strategy (Jegou et al., 2008). In this section, efficient application of AED in content-based image retrieval is described. Then, the performances of the proposed method are compared with those of SIFT and PCA-SIFT in retrieval of images in two image databases.

Our goal is not to show that our image retrieval system can beat state-of-the-art image retrieval systems. Usually, an image retrieval system in practice often combines multiple techniques to improve its accuracy and efficiency. Therefore, our purpose will be to show that AED is an alternative choice to SIFT in image retrieval, and is highly competitive to SIFT in terms of the accuracy, and more efficient than SIFT in terms of computational complexity.

5.1 Datasets

In order to evaluate the performances of the proposed descriptor, the INRIA Holidays dataset and ORL database are used.

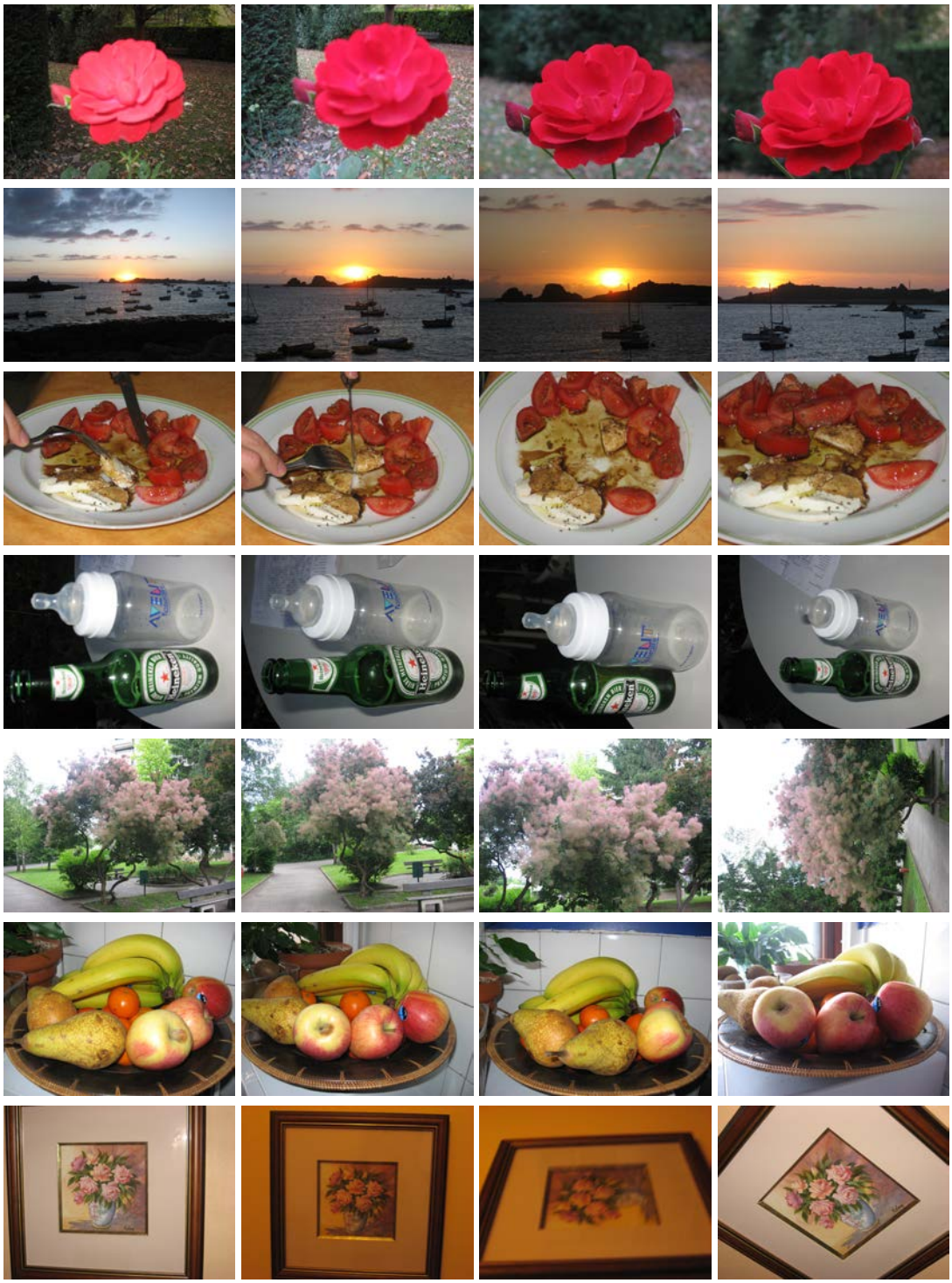


Figure 5.1: Example images of INRIA Holidays dataset.

INRIA Holidays dataset: The Holidays dataset [1] was created for the ANR RAFFUT project. It contains 1491 personal vacation photos with a very large variety of scene types, including natural, food, water and building. There are totally 500 image groups in this dataset, each of them represents a distinct scene of object. For each image group there are 2 to 10 images, and these images were taken from different pointview or by varying the lighting. We choose the first image of each group as the query, and other 1 to 9 images of the group are served as the correct retrieval results. Figure 5.1 are example images of this dataset for the variety of visual content.

ORL Database of Faces: The ORL database was created by AT&T Laboratories [2]. It contains a set of face images of 40 individuals taken between April 1992 and April 1994 at the lab. For each person, there are 10 different face images, and each image contains just one face. The 10 face images for a person are different due to four factors: 1) images were taken at different times; 2) varying the lighting; 3) facial expressions, such as open/closed eyes, smiling/not smiling; 4) facial details, such as glasses/no glasses. All the images have black homogeneous background. The size of each image is 92×112 pixels, with 256 grey levels per pixel. We divide the face images into 40 groups, each groups contains 10 images for one person. For each group, we choose the first image as the query image, and thus there are 9 relevant images for any query image. Figure 5.2 are example face images of this dataset for 10 individuals.

5.2 The retrieval procedure

For a given dataset, we first divide a dataset into two parts. The first part contains all the query images, and the rest is used to search the database. For a query image, the goal is to try to find all relevant images for the query. Taking the ORL dataset as an example, there

[1] https://lear.inrialpes.fr/ jegou/data.php
[2] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

Figure 5.2: Example face images of the ORL Database.

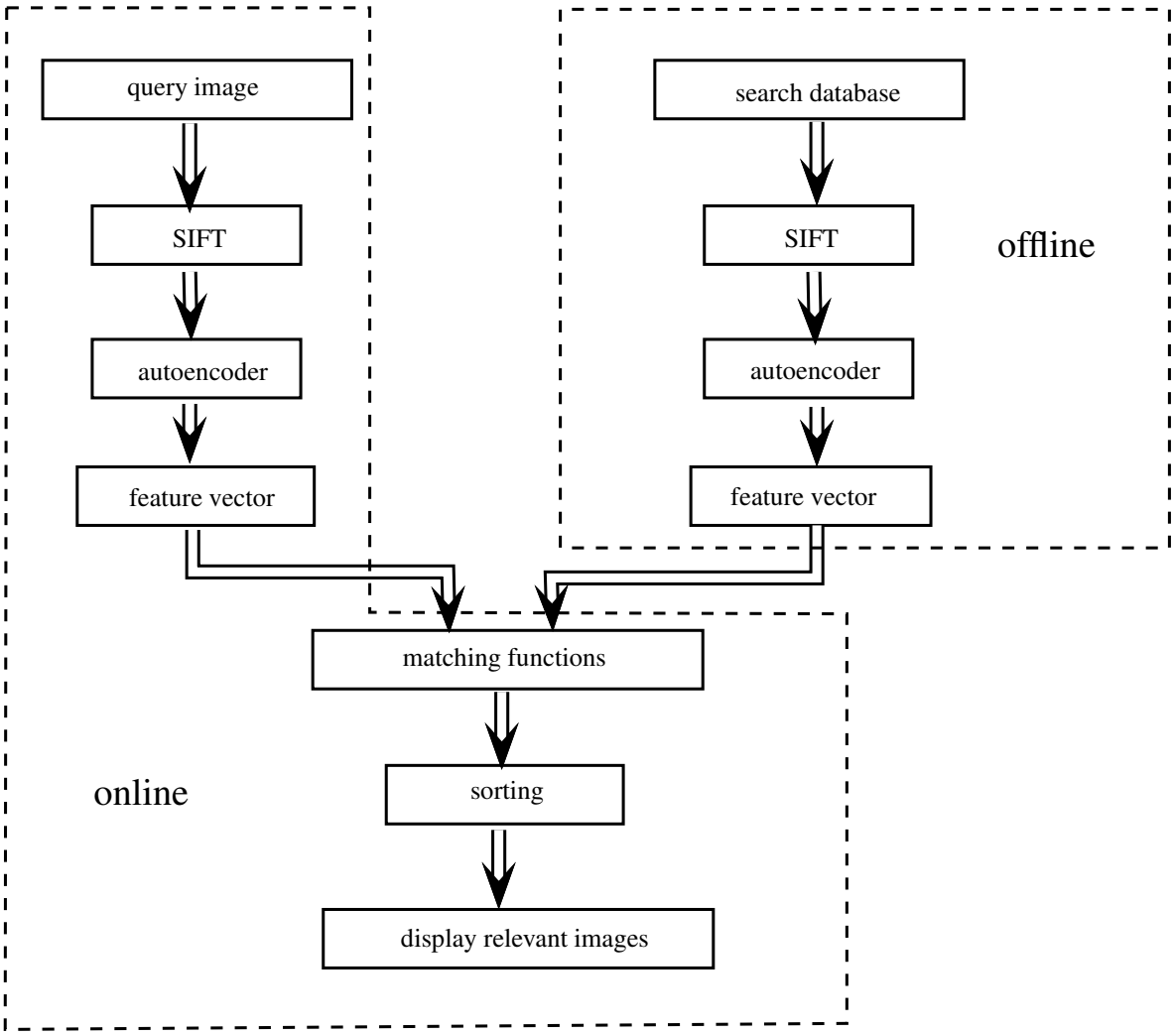


Figure 5.3: Flowchat of content-based image retrieval system using the AED method.

are 40 query images in total, and the remaining 360 images represent the search database. For a query image, we intend to find the other 9 images of the same person from the entire 360 images.

The whole image retrieval procedure consists of two steps: offline step and online step. The offline step can be performed once ahead of time, and updated as new images arrive periodically. Thus, the efficiency of the offline step is not crucial. Since the online step will be performed when a query image is given, this step should be implemented very efficiently. Figure 5.3 illustrate the whole image retrieval task. In the following, the details

of the offline step and the online step are provided.

5.2.1 Offline processing step

Given a database containing a large number of images, the offline step including following:

- Using the SIFT detector extract interest points in all images in search database.
- For each image, extract the local gradient patch centered at each interest point, and generate the gradient vector for each interest point.
- Project each gradient vector to a compressed feature vector by using the pre-trained autoencoder.
- Store the compressed feature vectors with the images.

5.2.2 Online processing step

In the online step, a query image is given, and it is required to find all relevant images in the search database. This can be achieved as follows:

- Using the SIFT detector extract interest points of the query image.
- Extract the local gradient patch centered at each interest point, and generate the gradient vector for each interest point.
- Project each gradient vector to a compressed feature vector by using the pre-trained autoencoder.
- For each image in the search database:
  - Compute the similarity between this image and the query image. For two images, we compare each feature vector in one image with all feature vectors in the other image. If the Euclidean distance between two feature vectors is smaller

than the chosen threshold, there is a match. The similarity of the two images is calculated by the number of matches. If the number of matches is greater than 0, return this image.

- Sort the retrieved images in descending order according to their similarities with the query image.
- Display the relevant images.

In this straightforward image retrieval process, the bottleneck in computations is the computational of the similarity between the query image and all the images in the search database. By employing AED, the relevant images can be obtained more efficiently than by SIFT. We will describe a strategy to further speed up the image retrieval process in Section 5.5.

5.3 Evaluation metrics

In the experiments, we use the popular metrics Precision vs. Recall to evaluate the performance of an image retrieval system. For any query image, Precision is the ratio of the number of relevant images retrieved to the total number of images (including relevant and irrelevant) retrieved:

$$Precision = \frac{\text{Number of relevant images retrieved}}{\text{Total number of images retrieved}}$$

where an image is considered relevant if and only if it is in the same group as the query image. Intuitively, Precision measures the quality of the retrieved images. The larger its value, the better the quality of the retrieved images. For simplicity, Precision of a query image is set to 1 if no images are retrieved.

Recall is the ratio of the number of relevant images retrieved to the total number of relevant images retrieved in the entire search database:

$$Recall = \frac{\text{Number of relevant images retrieved}}{\text{Total number of relevant images}}$$

In some real-world applications, the total number of relevant images is usually unknown. For example, if the search space is the entire Internet, it is hard to know how many images are relevant to a query images. However, in our application, Recall is computable as we assume that only if images in the same group are relevant. Recall measures the ability to find relevant images by the image retrieval system.

With the Precision and Recall for a query image, we can compute the average Precision and Recall over all the queries. Ideally, it is desired that an image retrieval system provide both a high Precision and a high Recall rate, but this is often an unattainable goal in practice. For the extreme cases, if no images are retrieved, the Precision is set to 1, but obviously this image retrieval system is useless. On the other hand, if an image retrieval system retrieve all the images in the search database, then Recall is always 1, but Precision can be close to 0e. Therefore, there is usually a trade-off between precision and recall, and we employ the Precision vs. Recall curves to measure the performance of an image retrieval system. The closer the curve is to the top of the chart it indicates a better performance.

5.4 Experimental results

The experiment is to compare the retrieval performance between our descriptor and SIFT and PCA-SIFT. For the image retrieval systems that use SIFT and PCA-SIFT, they are very similar to the process of Figure 5.3. The only difference is replacing the AED module to SIFT or PCA-SIFT.

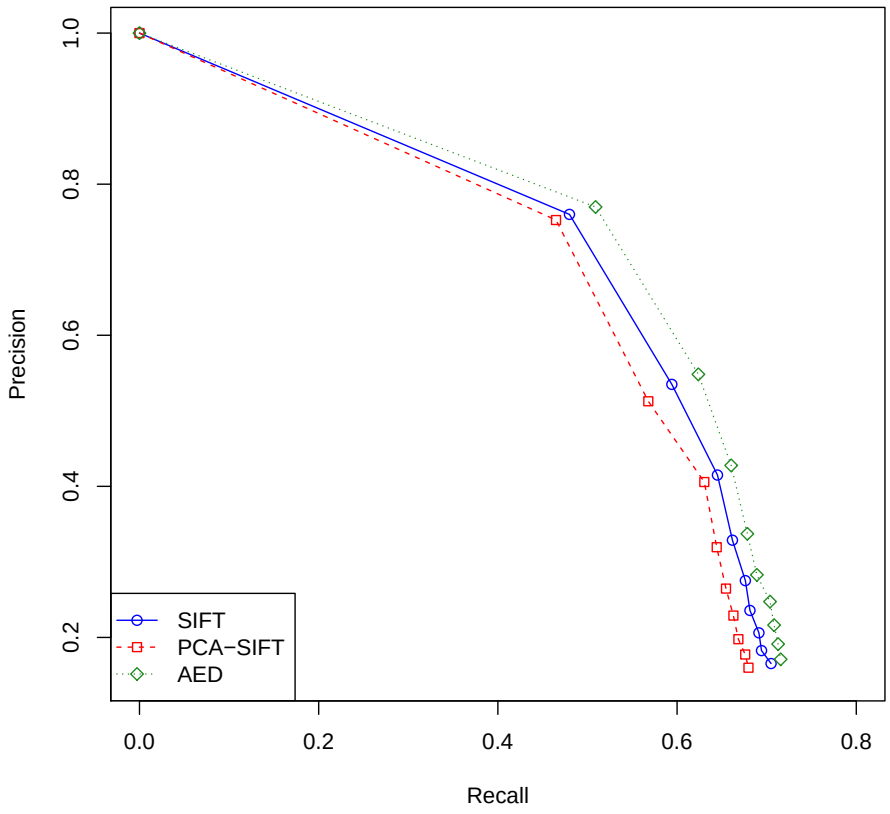
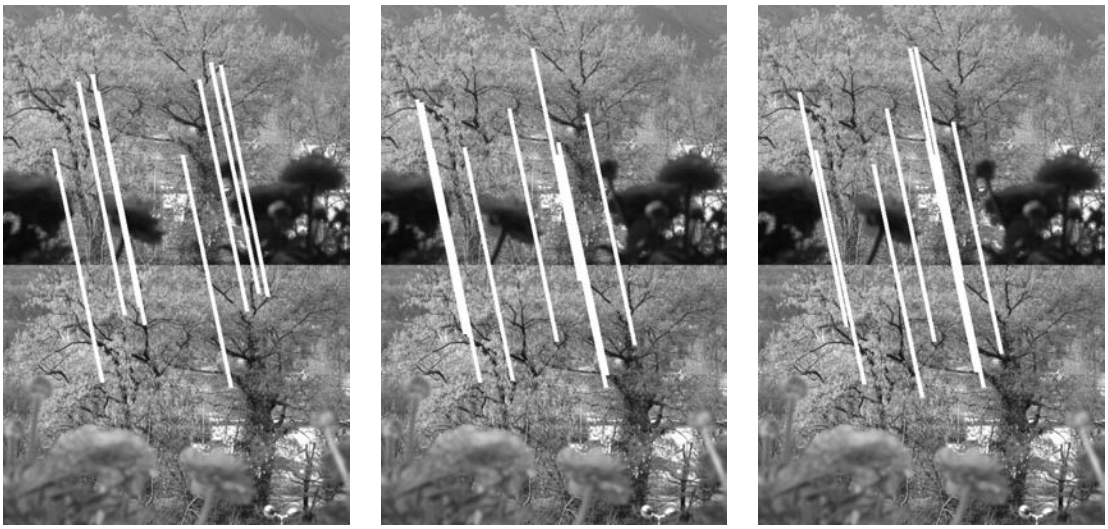


Figure 5.4: Precision vs. Recall curves of SIFT, PCA-SIFT, and AED using the Holidays dataset.

5.4.1 Image retrieval results using the Holidays dataset

The Precision vs. Recall curves of SIFT, PCA-SIFT, and AED are shown in Figure 5.4. For each query image, each algorithm is allowed to return at most 9 images. This result shows that AED is comparable or better than SIFT and PCA-SIFT in image retrieval when using the Holidays dataset. We believe this is due to AED's high matching accuracy at the interest point level, which also translates to high retrieval results.

Figures 5.8, 5.9, and 5.10 illustrate the retrieved images of SIFT, PCA-SIFT and AED algorithms. For figures 5.8 and 5.9, the corresponding image retrieval tasks are relatively easy, and all the algorithms perform well on these two query images. Note that AED per-



SIFT PCA-SIFT AED

Figure 5.5: The best 10 matching keypoints between the query image 136000.pgm and the retrieved image 136002.pgm.

forms especially well, finding all retrieved images. For the query image 136000.pgm in figure 5.8, 2967 keypoints are detected by SIFT. Figure 5.5 shows that the first 10 matching keypoints between the query image 136000.pgm and the retrieved image 136002.pgm. There are in total 631, 434, and 554 matches for SIFT, PCA-SIFT, and AED, respectively. For the query image 132500.pgm in Figure 5.9, 723 keypoints are detected by SIFT. Figure 5.6 shows the first 10 matching keypoints between the query image 132500.pgm and the retrieved image 132503.pgm. There are in total 143, 107, and 147 matches for SIFT, PCA-SIFT, and AED, respectively.

For Figure 5.10, the corresponding image retrieval task is more difficult due to detection of only 145 keypoints by SIFT. On the other hand, the relevant images are very different to the query image. Figure 5.7 shows the matching keypoints between the query image 139500.pgm and the relevant image 139505.pgm. There are 6, 5, 10 matches for

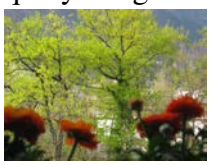
SIFT PCA-SIFT AED

Figure 5.6: The best 10 matching keypoints between the query image 132500.pgm and the retrieved image 132503.pgm.

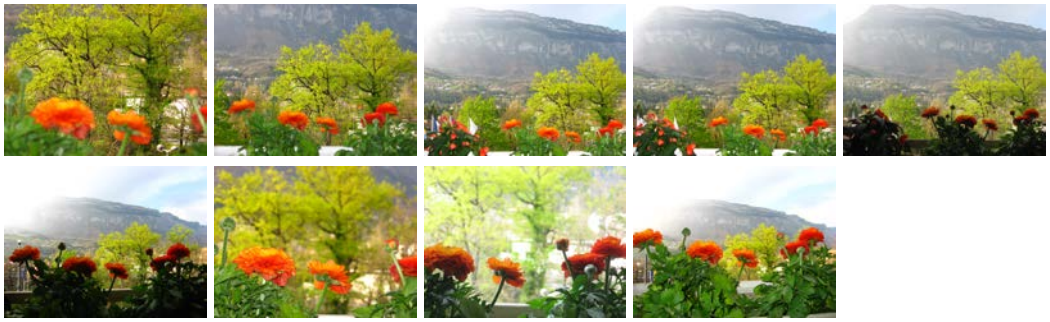
SIFT PCA-SIFT AED

Figure 5.7: Matching keypoints between the query image 139500.pgm and the relevant image 139505.pgm.

query image:



retrieved images by SIFT:



retrieved images by PCA-SIFT:

retrieved images by AED:

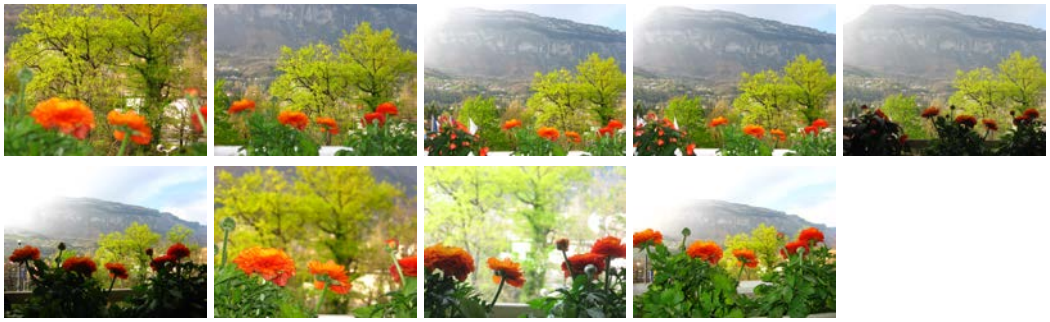
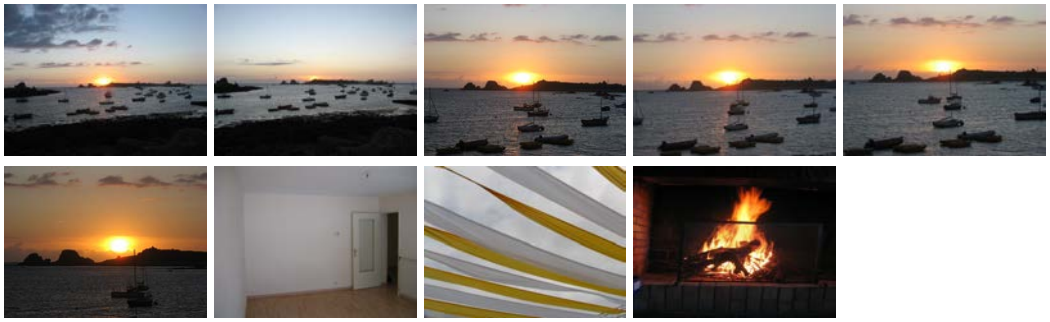


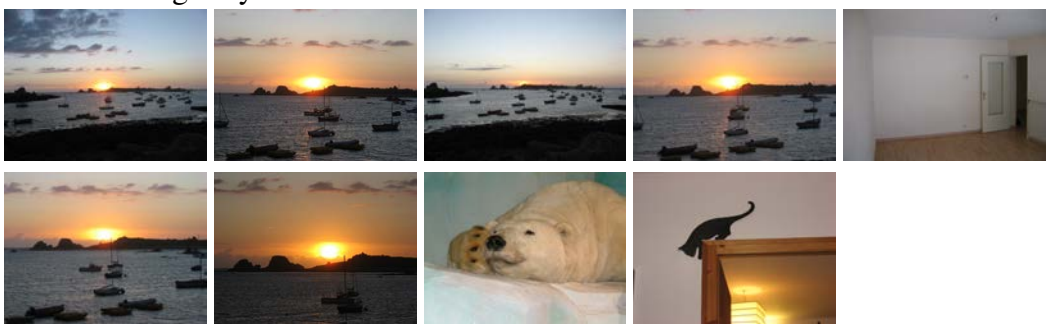
Figure 5.8: Retrieved images by SIFT, PCA-SIFT, and AED for the query image 136000.pgm.

query image:

retrieved images by SIFT:



retrieved images by PCA-SIFT:



retrieved images by AED:

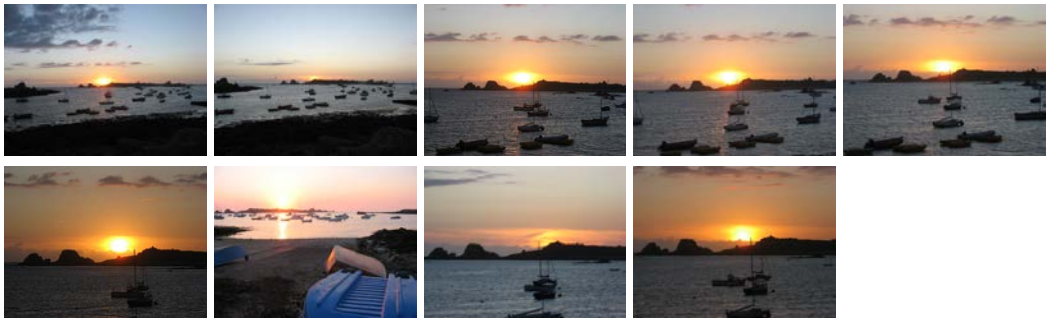


Figure 5.9: Retrieved images by SIFT, PCA-SIFT, and AED for the query image 132500.pgm.

query image:

retrieved images by SIFT:

retrieved images by PCA-SIFT:

retrieved images by AED:

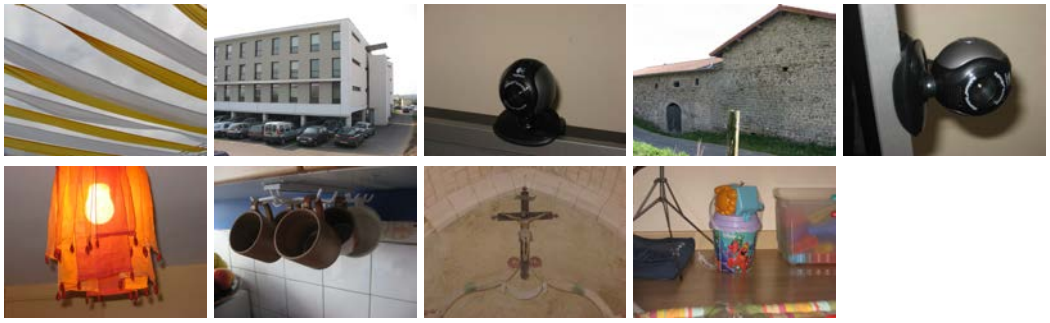


Figure 5.10: Retrieved images by SIFT, PCA-SIFT, and AED for the query image 139500.pgm.

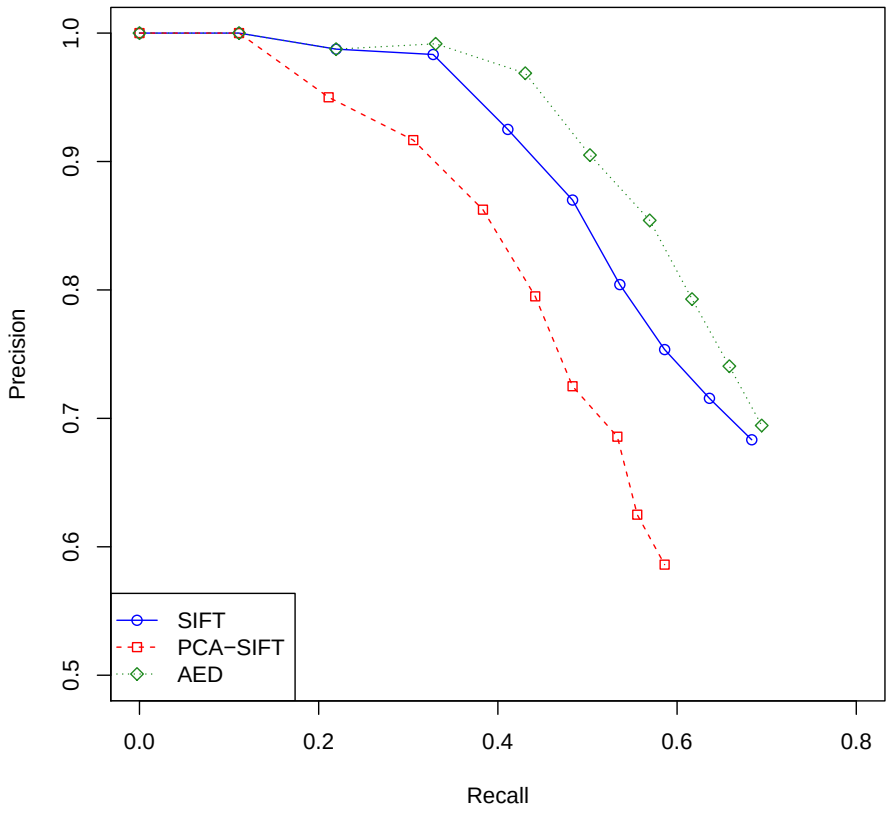


Figure 5.11: Precision vs. Recall curves of SIFT, PCA-SIFT, and AED when using the ORL dataset.

SIFT, PCA-SIFT, and AED. respectively.

5.4.2 Image retrieval result of the ORL dataset

From the experimental results using the ORL dataset, the Precision vs. Recall curves of SIFT, PCA-SIFT, nd AED are plotted in Figure 5.11. As before, at most 9 images were retrieved for each query image. The image retrieval task on this dataset is easier than the previous Holidays dataset. The retrieved results are perfect for each algorithm if we only consider the top one retrieved image. Considering the overall retrieval results, AED

performs better than SIFT and PCA-SIFT on this dataset.

Figure 5.12 shows the retrieved images for the query image of 7th person and the query image of 2nd person. For all the three algorithm, the 9 relevant images are retrieved although their rankings are slightly different. Considering the query image of the 7th person, 92 key points were detected by SIFT, and there are 48, 42, 48 matches for SIFT, PCA-SIFT and AED, respectively, for the top one retrieved image.

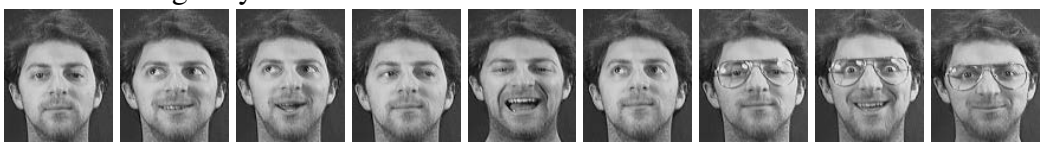
Figure 5.13 shows the retrieved images for the query image of 8th person and the query image of 3rd person. The image retrieval tasks for these two query images are more difficult, and each algorithm retrieved some irrelevant results. For the query image of 8th person, 7 relevant images were retrieved by all the three algorithms. However, PCA-SIFT performs worse than the other two since the irrelevant images ranked higher. For the query image of 3rd person, 6 relevant images were retrieved for both SIFT and PCA-SIFT. AED performs better for this query image since 8 relevant images are retrieved.

5.4.3 Run-time analysis

Besides the accuracy, an important performance measure of an image retrieval system is its run time. As was discussed before, the matching process is the bottleneck in image retrieval since it has to be executed online. To evaluate the time performance of the image retrieval systems time, we tested each method on a Red Hat linux server with Intel Xecon with 5650 CPUs. The matching algorithm was implemented in C++. Here an example using two images from the Holidays dataset is given. Overall, 2967 and 3284 keypoints were detected by the SIFT detector, requiring about 9.7 million point comparisons during the matching process. Table 5.1 shows the run times in ms for SIFT, PCA-SIFT, and AED. PCA-SIFT and AED spent almost the same time in matching as the feature vector in each keypoint had 36 dimensions by both methods. SIFT is about 3 times slower than PCA-SIFT and AED. This is due to associating a 128-dimensional feature vector to each keypoint. For

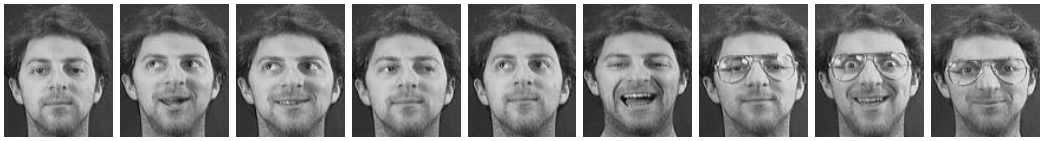
query image:

retrieved images by SIFT:



retrieved images by PCA-SIFT:

retrieved images by AED:



query image:

retrieved images by SIFT:

retrieved images by PCA-SIFT:

retrieved images by AED:

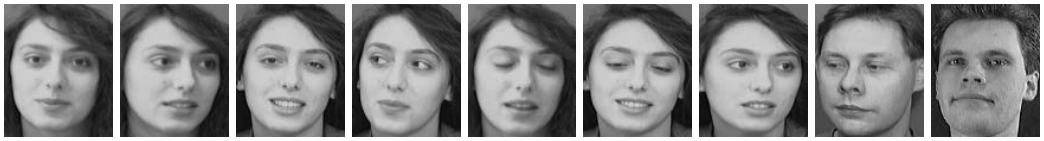
Figure 5.12: Retrieved images by SIFT, PCA-SIFT, and AED for query image of the 7th and 2nd persons.

query image:

retrieved images by SIFT:

retrieved images by PCA-SIFT:

retrieved images by AED:



query image:

retrieved images by SIFT:

retrieved images by PCA-SIFT:

retrieved images by AED:

Figure 5.13: Retrieved images by SIFT, PCA-SIFT, and AED for the query image of 8th and 3rd persons.

	Run time per image	Run time per point
SIFT	7.246	7.44e-7
PCA-SIFT	2.206	2.26e-7
AED	2.182	2.24e-7

Table 5.1: Matching run time in seconds for SIFT, PCA-SIFT, and AED when using two images from the Holidays dataset.

each query image, there was a need to compare it to all 1491 images in Holidays dataset, requiring about one hour matching time for AED and PCA-SIFT, and three hours for SIFT. Obviously this run time is not practical in real applications. In the next section, an efficient means to speed up the matching process is described.

5.5 Codebook image representation

Consider the matching process shown in Figure 5.3. Each image in the search database is compared with the query image. This is intractable in practice because a search database often contains a very large number of images. In order to speed up the matching process, we employ a codebook mapping method (Collins and Okada, 2012; Yuan et al., 2011). The main idea is to map each image into a fixed-length vector, so that the similarity of two images can be efficiently determined by using two vector. Then, perform matching only on images that produce high similarity scores.

Figure 5.14 shows the image retrieval process using the codebook mapping approach. In order to map an image to a fixed-length vector, we first cluster the (36-dimensional) AED feature vector of each keypoint for all images in the search database. We employ k-means clustering and set the number of clusters to L[3]. Each AED feature vector can then be assigned to one of the L clusters. In this manner, an image can be described as a frequency distribution of these L labels. This L-dimensional vector is called *codebook*

[3]We choose L to be 1000, suggested in (Collins and Okada, 2012).

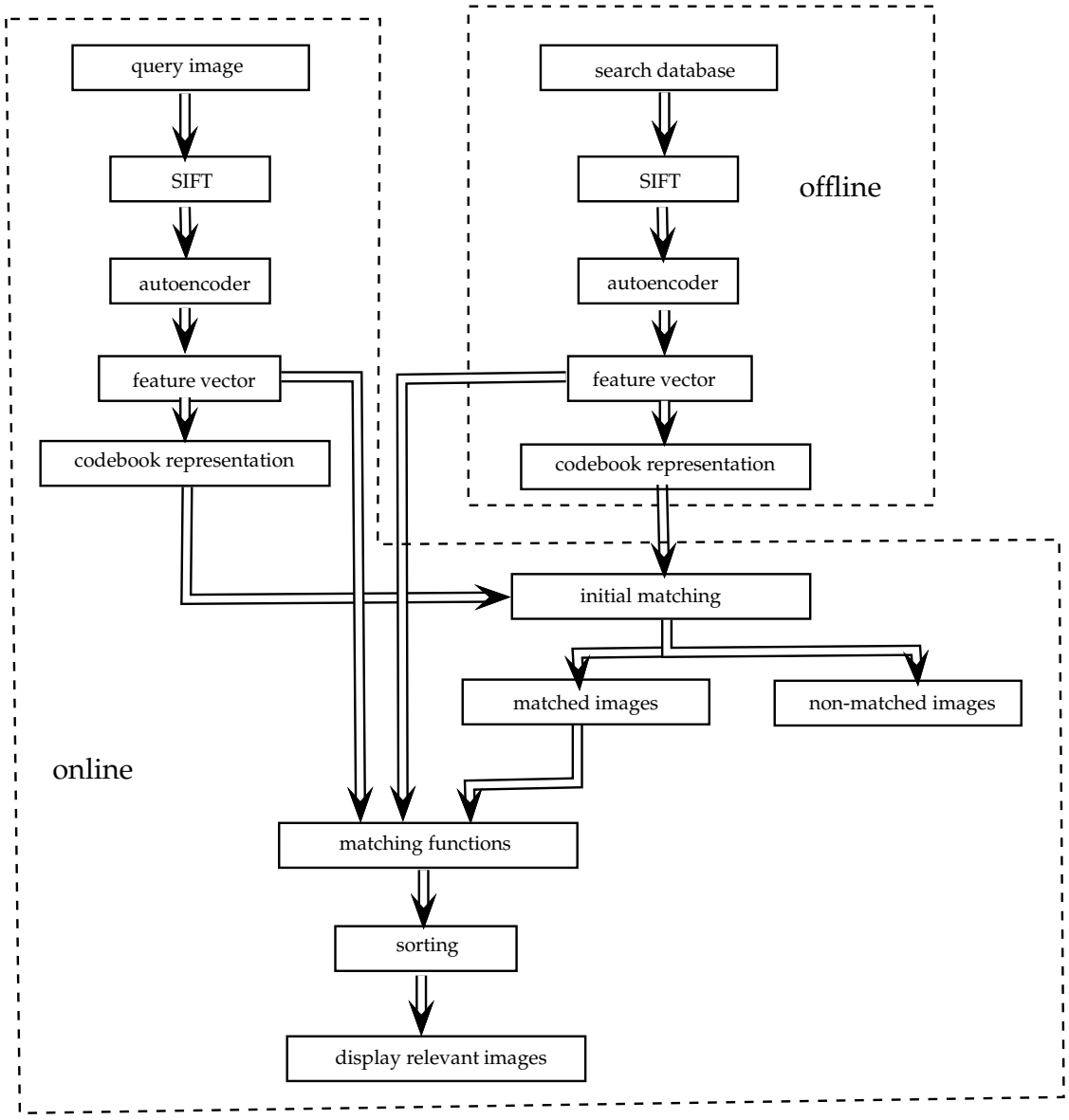


Figure 5.14: Flowchat of content-based image retrieval system using AED algorithm and codebook representation.

or *bag-of-words* representation. This notion comes from the natural language processing area, and is a popular way of representing a document by it's word frequency distribution, ignoring order.

To employ codebook mapping, there is one more step in the image retrieval's offline step. After AED vectors of all images in the search database are obtained and stored, we apply k-means on the AED feature vectors and group them into L clusters. Then, store these L-dimensional codebook vectors and the clustering model.

In the new image retrieval process, the online step is updated as follows:

- Using the SIFT detector extract interest points of the query image.
- Extract the local gradient patch centered at each interest point, and generate the gradient vector for each interest point.
- Project each gradient patch to an AED vector by using the pre-trained autoencoder.
- Map the query image to an L-dimensional codebook representation. More specifically, compute the distance between each AED vector in the query image with the L cluster centers generated by the k-means algorithm in the offline step. Then, assign each AED vector into the cluster which has the closest distance between its center and this AED vector.
- Compute the similarity between the query image and all the images in the search database using their codebook representations. Select N images with the highest similarity scores as the initial matched images.
- For each initial matched image in the search database:
  - Compute the similarity between this image and the query image. For two images, we compare each feature vector in one image with all feature vectors in the other image. If the Euclidean distance between two feature vectors is smaller than the chosen threshold, a match is declared. The similarity of the

two images is calculated by the number of matches. If the number of matches is greater than 0, the image is returned.

- Sort the retrieved images in descending order of their similarity with the query image.
- Display the relevant images.

We set N to 100 in our image retrieval system, and thus the query image is compared to 100 images instead of all images in search database. For the Holidays dataset, this speeds up the process by about 15 times in matching.

Since the codebook mapping is an approximate method, the matching performance may worsen when compared to the naive matching approach. Figure 5.15 shows the Precision vs. Recall curves of AED and AED with codebook mapping. After initial matching with codebook representation, some relevant images are dropped incorrectly. However, this loss of accuracy may be acceptable when considering improvement run time.

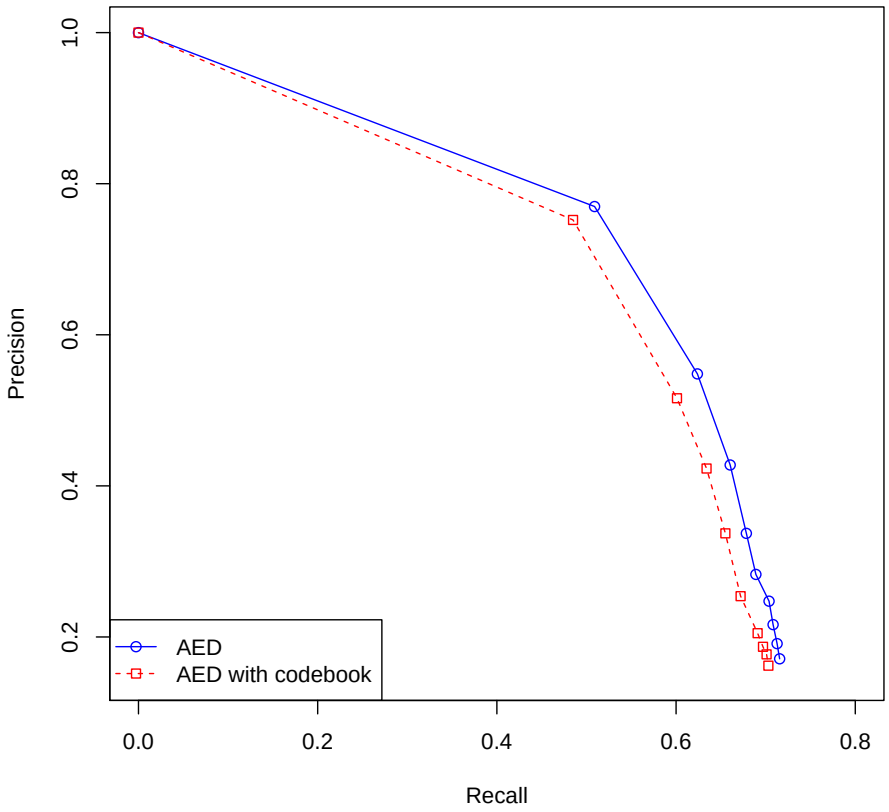


Figure 5.15: Precision vs. Recall curves of AED and AED with codebook mapping when using the Holidays dataset.

Conclusions

With development of various image descriptors, significant progress has been made in image matching and image retrieval. Image features generated by local image descriptors are generally invariant to different image transformations. This dissertation developed and evaluated a novel antoencoder-based image descriptor.

The autoencoder concept is used to reduce the dimension of the feature vector used to describe the properties of a keypoint. Compared to the SIFT descriptor, which has a 128-dimensional feature vector, the proposed encoder has a 36-dimensional feature vector. As a result, the proposed descriptor has a considerable computational advantage over the SIFT descriptor.

When used in image matching, the new descriptor is shown to have a higher combined precision and recall rate than SIFT under various image transformation, such as scaling, rotation, change in viewpoint, image blurring, and JPEG compression degradation.

When used in image retrieval, the proposed descriptor again provides a higher combined precision and recall rate than SIFT. Moreover, the proposed descriptor is about three times faster than the SIFT descriptor. By using the codebook method the speed of the proposed descriptor in image retrieval can be further increased.

Although AED uses fewer coefficients than SIFT, the coefficients used by AED are compact, while those used by SIFT represent raw values at gradient direction histogram bins. By keeping a fixed window of 41 41 that can be adjusted to the coarseness of an image sufficient information about the neighborhood of a keypoint can be captured.

The SIFT descriptor by changing the size of the neighborhood of a keypoint based on the estimated scale at the keypoint selects small windows at many keypoints. Although SIFT generates a fixed 128-dimensional feature vector, in many situations the selected window does not contain 128 pixels, including redundant information in a generated feature vector. Feature vectors generated by SIFT are not compact, often contains redundant information, consequently, producing performance measures that fall below those of AED.

The newly developed image descriptor in this dissertation has many advantages over the SIFT descriptor. However, there is still space for improvement. The main bottleneck of the proposed descriptor is in the offline training step. There is a need to speed up the offline training step by perhaps using a faster clustering algorithm.

Bibliography

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

John Collins and Kazunori Okada. A comparative study of similarity measures for content-based medical image retrieval. In *CLEF (Online Working Notes/Labs/Workshop)*, 2012.

Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (CSUR)*, 40(2):5, 2008.

William T. Freeman and Edward H Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):891–906, 1991.

Ian Goodfellow, Honglak Lee, Quoc V Le, Andrew Saxe, and Andrew Y Ng. Measuring invariances in deep networks. In *Advances in neural information processing systems*, pages 646–654, 2009.

James Hafner, Harpreet S Sawhney, Will Equitz, Myron Flickner, and Wayne Niblack. Efficient color histogram indexing for quadratic form distance functions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(7):729–736, 1995.

Martin T Hagan and Mohammad B Menhaj. Training feedforward networks with the marquardt algorithm. *Neural Networks, IEEE Transactions on*, 5(6):989–993, 1994.

Robert M Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, 1979.

Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, (6):610–621, 1973.

Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer, 1988.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Peter Howarth and Stefan Rüger. Evaluation of texture features for content-based image retrieval. In *Image and Video Retrieval*, pages 326–334. Springer, 2004.

Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Computer Vision–ECCV 2008*, pages 304–317. Springer, 2008.

Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–506. IEEE, 2004.

Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *Neural Networks, IEEE Transactions on*, 8(1): 98–113, 1997.

Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262–282, 2007.

David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

Bangalore S Manjunath, Philippe Salembier, and Thomas Sikora. *Introduction to MPEG-7: multimedia content description interface*, volume 1. John Wiley & Sons, 2002.

Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 52–59. Springer, 2011.

Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 525–531. IEEE, 2001.

Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, 2005.

Jean-Michel Morel and Guoshen Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009.

Greg Pass and Ramin Zabih. Histogram refinement for content-based image retrieval. In *Applications of Computer Vision, 1996. WACV'96., Proceedings 3rd IEEE Workshop on*, pages 96–102. IEEE, 1996.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 833–840, 2011.

Yong Rui, Thomas S Huang, and Shih-Fu Chang. Image retrieval: Current techniques, promising directions, and open issues. *Journal of visual communication and image representation*, 10(1):39–62, 1999.

Arnold WM Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(12):1349–1380, 2000.

Joh R Smith and Shih-Fu Chang. Automated binary texture feature sets for image retrieval. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 4, pages 2239–2242. IEEE, 1996a.

John R Smith and Shih-Fu Chang. Single color extraction and image query. In *Image processing, 1995. Proceedings., International conference on*, volume 3, pages 528–531. IEEE, 1995.

John R Smith and Shih-Fu Chang. Tools and techniques for color image retrieval. In *Electronic Imaging: Science & Technology*, pages 426–437. International Society for Optics and Photonics, 1996b.

Stephen M Smith and J Michael Brady. Susana new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997.

Markus A Stricker and Markus Orengo. Similarity of color images. In *IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology*, pages 381–392. International Society for Optics and Photonics, 1995.

Michael J Swain and Dana H Ballard. Color indexing. *International journal of computer vision*, 7(1):11–32, 1991.

Richard Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006.

Hideyuki Tamura, Shunji Mori, and Takashi Yamawaki. Textural features corresponding to visual perception. *Systems, Man and Cybernetics, IEEE Transactions on*, 8(6):460–473, 1978.

Tomasz Trzcinski, Mario Christoudias, and Vincent Lepetit. Learning image descriptors with boosting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(3): 597–610, 2015.

Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends® in Computer Graphics and Vision*, 3(3):177–280, 2008.

Tinne Tuytelaars and Luc Van Gool. Matching widely separated views based on affine invariant regions. *International journal of computer vision*, 59(1):61–85, 2004.

Abhishek Verma, Sugata Banerji, and Chengjun Liu. A new color sift descriptor and methods for image category classification. In *International Congress on Computer Applications and Computational Science, Singapore*, pages 819–822, 2010.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

Xiaoli Yuan, Jing Yu, Zengchang Qin, and Tao Wan. A sift-lbp image retrieval model based on bag of features. In *IEEE International Conference on Image Processing*, 2011.