5-2011

Feature-based Image Comparison and Its Application in Wireless Visual Sensor Networks

Yang Bai
University of Tennessee Knoxville, Department of EECS, ybai2@utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Yang Bai entitled "Feature-based Image Comparison and Its Application in Wireless Visual Sensor Networks." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Engineering.

Hairong Qi, Major Professor

We have read this dissertation and recommend its acceptance:

Mongi A. Abidi, Qing Cao, Steven Wise

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Feature-based Image Comparison and Its Application in Wireless Visual Sensor Networks

A Dissertation
Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Yang Bai

May 2011

Dedication

I dedicate this dissertation to my parents, who gave me the opportunity to explore the world, and to my wife, who has been always supportive no matter what happens.

Acknowledgments

I would like to thank all the individuals who have inspired, encouraged, and advised me in the preparation of this dissertation.

First and foremost, I would like to thank my advisor, Dr. Hairong Qi. Her willingness to support my work and her guidance throughout my studies has allowed me to develop my skills as a researcher within a supportive team environment. I thank her for that opportunity. I would further like to thank the other members of my committee: Dr. Mongi A. Abidi, Dr. Qing Cao, and Dr. Steven Wise. I greatly appreciate their time and input to this dissertation.

Within the AICIP Lab, I owe many thanks to my fellow graduate students. I enjoyed the many conversations and discussions that have had a tremendous impact on my research and myself as a person. Dayu Yang, Sangwoo Moon, Mahmut Karakaya, Li He, Jiajia Luo, Shuangjiang Li, Bryan Bodkin, Wei Wang, Zhibo Wang, Austin Albright and Paul Donnelly thank you very much.

Last but not the least, I express my deepest appreciation to my parents and my wife, for their unconditional love, support and encouragement.

Abstract

This dissertation studies the feature-based image comparison method and its application in Wireless Visual Sensor Networks.

Wireless Visual Sensor Networks (WVSNs), formed by a large number of low-cost, small-size visual sensor nodes, represent a new trend in surveillance and monitoring practices. Although each single sensor has very limited capability in sensing, processing and transmission, by working together they can achieve various high level tasks. Sensor collaboration is essential to WVSNs and normally performed among sensors having similar measurements, which are called neighbor sensors. The directional sensing characteristics of imagers and the presence of visual occlusion present unique challenges to neighborhood formation, as geographically-close neighbors might not monitor similar scenes. Besides, the energy resource on the WVSNs is also very tight, with wireless communication and complicated computation consuming most energy in WVSNs. Therefore the feature-based image comparison method has been proposed, which directly compares the captured image from each visual sensor in an economical way in terms of both the computational cost and the transmission overhead.

The feature-based image comparison method compares different images and aims to find similar image pairs using a set of local features from each image. The image feature is a numerical representation of the raw image and can be more compact in terms of the data volume than the raw image. The feature-based image comparison contains three steps: feature detection, descriptor calculation and feature comparison.

For the step of feature detection, the dissertation proposes two computationally efficient corner detectors. The first detector is based on the Discrete Wavelet Transform that provides multi-scale corner point detection and the scale selection is achieved efficiently through a Gaussian convolution approach. The second detector is based on a linear unmixing model, which treats a corner point as the intersection of two or three "line" bases in a 3×3 region. The line bases are extracted through a constrained Nonnegative Matrix Factorization (NMF) approach and the corner detection is accomplished through counting the number of contributing bases in the linear mixture.

For the step of descriptor calculation, the dissertation proposes an effective dimensionality reduction algorithm for the high dimensional Scale Invariant Feature Transform (SIFT) descriptors. A set of 40 SIFT descriptor bases are extracted through constrained NMF from a large training set and all SIFT descriptors are then projected onto the space spanned by these bases, achieving dimensionality reduction.

The efficiency of the proposed corner detectors have been proven through theoretical analysis. In addition, the effectiveness of the proposed corner detectors and the dimensionality reduction approach has been validated through extensive comparison with several state-of-the-art feature detector/descriptor combinations.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Image Comparison from Visual Features

Image comparison is an important preprocessing step for many computer vision applications, such as object recognition and tracking, target localization, image registration and 3D reconstruction [Faugeras 1993]. A general image comparison problem starts with several images of the same object taken from different perspectives, and one of them is used as a "query". Then a successful image comparison algorithm would find all images from the same object, among many other images from different objects.

Image comparison is similar to the *query by visual example* paradigm in **Content-based Image Retrieval** (CBIR) [Vasconcelos 2007]. However, in image comparison, the requirement is more restrictive, where only images from the exact same object are considered as correct solutions.

Image comparison has to rely on visual features to measure the image similarity. The image feature is defined as a numerical representation of an image which can capture a certain visual property. The image feature can be classified into global features which describe the entire image or local features which describe only a small group of pixels. Global features include the image metadata, such as captions, tags, keywords or other descriptions to the image. However, the metadata has to be added manually, which is time-consuming, laborious and expensive. Another type of the global feature, such as histograms or other image statistics, can be generated automatically from the image content. The global features are usually too rigid to represent an image. Specifically, they tend to be over sensitive to image variations. For example, the histogram is sensitive to illumination changes.

On the other hand, local features, which is calculated for a single pixel (for example, the local gradient histogram for a small region centering at that pixel) and then summarized from many pixels for the whole image, have the flexibility that although each feature only describes a small region, when put together, they can be representative for the whole image.

Color features and texture features are popular visual image features that can be used as either global features or local features, depending on the region from which they are calculated.

Color features were originally focusing on exploiting color spaces (for example, the LUV space) which was believed to better coincide with human vision system than the RGB color space. Recently, research on color features focused more on how to construct signatures from colors, including histogram-based descriptors, spatial color descriptors, and texture descriptors [Datta et al. 2008; Manjunath et al. 2001].

Texture features reflect the visual pattern that has properties of granularity or repetition from either the whole image or a small region. For example, grassland, brick walls, teddy bears, and flower petals differ in texture, by smoothness as well as patterns [Datta et al. 2008]. Texture features have been studied since a long while ago in both computer vision and computer graphics [Haralick 1979; Malik and Perona 1990; Unser 1995].

The salient point based features are local features that gain their popularity recently. Salient point based features, such as corner points, keypoints or interest points, originally used for stereo matching, are also being used for image comparison. A recent trend is to develop scale and affine invariant interest points that are robust to both affine transformations and illumination changes for image comparison [Bay et al. 2006; Lowe 2004; Mikolajczyk and Schmid 2004]. These special feature points are important because their locations usually mark important regions of an image, which lead to efficient indexing and sufficient discriminative power. Once salient points have been detected from an image, a local feature descriptor would be calculated based on the intensity information in the neighborhood. The popularity of salient point based feature in image comparison has been witnessed by a large number of recent publications [Bouchard and Triggs 2005; Carneiro and Lowe 2006; Fergus et al. 2003, 2005; Lazebnik et al. 2003].

In this dissertation, the salient point based features will be used for image comparison purpose, in favor of its robustness to scale, rotation, translation, affine and illumination variations.

1.2 Wireless Visual Sensor Networks

A Wireless Visual Sensor Network (WVSN) is a group of networked smart cameras with image/video capturing, computing and wireless communication capabilities powered by on-board batteries [Kundur et al. 2007]. WVSNs are also referred to as Distributed Smart Cameras (DSCs), although their focus is a bit different. For example, a WVSN tends to refer to the macrostructure and a DSC, emphasizing more on the single camera node. Besides WVSN and DSC, some other terminologies [Rinner and Wolf 2008] have also been used to refer to the same or similar concept, such as Wireless Multimedia Networks [Akyildiz et al. 2008]. The application of WVSNs has spanned a wide spectrum of domains, including environmental surveillance, human behavior monitoring, object tracking and recognition, etc. [Obraczka et al. 2002; Sankaranarayanan et al. 2008].

From the perspective of a single sensor node, since each node is built at low cost, many of its capabilities would have to be limited. For example, the sensing range is limited, the sensor reading sometimes may be unreliable, the on-board processor is not powerful enough to handle complicated processing, the storage space is limited, the communication is restricted to a short range, and its lifetime is constrained by the on-board battery which is impractical to replace or recharge. But on the other hand, the low cost of each individual sensor node also facilitates its massive production and deployment. When tens of or even hundreds of sensor nodes are deployed, they can form an ad hoc network and perform collaboration in sensing, processing and communication, thus providing capabilities greater than the sum of individual nodes.

From the perspective of the WVSN as a whole, the collaboration among sensors is a key in compensating for each sensor's limitations. Through collaboration, the sensing area will be determined by the extent and the density how the sensors are deployed, the sensor reading can be corrected by fusing more than one measurement, a complex processing task can be spread among several sensor nodes or executed at a central processor and each sensor node only needs to relay the measurements, the storage limit can be accommodated by transmitting measurements on each sensor node back to a central storage unit, and the communication can be bridged by adjacent sensor nodes relaying each other's message to reach a longer range.

The major difference between Wireless Sensor Networks (WSNs) using scalar sensors (such as thermo sensors and acoustic sensors) and WVSNs lies in the unique properties of the image sensor, i.e., directional sensing and intensive data processing.

The sensing range of scalar sensors can be modeled by a round disk with a fixed radius. However, the sensing range of a camera is a sector (called the Field Of View, or FOV), specified by the orientation and radius parameters. Therefore, while a scalar sensor collects data from its vicinity, a visual sensor can only collect data residing in its FOV. This fact results in an important difference in sensor clustering algorithms between general WSNs and the WVSNs. In a general WSN, when sensors are geographically close, they would sense the similar scene and have similar measurements; however, in a WVSN, sensors being geographically close may not sense the similar scene due to orientation differences or visual occlusions.

The readout from a scalar sensor is a 1D signal, which is relatively cheap in terms of processing, storage and transmission. However, the readout from a camera is a 2D image, which provides richer information but is also more demanding in data processing, storage and transmission. Considering a typical acoustic sensor working at the sampling rate of 1024 samples per second, even a low resolution camera of 640×480 pixels working at a low sampling rate of 5 frames per second would generate $1,536,000$ samples every second, which is 1500 times of the data generated by a scalar sensor.

The wireless setup and the visual sensors' property bring up four major challenges for WVSNs:

Resource Requirements. The lifetime of each camera node is determined by the on-board battery, whose usage is proportional to the energy required by the sensing, processing and transceiving modules. Considering the huge amount of raw data generated by the cameras, both processing and transceiving are much expensive in terms of energy consumption, much more than other scalar sensor networks. Moreover, image data transceiving is more bandwidth intensive. The last but not the least, storing the raw image data from a long period of measurement on the node requires much space in storage, which puts the third resource constraint on a WVSN. Therefore, the energy, the bandwidth and the storage space are three even more scarce resources in WVSNs than in any other scalar WSNs.

Local Processing. Local processing refers to the technique of processing the raw data (image) on each camera node right after the image has been captured. Local processing helps to reduce the data amount needs to be stored and transmitted. Local processing can be any type of image processing techniques from background subtraction for motion detection, feature (edge, corner or interest point) extraction to object classification. Basically, the local processing algorithm depends on the specific application of a WVSN and the algorithm complexity can vary a lot.

Camera Location and Orientation. Due to the unique property of the visual sensing modality, the camera location and orientation information is a must-have for sensor node collaboration. This information can always be obtained through a camera calibration process, which retrieves the camera's intrinsic and extrinsic parameters. Based on this information, a deployment map of the WVSN can be built. A deployment map contains the information of each sensor node's location and orientation, and is useful for sensor clustering in WVSNs. However, the deployment map can not handle the occlusion condition, where an object stays at the front of a camera and keep the camera from seeing other objects. Besides, the calibration process for a huge amount of camera nodes tends to be time consuming and sometimes even impractical.

Camera Collaboration. Camera collaboration should rely on exchanging the information from each camera node. Through collaboration, the cameras relate the events captured in the images and enhance their understanding toward the environment. The exchanged information could be low level image features or simple description of the images.

1.3 Motivations

To address the four major challenges posed by the WVSNs, the feature-based image comparison method has been proposed that can: alleviate the **resource requirements** by locally extracting image features such that the more compact image features are stored and transmitted, therefore achieving energy conservation from transmission; implement the **local processing** effectively by designing computationally light-weight feature detectors; avoid acquiring accurate **camera location and orientation** information by comparing the images captured by visual sensors directly, which can also handling visual occlusions; assist **camera collaboration** efficiently by letting each sensor node exchange the image features instead of the bulky raw image data.

1.4 Contributions

The dissertation work presents a set of innovative solutions to answer two questions for a WVSN: How to process the image locally so that the useful information can be extracted efficiently; What kind of information should be propagated in the WVSN so that the resource constraints could be satisfied.

Firstly, a computationally light-weight corner detector has been designed. The detector can be used for feature point detection on the raw image. Then for each feature point, a feature descriptor can be calculated to describe the local pattern. An effective dimensionality reduction technique has also been provided to compress the high-dimensional feature descriptors. The corner detector design is based on a linear unmixing model that defines a corner point as the intersection of two or three “line” patterns and examines the number of contributing line bases for corner identification. The set of line bases in the linear unmixing model is extracted through a constrained Nonnegative Matrix Factorization (NMF) approach. The corner detector is named **LUC**, standing for the Linear Unmixing based Corner detector. By utilizing the existing powerful local image feature descriptor, the Scale Invariant Feature Transform (SIFT) descriptor, a feature-based image representation can be formulated where each feature descriptor corresponds to a detected corner point, and a set of feature descriptors are used to represent an image. Then through the constrained NMF approach, a smaller set of bases for the SIFT descriptor can be extracted. By projecting the SIFT descriptors onto the space spanned by these bases, the dimensionality reduction can be achieved. In a word, on each visual sensor node, by performing the above mentioned process, a compact, feature-based image representation is computed and ready to be transmitted.

Secondly, in the WVSN, the image feature sets will be propagated, which is in a more compact form than the original image. Therefore the communication bandwidth and the energy can be consumed in a more efficient way. An additional benefit of propagating the image feature sets is that the image feature calculation is carried out on each individual sensor, thus distributing the computation burden over the entire network.

Besides, the effectiveness of the LUC corner detector and the feature-based image comparison scheme have been evaluated through thorough experiments on three datasets.

1.5 Dissertation Outline

The dissertation is organized as follows:

Chapter 2 provides a literature survey on classical and state-of-the-art approaches used in the three components involved in feature-based image comparison method; Chapter 3 introduces two computationally light-weight corner detectors, the DWT-based corner detector and the LUC detector; Chapter 4 describes how to perform dimensionality reduction for the high dimensional SIFT descriptors; Chapter 5 explains the evaluation metrics for feature detectors and feature descriptors, as well as the experimental results of the feature-based image comparison method using different feature detector/descriptor combinations on the Oxford dataset [Oxford 2002]; Chapter 6 shows the results of the feature-based image comparison on two image datasets collected from a simulated WVSN and a real WVSN, i.e., the Columbia object image library (COIL-100) dataset [Nene et al. 1996] and the Mobile Sensor Platform (MSP) dataset collected by ourselves. Finally, the dissertation is concluded with accomplished and future work in Chapter 7.

Chapter 2

Literature Survey

Feature-based image comparison contains three major steps, feature detection, feature description and feature comparison. This chapter provides a literature survey covers classical and state-of-the-art algorithms used in these three steps.

Image feature is a numerical description of the properties of an image. Global features and local features are two types of image features that are generally used for image classification and object recognition purpose. The histogram, shape, texture and eigenspace are all global features. For each image, global feature generates one feature vector, which is more compact than the original image and can be easily treated as an observation and used for standard classification algorithms. But the shortcoming of the global features is that they could only provide a coarse description of the image and thus cannot distinguish subtle differences between images [Oliva and Torralba 2006].

Local image features describe a local region (a "patch") surrounding a feature point using a feature descriptor. A descriptor is a vector that describes the intensity variation in a local region. The disadvantage of local features is that the numbers of features (and accordingly, descriptors) are not the same for different images, and therefore they cannot be used for standard classification procedure directly.

In the WVSN environment most images are similar, and therefore would be difficult to be distinguished by global features, local features will be used for image comparison purpose. The following feature detectors and descriptors are all examples of local features, and the feature comparison method is suitable for local features, too.

2.1 Feature Detection

This section reviews several corner detectors and two blob-like feature detectors.

In general, corner detection algorithms can be classified into two categories, i.e., the contour curvature based methods and the intensity based methods.

Corner detectors of the first kind work on contour images where contours are often encoded in chains of points or represented in a parametric form using splines [Langridge 1982; Medioni and Yasumoto 1987; Mokhtarian and Suomela 1998; Wang and Brady 1995]. Corner points are detected along the contours with high curvatures where the curvature of a curve is defined as the changing rate of the unit tangent vector with respect to the arc length. Contour curvature based corner detection methods can accurately locate the corner point. However, they cannot be applied on natural images directly, but rely on the preprocessing of edge detection methods. Therefore, their performance is heavily affected by the edge detection step and the algorithm efficiency is low. These methods were most popular in the 1980s and 1990s. For a comprehensive list of the literatures related to these detectors, please refer to [Dutta et al. 2008; Tuytelaars and Mikolajczyk 2007].

Corner detectors of the second kind are intensity based methods, including the Moravec's detector [Moravec 1980], Harris detector and its many variations [Harris and Stephens 1988; Mikolajczyk and Schmid 2004; Shi and Tomasi 1994], the Smallest Univalue Segment Assimilating Nucleus (SUSAN) detector [Smith and Brady 1997], and the Features from Accelerated Segment Test (FAST) detector [Rosten and Drummond 2005, 2006]. This section reviews the Moravec's detector, the Harris detector, its several variations and the FAST detector.

In recent years, blob-like feature detectors, as well as their corresponding feature descriptors, are attracting more attentions. The Scale Invariant Feature Transform (SIFT) [Lowe 2004] and the Speeded-Up Robust Feature (SURF) [Bay et al. 2008] are two typical examples of this kind. The detectors in SIFT and SURF both detect blob-like features from multi-scale image space and use the blob center as the feature points. This section will also review the detectors in SIFT and SURF.

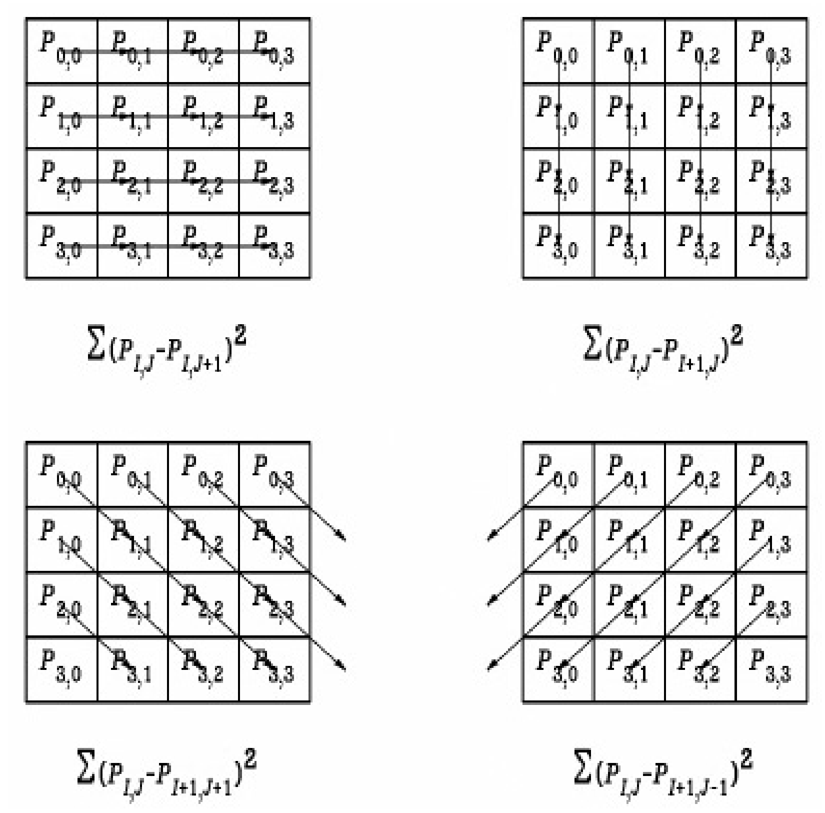

$\sum(P_{I,J}-P_{I,J+1})^2$
$\sum(P_{I,J}-P_{I+1,J})^2$
$\sum(P_{I,J}-P_{I+1,J+1})^2$
$\sum(P_{I,J}-P_{I+1,J-1})^2$

Figure 2.1: Four windows and their corresponding SSDs (adapted from [Moravec 1980]).

2.1.1 Moravec's Corner Detector

Moravec proposed a corner detector used in the navigation of a robot rover for obstacle avoidance [Moravec 1980]. The corners are found through a two-step procedure: In the first step, for every pixel, four Sums of Squares of Differences (SSDs) along four directions (horizontal, vertical and two diagonals) are calculated within rectangular windows and the minimum of these four is selected as the corner strength measure; in the second step, the corner strength measure is compared with its neighbors' and a corner point will be found if its corner strength measure is the local maximum. Figure 2.1 illustrates the first step where SSDs are calculated in the four windows. Here $P_{I,J}$ stands for the grayscale value at coordinate (I, J). Every arrow connects two corresponding pixels whose difference is to be calculated.

Another way to interpret the SSD calculation is to treat the upper-left square window in Fig. 2.1 as a 4×3 rectangular window shifting one pixel to the east and the rest three

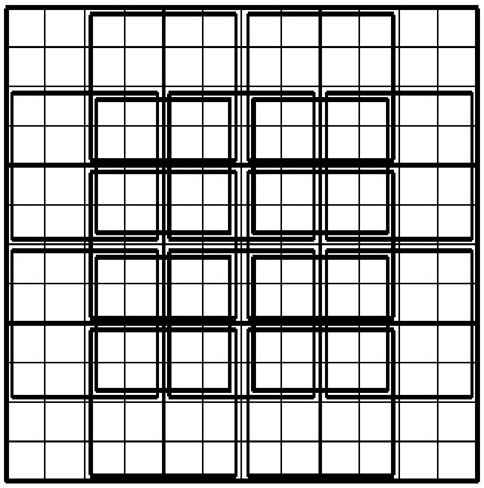


Figure 2.2: The 25 4×4 square windows spread within a 12×12 square area. The local maximum of corner strength measure is considered over this area (adapted from [Moravec 1980]).

square windows as 3×4 rectangular windows shifting one pixel to the south, southeast and southwest, respectively. Then the SSDs are calculated between the corresponding pixels within the prior-shifting windows and post-shifting windows.

Fig. 2.2 illustrates how to find the local maximum of the corner strength measures in a 12×12 square area. 4×4 windows are spaced 2 pixels apart over the entire area, and there are 25 windows in total. The 4×4 square window at the center will be declared to contain a corner point if its corner strength measure is greater than those of the other 24 neighbor windows.

By taking the minimum value of four SSDs, the strong SSD responses corresponding to edges along the four directions can be eliminated. By taking the local maximum of all corner strength measures, the weak corner strength corresponding to smooth image areas can be further filtered out. So the remaining would imply corner points. Although there are only four directions of shifting the rectangular window, by uniformly and densely placing these windows, the Moravec corner detector can filter out the edges with strong SSD responses along all eight directions.

However, there are several fundamental limitations for the Moravec's corner detector [Harris and Stephens 1988; Parks 2006]. The first is that it responds too readily to edges along the directions at the multiples of 45°, or in short, its response is anisotropic. The second, as pointed out by Moravec, is that the windows used in Moravec's detector is rectangular and binary, which leads to a noisy response [Moravec 1980]. The third is that the Moravec's detector responds to edges strongly since some imperfections in an edge due to the noise, pixelation, or intensity quantization may result in the local minimum of intensity variation over all shift directions being relatively large.

In Moravec's original work, the windows used for calculating SSDs usually had sizes of 4×4 (as shown in Fig 2.1) or 8×8. However, most literatures since then tended to use 3×3 or 5×5 windows. By using odd size windows, it is easier to define the window center.

2.1.2 Harris Corner Detector

The mathematical definition of SSD was not explicitly provided in [Moravec 1980], although Fig. 2.1 shows four equations for SSD calculation in each special case. Harris gave the general equation for calculating SSD in [Harris and Stephens 1988]

$$E_{x,y} = \sum_{u,v} w_{u,v} |I(u+x, v+y) - I(u,v)|^2 \tag{2.1}$$

where E denoted the SSD, I stood for the grayscale value, u and v were the coordinates of the reference point (the current pixel being examined), x and y indicated the shift, w specified the window used to calculate the SSD: it was uniform within the window and zero elsewhere. The shifts, i.e., values of (x, y) corresponding to the four directions were comprised of $(0,1), (1,1), (1,0), (-1,1)$.

By realizing the three limitations of the Moravec's corner detector, and based on the definition of the general SSDs, Harris proposed his modification toward Moravec's detector and this modification was named Harris corner detector. Starting from Eq. 2.1, plug in the Taylor expansion

$$I(u+x, v+y) = I(u,v) + I_x(u,v)x + I_y(u,v)y + O(x^2, y^2) \tag{2.2}$$

to obtain

$$\begin{aligned} E_{x,y} &\approx \sum_{u,v} w_{u,v}\ (I_x(u,v)x + I_y(u,v)y)^2 \\ &= \sum_{u,v} w_{u,v} \left([I_x(u,v)\ I_y(u,v)] \begin{bmatrix} x \\ y \end{bmatrix} \right)^2 \\ &= [x\ y] \begin{bmatrix} \sum_{u,v}(I_x(u,v))^2 & \sum_{u,v} I_x(u,v)I_y(u,v) \\ \sum_{u,v} I_x(u,v)I_y(u,v) & \sum_{u,v}(I_y(u,v))^2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\ &= [x\ y]\ C(u,v) \begin{bmatrix} x \\ y \end{bmatrix} \end{aligned} \tag{2.3}$$

where I_x and I_y are partial derivatives of I. $E_{x,y}$ is closely related to the local autocorrelation function, with $C(u,v)$, a second moment matrix, describing its shape at the origin. Let λ_1, λ_2 be the eigenvalues of matrix $C(u,v)$. The eigenvalues form a rotation-invariant description. Harris corner detector is based on the properties of the two eigenvalues and has three scenarios to be considered:

1. If both λ_1 and λ_2 are small, the local autocorrelation function is flat, and the windowed image region has approximately constant intensities.
2. If one eigenvalue is large and the other is small, the local autocorrelation function is ridge shaped, then this indicates an edge.
3. If both eigenvalues are large, then this indicates a corner.

In order to quantify the corner strength and avoid the explicit eigenvalue decomposition of matrix $C(u,v)$, which is computationally expensive, Harris proposed to use

$$R(u,v) = det(C(u,v)) - k \times\ trace^2(C(u,v)) \tag{2.4}$$

as the corner strength measure at coordinate (u,v). $det(\cdot)$ is the determinant of a matrix and $trace(\cdot)$ is the trace of a matrix. The value of k has to be determined empirically and has been suggested to be in the range of $[0.04, 0.15]$ [Schmid et al. 2000]. So for corner regions, R has large positive values; for edge regions, R has negative values; for flat regions,

R has small positive values.

Through the derivation from Eq. 2.1 to Eq. 2.3, all possible small shifts can be covered and thus the Harris detector is isotropic. Through the introduction of a new corner strength measure defined in Eq. 2.4, the Harris detector can exclude responses to edges and leave only corners. To overcome the noise in Moravec's detector, Harris suggested a smooth circular window instead of a uniform square window. For example, a Gaussian window

$$w_{u,v} = exp(-(u^2 + v^2)/2\sigma^2). \tag{2.5}$$

2.1.3 Variations of the Harris Corner Detector

After Harris introduced his corner detector, several variants have been proposed.

Shi and Tomasi corner detector. Shi and Tomasi proposed a corner detector that also based on the second moment matrix $C(u, v)$ [Shi and Tomasi 1994]. Instead of using Eq. 2.4, they chose $min(\lambda_1, \lambda_2)$ as the corner strength measure and showed, through experiments, that it was superior to the Harris detector for object tracking purpose.

Multi-scale Harris detector. Scale-space representation of an image is obtained by convolving an image $I(u, v)$ with a 2D Gaussian kernel

$$g(u, v; t) = \frac{1}{2\pi t} e^{-(u^2+v^2)/2t^2} \tag{2.6}$$

such that

$$\begin{aligned} L(u, v; t) &= I(u, v) * g(u, v; t) \\ &= (g_t * I)(u, v) \end{aligned} \tag{2.7}$$

where $L(u, v; t)$ is the scale-space representation of image $I(u, v)$ and t is the scale variable. The partial derivatives defined in the scale-space are

$$\begin{aligned} L_u(u, v; t) &= \frac{\partial}{\partial u} L(u, v; t) \\ L_v(u, v; t) &= \frac{\partial}{\partial v} L(u, v; t) \end{aligned} \tag{2.8}$$

The computation of the second moment matrix $C(u, v)$ in the original Harris detector requires the computation of image derivatives I_x and I_y as well as the summation of non-linear combinations of these derivatives over local windows. This corresponds to, under the scale-space representation, two additional variables: t indicates the scale and s indicates the size of the Gaussian window function $g(u, v; s)$ [Lindeberg and Garding 1997].

$$\begin{aligned} C(u,v;t,s) &= \begin{bmatrix} L_x^2(u,v;t) & L_x(u,v;t)L_y(u,v;t) \\ L_x(u,v;t)L_y(u,v;t) & L_y^2(u,v;t) \end{bmatrix} * g(u,v;s) \\ &= \int_{\xi=-\infty}^{\infty}\int_{\eta=-\infty}^{\infty} Tg(\xi,\eta;s)d\eta d\xi \end{aligned} \tag{2.9}$$

where

$$T = \begin{bmatrix} L_x^2(u-\xi,v-\eta;t) & L_x(u-\xi,v-\eta;t)L_y(u-\xi,v-\eta;t) \\ L_x(u-\xi,v-\eta;t)L_y(u-\xi,v-\eta;t) & L_y^2(u-\xi,v-\eta;t) \end{bmatrix} \tag{2.10}$$

Then, the corner strength measure $R(u, v; t, s)$ can be computed in a similar way as that in the Harris detector

$$R(u,v;t,s) = det(C(u,v;t,s)) - k \times trace^2(C(u,v;t,s)) \tag{2.11}$$

The relationship between the scale parameter t and the integration window parameter s is given by

$$s = \gamma^2 t \tag{2.12}$$

where γ is usually chosen in the range $[\sqrt{2}, 2]$. Thus the corner strength measure is a function of scale variable t as $R(u, v; t)$ and can be used to evaluate the strength of varying size corners at different scale t [Baumberg 2000].

In practice, this multi-scale corner detector is often complemented by a scale selection step, where the scale-normalized Laplacian operator

$$\begin{aligned} \bigtriangledown_{norm}^2 L(u,v;t) &= t \bigtriangledown^2 L(u,v;t) \\ &= t(L_{uu}(u,v;t) + L_{vv}(u,v;t)) \end{aligned} \tag{2.13}$$

is computed at every scale t in scale-space [Lindeberg 1998]. And the scale adapted corner points with automatic scale selection (the "Harris-Laplace operator") are computed from the points that satisfy the following two conditions at the same time [Mikolajczyk and Schmid 2004]:

- spatial maximum of the multi-scale corner strength measure $R(u, v; t)$ at the vicinity of (u, v)

$$(\hat{u}, \hat{v}; t) = \arg \max_{u,v} \ R(u, v; t) \tag{2.14}$$

- local maximum or minimum over scales of the scale-normalized Laplacian operator $\bigtriangledown^2_{norm}(u, v; t)$:

$$\hat{t} = \arg \max_{t} \min \ \bigtriangledown^2_{norm} L(u, v; t) \tag{2.15}$$

2.1.4 Harris Detector for Color Images

There are two ways to extend the Harris corner detector for color images. The first is to treat each plane of the color space separately and combine the corner detection results afterwards; The other way is to define the color image as a vector-valued function $I : Z^2 \rightarrow Z^3$. Then the SSD is defined as

$$E_{x,y} = \sum_{u,v} w_{u,v}(I(u + x, v + y) - I(u, v))^2 \tag{2.16}$$

where $I(u, v) = (f^1(u, v) \ f^2(u, v) \ f^3(u, v))'$, and $f^j, \quad j = 1, 2, 3$ represents each color plane. The Taylor expansion for vector-valued function is [Wilson 2008]:

$$I(u + x, v + y) = I(u, v) + I_x x + I_y y + \frac{1}{2}(I_{xx}x^2 + 2I_{xy}xy + I_{yy}y^2) + \ldots \tag{2.17}$$

Plug Eq. 2.17 into Eq. 2.16, will have

$$
\begin{aligned}
E_{x,y} &\approx \sum_{u,v} w_{u,v}(I_x(u,v)x + I_y(u,v)y)^2 \\
&= \sum_{u,v} w_{u,v} \left(\begin{bmatrix} f_x^1(u,v)x \\ f_x^2(u,v)x \\ f_x^3(u,v)x \end{bmatrix} + \begin{bmatrix} f_y^1(u,v)y \\ f_y^2(u,v)y \\ f_y^3(u,v)y \end{bmatrix} \right)^2 \\
&= \sum_{u,v} w_{u,v} \left(\begin{bmatrix} f_x^1(u,v) & f_v^1(u,v) \\ f_x^2(u,v) & f_v^2(u,v) \\ f_x^3(u,v) & f_v^3(u,v) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \right)^2 \\
&= [x\ y] \begin{bmatrix} \sum_{u,v} w_{u,v} \sum_{i=1}^3 f_x^i(u,v)^2 & \sum_{u,v} w_{u,v} \sum_{i=1}^3 f_x^i(u,v) f_y^i(u,v) \\ \sum_{u,v} w_{u,v} \sum_{i=1}^3 f_x^i(u,v) f_y^i(u,v) & \sum_{u,v} w_{u,v} \sum_{i=1}^3 f_y^i(u,v)^2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \\
&= [x\ y] C(u,v) \begin{bmatrix} x \\ y \end{bmatrix} \qquad (2.18)
\end{aligned}
$$

And the same conclusions on the properties of $C(u,v)$ can be used to analyze the existence of corner points. The derivation was first developed by Zenzo [Zenzo 1986].

2.1.5 FAST Corner Detector

Rosten and Drummond firstly proposed the Features from Accelerated Segment Test (FAST) corner detector for the purpose of real-time 3D model-based tracking [Rosten and Drummond 2005], and then refined the implementation of FAST through a decision tree [Rosten and Drummond 2006].

The FAST corner detector examines, for a pixel c, a circle of 16 pixels (a Bresenham circle of radium 3) surrounding c. A corner point will be detected at c if at least 12 contiguous pixels are brighter or darker than c by some threshold t, as shown in Fig. 2.3. The test for this example can be further optimized by firstly examing pixel 1, 9, 5 and 13, to reject candidate pixels more quickly. Because a corner point can only exist at c if three of these test pixels are all brighter all darker than c by the threshold.

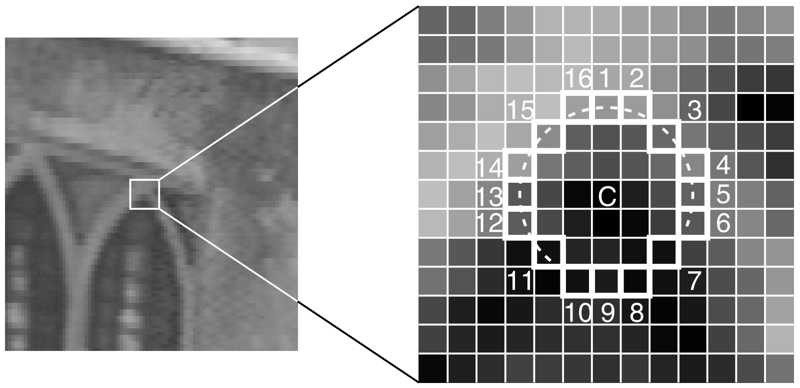


Figure 2.3: FAST corner detector in an image patch. The highlighted region contains the pixel c currently under test. The dashed line connects 12 pixels all brighter than c above the threshold, and c is detected as a corner point [Rosten and Drummond 2005]).

2.1.6 The Feature Detector in Scale Invariant Feature Transform (SIFT)

SIFT was introduced by Lowe in 1999 [Lowe 1999], and the feature points are named keypoints. SIFT includes four major stages for keypoint detection and keypoint descriptor calculation [Lowe 2004]:

Stage 1: Scale-space extrema detection. The image scale space is constructed by convolving the image with the Difference-of-Gaussian (DoG) functions of varying sizes. Then local extrema at all scales are located.

Stage 2: Keypoint localization. At each local extreme found from stage one, a detailed model is fit to determine the location and scale in sub-pixel/sub-scale accuracy. Keypoints detected on edges are removed because of their low stability.

Stage 3: Orientation assignment. One or more orientations are assigned to each keypoint based on directions of local gradients. The following descriptor calculation is performed in a square window confined to the assigned orientation, scale, and location for each keypoint, therefore providing invariance to these transformations.

Stage 4: Keypoint descriptor. The local gradients are calculated at the selected scale in a square window surrounding each keypoint. Then the histogram of the gradient orientations in each window are recorded in a 128-dimensional vector, used as the feature descriptor for that keypoint.

In Stage 1, the scale space of an image is defined as a function, $L(x, y, \sigma)$, which is the convolution of a scale-varying Gaussian function, $G(x, y, \sigma)$, with an input image, $I(x, y)$

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.19}$$

where $*$ is the convolution operator, and

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}. \tag{2.20}$$

The Difference-of-Gaussian (DoG) function is defined as

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= (L(x, y, k\sigma) - L(x, y, \sigma)) \end{aligned} \tag{2.21}$$

where k is a constant.

Choosing the DoG function has two reasons. The first is that computing DoG can be very efficient, and the second is that it approximates the scale-normalized Laplacian of Gaussian, $\sigma^2 \bigtriangledown^2 G$ [Lindeberg 1994], which is a multi-scale edge detector and robust to noise. An efficient way to build the DoG functions, $D(x, y, \sigma)$, is shown in Fig. 2.4. The original image is incrementally convolved with Gaussian functions to produce images separated by a constant factor k in scale space, as shown on the left side of Fig. 2.4. Each octave of scale space (i.e., doubling of σ) is divided into an integer number, s, of intervals, so $k = 2^{1/s}$. Consecutive image scales are subtracted to generate the DoG images as shown on the right side of Fig. 2.4.

To find the local maxima and minima of the DoG function, $D(x, y, \sigma)$, each sample point will be compared to its eight neighbors in the image space (at the same scale) and eighteen neighbors in the scale above and below, as shown in Fig. 2.5. Then a quadratic function is used to determine the local extrema at subpixel accuracy [Brown and Lowe 2002]. Since the DoG function approximates the Laplacian of Gaussian function, which in essence is an edge detector, the local extrema in DoG contain not only stable keypoints, but also unstable points on edges. Therefore, the Hessian matrix at each local extreme is checked and those with two eigenvalues varying a lot are treated as unstable keypoints, therefore will be removed. The scale of a keypoint is assigned as the scale at which the

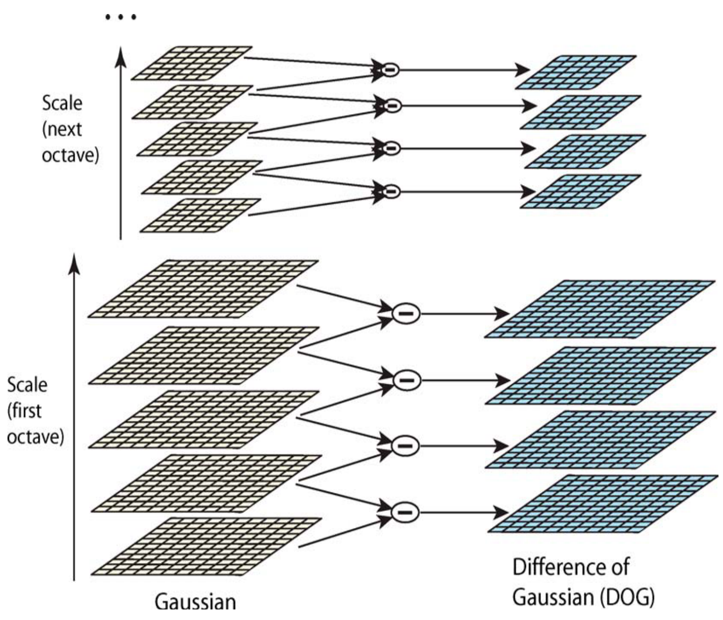


Figure 2.4: The octaves and DoGs (adapted from [Lowe 2004]).

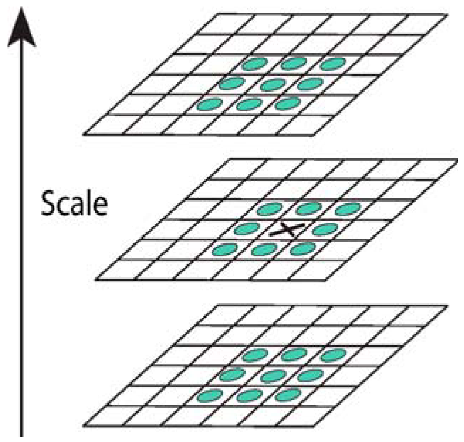


Figure 2.5: A local extreme in scale space (adapted from [Lowe 2004]).

local extreme has been detected.

In stage 4, a histogram is formed from the orientations of sample points' gradients within a square window surrounding the keypoint. The histogram has 36 bins covering the 360 degree range of orientations, and the dominant bin(s) correspond(s) to the orientation(s) of this keypoint.

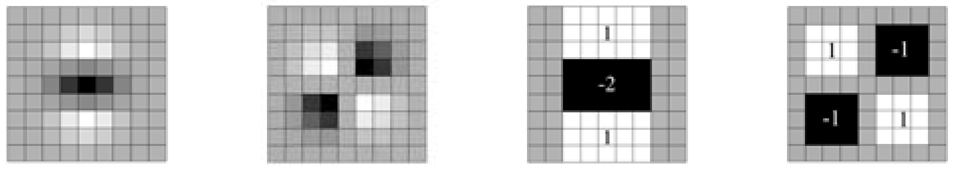


Figure 2.6: Discrete 2nd order Gaussian partial derivatives (left two) and approximation templates (right two) (adapted from [Bay et al. 2006]).

2.1.7 The Feature Detector in Speed-Up Robust Features (SURF)

SURF, following the similar principle as SIFT, was proposed by Herbert Bay in 2006 [Bay et al. 2006]. The feature points in SURF are named interest points. The interest points defined in SURF are detected by Hessian matrix — the local maximum of the Hessian matrix determinant indicates a blob-like structure. SURF adopted a set of templates as approximations of second order Gaussian derivatives, as shown in Fig. 2.6.

Viola proposed the "integral image" approach [Viola and Jones 2001] in order to achieve a fast convolution for rectangular features. The concept of integral image is similar to the Summed-Area Table (SAT) used in computer graphics [Crow 1984]. Using the SAT serves as a vital role for fast convolution, thus speeding up SURF's computation.

The SAT value at coordinate (x, y) is the sum of all pixels above and to the left of (x, y), assuming the origin is at the upper-left corner of the image (the shaded region), as shown in Fig. 2.7 (a),

$$SAT(x,y) = \sum_{x' \leq x, y' \leq y} I(x', y'), \tag{2.22}$$

where $I(x, y)$ is the original image. The computation of SAT can be done in one sweep from left to right and from top to bottom by

$$\begin{aligned} SAT(x,y) &= SAT(x,y-1) + SAT(x-1,y) + \\ &\quad I(x,y) - SAT(x-1,y-1) \end{aligned} \tag{2.23}$$

with $SAT(-1, y) = SAT(x, -1) = 0$.

Once the SAT has been built, it can be used to calculate the area of any rectangular region, as shown in Fig. 2.7 (b). To calculate the area of the shaded region (marked by the "plus" sign at the lower right), the SAT value at its lower right corner needs to minus

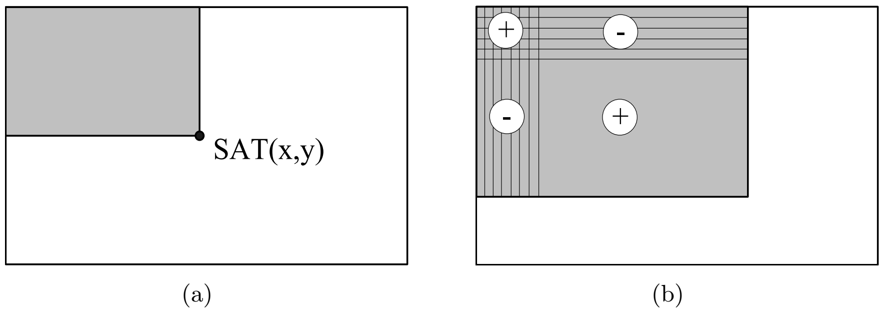


Figure 2.7: Definition of SAT (a) and how to calculate the area for a rectangular region using SAT (b) (redrawn from [Viola and Jones 2001]).

the SAT values at its lower left and upper right corners, and then add the SAT value at its upper left corner.

Since the second order Gaussian derivatives can be approximated through convolution with certain templates, that is, the addition/subtraction of several rectangular areas, the fast convolution can be implemented through SAT.

Scale-space maxima detection and interest point localization. To find the interest points at different scales, templates shown in Fig. 2.6 are up-scaled in size to convolve with the SAT image and thus generate the scale space representation. Then a 3D non-maximum suppression is applied to find the local maxima of the Hessian matrix determinants. After an interpolation in both the image space and the scale space, the locations of interest points are found at sub-pixel accuracy.

Orientation assignment. To find the orientation for each detected interest point, the SAT image at the scale where the local maximum has been detected will again convolve with two Haar wavelet templates. Once the wavelet responses of pixels within a circular neighborhood of the interest point are calculated and weighted with a Gaussian window ($\sigma = 2s$, where s is the scale parameter) centered at the interest point, the responses are represented as points in a 2D space with the horizontal response strength along the abscissa and the vertical response strength along the ordinate. The dominant orientation is estimated by calculating the sum of all responses within a sliding window of size (spanning angle) $\frac{\pi}{3}$, as shown in Fig. 2.8.

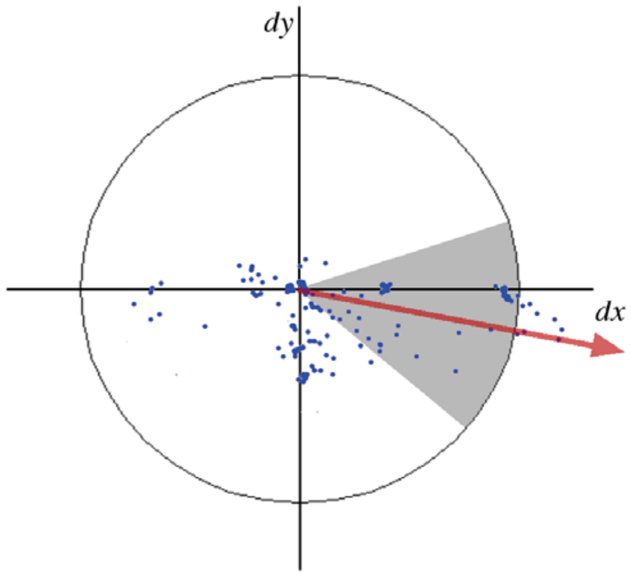


Figure 2.8: Orientation assignment for an interest point: A sliding window (the shaded region) of the size $\frac{\pi}{3}$ detects the dominant orientation of the Gaussian weighted Haar wavelet responses within a circular neighborhood (adapted from [Bay et al. 2008]).

2.2 Feature Description

This section reviews how to calculate the feature descriptors in SIFT [Lowe 2004] and SURF [Bay et al. 2008]. A much compact feature descriptor, the Moment Invariants [Hu 1962] based descriptor, is also reviewed.

2.2.1 Feature Descriptor in Scale Invariant Feature Transform (SIFT)

In SIFT, the feature descriptor for each keypoint is calculated on the image at the same scale as the keypoint has been detected. To achieve the orientation invariance, the calculation of the gradient orientations are aligned with the keypoint orientation. As an example, the gradient amplitudes and orientations are shown on the left side in Fig. 2.9.

The amplitudes of the gradients are weighted by a Gaussian function with σ equal to one half of the width of the square window. These weighted gradients are then accumulated into gradient orientation histograms summarizing the contents over the 4×4 subregions, as shown on the right of Fig. 2.9, with the length of each arrow corresponding to the sum of the gradient amplitudes on that direction within the subregion. Figure 2.9 shows a 2×2 descriptor array computed from 8×8 samples, for illustration purpose. In practise, a 4×4

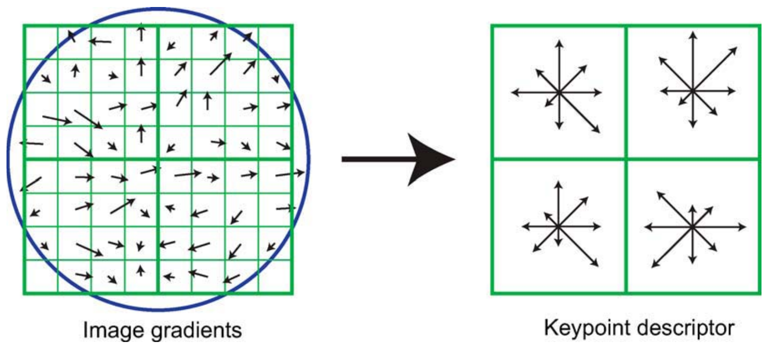


Figure 2.9: Calculation of the keypoint descriptor (adapted from [Lowe 2004]).

descriptor array computed from 16×16 samples is used, making each feature descriptor a 128-dimensional vector.

2.2.2 Feature Descriptor in Speed-Up Robust Feature (SURF)

The descriptor in SURF can also be treated as a gradient histogram of the pixels around the interest point. It is composed of the Haar wavelet responses in two perpendicular directions.

The first step to calculate the descriptor is to define a square region around the interest point and align it with the orientation determined from the detection step, as shown in Fig. 2.10. Then the square region is divided into 4×4 sub-regions. For each sub-region, there is a 4-entry descriptor v representing its underlying intensity patterns $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. dx and dy are Haar wavelet responses along the horizontal and vertical directions at one of the sample points, respectively. In each sub-region, there are 5×5 regularly spaced sample points and the summation is for all the 25 sample points in a sub-region. Concatenating v from each sub-region for all 16 of them would result in a descriptor vector with 64 entries.

Figure 2.11 shows the two Haar wavelet templates and Fig. 2.12 shows the Haar wavelet responses to four different image patterns.

2.2.3 Moment Invariants

The Moment Invariants is a statistical texture descriptor that contains only 7 elements, and can be used as a compact feature descriptor. The 7 moments are invariant to translation,

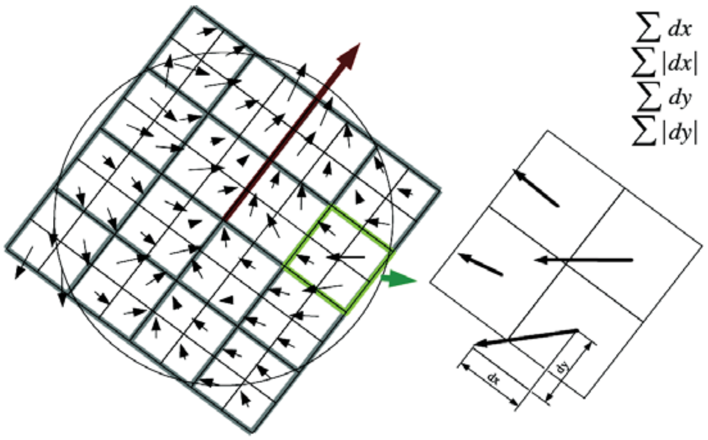


Figure 2.10: To build the descriptor, an oriented quadratic grid with 4×4 sub-regions is laid over the interest point (left). The wavelet responses are calculated for each sub-region. The 2×2 sub-divisions of each sub-region correspond to the actual fields of the descriptor. These are the sums of dx, $|dx|$, dy, $|dy|$, computed relatively to the orientation of the grid (right) (adapted from [Bay et al. 2008]).

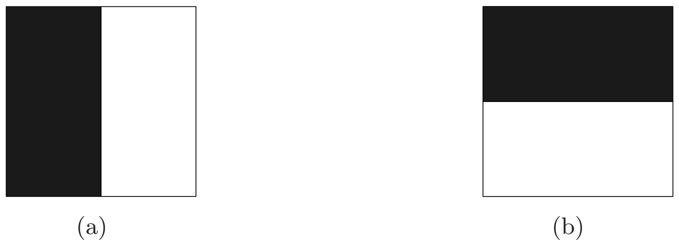


Figure 2.11: Haar wavelet templates to compute the responses along horizontal (a) and vertical (b) directions.

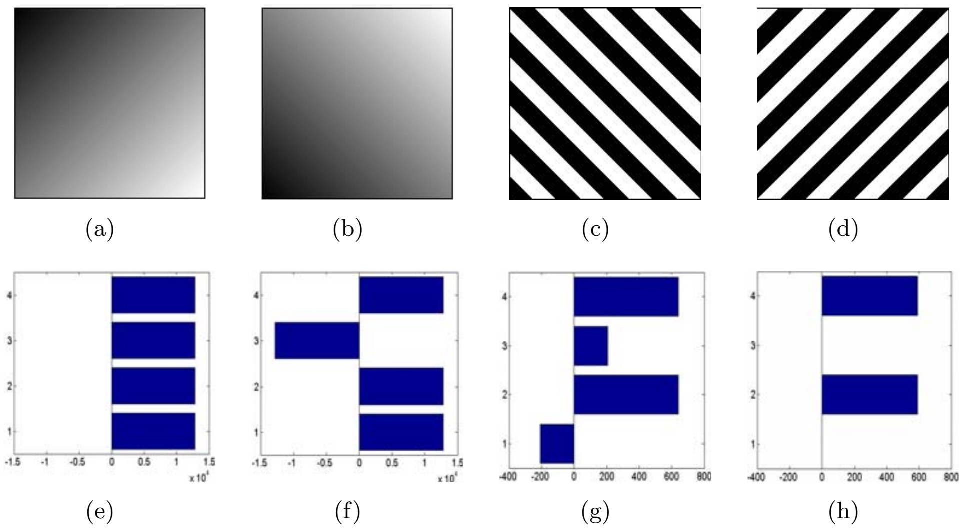


Figure 2.12: Haar wavelet responses to various patterns. Top row: the four different image patterns; Second row: corresponding responses using the horizontal and vertical Haar wavelets. In each subfigure, the four horizontal bars represent $\sum dx$, $\sum |dx|$, $\sum dy$, and $\sum |dy|$ from bottom to top.

rotation and scale changes.

The 2-D moment of order $(p+q)$ for an image patch of size $M \times N$ is defined as [Hu 1962]

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q I(x,y) \tag{2.24}$$

where $p = 0, 1, 2, ...$ and $q = 0, 1, 2, ...$ are integers, M and N are the height and the width of the region to calculate the moments. The corresponding central moment of order $(p+q)$ is defined as

$$\mu_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x-\bar{x})^p (y-\bar{y})^q I(x,y) \tag{2.25}$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}} \qquad \text{and} \qquad \bar{y} = \frac{m_{01}}{m_{00}} \tag{2.26}$$

The normalized central moments, denoted η_{pq} are defined as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \tag{2.27}$$

where

$$\gamma = \frac{p+q}{2} + 1 \tag{2.28}$$

for $p + q = 2, 3, ...$

A set of seven moment invariants can be derived from the second and third moments

$$\begin{aligned}
\phi_1 &= \eta_{20} + \eta_{02} && (2.29)\\
\phi_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 && (2.30)\\
\phi_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 && (2.31)\\
\phi_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 && (2.32)\\
\phi_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\
&\quad (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] && (2.33)\\
\phi_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + \\
&\quad 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) && (2.34)\\
\phi_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\
&\quad (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] && (2.35)
\end{aligned}$$

This set of moment invariants can be calculated from a local region of an image and used as a compact local feature descriptor. Usually the local region is defined from a feature point, for example, a corner point or a blob-like feature point.

2.2.4 Comparison of Feature Detectors and Descriptors

Tables 2.1 and 2.2 summarized properties from detectors and descriptors reviewed above.

The second moment matrix (Harris matrix) is a generalization of Sums of Squares of Differences (SSD). Although both SIFT and SURF use Hessian matrix in feature point detection, they use it in different ways. For SIFT, since the DoG filter responses readily to feature points on edges, Hessian matrix is used to filter them out. For SURF, the local maximum of the determinant of the Hessian matrix is treated as a feature point and the actual calculation of the Hessian matrix is through the SAT image and the templates for the second order Gaussian derivatives.

Method	Moravec	Harris	Multi-scale Harris	FAST	SIFT Detector	SURF Detector
Detector	SSD	Harris matrix	Harris matrix+ Gaussian scale space	Accelerated Segment Test	DoG + Hessian	Hessian
Detection result	Location	Location	Location+scale	Location	Location +scale +orientation	Location +scale +orientation
Computation Cost	Low	Low	Medium	Low	High	Medium
Rotational Invariant	No	Yes	Yes	Yes	Yes	Yes
Perspective Invariant	Yes	Yes	Yes	Yes	Yes	Yes
Robust to Noise	No	Yes	Yes	Yes	Yes	Yes
Illumination Invariant	Yes	Yes	Yes	No	Yes	Yes

Table 2.1: Comparison among several feature detectors.

Method	SIFT Descriptor	SURF Descriptor	Moment Invariants
Descriptor	Gradient histogram	Gradient histogram	Moment Invariants
Descriptor Length	128	64	7
Computation Cost	High	Medium	Low
Rotational Invariant	Yes	Yes	Yes
Perspective Invariant	Yes	Yes	Yes
Robust to Noise	Medium	High	High

Table 2.2: Comparison among several feature descriptors.

2.3 Feature Comparison

The similarity between images is measured by the number of matching feature pairs they have. The definition of a matching feature pair depends on the comparison method. Mikolajczyk et al. proposed three comparison methods [Mikolajczyk and Schmid 2005]. The first is a threshold based matching: Two feature points are claimed to be matching pairs if the distance between their descriptor vectors is below a threshold. Usually the Euclidean distance is used. In this case, a feature point can have several matches. The second is a nearest neighbor-based matching: two feature points A and B are claimed to be matching pairs if the descriptor vector D_B is the nearest neighbor to D_A and if the distance between them is below a threshold. In this case, a feature point has at most one match (unless a tie happens during the nearest neighbor search). The third is similar to the nearest neighbor-based approach except that the threshold is applied to the distance ratio between the first and the second nearest neighbors. Therefore, two feature points A and B are claimed to be a match if $||D_A - D_B||/||D_A - D_C|| < t$, where D_B is the first nearest neighbor and D_C is the second nearest neighbor to D_A.

According to Mikolajczyk, the precision (defined as the ratio between the number of correct matches and the summation of the number of correct matches and the number of false matches) is higher for the nearest neighbor-based matching than for the threshold-based approach [Mikolajczyk and Schmid 2005]. This is because the nearest neighbor is mostly correct, although the distance between similar descriptors varies greatly due to image transformations.

Image feature points can be found consistently under scale, perspective and rotation variations using various feature detectors/descriptors. But sometimes using them directly for image comparison is still unreliable. Because it is quite possible that different scenes have silimar feature descriptors and thus would result in inaccurate image comparison results.

To remove these false alarms, the perspective geometry [Coxeter 1969] can be used as an additional constraint, if the scenes are planar. According to the theory from the perspective geometry, if two images are really from the same planar scene but from different view points, the two image planes can be related by a homography transform; if not, then this homography transform relationship does not exist. Moreover, if the two image planes are related by a homography transform, the corresponding matching feature points are also related by the same homography transform.

Chapter 3

Light-weight Feature Detectors

Corner points are low-level image features that represent meaningful image regions and can be uniquely localized, thus becoming ideal candidates for feature-based image representation. Corner point detection is an essential pre-processing step for high-level computer vision tasks involving feature-based image understanding. However, existing corner detection algorithms are either computationally intensive or inaccurate in localizing corner points.

In this chapter, two computationally light-weight corner detectors will be introduced. The first is based on the Discrete Wavelet Transform (DWT) and robust to scale change and image noise. The second is based on a linear mixing model and has good corner point localization capability as well as computational efficiency.

The content in this chapter has been appeared in [Bai and Qi 2011a,b].

3.1 DWT-based Corner Detector

This section proposes a new corner detection method that is robust against scaling and noise, as well as being light-weighted and requiring little data storage. The detector calculates the corner strength measure from the DWT coefficients at all scales. The scale of a corner point is defined as a function of the corner distribution at its vicinity. To make the scale calculation light-weighted, an approximate solution using a Gaussian kernel convolution is provided. By introducing the so-called "Polarized Gaussian" kernels, the proposed corner detection method is robust to “false” corners on inclined discrete edges.

3.1.1 Localize the Corner Points using Gaussian Interpolation from Decimated DWT Decomposition

In images, a set of connected pixels with different pixel intensities on their two sides is defined as an “edge” and the intersection of edges as a “corner”. By using the 2-D DWT decomposition, the edges can be found by examining the local extremes of wavelet coefficients along the horizontal, vertical or diagonal directions. Then a corner point can be found by looking at the coordinates where two or three wavelet coefficients along different directions both/all reach a local extreme.

The DWT decomposition of an image of size $m \times n$ using Haar wavelet bases results in a decimated dyadic decomposition up to scale J. This means for each pixel in the original image, at each decomposition scale s, we have three wavelet coefficients denoted as $W1_s(i,j)$, $W2_s(i,j)$ and $W3_s(i,j)$, respectively. The same formulation as in [Fauqueur et al. 2006] has been followed to define the corner strength map C_s at scale s

$$C_s(i,j) = (\prod_{t=1}^{3} |Wt_s(i,j)|)^{\frac{1}{3}} \tag{3.1}$$

where (i,j) stands for the image coordinate, and $i = 1, 2, \cdots, m/2^s$, $j = 1, 2, \cdots, n/2^s$.

Due to the decimated decomposition, the sizes of C_ss differ by 2^s. Detecting local maxima on C_s separately at each scale would result in poor corner point localization in the original image. As suggested in [Fauqueur et al. 2006], C_s is interpolated at each scale up to the original image size $m \times n$ using a Gaussian kernel (i.e., upsample by the factor of 2^s for C_s) with the standard deviation proportional to the scale factor 2^s. Denote the interpolated corner strength map C_s as IC_s. The corner strength measure at the original resolution is then

$$C = \sum_{s=1}^{J} IC_s. \tag{3.2}$$

For example, for a test image shown in Fig. 3.1 (a), the corner strength measure C is illustrated in Fig. 3.1 (b), and Fig. 3.1 (c) presents the corner point localization by finding the local maxima within a 3×3 window on the corner strength measure. The center of a

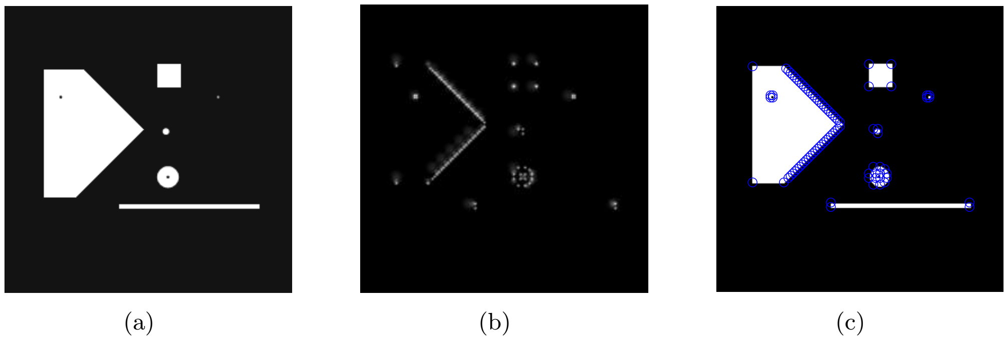


Figure 3.1: (a) The test image; (b) The corner strength measure; (c) The local maxima detected from (b).

circle in the figure indicates the location of a detected corner point. The 13 corners can all be correctly picked up, although many "false alarms" along the inclined edges are also detected. This is because the discretization of a straight line along directions other than the horizontal/vertical will create a "zigzag" edge, and this pattern, if being looked at in a finer scale, shows the same property as a corner point. The next section will show that these "false alarms" could be separated from other "real" corners by assigning them small scales.

3.1.2 Estimate Corner Point Scales from Gaussian Convolution on the Corner Map

Determine an appropriate scale parameter for a corner point is essential in our method to get rid of small-scale corners on the discrete edges. This section proposes a new scale selection method for each corner point based on an approximation to the corner distribution within its neighborhood.

Our scale selection algorithm is designed based on the observation that the corners on an inclined discrete edge are usually clustered together and these corners become more apparent at finer scales. Therefore, it is appropriate to assign small scale parameters to the corners where corners are densely distributed. In other words, the corner scale is related to the corner distribution at its vicinity. The two factors involved in describing the corner distribution for a reference corner are: How many corners are within the neighborhood and how far away these neighbor corners are from the reference corner. Then the corner scale map $S(i,j)$ is defined as

$$S(i,j) = k \times \frac{\sum_{c \in N_c(i,j)} d_c(i,j)}{|N_c(i,j)|} \tag{3.3}$$

where $d_c(i,j)$ is the Euclidean distance between the reference corner at (i,j) and its neighbor corner c, $N_c(i,j)$ is the set of all neighbor corners of the reference corner, $|N_c(i,j)|$ is the number of corners in $N_c(i,j)$, and k is a scalar that needs to be determined experimentally.

Although the two unknown factors $d_c(i,j)$ and $N_c(i,j)$ in Eq. 3.3 can be calculated explicitly, finding these accurate values would be time consuming. For example, the image in Fig. 3.1 (a) is of size 256×256, there are 85 corner points detected as shown in Fig. 3.1 (c). For each corner point, finding the distances to other corners requires 84 times of distance calculation and for the whole image, a total of $3,570$ times of distance calculations are required.

To speed up the scale parameter calculation process, a new method is proposed to approximate Eq. 3.3. C_m is named the "Corner Map", which is an $m \times n$ matrix with binary entries, with 1 indicating a corner point at that coordinate and 0 indicating no corner. The Corner Map C_m is the resulting matrix of taking the local maximum from the corner strength measure C and then binarizing. C_m is convolved with a Gaussian kernel, and only record the convolution results at the coordinates that correspond to corners, and use the reciprocals of the results as scale parameters. For a single corner point on C_m, the convolution result equals to a weighted sum of 1s (if there is a neighbor corner) and 0s (if there is no neighbor corner) at its vicinity and the weight is decided by the Gaussian kernel. For a specific corner, the closer other corners to it, the larger the summation is; the more neighbor corners it has, the larger the summation is.

However, this approximation of Eq. 3.3 cannot distinguish a "real" corner at the junction of two inclined discrete edges (the rightmost corner on the pentagon in Fig. 3.1) and a "false" corner on an inclined discrete edge. This is because the two kinds of corners would have the same response for the Gaussian kernel convolution. Therefore, for the corner points that have small scale assignments from Eq. 3.3, which are mostly the corners on the inclined discrete edges (both "real" and "false" corners), they are further convolved with the so-called "Polarized Gaussian" kernels. By examining responses to these kernels, the "real" corners can be separated.

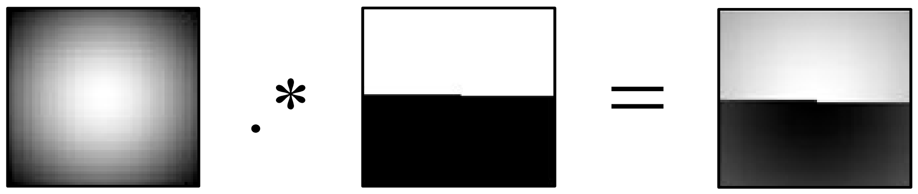


Figure 3.2: Building a polarized Gaussian kernel by multiplying a Gaussian kernel with a binary mask, where ".∗" represents the element-wise multiplication.

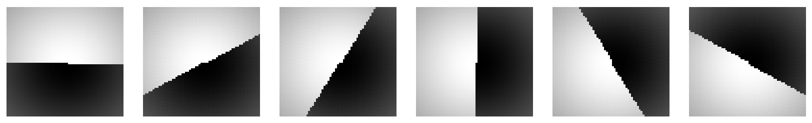


Figure 3.3: The 6 polarized Gaussian kernels.

A polarized Gaussian kernel is created by element-wise multiplying a Gaussian kernel with a binary mask. The elements in the binary mask are either "1"s (the elements in the upper half and the right side of the horizontal central line of the binary mask) or "−1"s (the rest elements). Figure 3.2 shows the calculation for a polarized Gaussian kernel. The element summation inside a polarized Gaussian kernel along any direction passing the center is zero, thus making this kernel have small responses to "false" corners, and large responses to "real" corners when convolve with the corner map C_m. A "real" corner may have small response for a polarized Gaussian kernel if its two rays fall into the two polarizations separately, but by providing another five rotated polarized Gaussian kernels, this possibility has been significantly minimized. Figure 3.3 shows six polarized Gaussian kernels by rotating the first one 30°, 60°, 90°, 120°, 150°, respectively.

The proposed scale selection algorithm is summarized as follows

- Step 1: Convolve the binary corner map C_m with a Gaussian kernel, and the result of the reciprocal is used as an approximation to compute Eq. 3.3 so that every corner has a scale parameter assigned.

- Step 2: For corners whose scales are smaller than a threshold, convolve them with the six polarized Gaussian kernels, respectively. Remove corners that have small responses for all 6 polarized Gaussian kernels.

The experimental results of this DWT-based corner detector are shown in Chapter 5.

3.2 An Efficient Corner Detector Based on Linear Unmixing

The most intuitive definition of corners has been that "A corner is the place where two lines meet at an angle [Wikipedia 2010]." However, none of the existing corner detectors could take advantage of this simple intuition directly. This section interprets corner formation from this perspective and treat corners as line junctions. Then a corner patch (a small region in the image, e.g., 3×3, with the corner point located at the center of the region) is formulated as a linear mixture (or weighted sum) of "line bases". To recognize corners based on this formulation, two issues have to be resolved. The first is to find the set of line bases. The second is to calculate the set of mixing weights or coefficients. The Nonnegative Matrix Factorization (NMF) method is used with the sparseness and uncorrelatedness constraints to solve for the optimal set of line bases. Then for each patch within the image, a set of coefficients using the optimal line bases can be derived, from which patterns are recognized that correlate to the existence of a corner. This method is referred as Linear Unmixing-based Corner detector (LUC).

3.2.1 Corner Detection using Linear Mixing Model

The definition of the corner on Wikipedia [Wikipedia 2010] states that "A corner is the place where two lines meet at an angle." Dias et al. [Dias et al. 1995] designed a neural network-based corner detector by identifying line junctions. They generated a series of 8×8 binary templates containing various line junctions forming angles of multiples of $45°$. These templates were used to train a neural network to recognize corners formed with these patterns. Then an edge image with contours of 1-pixel width was generated and every 8×8 region in the contour image was examined through the trained neural network to make a decision on whether there is a corner point or not. However, their corner detector failed to provide a explicit corner definition in the mathematical form as the junction point of lines. And their detector suffered from two major problems: First or all, the algorithm relies on the edge image and introducing additional uncertainty and computational burden to the corner detection process. In addition, since an 8×8 window was adopted to best capture the line junctions, the algorithm can only localize corner points within a 4×4 region, making accurate corner detection difficult.

In the following, the first attempt is presented that formulates the corner using a linear mixture model where a 3×3 patch with a corner at the center is represented as a linear combination (or weighted sum) of basic "line" patterns of the same size. For example, a corner patch in Fig. 3.4 (a) can be treated as the overlapping of two line patches in Fig. 3.4 (b) and (c). These line patterns are used as bases for representing a corner, leading to the design of an efficient corner detector that uses the information on how the bases contribute to the corner formation (or mixing coefficients) to detect corners. This detector is referred as the Linear Unmixing-based Corner detector (LUC). Note that LUC works on the original grayscale image *directly* and can identify corner points accurately with high computational efficiency.

Two linear unmixing problems will be studied: The first is to identify the bases that can constitute a corner; and the second is to solve for the mixing coefficients. The first unmixing problem is a matrix factorization problem where the observation matrix is known and the two matrices, the basis and mixing coefficient matrices, whose product approximates the observation matrix, are to be found. The second unmixing problem is a equation-solving problem where both the observation and the basis matrices are known, and the coefficient matrix is to be found.

Let $x \in R^{9\times1}$ denote the lexicographical representation of a 3×3 corner patch. Assume there exists a set of bases and corresponding coefficients, such that x can be represented by

$$x = Es \tag{3.4}$$

where each column in $E \in R^{9\times8}$ denote the lexicographical representation of a line patch (basis) and $s \in R^{8\times1}$ denotes the coefficient vector. $x \succeq 0$ because it comes from an image, whose grayscale values are nonnegative. The symbol $\succeq$ denotes component-wise inequality, e.g., $x_i \geq 0$ for $i = 0, \ldots, 9$. The basis matrix E and the mixing coefficients in s are constrained to be nonnegative.

Eq. 3.4 explains that a corner patch of a 3×3 block can be treated as a linear mixture of a set of bases. The contribution of each basis in the mixture is represented by the mixing coefficient vector s. Therefore line junction based corner detectors can be implemented by examining, in each patch, how many bases are contributing to the linear mixture and

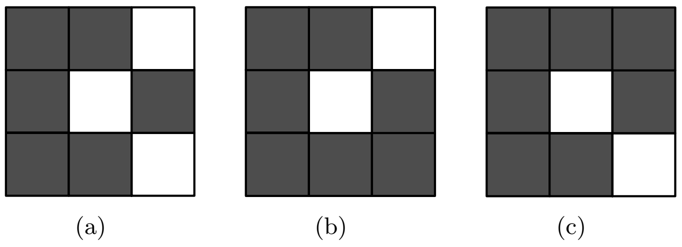


Figure 3.4: A corner pattern in a binary image (a) represented by the sum of two line patterns in (b) and (c). A white pixel denotes 1 and a dark pixel denotes 0.

how much they contribute. For example, a patch from a smooth area would result in all coefficients in s being similar and a block containing a corner point would only have few elements in s with large values and others with smaller values. The merit of formulating a corner patch using a linear mixture model is that once the basis matrix E has been found, the mixing coefficients s can be solved simply from a least squares solution $s = (E^T E)^{-1} E^T x$, and the corner detection can be achieved through simply checking the magnitude distribution of s.

To find matrix E, we need a training set X containing various corner patterns and the linear mixture model in Eq. 3.4 can be represented in the matrix form as

$$X = ES \tag{3.5}$$

where $X \in R^{9 \times l}$ and $S \in R^{8 \times l}$ denote the training set with each column representing a corner patch and the mixing coefficients matrix, respectively. The number l stands for the number of corner patches used in the training set. Now the problem of finding matrix E becomes a nonnegative matrix factorization (NMF) problem that factorizes matrix X into the production of two nonnegative matrices E and S such that $X \approx ES$.

3.2.2 Constrained NMF for Basis Calculation

A standard NMF problem is to minimize the objective function in Eq. 3.6 with nonnegative constraints. Since the factorization to the standard NMF problem is not unique, additional application-specific constraints are needed in order to find the optimal solution.

$$f(E,S) = \frac{1}{2}\|X - ES\|^2 = \frac{1}{2}\sum_{i,j}(X_{ij} - (ES)_{ij})^2 \tag{3.6}$$

The sparseness constraint. The column vectors (or bases) in E represent line patterns as illustrated in, e.g., Fig. 3.4 (b) and (c), where only two out of nine pixels have intensities much higher than the rest, which manifests that "sparseness" should be added as one of the constraints in the standard NMF process.

There are two possible ways to incorporate the sparseness constraint: The first is to implement, in each iteration step, a nonlinear projection that increases the sparseness of the estimated components [Hoyer 2004]. The other is to add to the cost function a penalty term [Hoyer 2002]. The problem with the second method is that sparseness can only be adjusted implicitly by the regularization coefficients, while the first method provides a way to explicitly adjust the sparseness. The sparseness measure in the first approach is [Hoyer 2004]

$$\text{sparseness}(v) = \frac{\sqrt{m} - ||v||_1/||v||_2}{\sqrt{m} - 1} \tag{3.7}$$

where m is the length of vector v, $||v||_1 = \sum_{i=1}^{m}|v_i|$ is the L_1 norm, and $||v||_2 = \sqrt{\sum_{i=1}^{m}|v_i|^2}$ is the L_2 norm. The sparseness measure varies from unity to zero when v contains only a single non-zero element and when all elements have equal non-zero values.

The uncorrelatedness constraint. The second property the set of bases in E should have is the uncorrelatedness, which is a common condition that all bases should satisfy. The least correlation constraint is defined as [Zhang and Fang 2007]

$$J(E) = \sum_{i=1}^{n} log((E^TE)_{ii}) - log|E^TE| \tag{3.8}$$

where $(E^TE)_{ii}$ is the ith diagonal element of the square matrix E^TE and $|E^TE|$ is the determinant of matrix E^TE. Let $B = E^TE \in R^{n\times n}$ which is a positive definite matrix and the following inequality holds

$$\sum_{i=1}^{n} log(B_{ii}) - log|B| \geq 0. \tag{3.9}$$

$J(E)$ is a nonnegative function that takes on the minimum if and only if $< e_i, e_j >= 0$, $\forall i, j$, $i \neq j$, where $< \cdot >$ represents the dot product and e_i, e_j are column vectors of E.

Learning algorithm. Combining the objective function in Eq. 3.6 and the sparseness and uncorrelatedness constraints, the following optimization problem is formulated:

$$\begin{aligned} \min_{E,S} \quad & f(E,S) = \frac{1}{2}\left\|X - ES\right\|^2 \\ s.t. \quad & \text{sparseness}(e_j) = c \quad \forall j \\ & J(E) = 0 \\ & E, S \succeq 0 \end{aligned} \tag{3.10}$$

where e_j is the jth column of matrix E, $j = 0, \ldots, 7$. By introducing the regularization coefficient λ, the uncorrelatedness constraint can be combined with the cost function and the optimization problem becomes:

$$\begin{aligned} \min_{E,S} \quad & \mathcal{L}(E,S) = \frac{1}{2}\left\|X - ES\right\|^2 + \lambda J(E) \\ s.t. \quad & \text{sparseness}(e_j) = c \quad \forall j \\ & E, S \succeq 0 \end{aligned} \tag{3.11}$$

where λ is usually chosen to be between 0.01 and 0.5 [Cichocki et al. 2006].

Taking the partial derivatives of $\mathcal{L}(E,S)$

$$\begin{aligned} \frac{\partial \mathcal{L}(E,S)}{\partial E_{ij}} &= [(ES - X)S^T]_{ij} + \lambda \frac{\partial J(E)}{\partial E_{ij}} \\ \frac{\partial \mathcal{L}(E,S)}{\partial S_{jk}} &= [E^T(ES - X)]_{jk} \end{aligned} \tag{3.12}$$

The gradient descent method is applied to update E and S

$$\begin{aligned} E_{ij} &\leftarrow E_{ij} + \mu_{ij}\{[(X - ES)S^T]_{ij} - \lambda((EE^T)^{-T}E)_{ij}\} \\ S_{jk} &\leftarrow S_{jk} + \eta_{jk}\{[E^T(X - ES)]_{jk}\} \end{aligned} \tag{3.13}$$

where μ_{ij} and η_{jk} are small positive learning stepsizes. The nonnegativity of both E and S are achieved through this additive updating rule. Lee and Seung developed a multiplicative updating rule which guarantees the nonnegativity of both E and S by setting [Lee and

Seung 1999]

$$\mu_{ij} = \frac{E_{ij}}{(ESS^T)_{ij}}, \quad \eta_{jk} = \frac{S_{jk}}{(E^TES)_{jk}} \tag{3.14}$$

Substituting Eq. 3.14 into Eq. 3.13

$$\begin{aligned} E_{ij} &\leftarrow E_{ij}\frac{(XS^T)_{ij} - \lambda[(EE^T)^{-T}E]_{jk}}{(ESS^T)_{ij}} \quad \forall i,j \\ S_{jk} &\leftarrow S_{jk}\frac{(E^TX)_{jk}}{(E^TES)_{jk}} \quad \forall j,k \end{aligned} \tag{3.15}$$

To incorporate the sparseness constraint, a projected gradient descent algorithm developed by Hoyer [Hoyer 2004] is adopted, which takes a step toward the negative gradient direction and then projects onto the constraint space of sparseness, making sure that the taken step is small enough such that the cost function $\mathcal{L}(E,S)$ is reduced at every step, therefore achieving the desired degree of sparseness.

A geometric explanation of the projection operator is provided as follows: The range of the sparseness of a normalized vector v always falls into $[0,1)$, which guarantees that, according to Eq. 3.7, the hyperplane $\sum_i v_i = l_1$ (assume $v_i \geq 0$) and the unit hypersphere $\sum_i v_i^2 = 1$ always have intersections where all elements of v_i are nonnegative. At the beginning, a vector with unit L_2-norm is on the unit hypersphere $\sum_i v_i^2 = 1$, then it is projected onto the hyperplane $\sum_i v_i = l_1$. Next, on the hyperplane, the vector will be projected to the closest point on the joint constraint hypersphere (intersection of the L_1 and L_2 norm constraints). This is done by moving radically outward from the center of the hyperplane $\sum_i v_i = l_1$. Since at the intersections all v_is are nonnegative, this projection operator will not violate the nonnegative constraint.

The projection operator is summarized in Algorithm 1 [Hoyer 2004].

3.2.3 Training Set and the Extracted Bases

In order to extract proper bases representing lines, a dataset containing all corner patches in a 3×3 block is created, forming matrix X. Among all possible patterns, i.e., $2^9 = 512$, within a 3×3 block of binary pixels, those containing 2 or 3 lines are chosen as corners, excluding the cases where two lines are co-linear. The reason for this is that the number of corners containing 2 or 3 lines significantly outnumbers those containing 1 or more than

Algorithm 1 Projector that finds any vector v the closest nonnegative vector w with given L_1 and L_2 norm (sparseness)

```
1. Set $w_i := v_i + (L_1 - \sum v_i)/\dim(v), \quad \forall i$
2. Set Z:={}
3. Iterate
   (a) Set $m_i := \begin{cases} L_1/(dim(v) - size(Z)) & \text{if} \quad i \notin Z \\ 0 & \text{if} \quad i \in Z \end{cases}$
   (b) Set $w := m + \alpha(w - m)$, where $\alpha \geq 0$ is selected such that the resulting $w$
       satisfies the $L_2$ norm constraint. This requires solving a quadratic equation.
   (c) If all components of $w$ are nonnegative, return $w$, end
   (d) Set $Z := Z \cup \{i : w_i < 0\}$
   (e) Set $w_i := 0, \quad \forall i \in Z$
   (f) Calculate $c := (\sum w_i - L_1)/(\dim(v) - \text{size}(Z))$
   (g) Set $w_i := w_i - c, \quad \forall i \notin Z$
   (h) Go to (a)
```

3 lines, as shown in Fig. 3.5, which illustrates the number of different corner patterns as a function of how many lines are involved in the corner formation. Based on this definition, 160 patterns are selected as corner patches, among which 80 are made of lines represented by 1s (or foreground corners), and the other 80 are made of lines represented by 0s (or background corners). The foreground/background corner patterns are complementary to each other. Fig. 3.6 shows 4 foreground corner patterns used in the training set.

The 80 foreground corner patterns are used as the training set, constructing matrix X in the optimization problem Eq. 3.11 with each column of X representing a corner pattern. The additive rule in Eq. 3.13 is used to update E, to control the stepsize μ when projecting each column of E onto the sparseness constrained space, making the cost function $\mathcal{L}(E, S)$ non-increasing. The multiplicative rule in Eq. 3.15 is used to update S in favor of its fast convergence [Lee and Seung 2001]. Fig. 3.7 shows the 8 bases extracted from the constrained NMF problem, clearly indicating a set of 8 lines can be used to construct the corner patterns. Table 3.1 lists the numerical values of each basis in E, showing that the 8 bases all have values close to 0.36 at the center, and a large value around 0.93 in different positions surrounding the center.

It is easy to verify that matrix $E \in R^{9\times8}$ has rank 8 and therefore a stable least squares solution always exists.

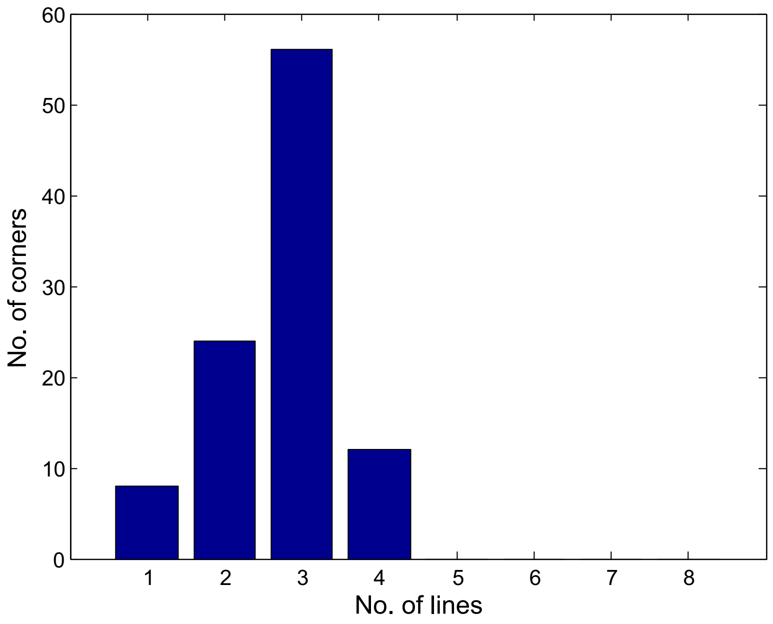


Figure 3.5: The number of corners in the binary 3×3 template as a function of the number of line bases involved.

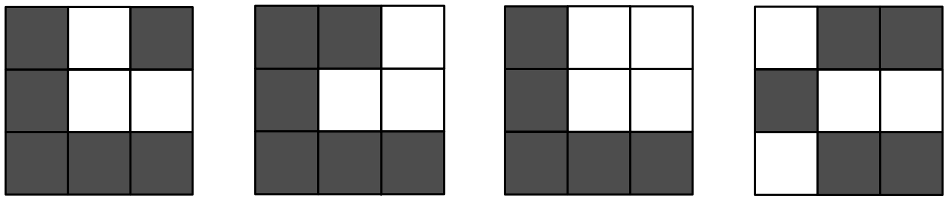


Figure 3.6: 4 corner patterns used in the training set containing 2 and 3 line bases, respectively.

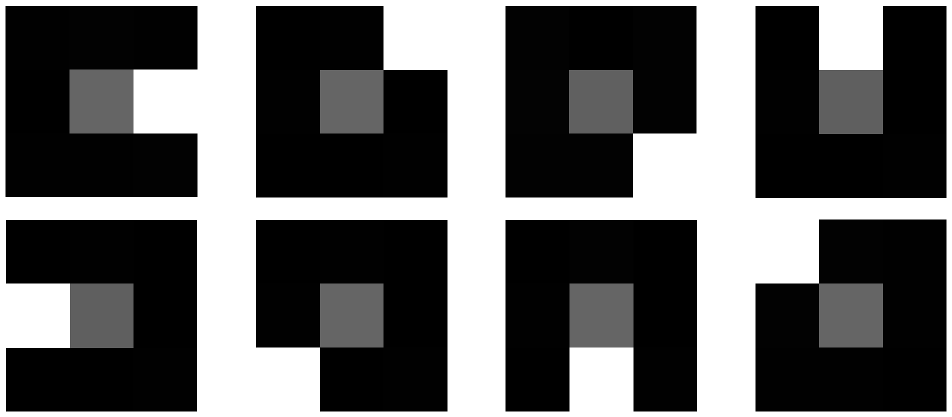


Figure 3.7: The 8 bases from the constrained NMF represented in blocks. Brighter pixels indicate larger magnitude.

0.017	0.017	0.016	0.017	0.018	0.011	0.017	0.933
0.016	0.019	0.017	0.011	0.016	0.934	0.017	0.016
0.018	0.933	0.016	0.017	0.017	0.011	0.018	0.016
0.009	0.018	0.017	0.933	0.017	0.018	0.016	0.016
0.361	0.357	0.358	0.357	0.361	0.355	0.358	0.357
0.017	0.017	0.933	0.018	0.009	0.018	0.016	0.018
0.932	0.015	0.013	0.015	0.014	0.017	0.013	0.013
0.018	0.009	0.016	0.017	0.017	0.019	0.933	0.016
0.014	0.015	0.014	0.015	0.932	0.017	0.013	0.013

Table 3.1: The numerical values of the 8 bases extracted from the constrained NMF.

3.2.4 Robust Detection of Corners

Analyzing mixing coefficient patterns for corner detection. Given E (or line bases) obtained from the constrained NMF, for any 3×3 block in the image whose lexicographical representation is x, the mixing coefficients in Eq. 3.4 can be solved using the least squares method, resulting in $s = (E^T E)^{-1} E^T x$. The number of large coefficients in s provides a good indicator on how many line bases are involved in forming the mixture, x. The following definition of corners based on this information is provided.

Definition When the number of large (or small) values in s falls into the range of $[2, 3]$, the corresponding block contains a foreground (or background) corner, respectively.

Algorithm 2 details this process where large (or small) values of coefficients are judged by a parameter, θ.

Algorithm 2 Corner detection based on mixing coefficient patterns.

```
1:  Sort $s \in R^{8\times 1}$ in ascending order, take the 1st order difference and record the result as
    $t \in R^{7\times 1}$
2:  Find the largest two values in $t$, denoted as $d_1$ and $d_2$ (indices are $id_1$ and $id_2$, $id_1 < id_2$)
3:  if $\min(d_1, d_2) \geq \theta$ then
4:      if $2 \leq id_1 \leq 3$ or $6 \leq id_2 \leq 7$ or $2 \leq id_2 - id_1 \leq 3$ then
5:          The current pixel is a corner point
6:      end if
7:  else if $\min(d_1, d_2) \leq \theta$ and $\max(d_1, d_2) \geq \theta$ then
8:      Let $d = max(d_1, d_2)$ and its index be $id$
9:      if $2 \leq id \leq 3$ or $6 \leq id \leq 7$ then
10:         The current pixel is a corner point
11:     end if
12: end if
```

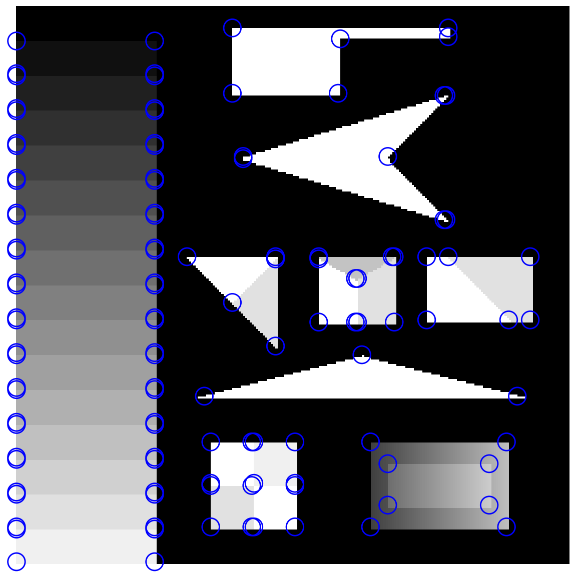


Figure 3.8: Corners detected on a synthetic grayscale image of the size 256×256.

The corner detection result on a synthetic grayscale image of the size 256×256 is shown in Fig. 3.8. There are 119 pixels that satisfy our corner definition and the corner detector has successfully picked all of them. The parameter θ is set as 0.05.

Remove unstable corner detections. The corner detection from the previous step may contain false alarms along discrete edges and therefore an additional step is required to filter them out. The corner strength measure calculated from the second moment matrix for each corner detected from the first step is used and those with lower corner strength measure are filtered out. The second moment matrix and the corner strength measure are the same as those in [Harris and Stephens 1988], however the calculation for the second moment matrix is different.

When calculating the image derivatives as in Eq. 2.3, instead of performing the template operation using the Prewitt templates on the image directly, the basis matrix E can be used,

$$\begin{aligned} I_x &= p_x * I(u,v) \\ &= p_x * \begin{bmatrix} E_0 s & E_1 s & E_2 s \\ E_3 s & E_4 s & E_5 s \\ E_6 s & E_7 s & E_8 s \end{bmatrix} \\ &= [(E_6 - E_0) + (E_7 - E_1) + (E_8 - E_2)]s \end{aligned} \tag{3.16}$$

where $I(u,v)$ is the 3×3 block centered around (u,v), p_x is the Prewitt operator along the vertical direction, $*$ is the correlation operator, and E_i, $i = 0, \ldots, 8$ is the $(i+1)$th row of matrix E. Similarly,

$$I_y = [(E_2 - E_0) + (E_5 - E_3) + (E_8 - E_6)]s \tag{3.17}$$

Note that in both Eqs. 3.16 and 3.17 the terms inside the bracket are constant, therefore the image derivatives can be computed by two vector-vector multiplications, which is more efficient than applying the Prewitt operator. And the corner strength measure is calculated as in Eq. 2.4.

3.2.5 Computational Efficiency

Since efficiency is one of the main goals of the proposed corner detector, this section studies in further detail the computational efficiency of six corner detectors, including the proposed DWT-based detector, the LUC detector, SIFT, SURF, FAST, and Harris detector. The image is assumed to have N pixels (row $\times$ column), out of which 25% would be detected as corner (feature) point candidates by various detectors. Note that only a small portion of these candidates would be finally detected as corner (feature) points. And the window size is $n = 3$ whenever a Gaussian function is needed. The computation is estimated for each pixel first and then times the size of the image.

For Harris detector, in the pre-smoothing step, it needs n^2 multiplications and $n^2 - 1$ additions; in the image derivative calculation step, to compute I_x and I_y, it needs 5 additions each, totally 10 additions; in the step to calculate the second moment matrix, it needs $2 \times n^2$ multiplications and $2 \times n^2 - 2$ additions to calculate the averages of I_x and I_y in a Gaussian window; and 3 multiplications and 3 additions to calculate the corner strength measure; then the non-maximum suppression adds 8 comparisons. So in total $3 \times n^2 + 3$ multiplications and $3 \times n^2 + 10$ additions are needed; for the whole image, $N \times (3n^2 + 3$ multiplications, $3n^2 + 10$ additions and 8 comparisons) are needed.

For the detector in SIFT, pre-smoothing adds n^2 multiplications and $n^2 - 1$ additions; then subtracting consecutive image planes in scale space adds 1 addition; The non-maximum suppression needs 26 comparisons. Note that the following computations apply only to the keypoint candidates. The sub-pixel accuracy calculation requires 10 multiplications and 28 additions; the determinant of the Hessian matrix adds another 3 multiplications and 2 additions. Assume the Difference of Gaussian space is calculated at 4 octaves and each of them contains 4 scales and then there are 16 image planes. So in total $16 \times \{N \times [(n^2$ multiplications, n^2 additions and 26 comparisons$) + 0.25 \times ($13 multiplications and 30 additions$)]\}$ are needed.

For the detector in SURF, calculating the integral image needs 3 additions; calculating the Hessian matrix convolution through integral image needs 2 multiplications and 30 additions; calculating the Hessian matrix determinant requires 2 multiplications and 1 additions; then non-maximum suppression adds 26 comparisons. The next step computation applies only to the interest point candidates. The sub-pixel accuracy calculation adds 10 multiplications and 28 additions. Assume the scale space is calculated at 4 octaves and each of them contains 4 scales, then there are 16 image planes. In total $16 \times \{N \times [($2 multiplications, 33 additions and 26 comparisons$) + 0.25 \times ($10 multiplications and 28 additions$)]\}$ are needed.

For each pixel of an image, the FAST detector first compares 4 pixels on its Bresenham circle to two thresholds, which takes 8 comparisons. Since the pixel addresses on a Bresenham circle is not continuous in storage, the address calculation takes another 4 additions. The first step will reject most pixels and the remaining are corner point candidates. For each candidate, all 16 pixels on the Bresenham circle will be compared to two thresholds, which takes 32 comparisons and another 32 additions for address calculation. To determine whether there are 12 consecutive pixels have brighter or darker intensity levels than the central pixel, 41 additions are needed. At last, to calculate the corner response function [Rosten and Drummond 2006], 32 additions, 16 comparisons and 1 comparison are required. In total $N \times [(4$ additions and 8 comparisons $) + 0.25 \times (16$ multiplications, 105 additions and 49 comparisons)] are needed.

for the DWT-based corner detector, a 3-level DWT decomposition using Haar wavelet requires $\frac{7}{2} \times N$ multiplications and N additions; to build the corner strength map at each decomposition level needs $\frac{21}{32} \times N$ multiplications; to find the local maximum at each level requires $\frac{21}{8} \times N$ comparisons. In the Gaussian interpolation step, assume the Gaussian function window size is n_1, then it needs $\frac{21}{64} \times N \times n_1^2$ multiplications and $\frac{21}{64} \times N \times (n_1^2 - 1)$ additions; then to add the 3 interpolated corner strength maps needs $2 \times N$ additions. Assume the polarized Gaussian function has the window size n_2, then the convolution takes $6 \times N \times 0.25 \times n_2^2$ multiplications and $6 \times N \times 0.25 \times (n_2^2 - 1)$ additions. In total $N \times [((\frac{266}{64} + \frac{21}{64} \times n_1^2)$ multiplications, $(\frac{171}{64} + \frac{21}{64} \times n_1^2)$ additions, $\frac{21}{8}$ comparisons $) + 0.25 \times 6 \times (n_2^2$ multiplications, $n_2^2 - 1$ additions)] are needed. $n_1 = 11$ and $n_2 = 11$ are chosen and the same values are used for the plot in Fig. 3.9.

For the proposed linear unmixing based corner detector, solving the Least Square equation requires 8 multiplications and 64 additions; calculating A_1 and A_2 requires 8 comparisons. For each corner point candidate, calculating image derivatives I_x and I_y needs 9 multiplications and 8 additions; calculating the averages of I_x and I_y in a Gaussian window needs n^2 multiplications and $n^2 - 1$ additions; finally calculating the corner strength adds another 3 multiplications and 3 additions. In total $N \times [(8$ multiplications, 64 additions and 8 comparisons$) + 0.25 \times (n^2 + 12$ multiplications and $n^2 + 10$ additions)] are needed.

Fig. 3.9 shows the number of operations (multiplications, additions and comparisons) required to process images of varying sizes. Reference Trajkovic and Hedley [1998] used the same approach to evaluate the computational complexity of several corner detectors. All these six feature point detectors have a computational complexity that is a linear function of the image size. Among the six detectors, the detector in SIFT is most computationally intensive, followed by the detector in SURF, which is designed to be more efficient than the detector in SIFT. The DWT-based corner detector, although is a multi-scale detector, is more efficient than SIFT and SURF detectors because the scale space from a DWT decomposition is more compact than the Gaussian scale space. The LUC detector has a better

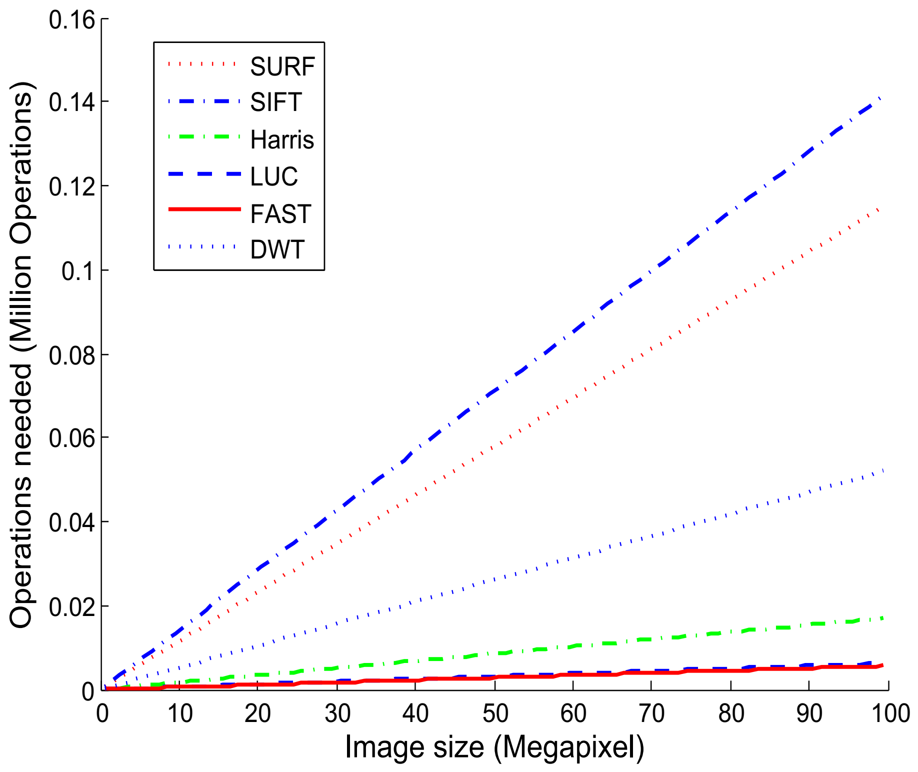


Figure 3.9: Number of operations (multiplications, additions and comparisons) as a function of the image size. Image sizes varies from low resolution of 600×800 to high resolution up to 100 Megapixels.

computational cost than Harris detector because it uses matrix multiplication instead of Gaussian convolution to compute the image derivatives. The FAST corner detector has the lowest computational complexity, but the margin between the LUC detector and the FAST detector is considerably small even for a very large image size.

3.3 Discussion

This chapter presented two computationally light-weight corner detectors.

The first is a multiresolution corner detector using Haar wavelet based DWT. It includes two steps: The first step is to localize the corner points and the second step is to determine the scale parameter for each corner detected. A corner strength map at each scale is built from the wavelet coefficients, differing in size by 2^s, and the Gaussian kernel interpolation is used for upsampling these decimated corner strength maps to the original image size, building the corner strength measure. The local maxima on the corner strength measure can then be detected as corners, and the scale of the corner point is defined as a function of the corner distribution in its vicinity. A fast calculation of the scale parameters is provided using a Gaussian kernel convolution.

The second is a linear unmixing-based corner detector, which is named as “LUC”. The LUC detector relies on a linear mixing process to describe a corner pattern in a 3×3 patch. A set of “line” bases is found for this linear mixing model through the constrained NMF. Then for each 3×3 block in the image the mixing coefficients are solved and corner detection is performed based on how many “line” bases are involved in the mixture. A corner strength measure function is also used to reject unstable corner detections.

Chapter 4

Dimensionality Reduction for SIFT Descriptors

For a typical image of the size 640×480 pixels, there could be more than 1000 SIFT feature points, and therefore, the SIFT feature descriptor matrix (each column is a SIFT descriptor, and assume there are 1000 feature points) for that image would have 128×1000 values. In terms of the data volume saving, the reduction from $307,200$ to $128,000$ is not considered effective enough. Therefore, performing dimensionality reduction on SIFT feature descriptors is necessary.

Performing dimensionality reduction on the feature descriptors has two obvious advantages for WVSNs: Firstly, on each individual sensor node, the storage space required for saving the feature descriptor of its captured image could be significantly reduced; Secondly, the transmitted data volume will be kept at a lower level, therefore achieving energy conservation.

A fact demonstrates the necessity of performing dimensionality reduction on the feature descriptor is that there is much redundancy existing in the feature descriptors. Fig. 4.1 (a) and (b) show the covariance matrices of the original 128-dimensional SIFT descriptor and the dimensionality reduced 40-dimensional SIFT descriptors, respectively, for an image from the Oxford dataset, as shown in Fig. 5.1 (a). The large magnitudes of some off-diagonal entries indicates that some dimensions are strongly correlated with other dimensions. The correlation is due to the fact that the descriptor components are computed

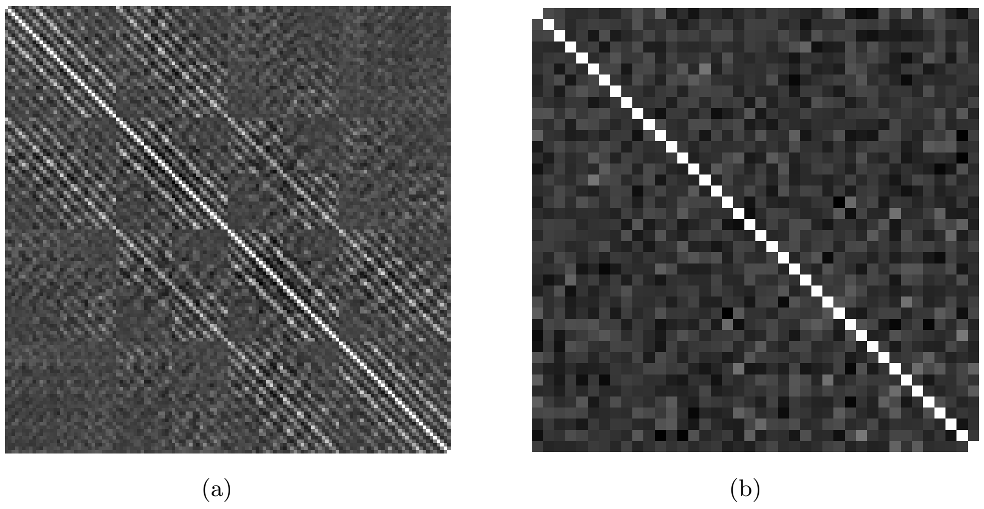


Figure 4.1: Covariance matrices for 128-dimensional SIFT (a) and the dimensionality reduced 40-dimensional SIFT descriptors (b). Mid-gray represents zero.

from gradients and their absolute values are weighted by a two-dimensional Gaussian kernel [Chandrasekhar et al. 2009].

4.1 Constrained NMF for Calculating Descriptor Bases

According to the observation that there is much redundancy for the SIFT descriptor set for an image, it is reasonable to assume that there exists a smaller set of 128-dimensional bases that can be used to represent the SIFT descriptor set of that image. In other words, a matrix $X \in R^{128\times 1000}$ of the SIFT descriptors for an image can be decomposed linearly into

$$X = ES \tag{4.1}$$

where $E \in R^{128\times n}$ is the basis matrix and $S \in R^{n\times 1000}$ is the mixing coefficient matrix, $n < 128$. Our dimensionality reduction is based on the assumption that there exists a set of bases in E constant for all images, and the differences between different SIFT descriptor matrices (corresponding to different images) can be treated as all reside in different S matrices. Therefore transmitting and comparing matrix S alone will be enough to measure the similarity between images, but the data volume of S is more compact than that of X.

A constrained NMF problem has been formulated for calculating the "line" bases in forming corner patterns in a 3×3 patch in Chapter 3.2. In fact, the same mathematical model can also be used for calculating the basis matrix E for SIFT descriptor matrix X, and then these bases can be used for calculating coefficient matrix S, therefore achieving dimensionality reduction.

For description convenience, Eq. 3.11 is written as

$$\begin{aligned} \min_{E,S} \quad & \mathcal{L}(E,S) = \frac{1}{2}\left\|X - ES\right\|^2 + \lambda J(E) \\ s.t. \quad & \text{sparseness}(e_j) = c \quad \forall j \\ & E, S \succeq 0 \end{aligned} \tag{4.2}$$

where $X \in R^{128\times l}$ is a matrix with each of its column representing a SIFT descriptor; l is the number of SIFT feature points; $E \in R^{128\times n}$ is the basis matrix with each of its column being a basis; $S \in R^{n\times l}$ is the mixing coefficient matrix and each column of S corresponds to a SIFT descriptor, but could be in a much lower dimension n where $n < 128$.

Although the constrained optimization problem in Eq. 4.2 has the same mathematical form as that in Eq. 3.11, and the same learning algorithm as in Eqs. 3.13 and 3.15 in Chapter 3.2 can be used, the interpretation of the constraints are different.

Sparseness constraint. Essentially, the SIFT descriptor is the histogram of local gradients around a feature point and therefore suffers from image noise, as shown in Fig. 4.2. When extracting the bases for SIFT descriptor, the sparseness constraint is intentionally added to force the small values in the gradient histogram to be zero, so that the extracted bases has a better suppression over the image noise.

Uncorrelatedness constraint. As shown in Fig. 4.1 (a), the 128-dimensional SIFT descriptor has a lot correlation between different dimensions. Therefore, the uncorrelatedness constraint is put on column vectors of matrix E so that the extracted bases could be a set of uncorrelated vectors. Fig. 4.1 (b) shows the dimensionality reduced SIFT descriptor where all off-diagonal entries of the covariance matrix are close to zero, demonstrating the redundancy from the correlation has been removed.

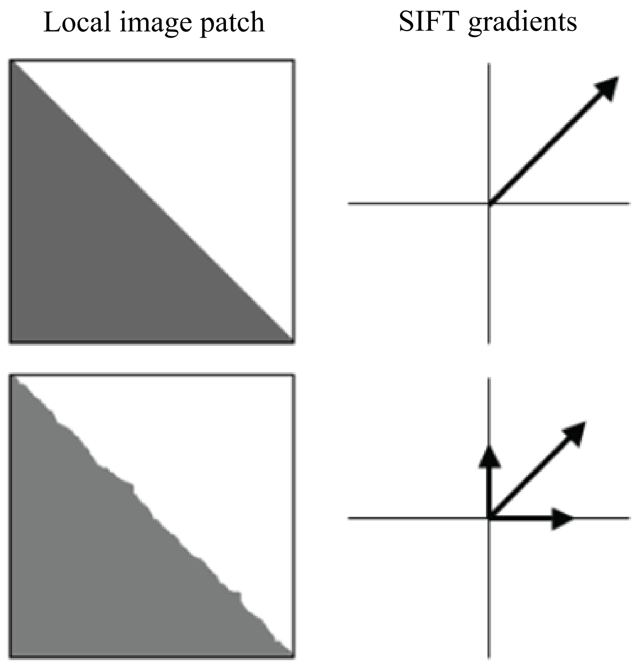


Figure 4.2: The SIFT descriptor is composed of local gradients and therefore vulnerable to noise (adapted from [Bay et al. 2008]).

4.2 Offline Training and Dimension Selection

To build the training set, 24 images are chosen from the Oxford dataset [Oxford 2002], where 3 images from each group are used and there were 8 groups of images from various scenes. 24027 SIFT descriptors are collected from these 24 images, that is, around 1000 SIFT descriptors from each image. These SIFT descriptors were put into the matrix X such that $X \in R^{128\times 24027}$.

The training process has been performed to acquire the basis matrix $E \in R^{128\times n}$. Then the least squares solution can be used to get

$$S = (E^T E)^{-1} E^T X. \tag{4.3}$$

Finally matrix S will be used as a dimensionality reduced descriptor set for storage and transmission.

To select a proper dimension parameter n, the following experiment has been carried out. The feature-based image comparison has been applied on the Oxford dataset, using

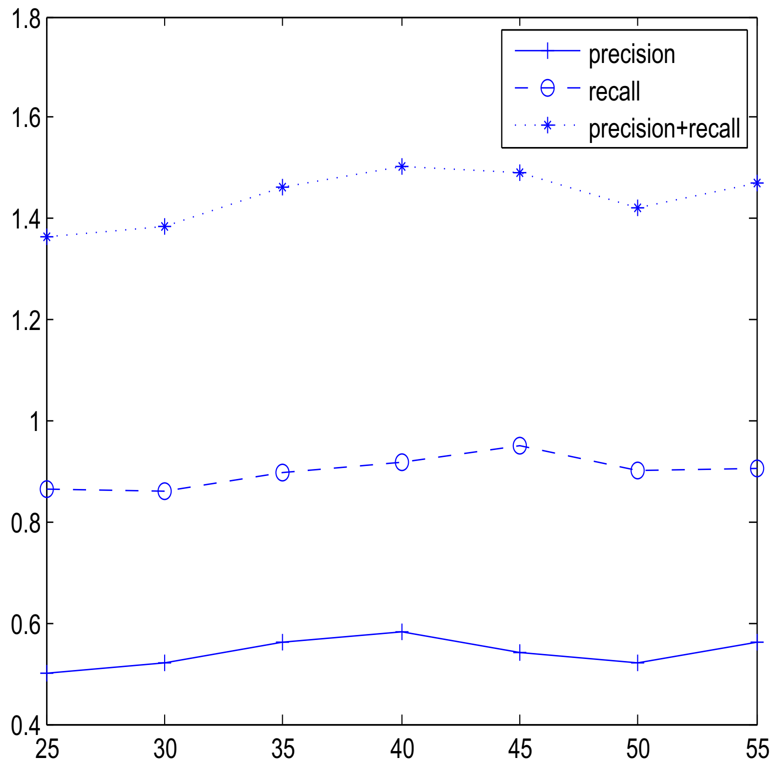


Figure 4.3: Recall and precision varies as the number descriptor dimension changes.

Harris detector plus the SIFT descriptor, then the dimensionality reduction is applied on the 128-dimensional SIFT descriptor. The reduced dimensionality varied from 25 to 55 and the precision and recall values (the definitions of precision and recall will be explained in Chpater 5.2.1) are recorded. The ground truth is that each group in the Oxford dataset contain images from the same scene but under imaging condition variations, therefore the images from the same group are "similar" images but those from different groups are not. By thresholding the number of matched feature descriptors, the feature-based image comparison algorithm claims those image pairs having larger number of matching feature descriptors are similar. The precision and recall are calculated by comparing the results with the ground truth, and the results are shown in Fig. 4.3.

The precision measures among all the detected similar image pairs, how many of them are really "similar"; the recall measures among all the "similar" image pairs, how many of them can be correctly detected. Depending on how to choose the threshold, the precision will increase but the recall will decrease, or vice versa. Therefore their summation is also plotted for a fair comparison. Fig. 4.3 shows that when compressing the 128-dimensional SIFT descriptor into a 40-dimensional vector, the precision/recall measured performance peaks. Therefore the 128-dimensional SIFT descriptors is chosen to reduced into 40-dimensional descriptors and name it the **"SIFT-**40**"** descriptor.

Chapter 5

Experimental Results

In this chapter, thorough evaluation has been performed on various feature detectors, including the Harris corner detector, Shi-Tomasi corner detector, Multi-Harris detector, FAST corner detector, the detector in SIFT, the detector in SURF, the proposed DWT-based corner detector and the LUC corner detector.

Also various detector-descriptor combinations are evaluated when used in feature-based image comparison. These combinations include SIFT, SURF, Harris detector with SIFT descriptor, Harris detector with SURF descriptor, DWT-based corner detector with SIFT descriptor, DWT-based corner detector with SURF descriptor, the detector in SIFT with the moment invariants descriptor, the detector in SURF with moment invariants descriptor, Harris detector with moment invariants descriptor, DWT-based corner detector with moment invariants based descriptor, LUC detector with SIFT descriptor, SURF descriptor, the moment invariants descriptor, and the SIFT-40 descriptor.

The content in this chapter has been appeared in [Bai and Qi 2009, 2010, 2011a,b].

5.1 Performance Comparison for Feature Detectors

To evaluate the performance of feature point detectors, the metric called *repeatability* is used, which measures the geometrical stability of the detected feature points between different images of the same scene taken under different viewing conditions [Schmid et al. 2000]. The repeatability of several feature point detectors are calculated, including Harris detector, Shi-Tomasi detector, Multi-Harris detector, FAST detector, SIFT detector, SURF detector, the DWT-based corner detector and the LUC detector. The performance of the eight detectors are compared using the 6 “graffiti” images and 6 “wall” images from the Oxford dataset [Oxford 2002].

5.1.1 Repeatability

Given two images I^1 and I^2 taken from the same planar scene under a viewpoint change, assume a real scene point X^r appears in both images, denoted as X^1 and X^2, respectively. Then the relation between X^1 and X^2 can be represented by

$$w \times X^2 = H_{12}X^1 \tag{5.1}$$

where $H_{12} \in R^{3\times3}$ is a *homography matrix*, $X^1 = [x_1^1, x_2^1, 1]'$ and $X^2 = [x_1^2, x_2^2, 1]'$ are homogeneous coordinates and w is a non-zero constant [Semple and Kneebone 1952]. If there are many real world points from a planar scene, the relations between two images of real world points can all be represented by the same homography matrix.

A point X^1 detected in image I^1 is said to be repeatedly detected in image I^2 if the corresponding point X^2 is detected in image I^2. The repeatability rate is defined as the number of feature points repeatedly detected between two images with respect to the number of feature points that should be repeatedly detected. Due to the viewpoint change in the imaging step, such as rotation and scale variation, the observed scene parts in the two images may differ. And the number of repeated points is only considered from the common scene part of the two images. This common scene part is determined by the homography matrix. For example, to count the number of repeatedly detected points between images I^1 and I^2, assume the homography matrix H_{12} that projects points in I^1 to I^2 is known, and thereafter $H_{21} = H_{12}^{-1}$ that projects points in I^2 to I^1. Denote the points that lie in the common part of images I^1 and I^2 as

$$\tilde{X}^1 = \{X^1 | H_{12}X^1/w \in I^2\} \tag{5.2}$$

$$\tilde{X}^2 = \{X^2 | H_{21}X^2/w \in I^1\} \tag{5.3}$$

where X^1 and X^2 stand for points detected in images I^1 and I^2, respectively. Then the number of points detected in the common part of images I^1 and I^2 are denoted as n_1 and n_2, respectively where $n_1 = |X^1|$ and $n_2 = |X^2|$.

Moreover, the repeatability rate is a function of the tolerance measuring the uncertainty of the detection. A repeatedly detected point X^1 in image I^1 will in generally not be detected exactly at position X^2, i.e., the projection of X^1, but several pixels away from it. The magnitude of this displacement is denoted as σ and the repeatability rate measured under such a displacement is called *σ-repeatability*. Two point correspondence between X^1, X^2 under the σ-tolerance is represented by

$$R_{12}(\sigma) = \{(X^1, X^2) \quad | \quad ||H_{12}X^1/w - X^2|| < \sigma\} \tag{5.4}$$

And the repeatability rate $r_{12}(\sigma)$ between images I^1 and I^2 is defined as:

$$r_{12} = \frac{|R_{12}|}{min(n_1, n_2)} \tag{5.5}$$

where the $min(\cdot)$ takes care of the inconsistence between the numbers of detected feature points in the two images. For example, in the case of a scale change, the image with a high resolution would have more feature points detected in the common scene part.

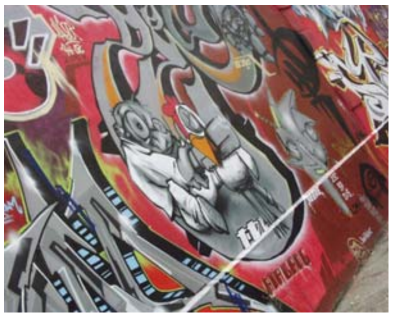
5.1.2 Dataset and the Ground Truth

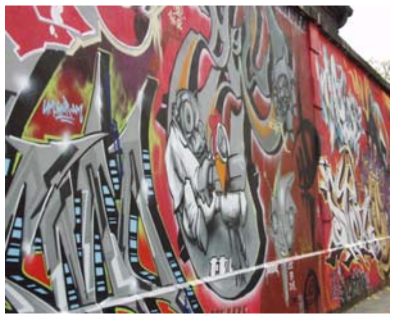
The σ-repeatability for various feature point detectors are measured on the 6 "graffiti" images and 6 "wall" images from the Oxford dataset [Oxford 2002]. The Oxford dataset contains 48 images from 8 different scenes (with each scene contains 6 images) and the same dataset was used in [Mikolajczyk and Schmid 2005] to provide a performance evaluation of local descriptors. The 6 "graffiti" images are taken from a graffiti wall under scale, viewpoint and rotational changes, as shown in Fig. 5.1. The 6 "wall" images are taken from a brick wall under scale, viewpoint and rotational changes, as shown in Fig. 5.2. The "graffiti" and "wall" images represent two different scene types, with the former containing structured scenes (homogeneous regions with distinctive edge boundaries) and the latter containing repeated textures.

(a)

(b)

(c)


(d)


(e)

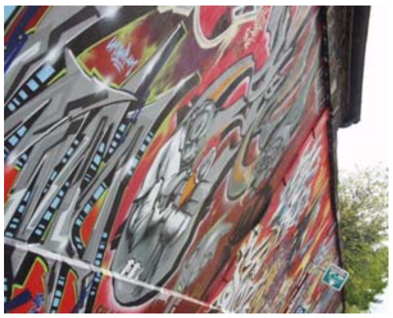

(f)

Figure 5.1: The 6 “graffiti” images from the Oxford dataset. The image sizes are all 640×800.

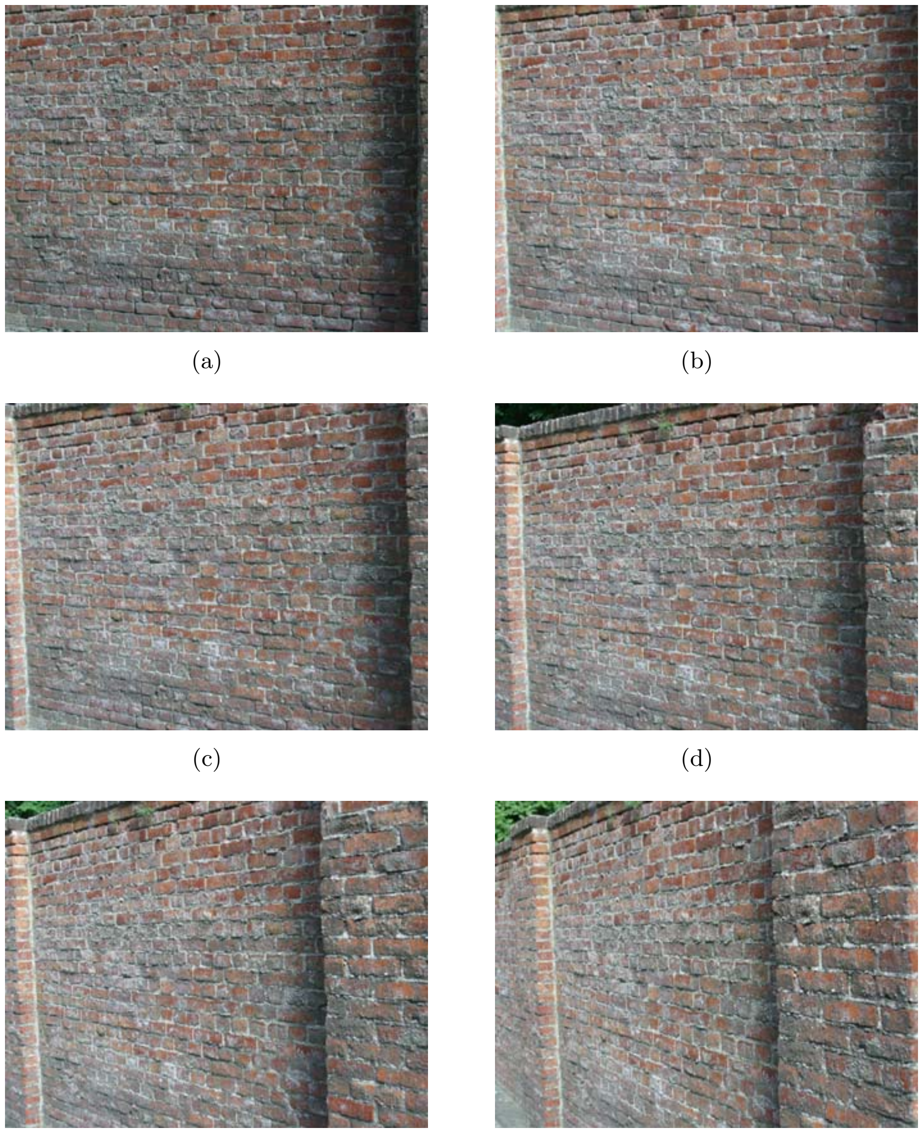


Figure 5.2: The 6 “wall” images from the Oxford dataset. The image sizes are all 680×880.

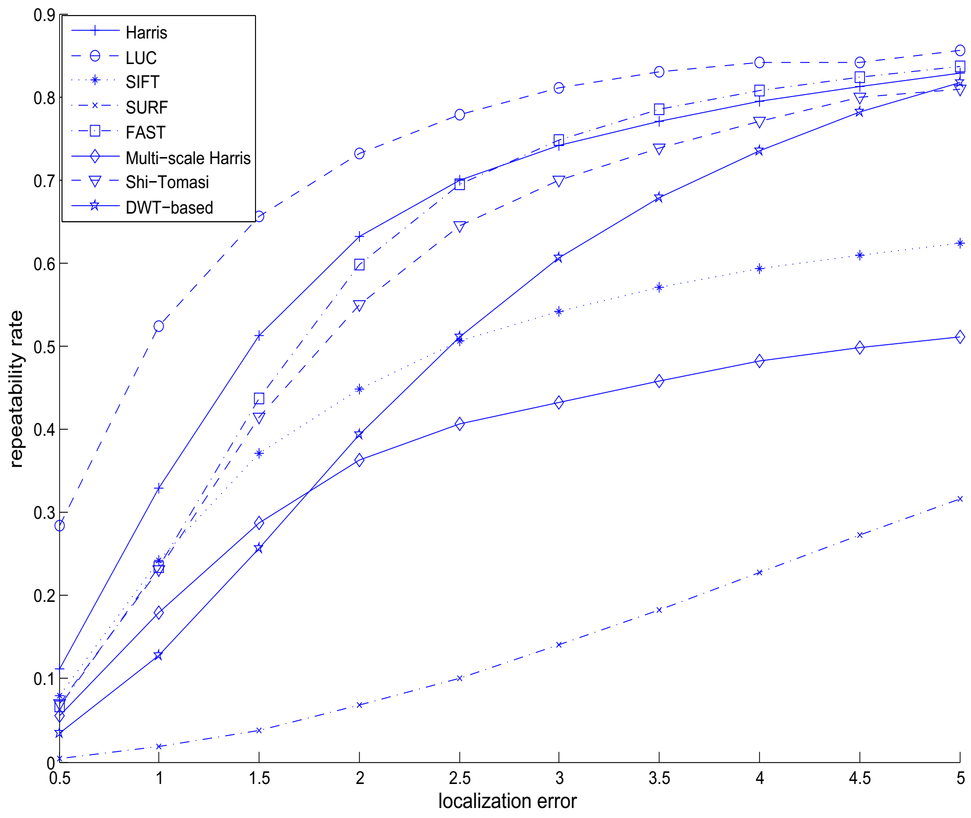


Figure 5.3: Repeatability rate as a function of the tolerance σ for the "graffiti" image set.

Both the graffiti and the wall are ideal planar scenes and a homography matrix can describe the geometric relation of every image pair. Among the 6 images, 15 image pairs can be formed with each including two images. The homography matrix for each image pair is pre-calculated and used as the ground truth in calculating the σ-repeatability.

5.1.3 Experimental Results for the Repeatability

Figs. 5.3 and 5.4 list the σ-repeatability for Harris detector, Shi-Tomasi detector, Multi-Harris detector, SIFT detector, SURF detector, FAST detector, the LUC detector and the DWT-based detector. The tolerance parameter σ varies from 0.5 to 5.

The results shown in Figs. 5.3 and 5.4 reveal three facts: Firstly, all corner detectors working on single scale images demonstrate better repeatability rate than blob-like feature detectors; Secondly, the LUC detector has achieved the best repeatability rate under most σ values; Thirdly, for most feature detectors, the repeatability increases fast when the

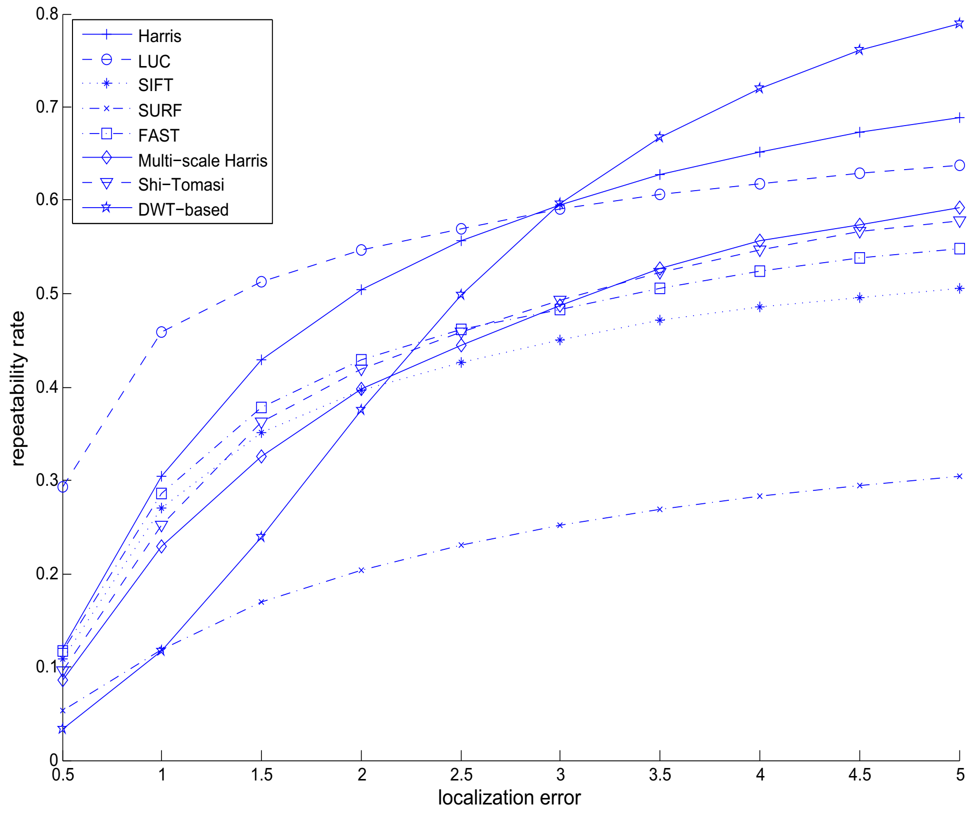


Figure 5.4: Repeatability rate as a function of the tolerance σ for the “wall” image set.

localization error (σ) reaches 3, and slowly afterwards, meaning the localization error for most feature detectors is within 3 pixels.

The first observation is due to the fact that corners are better feature points than the blob-like features because each corner point has unique location and is not subject to the scale variations. On the other side, the blob-like feature points have a strong relation with the scale variations — A blob feature point can be detected only at a certain scale. And the construction of a scale space for an image involves Gaussian blur, which decreases the localization accuracy. For the DWT-based corner detector, the localization inaccuracy comes from the Gaussian interpolation when building the corner strength map, an indirect consequence from using the scale-space. The second observation is due to the fact that the proposed LUC detector has a better corner point localization accuracy than other corner detectors. The third observation actually shows that 3 pixels are large enough to accommodate the inaccuracy of the feature point localization.

Besides, the overall repeatability rate for the "wall" image set is lower than that from the "graffiti" image set. This is because the actual wall in the scene was made up of bricks and some bricks wore and tore, making the wall not an ideal planar object, therefore larger error is resulted in the homography transformation in Eq. 5.4. The repeatability of the LUC detector is inferior to that of the Harris detector and this is because of the fact that when the σ value is greater than the error, more corners in each image pairs will be considered as corresponding points, making the increase of the repeatability for Harris detector stronger than that of the repeatability for the linear unmixing based detector.

5.2 Performance Comparison for Feature-based Image Comparison

To evaluate the feature-based image comparison scheme, the performance of using combinations of several feature detectors and descriptors for image comparison has been tested on the Oxford dataset [Oxford 2002]. These combinations include SIFT, SURF, Harris detector with SIFT descriptor, Harris detector with SURF descriptor, DWT-based corner detector with SIFT descriptor, DWT-based corner detector with SURF descriptor, the detector in SIFT with the moment invariants descriptor, the detector in SURF with moment invariants descriptor, Harris detector with moment invariants descriptor, DWT-based corner detector with moment invariants based descriptor, LUC detector with SIFT descriptor, SURF descriptor, moment invariants descriptor and the compressed SIFT-40 descriptor.

5.2.1 Evaluation Metrics

Four metrics are used for performance evaluation purpose, i.e., number of feature points detected, number of bytes transferred, recall and precision. The first two metrics are straightforward, that used to evaluate resource consumption of the algorithm. The latter two reflect algorithm performance compared to the ground truth.

Let True Positive (TP) represent the detected correct matching image pairs, False Positive (FP) the unmatching image pairs but have been detected as matching, and the False Negative (FN) the matching image pairs but have been detected as unmatching. Then *Recall* is defined as the ratio between the number of TPs and the total number of TPs and FNs. *Precision* is defined as the ratio between the number of TPs and the total number of TPs and FPs.

$$\begin{aligned} Recall &= \frac{TP}{TP + FN} \\ Precision &= \frac{TP}{TP + FP} \end{aligned} \tag{5.6}$$

The higher the recall, the less "misses" compared to the true match; and the higher the precision, the less "false alarms".

5.2.2 Datasets and the Ground Truth

The performance is evaluated on the Oxford datasets [Oxford 2002], that contains images from different scenes and is for general image feature evaluation purpose. The same dataset was used in [Mikolajczyk and Schmid 2005] to provide a performance evaluation of local descriptors.

Window size. A free parameter in the feature-based image comparison system is the size of a local area (or window size) used to calculate the feature descriptor for a detected feature point. The default window size from the SIFT and SURF descriptors is used, which is 16×16 and 20×20, respectively. The window size for calculating the moment invariants

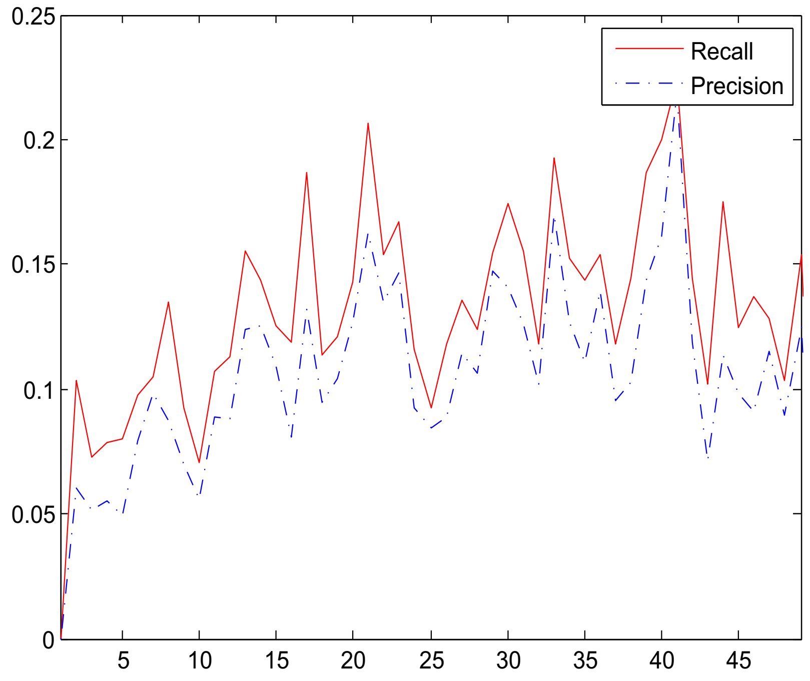


Figure 5.5: Precision and recall as functions of the half window size for calculating the moment invariants in feature description.

is determined through empirical study. The 6 "graffiti" images from the Oxford dataset is used as references, and compare them with all other images in the Oxford dataset, as varying the half window size from 2 to 50.

Fig. 5.5 shows the *recall* and *precision* as functions of the half window size for calculating the moment invariants for feature description. The consistent trend can be observed between "recall versus window size" and "precision versus window size", and that different window size does affect both the *recall* and *precision*. 23 is used as the half window size because it provides a good enough recall/precision rate. While increasing the window size to, e.g., 38, it would improve the performance to some extent, it would also incur more computational cost.

Finding Correspondence Between Feature Points. When comparing the similarity between two images, the first step is to find the correspondence of detected feature points across the two images. The second step is to count the number of corresponding feature points. If this number is large enough, the two images are claimed to be "similar".

Given two sets of feature descriptors generated from two images, for each feature descriptor in the first image, the distances between this descriptor and all feature descriptors in the second image are calculated. If the ratio between the smallest distance and the second smallest distance is below a threshold, the feature pair with the smallest distance is claimed to be a matching feature pair, or in other words, the two feature points correspond to each other. The threshold has been experimentally determined to be 0.3. Both SIFT and SURF use this threshold in their implementations.

5.2.3 Feature-based Image Comparison on the Oxford Dataset

The Oxford image dataset contains 8 groups of images from 8 totally different surroundings. Each group has 6 images taken from the same scene but under imaging condition changes. Figure 5.6 shows two images from each group of the Oxford dataset, where six imaging condition variations are present, including blur (Figs. 5.6 (a) and 5.6 (b)), viewpoint change (Figs. 5.6 (c) and 5.6 (d)), scale and rotation changes (Figs. 5.6 (e) and 5.6 (f)), illumination change (Fig. 5.6 (g)), and JPEG compression (Fig. 5.6 (h)).

Various combinations of feature detectors and descriptors are applied on the Oxford dataset. Firstly image features are detected and descriptors are calculated from the 48 images, out of which one set is selected as a query. Secondly the query feature set is compared to other 47 record feature sets and records that have large enough number of matching features with the query set will be returned as retrievals. Assume the maximum number of matching features between the query and the record is N, then the threshold is chosen to be $N \times 0.85$, such that any record having matching features more than this threshold will be "claimed" as a retrieval. Thirdly the retrieved feature set(s), or image(s) is(are) compared to the ground truth, and compute the TP, FP, FN, TN, recall and precision. The ground truth is based on the fact that all images from the same group are similar but not otherwise. A 48-fold cross validation is applied such that every image from the dataset will be used as the query once and all the results are averaged out, which is presented in Table 5.1.

Table 5.1 shows that in terms of resource consumption, SIFT is the worst, considering it actually creates even larger volume of data than the raw image. Except for the combination of the DWT-based detector and the SIFT descriptor, which generates comparable data

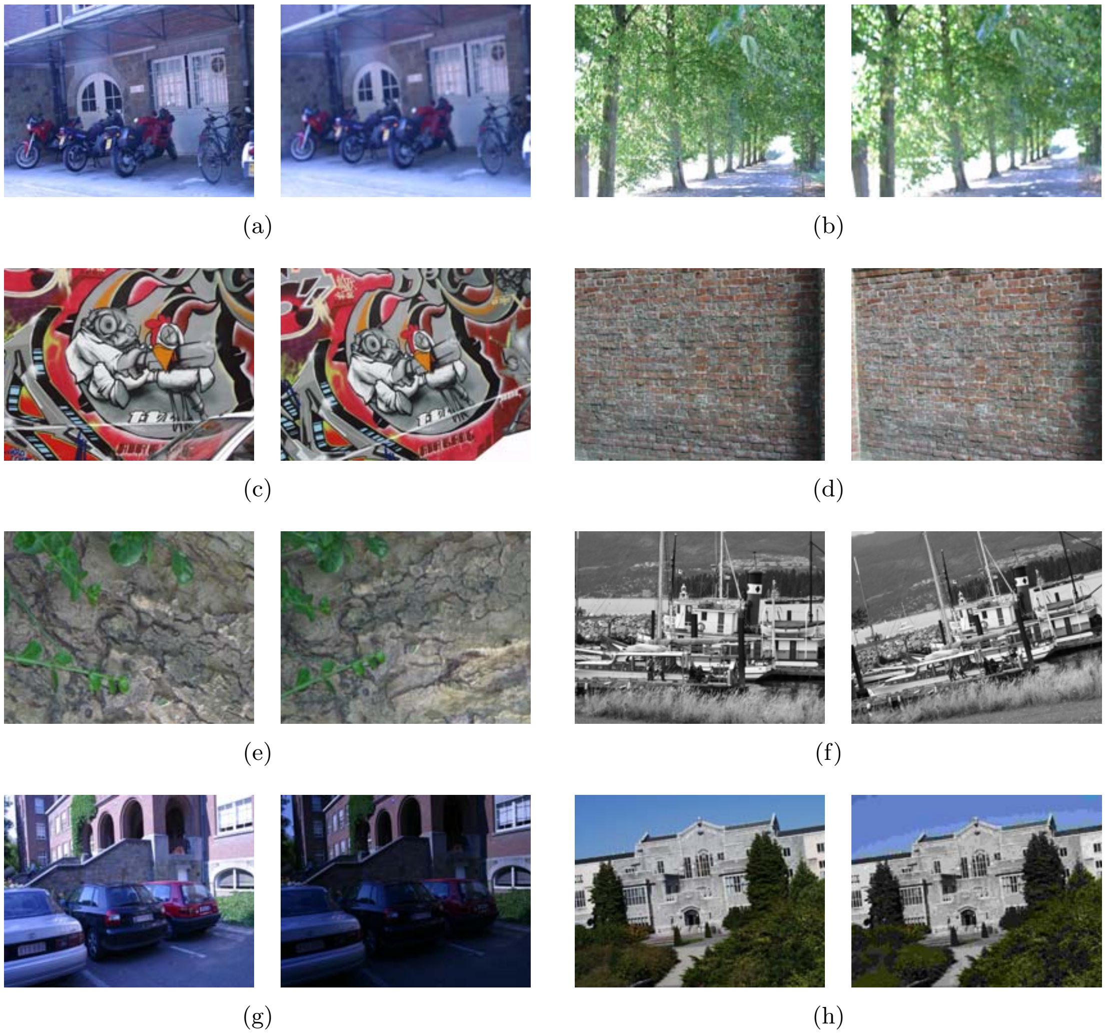


Figure 5.6: The Oxford dataset. Examples of images from each group shows the imaging condition changes: (a) and (b) blur; (c) and (d) viewpoint change; (e) and (f) zoom plus rotation; (g) illumination change and (h) JPEG compression.

	Descr. length	No. of feature points	Feature data vol. (KB)	Recall	Precision	Recall + Precision
SIFT	128	5938	3040.3(15)	0.3542	1.0000	1.3542(6)
SURF	64	1478	378.4(13)	0.3958	1.0000	1.3958(3)
Harris + SIFT descr.	128	442	226.3(10)	0.4583	0.9375	1.3958(3)
Harris + SURF descr.	64	442	113.2(7)	0.7083	0.6970	1.4053(2)
DWT + SIFT descr.	128	1145	586.2(14)	0.4375	0.9063	1.3438(7)
DWT + SURF descr.	64	1145	293.1(11)	0.4375	0.5758	1.0133(12)
SIFT detec.+ M. I.	7	5938	166.3(8)	0.4167	0.5084	0.9251(14)
SURF detec.+ M. I.	7	1478	41.4(4)	0.3958	0.4029	0.7987(15)
Harris + M. I.	7	442	12.4(2)	0.2014	0.7292	0.9306(13)
DWT + M.I.	7	1145	32.1(3)	0.1566	0.8958	1.0524(10)
LUC + SIFT	128	387	198.14(9)	0.4583	0.9841	1.4424(1)
LUC + SURF	64	387	99.07(6)	0.3667	1.0000	1.3667(5)
LUC +M.I.	7	387	10.84(1)	0.3333	0.6944	1.0277(11)
LUC + SIFT-40	40	387	61.92(5)	0.4500	0.8536	1.3036(8)
PCA-SIFT	36	2065	297.4(12)	0.5208	0.6110	1.1318(9)

Table 5.1: Recall and precision of each feature detector/descriptor combination. The test run on the Oxford dataset and results in the third and fourth columns are for each image on average. For reference, the average raw image volume is 574.2KB. Assume a float type number takes 4 Bytes in storage. The numbers in the parentheses in column 4 and 7 indicate the "rank" of the detector/descriptor combination in terms of the corresponding performance metric in that column.

volume as the raw image, all other combinations of feature detector/descriptor provide savings in data storage and transmission, especially those using moment invariants. The data volume is also ranked in the fourth column (the number in the parentheses) for an illustrative comparison on how compact the image feature sets are.

In terms of performance accuracy, the recall and precision are added together and ranked in the seventh column (the number in the parentheses). The LUC detector plus SIFT descriptor achieved the best performance in terms of accuracy, followed by Harris detector plus SURF descriptor and SURF. Harris detector performs well when used together with SIFT or SURF descriptor, generating better balance between recall and precision. The DWT-based corner detector does not perform as well as the Harris detector when used with SIFT or SURF descriptor. This is because the DWT-based detector is more sensitive to discretization noise caused by image rotations which happen to be very common in the Oxford dataset (4 out of the 8 groups contain strong rotational variations, another 3 contain weak rotational variations).

In addition, by observing the fourth to seventh rows, the SIFT descriptor is superior to the SURF descriptor in terms of the precision. And from the eighth to eleventh rows, the moment invariants-based feature descriptors are all inferior to SIFT or SURF descriptors in terms of precision. Considering the definitions of recall and precision, a comparable recall value but extremely low precision value implies there are too many False Positives (or false alarms).

5.3 Conclusion

This chapter lists the experimental results for evaluating different feature point detectors and different combinations of detector/descriptor in feature-based image comparison method.

The performance of feature detectors are measured by the repeatability, which quantifies the stability of the detected feature point under geometric variations. The results show the superiority of the proposed LUC detector in terms of both the repeatability and the computational efficiency. The proposed DWT-based corner detector provides another approach for feature point detection in scale-space, and shows its advantage under some circumstances, for example, when the images to be compared contain many repeated textures, as in the 6 “wall” images from the Oxford dataset.

The performance of the feature-based image comparison methods are measured by the recall and the precision. Recall measures among all the similar images, how many of them can be detected as “similar”; precision measures among all the detected “similar” images, how many of them are really similar. Different combinations of detectors and descriptors are tested on the Oxford dataset. The judgement is made based on both accuracy (as measured by the recall and the precision) and efficiency (data volume). The results show that the proposed LUC detector with the SIFT descriptor combination gave the best result in terms of accuracy, and has comparable performance in terms of efficiency. And the LUC detector plus the compressed SIFT-40 descriptor, although sacrifices in accuracy for a small amount, achieved much better balance in terms of efficiency.

Chapter 6

Feature-based Image Comparison on WVSN Datasets

In this chapter, the feature-based image comparison is applied on two image datasets from WVSNs.

The content in this chapter has been appeared in [Bai and Qi 2009, 2010].

6.1 Feature-based Image Comparison on the MSP Image Dataset

This section shows how to apply the feature-based image comparison in a small size WVSN testbed for semantic neighbor selection, and provide the performance evaluation. This WVSN is composed of 12 Mobile Sensor Platforms (MSPs) built at the AICIP lab [Beall and Qi 2006]. The structure of MSP is based on two 12×12 inch PVC sheets. The parts mounted on the platform include a Mini-ITX motherboard with 1.5GHz processor, 1GB RAM, 40GB hard drive, 802.11g wireless card, two 12VDC gearhead motors, and H-Bridge circuit to control the motors. The on-board sensing devices include a Logitech Quickcam 4000 Pro webcam and an ultra-sonic range sensor, connected via USB and the parallel port, respectively. A complete assembled MSP is shown in Fig. 6.1. The images from the second dataset are taken in an office setup. Unlike the Oxford dataset, where images from different groups bear distinctive differences, all images taken from the small-size WVSN

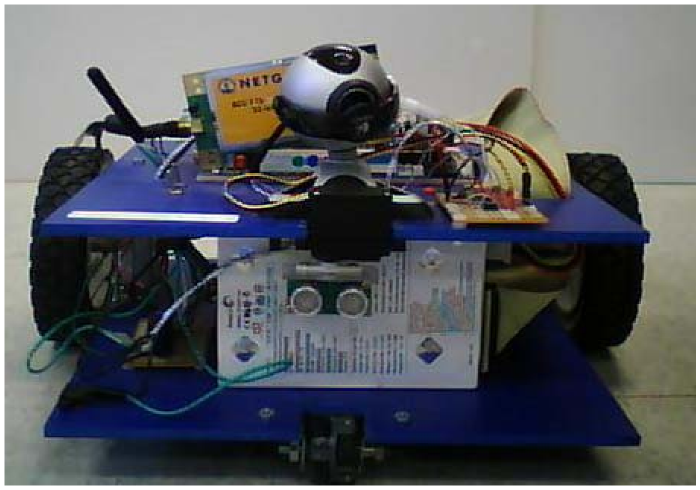


Figure 6.1: The Mobile Sensor Platform (MSP)

have similarity to some degree, making feature-based image comparison more challenging.

In this experiment, 12 MSPs are deployed within a 10 by 10 grid area in an office setup. The deployment map is shown in Fig. 6.2. The position and orientation of every MSP is randomly assigned.

Figure 6.3 shows the 12 images in this dataset. Different from the Oxford image dataset which contains images from totally different surroundings, the images in this dataset are all taken from the same office but from different viewpoints. Therefore, there are many common sub-regions (like the floor, the wall and the ceiling) in images even when the cameras are not shooting to the same direction. This would result in a higher False Positive rate compared to the Oxford dataset. Another noteworthy problem is that for some MSPs, even they are shooting the similar scene, the overlapped scene region among their images is quite small, due to the differences in viewpoints. This would results in a higher False Negative rate compared to the Oxford dataset.

The 14 combinations of feature detector/descriptor are applied on the 12 images and every feature set will be selected as the query to be compared with the others. The ground truth is found from the knowledge of the deployment map, through which it is easy to know which MSP is shooting at the same scene as other MSPs and therefore build the image matching relations.

Table 6.1 lists the results of resource consumption and performance accuracy. In terms

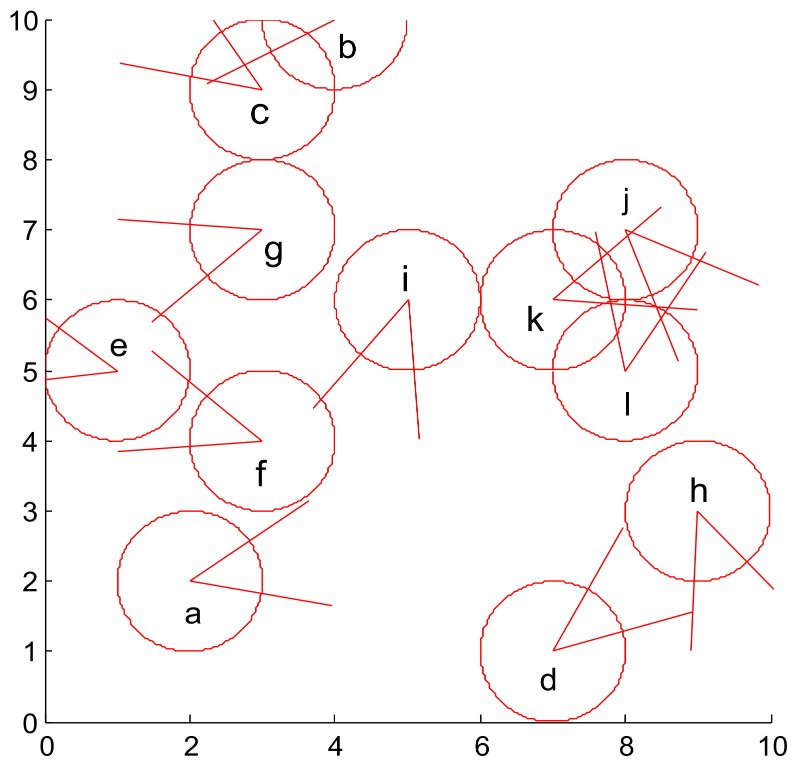


Figure 6.2: 12 MSPs in a 10 by 10 grid map. The letter within each circle indicates the image label in Fig. 6.3.

(a)

(b)

(c)

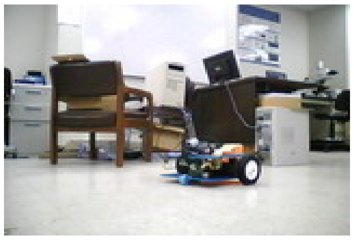
(d)

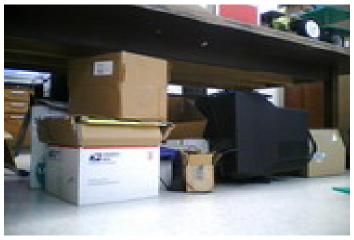
(e)

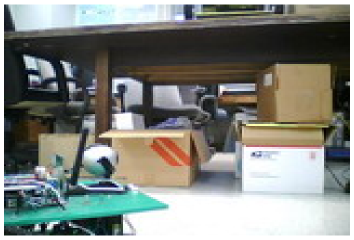
(f)

(g)

(h)


(i)


(j)


(k)

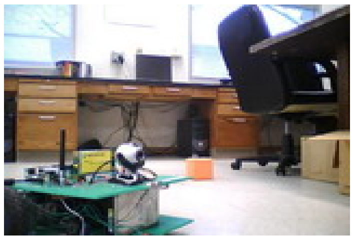

(l)

Figure 6.3: 12 images taken from 12 MSPs.

of resource consumption, SIFT and DWT-based detector plus SIFT descriptor give us the worst result by generating larger data volume than that of the original image. On the other hand, the moment invariants-based descriptor generates the most compact feature sets.

In terms of performance accuracy, as expected, all combinations of feature detector/descriptor give us worse results compared to those generated from the Oxford dataset. In general, SIFT descriptors are superior to SURF descriptors in terms of accuracy, which is consistent with the Oxford dataset, although the SURF gave us the best performance in terms of accuracy, followed closely by the LUC detector plus the SIFT descriptor. Again, the LUC plus the SIFT-40 gave us a good balance between accuracy and efficiency.

For detectors, Harris detector and DWT-based detector are comparable in terms of recall and precision. This conclusion is different from that of the Oxford dataset, because in the MSP dataset, all images are taken with MSPs laying down on the ground, and there is little rotational variation in images.

6.2 Feature-based Image Comparison on the COIL-100 Dataset

The feature-based image comparison is applied on another dataset, the Columbia Object Image Library (COIL-100) set [Nene et al. 1996]. The COIL-100 is a dataset containing images of 100 objects. Each object was placed on a motorized turntable against a black background and the turntable rotated 360° to vary the object's pose with respect to a fixed camera. An image was taken every time the turntable rotated 5° (separation angle), generating 72 images for each object with different poses.

25 images are selected from 5 objects, 5 images from each object, with the separation angle being 25°, as shown in Fig. 6.4.

The 25 images are in 5 groups with each group containing the images from the same object. The feature-based image comparison method is then applied on this dataset and the results are listed in Table 6.2.

Table 6.2 shows that in terms of data compactness, moment invariants descriptors showed its advantage, but their accuracy are not competitive. On the other hand, LUC detector plus either SIFT descriptor or SURF descriptor has higher accuracy, but sacrifices in data compactness. The LUC detector with SIFT-40 descriptor, however, again reached

	Descr. length	No. of feature points	Feature data vol. (KB)	Recall	Precision	Recall + Precision
SIFT	128	139	71.2(14)	0.4000	0.3077	0.7077(8)
SURF	64	72	18.4(9)	0.4500	0.5294	0.9794(1)
Harris + SIFT descr.	128	39	20.0(10)	0.3500	0.3043	0.6543(9)
Harris + SURF descr.	64	39	10.0(7)	0.4500	0.1915	0.6415(10)
DWT + SIFT descr.	128	87	44.5(13)	0.4000	0.3636	0.7636(7)
DWT + SURF descr.	64	87	22.2(11)	0.2500	0.2000	0.4500(12)
SIFT detec.+ M. I.	7	139	3.9(5)	0.1500	0.2000	0.3500(13)
SURF detec.+ M. I.	7	72	2.0(3)	0.1500	0.1071	0.2571(14)
Harris + M. I.	7	39	1.1(1)	0.4000	0.4211	0.8211(4)
DWT + M. I.	7	87	2.4(4)	0.3000	0.2609	0.5609(11)
LUC + SIFT	128	58	29.70(12)	0.5000	0.4706	0.9706(2)
LUC + SURF	64	58	14.85(8)	0.3500	0.4593	0.8093(5)
LUC +M.I.	7	58	1.62(2)	0.7000	0.2311	0.9311(3)
LUC + SIFT-40	40	58	9.28(6)	0.4500	0.3333	0.7833(6)

Table 6.1: Recall and precision of each feature detector/descriptor combination. The test run on the MSP dataset and results in the third and fourth columns are for each image on average. For reference, the average raw image volume is 19.2KB. Assume a float type number takes 4 Bytes in storage. The numbers in the parentheses in columns 4 and 7 indicate the "rank" of the detector/descriptor combination in terms of the corresponding performance metric in that column.

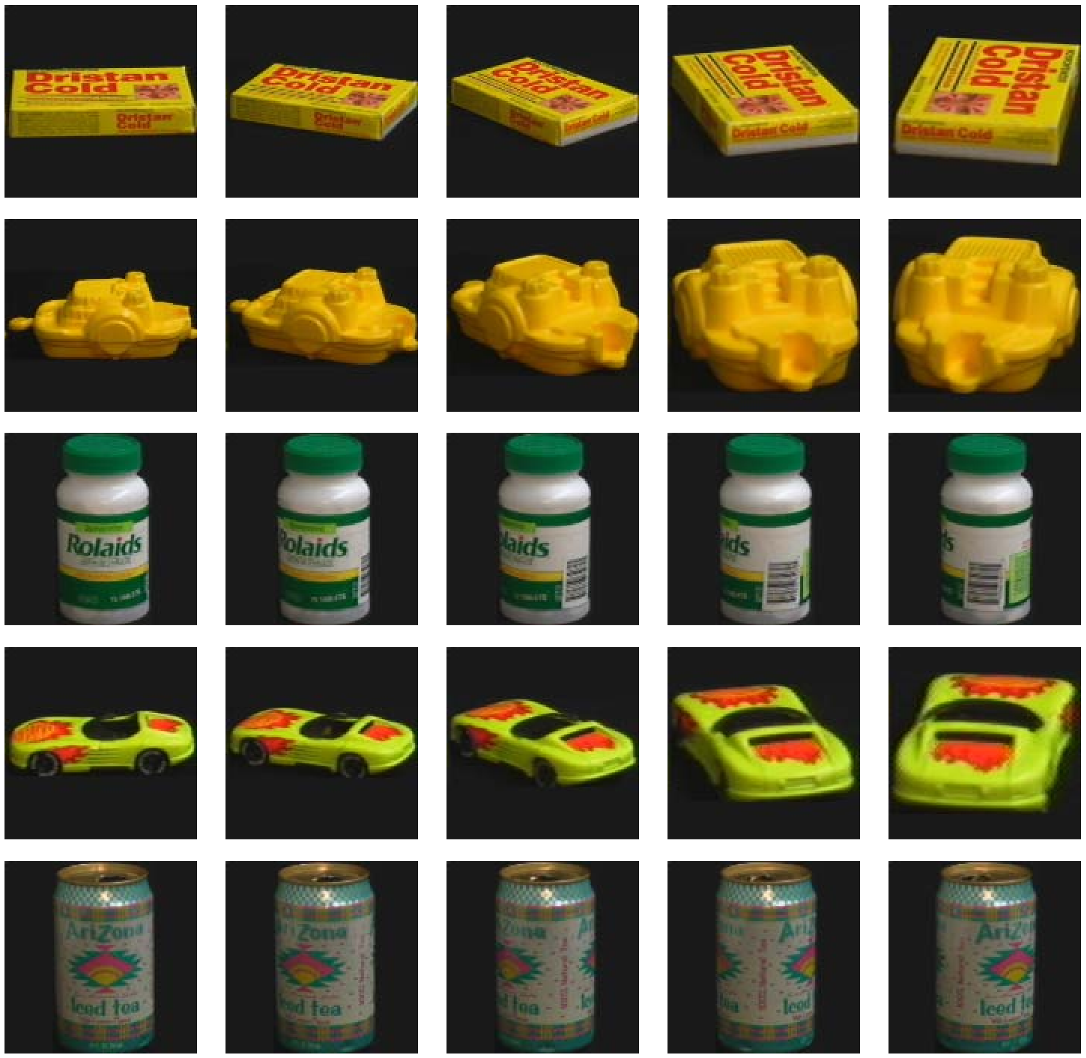


Figure 6.4: The 25 images from 5 objects 5 in the COIL-100 dataset, with the viewing angle 0°, 25°,...,100° (from left to right).

	Descr. length	No. of feature points	Feature data vol. (KB)	Recall	Precision	Recall + Precision
SIFT	128	104	53.25(15)	0.3817	0.9100	1.2917(2)
SURF	64	65	16.64(10)	0.5076	0.7200	1.2276(3)
Harris + SIFT descr.	128	54	27.65(12)	0.4500	0.7333	1.1833(8)
Harris + SURF descr.	64	54	13.82(8)	0.5200	0.6967	1.2167(5)
DWT + SIFT descr.	128	68	34.82(14)	0.2961	0.7800	1.0761(13)
DWT + SURF descr.	64	68	17.41(11)	0.3031	0.7000	1.0031(14)
SIFT detec.+ M. I.	7	104	2.91(5)	0.3151	0.8600	1.1751(9)
SURF detec.+ M. I.	7	65	1.82(3)	0.3253	0.8200	1.1453(11)
Harris + M. I.	7	54	1.51(1)	0.3171	0.6800	0.9971(15)
DWT + M. I.	7	68	1.90(4)	0.3600	0.8000	1.1600(10)
LUC + SIFT	128	56	28.67(13)	0.5000	0.8750	1.3750(1)
LUC + SURF	64	56	14.34(9)	0.3369	0.8800	1.2169(4)
LUC +M.I.	7	56	1.57(2)	0.3472	0.7500	1.0972(12)
LUC + SIFT-40	40	56	8.96(7)	0.4374	0.7600	1.1974(6)
PCA-SIFT	36	30	4.32(6)	0.5600	0.6333	1.1933(7)

Table 6.2: Recall and precision of each feature detector/descriptor combination. The test run on the COIL-100 dataset and results in the third and fourth columns are for each image on average. For reference, the average raw image volume is 16.4KB. Assume a float type number takes 4 Bytes in storage. The numbers in the parentheses in columns 4 and 7 indicate the "rank" of the detector/descriptor combination in terms of the corresponding performance metric in that column.

a better balance between accuracy and efficiency.

By comparing Tables 5.1, 6.1, and 6.2, the recall and precision rates for the MSP dataset is severely worse than those for the Oxford dataset. Although both datasets contain image pairs having similar scenes, the degree of overlap in the similar image is different. The similar images in Oxford dataset have much more overlap than those in the MSP dataset. Another experiment is used to reveal the relationship between the amount of image overlap and the algorithm performance in terms of recall and precision rate.

To further investigate into how the degree of overlap between images affect the performance of feature-based image comparison, the COIL-100 set is used again, because the image overlap can be evaluated by the separation angle between two images, the smaller the separation angle, the more overlap they have.

The first 10 objects are selected from the COIL-100 dataset, one of which at different view angles is shown in Fig. 6.5. For each object, two images are used: The first image is the one taken at angle 0° and the second is the one taken at angle θ. For the first object, its image taken at angle 0° is used as a reference and compared to the rest 18 images using feature-based image comparison, and the recall and precision rates are recorded. The Harris detector plus the SIFT descriptor combination is used in this experiment because it demonstrated good performance in the Oxford dataset. This process is repeated for all 10 objects and the average is taken. Then the θ value varied from 10° to 180° with a step size of 10° and the results are plotted in Fig. 6.6.

The recall and precision, as well as their summations, are plotted separately in Fig. 6.6. Recall and precision are two contradict indices in that when other conditions are the same, lower recall rate would result in higher precision rate and vice versa. Therefore their summation is used to illustrate the general trend of the performance degradation. The summation is used instead of the average of recall and precision to provide an offset so that the curve can be better visualized. The summation curve stays stable when there is a 10° separation angle between the two images of the same object; when the separation angle increases to 40°, the curve drops 11%. The curve drops to its minimum when the separation angle is 90°, with a performance degradation of 34%, after which the performance starts to improve with the curve climbing back up.

The conclusion from Fig. 6.6 is that the feature-based image comparison method can

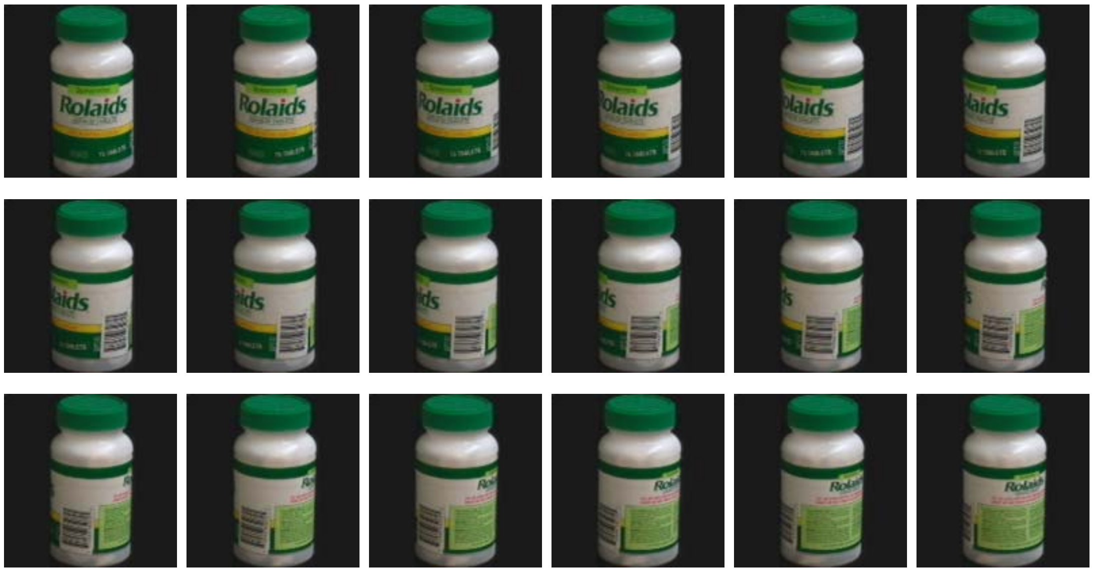


Figure 6.5: Images of object 5 from the COIL-100 dataset, at view angles 0°, 10°,...,170° (from left to right, from top to bottom), respectively.

perform well (within 11% degradation) as long as the similar images have a certain degree of overlap that corresponds to a 40° separation angle in viewpoints. But as the amount of overlap decreases, the performance drops to as many as 34%. Note that these measurements have not considered the factor of distance variations between the camera and the object, which would further reduce the performance. The performance improvements beyond the separation angle of 90° are due to the fact that the objects used in our experiment all show circular symmetry.

6.3 Discussion

A Wireless Visual Sensor Network environment is a more challenging setup for feature-based image comparison algorithms. First of all, it poses difficulties for image comparison itself by having to differentiate images of much similarity. Secondly, it requires the algorithms to be computationally light-weight so that the low end processor can afford running it. Finally, it requires the feature set to be compact enough so that the transmission overhead is low.

From our experiments, although LUC detector plus the SIFT descriptor could perform well on general image comparison problems or ideal cases from WVSNs, such as the Oxford

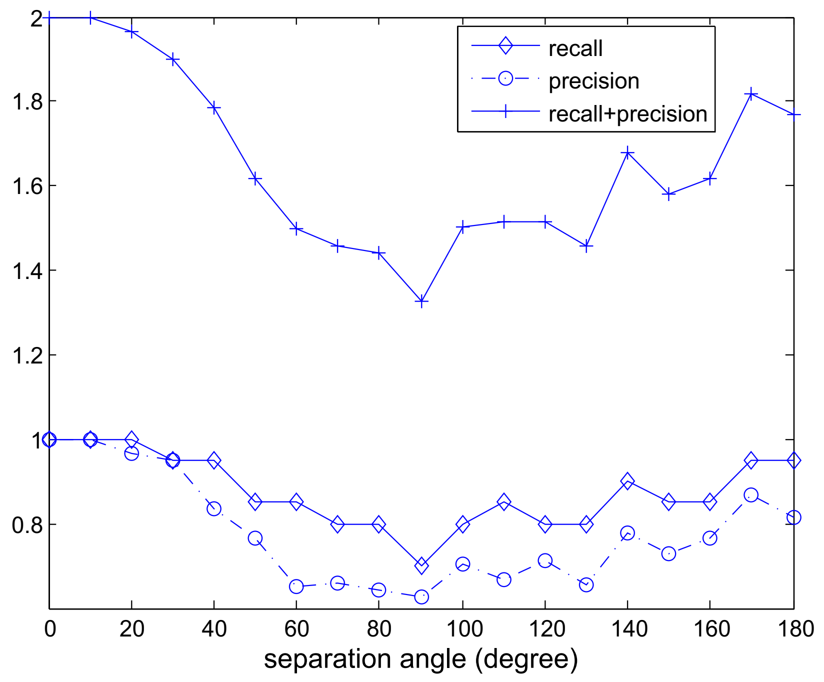


Figure 6.6: Recall and precision as a function of image overlap (measured by separation angle).

dataset and the COIL-100 dataset, their performances degrade severely in a more realistic WVSN environment, where images have high similarity, making it hard to distinguish different images. Therefore, one way to increase the performance accuracy is to perform background removal techniques prior to image feature detection.

Chapter 7

Conclusions and Future Works

In this dissertation, the feature-based image comparison method was described for resource-constrained WVSNs, where low-end visual sensors have very limited power, processing and wireless communication bandwidth resources, and therefore require carefully designed local processing techniques. The linear unmixing based corner detector, LUC and the DWT-based corner detector were proposed, which are two computationally light-weight corner detectors, and the LUC detector showed best performance in terms of the repeatability among several classical and state-of-the-art feature detectors. A dimensionality reduction method was also proposed that utilizes a set of bases extracted from nonnegative matrix factorization to represent high dimensional SIFT descriptors. Through thorough evaluation on three different image datasets, the feasibility of the feature-based image comparison method for using in WVSNs was validated.

7.1 Summary of Contributions

The principal contributions of this dissertation is the proposed feature-based image comparison method for WVSNs with the consideration of both transmitting more compact data for lowering transmission energy consumption and designing efficient feature detectors for lowering processing energy consumption. The proposed methods take into account all the constraints from the WVSNs, and therefore are more practical for real applications. The dissertation makes four major contributions as follows:

Feature-based image comparison method for WVSNs. The proposed comparison method that includes two light-weight feature detectors and 1 feature descriptor with reduced dimension, requires less data in transmission and for storage, facilitating the practical deployment of the technique in real-world WVSNs.

The linear unmixing based corner detector, LUC. The LUC corner detector is designed from a novel interpretation of corner points. Its efficiency has been proved by theoretical analysis. Its effectiveness has also been validated through thorough comparison with state-of-the-art feature detectors using three different image datasets.

The DWT-based corner detector. In multi-scale feature point detection, instead of using the scale-space from Gaussian function, the possibility of using the multi-scale wavelet analysis to construct the scale-space was explored. Although its performance in terms of repeatability is not as good as feature detectors from Gaussian function-based scale space, it demonstrated advantages in terms of computational efficiency.

The dimensionality-reduced SIFT descriptor. SIFT descriptors shows its advantages in representing local image patterns but its high dimensionality is not suitable for the WVSN environment. The proposed dimensionality reduction technique uses a set of pre-trained SIFT descriptor bases and projects the 128-dimensional SIFT descriptor onto the space spanned by this set of SIFT descriptor bases, requiring only 40-dimensional descriptors.

The proposed feature detectors and descriptor dimensionality reduction methods have been evaluated through thorough experiments on different image datasets. The feature-based image comparison framework for WVSNs has appeared in [Bai and Qi 2009, 2010]. The experimental evaluation for several feature detector and descriptors has appeared in [Bai and Qi 2010]. The DWT-based corner detector has appeared in [Bai and Qi 2011a]. The evaluation and comparison to other detectors of LUC has appeared in [Bai and Qi 2011b].

7.2 Directions for Future Research

For wireless sensor networks, a neighborhood is defined as a group of near-by sensors taking the same or similar measurements. Depending on the characteristics of sensing devices, the neighbor selection algorithm can be very different. In traditional sensor networks where scalar sensing is generally used, the sensors are omni-directional, and their measurements are mostly related to the distance between the event point and the sensor, which implies that the closer two sensors stand, the more similar their measurements are. That is, the neighborhood is determined by distances between sensors only.

On the other hand, visual sensors are usually directional. Therefore, their measurement similarity depends not only on their distance, but also on their orientation. For example, two surveillance cameras on the two sides of a wall pointing at opposite directions are geographically close, but they are not capturing the same scene and hence should not be defined as neighbors. In a WVSN, the **semantic neighbor** should satisfy *two* conditions, geographically close and with overlapped Field of Views (FOVs). The second condition essentially imposes the orientation constraint. For clarity, "g-neighbors" is used to refer to neighbors that only require geographical closeness and "s-neighbors" to refer to neighbors that satisfy both conditions.

The feature-based image comparison provides an efficient and effective approach for finding out the sensor position and orientation information, as well as solving sensor collaboration problems. By identifying the sensors having similar measurements, sensors can be organized into different semantic neighborhoods (clusters). Then the sensors in each s-neighborhood can collaborate in two different ways. Firstly, a carefully designed sleep-wakeup schedule can be applied so that the information redundancy within each cluster can be lowered, therefore prolong the network lifetime. Secondly, the information redundancy can be utilized to enhance the surveillance, for example, reconstructing super-resolution images.

7.2.1 Redundancy Removal for WVSNs

Neighbor selection in WSN can be used to prolong the network lifetime by arranging an efficient sleep-wakeup selection among neighbor nodes [Blough and Santi 2002]. The idea is to put some nodes within the neighborhood into the sleep mode for energy conservation when its neighbors are capable of taking measurements. Chen et al. introduced a power saving technique named SPAN which assumes every node is aware of its active neighbors

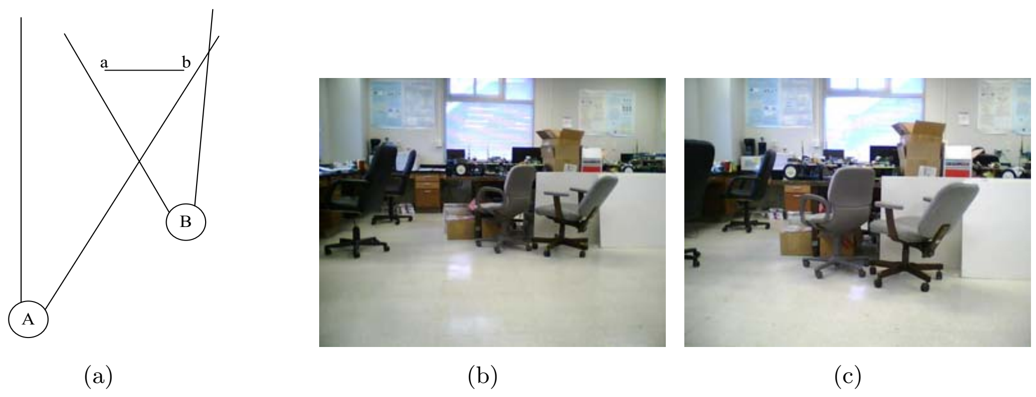


Figure 7.1: Overlapped FOV between smart cameras indicates information redundancy. (a) Two cameras have an overlapped FOV; (b) Image from camera A; (c) Image from camera B.

within a 2-hop distance and exchanges this active-neighbor information with its peers [Chen et al. 2002]. All nodes can thus use this to decide which node to turn off. Ye et al. proposed a Probing Environment and Adaptive Sleeping (PEAS) protocol that put the node into the sleep mode as long as possible if there is a nearby active node within its neighborhood [Ye et al. 2003]. Younis et al. presented a Hybrid Energy-Efficient Distributed (HEED) clustering protocol to organize the ad-hoc network in clusters and select cluster head based on node's power level and proximity to its neighbors [Younis and Fahmy 2004]. Among all these studies, the neighborhood information is regarded as trivial since it is only determined by the distance between node pairs. Neighbor selection in WVSNs is more difficult due to the directional sensing property of visual sensors.

The random and dense deployment of visual sensors implies the information redundancy within WVSNs. Figure 7.1 illustrates this scenario, where cameras A and B are facing a similar direction and their FOVs overlap. Thus the same scene (indicated by the line segment "ab") will appear in images captured by both A and B, with a slight change in scale and the angle of view. If the object of interest is within this scene, only one image from either A or B would be enough to meet the surveillance purpose. If the task is information retrieval or in-network image processing, transmitting/processing both images will not gain more information but would result in extra power consumption. Figure 7.1 is only a two-camera case. With the increase of the number of cameras in the network, this situation will become more severe.

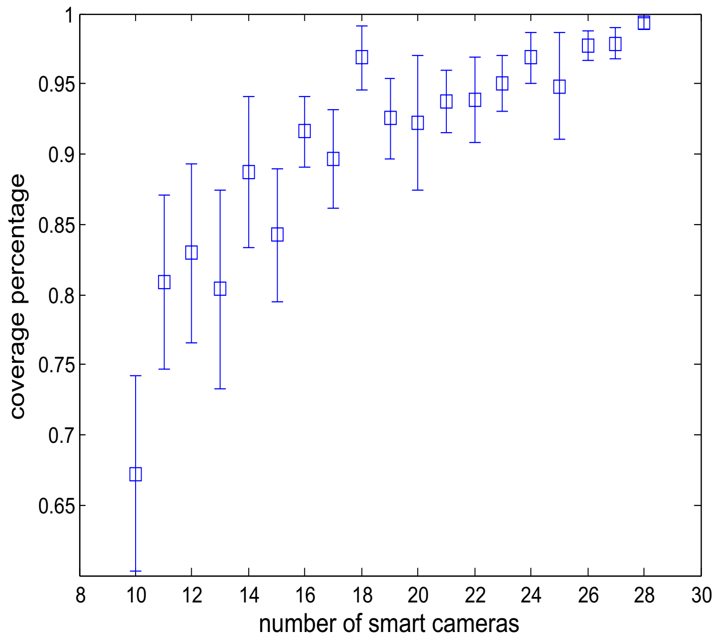
To ensure a k-coverage in WVSNs, the node density has to increase significantly, where k-coverage means any spot in the surveillance area should be covered by at least k visual sensors. Even with a 1-coverage WVSN, the information redundancy is already quite high. To demonstrate this phenomenon, a 100 pixel by 100 pixel square map with cameras randomly positioned at each pixel with randomly designated orientations is simulated. The number of smart cameras is gradually increased to attain a 1-coverage. Figure 7.2 shows the percentage of the covered area of the map and the average number of s-neighbors per node as a function of the number of smart cameras deployed. Camera B is defined as the s-neighbor of camera A if the size of their overlapped FOV constitutes at least 10% of the FOV of camera A. The results are averaged over 10 runs of the simulation with vertical bars indicating the variance. Figure 7.2 (a) shows that at least 21 smart cameras should be deployed in order to get a 1-coverage for 90% of the entire map area. But Fig. 7.2 (b) shows that when there are 21 smart cameras, on average, the FOV of every smart camera will overlap with those of other 5 cameras. To achieve a higher coverage rate, say more than 95%, then at least 27 cameras are needed and at the same time, there will be more than 6 s-neighbors for every camera on average.

On the other hand, by carefully designing a sleep-wakeup schedule, a significant number of visual sensors can be put into sleep mode without affecting the coverage requirement seriously. For example, on a 100×100 square unit area, randomly deploy 36 visual sensors to achieve a 1-coverage on 99.53% of the whole area. If wisely selected, only 16 sensors are required to reach a 1-coverage on 89.56% of the whole area. This implies a 56.1% saving of the number of sensors, or in other words, the energy consumption. This facts reveals the great potential of applying the feature-based image comparison method for selecting s-neighborhoods and implementing redundancy removal strategy.

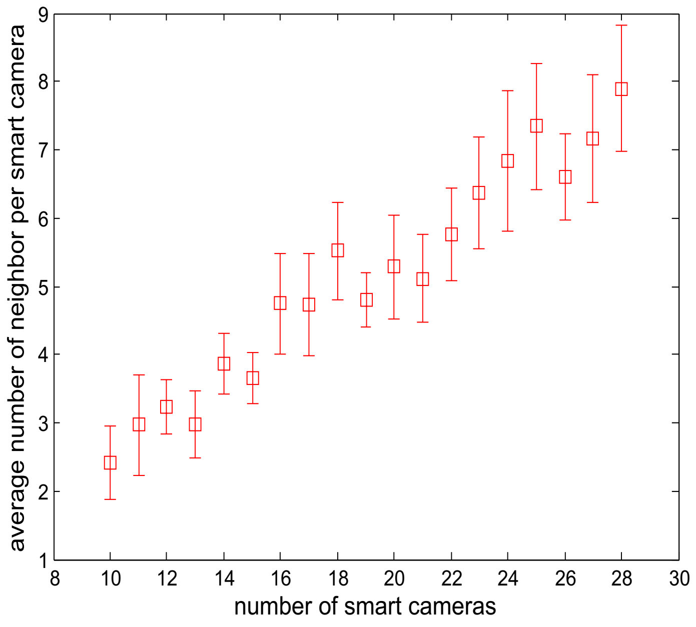
7.2.2 Super-resolution Image Reconstruction in WVSNs

Another direction of the future research work could be to exploit the potential of improving the surveillance quality from the information redundancy in each s-neighborhood.

The spatial resolution at which a WVSN can provide determines what application can be implemented. For example, Fig. 7.3 (a) shows an example scene from a surveillance



(a) Coverage area as a function of the number of cameras



(b) Semantic neighbors per node as a function of the number of cameras

Figure 7.2: Coverage and semantic neighbors as a function of the number of cameras.

(a) Scene (b) Low resolution (c) High resolution

Figure 7.3: Resolution determines feasible image processing (adapted from [Kansal et al. 2006]).

camera, and a small portion of this scene, specifically, the portion covering a vehicle at the stop line is shown in Fig. 7.3 (b). At this current resolution, the images captured can be used for vehicle detection and counting. But images at this resolution is not sufficient for applications like license plate recognition. Fig. 7.3 (c) shows the same portion from the scene as that in Fig. 7.3 (b) but at a much higher resolution. Such a resolution could enable applications such as vehicle recognition. This illustrative example shows how the resolution of the captured images from a WVSN can determine the applications can be implemented on that WVSN.

The technique of building a higher resolution image from a set of lower resolution images (or even a single lower resolution image) is named super-resolution image reconstruction [Park et al. 2003]. Among the many approach for super-resolution image reconstruction, there is an example-based method [Freeman et al. 2002; Glasner et al. 2009], which utilizes the correspondences between low and high resolution image patches. These correspondences are acquired by either the training result from a dataset or from the low resolution image itself.

Within each s-neighborhood of a WVSN, a training dataset can easily be built for finding the correspondences between low and high resolution image patches, because all images from an s-neighborhood are representing the same scene but from different angles and distances (scales). Therefore the example-based super-resolution image reconstruction can be applied in each s-neighborhood, if required.

7.3 Closing Remarks

This whole work was inspired by a plenary talk at the IEEE International Conference on Image Processing (ICIP) 2008, given by Professor Mark Levoy. In his talk, Prof. Levoy mentioned Hays and Efros's paper [Hays and Efros 2007], "Scene completion using millions of photographs", which demonstrated the success of using global image features in finding similar images from a large size of image database. However, when it comes to the WVSN environment, although facing the same mission of finding similar images, the conditions are a lot different. All images from a WVSN are similar, in some sense, and the objective is to find those taken from the same scene. Therefore, local image features are more useful, like corner points or blob-like points, as well as their corresponding feature descriptors. Thus the (local) feature-based image comparison method is developed for the WVSN environment.

Publications

Publications

1. **Y. Bai** and H. Qi,“An Efficient corner detector based on linear mixing model,” *IEEE Transactions on Image Processing*, in submission.

2. **Y. Bai** and H. Qi, “An Effective Approach to Corner Point Detection through Multiresolution Analysis,” *18th IEEE International Conference on Image Processing (ICIP)*, accepted, 2011.

3. **Y. Bai** and H. Qi, “Feature-based image comparison for semantic neighbor selection in resource-constrained visual sensor networks,” in *EURASIP Journal on Image and Video Processing*, special issue on Multicamera Information Processing: Acquisition, Collaboration, Interpretation, and Production, volume 2010, pp.1-11,2010.

4. **Y. Bai** and H. Qi, “Redundancy removal through semantic neighbor selection in Visual Sensor Networks,” in *Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, pp.1-8, 2009.

5. **Y. Bai** and H. Qi, “A new perspective on terahertz image reconstruction based on linear spectral unmixing,” *15th IEEE International Conference on Image Processing (ICIP)*, pp.2996-2999, 2008.

Bibliography

Bibliography

Akyildiz, I. F., Melodia, T., and Chowdhury, K. R. (2008). Wireless multimedia sensor networks: Applications and testbeds. *Proceedings of the IEEE*, 96:1588–1605.

Bai, Y. and Qi, H. (2009). Redundancy removal through semantic neighbor selection in visual sensor networks. *Third ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–8.

Bai, Y. and Qi, H. (2010). Feature-based image comparison for semantic neighbor selection in resource-constrained visual sensor networks. *Eurosip Journal on Image and Video Processing*, 2010:1–11.

Bai, Y. and Qi, H. (2011a). An effective approach to corner point detection through multiresolution analysis. *2011 International Conference on Image Processing (ICIP'11)*, accpeted.

Bai, Y. and Qi, H. (2011b). An efficient corner detector based on linear unmixing model. *IEEE Transactions on Image Processing*, in submission.

Baumberg, A. (2000). Reliable feature matching across widely separated views. *2000 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00)*, 1:1774–1781.

Bay, H., Ess, A., Tuyelaars, T., and Gool, L. V. (2008). Speeded up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359.

Bay, H., Tuytelaars, T., and Gool, L. V. (2006). Surf: Speeded-up robust features. *Proceedings of the 9th European Conference on Computer Vision, Springer LNCS*, 3951:404–417.

Beall, C. and Qi, H. (2006). Distributed self-deployment in visual sensor networks. *Control, Automation, Robotics and Vision, 9th International Conference on*, pages 1–6.

Blough, D. M. and Santi, P. (2002). Investigating upper bounds on network lifetime extension for cell-based energy conservation techniques in stationary ad hoc networks. *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, pages 183–192.

Bouchard, G. and Triggs, B. (2005). Hierarchical part-based visual object categorization. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:710–715.

Brown, M. and Lowe, D. (2002). Invariant features from interest point groups. *British Machine Vision Conference*, pages 656–665.

Carneiro, G. and Lowe, D. (2006). Sparse flexible models of local features. *In Proceedings of the European Conference on Computer Vision (ECCV)*, pages 29–43.

Chandrasekhar, V., Takacs, G., and Chen, D. (2009). Transform coding of image feature descriptors. *Proceedings of SPIE Conference on Visual Communication and Image Processing*, 7257(10).

Chen, B., Jamieson, K., Balakrishnan, H., and Morris, R. (2002). Span: An energy-efficient coordination algorithm for topology maintaenance in ad hoc wireless networks. *Wireless Networks*, 8(5):481–494.

Cichocki, A., Zdunek, R., and ichi Amari, S. (2006). New algorithms for non-negative matrix factorization in applications to blind source separation. *Proceedings of the 2006 International Conference on Acoustics, Speech and Signal Processing (ICASSP'06)*, pages 621–624.

Coxeter, H. S. M. (1969). *Introduction to Geometry.* New York: Wiley, second edition.

Crow, F. (1984). Summed-area tables for texture mapping. *Proceedings of SIGGRAPH*, 18(3):207–212.

Datta, R., Joshi, D., Li, J., and Wang, J. Z. (2008). Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2).

Dias, P., Kassim, A., and Srinivasan, V. (1995). A neural network based corner detection method. *Proceedings of IEEE International Conference on Neural Networks*, 4:2116–2120.

Dutta, A., Kar, A., and Chatterji, B. (2008). Corner detection algorithms for digital image in last three decades. *IETE Technical Review*, 25:123–133.

Faugeras, O. (1993). *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press.

Fauqueur, J., Kingsbury, N., and Anderson, R. (2006). Multiscale keypoint detection using the dual-tree complex wavelet transform. *IEEE International Conference on Image Processing*, pages 1625–1628.

Fergus, R., Perona, P., and Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2:264–271.

Fergus, R., Perona, P., and Zisserman, A. (2005). A sparse object category model for efficient learning and exhaustive recognition. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:380–387.

Freeman, W. T., Jones, T. R., and Pasztor, E. C. (2002). Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22:56–65.

Glasner, D., Bagon, S., and Irani, M. (2009). Super-resolution from a single image. *International Conference on Computer Vision (ICCV)*, pages 1–8.

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

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. *Proceedings of the 4th Alvey Vision Conference*, pages 77–116.

Hays, J. and Efros, A. A. (2007). Scene completion using millions of photographs. *ACM Transactions on Graphics (SIGGRAPH)*, 26(3):1–7.

Hoyer, P. O. (2002). Non-negative sparse coding. *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 557–565.

Hoyer, P. O. (2004). Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469.

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

Kansal, A., Kaiser, W., Pottie, G., Srivastava, M., and Sukhat, G. (2006). Virtual high-resolution for sensor networks. *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 43–56.

Kundur, D., Lin, C.-Y., and Lu, C. S. (2007). Visual sensor networks. *EURASIP Journal on Advances in Signal Processing.*

Langridge, D. (1982). Curve encoding and detection of discontinuities. *Computer Graphics Image Processing*, 20:58–71.

Lazebnik, S., Schmid, C., and Ponce, J. (2003). Affine-invariant local descriptors and neighborhood statistics for texture recognition. *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 649–655.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791.

Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13:556–562.

Lindeberg, T. (1994). Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, 21(2):224–270.

Lindeberg, T. (1998). Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30:79–116.

Lindeberg, T. and Garding, J. (1997). Shape-adapted smoothing in estimation of 3-d shape cues from affine distortions of local 2-d brightness structure. *Image and Vision Computing*, 15:415–434.

Lowe, D. (1999). Object recognition from local scale-invariant features. *Proceedings of the ICCV*.

Lowe, D. G. (2004). Distinctive image feature from scale-invariant keypoints. *International Journal of Compputer Vision*, pages 91–110.

Malik, J. and Perona, P. (1990). Preattentive texture discrimination with early vision mechanisms. *Journal of the Optical Society of America*, 7:923–932.

Manjunath, B. S., Ohm, J.-R., Vasudevan, V. V., and Yamada, A. (2001). Color and texture descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):703–715.

Medioni, G. and Yasumoto, Y. (1987). Corner detection and curve representation using cubic b-spline. *Computer Vision, Graphics and Image Processing*, pages 267–278.

Mikolajczyk, K. and Schmid, C. (2004). Scale and affine invariant interest point detectors. *Int. J. Comput. Vision*, 60(1):63–86.

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

Mokhtarian, F. and Suomela, R. (1998). Robust image corner detection through curvature scale-space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1376–1381.

Moravec, H. (1980). Obstacle avoidance and navigation in the real world by a seeing robot rover. *tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University and doctoral dissertation, Stanford University.*

Nene, S., Nayar, S., and Murase, H. (1996). Columbia object image library (coil-100). *Technical report*, (CUCS-006-96).

Obraczka, K., Manduchi, R., and Garcia-Luna-Aveces, J. J. (2002). Managing the information flow in visual sensor networks. *Proceedings of the Fifth International Symposium on Wireless Personal Multimedia Communications*, 3:1177–1181.

Oliva, A. and Torralba, A. (2006). Building the gist of a scene: The role of global image features in recognition. *Progress in Brain Research: Visual perception*, pages 23–36.

Oxford (2002). http://www.robots.ox.ac.uk/∼vgg/research/affine/.

Park, S. C., Park, M. K., and Kang, M. G. (2003). Super-resolution image reconstruction. *IEEE Signal Processing Magazine*, 20(3):21–36.

Parks, D. (2006). http://www.cim.mcgill.ca/∼dparks/CornerDetector/harris.htm.

Rinner, B. and Wolf, W. (2008). A bright future for distributed smart cameras. *Proceedings of the IEEE*, 96:1562–1564.

Rosten, E. and Drummond, T. (2005). Fusing points and lines for high performance tracking. *IEEE International Conference on Computer Vision*, 2:1508–1515.

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. *European Conference on Computer Vision*, 1:430–443.

Sankaranarayanan, A. C., Veeraraghavan, A., and Chellappa, R. (2008). Object detection, tracking and recognition for multiple smart cameras. *Proceesdings of the IEEE*, 96:1606–1624.

Schmid, C., Mohr, R., and Bauckhage, C. (2000). Evaluation of interest point detectors. *International Journal of Computer Vision*, 37:151–172.

Semple, J. and Kneebone, G. (1952). *Algebraic Projective Geometry.* Oxford Science Publication.

Shi, J. and Tomasi, C. (1994). Good features to track. *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593–600.

Smith, S. and Brady, J. (1997). Susan - a new approach to low level image processing. *International Journal of Computer Vision*, 23:45–78.

Trajkovic, M. and Hedley, M. (1998). Fast corner detection. *Image and Vision Computing*, 16:75–87.

Tuytelaars, T. and Mikolajczyk, K. (2007). Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3:177–280.

Unser, M. (1995). Texture classification and segmentation using wavelet frames. *IEEE Transactions of Image Processing*, 4(11):1549–1560.

Vasconcelos, N. (2007). From pixels to semantic spaces: Advances in content-based image retrieval. *IEEE Computer*, 40(7):20–26.

Viola, P. A. and Jones, M. J. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of CVPR*, pages 511–518.

Wang, H. and Brady, M. (1995). Real-time corner detection algorithm for motion estimation. *Image and Vision Computing*, 13:695–703.

Wikipedia (2010). http://en.wikipedia.org/wiki/Corner_detection.

Wilson, P. (2008). Curved spaces: from classical geometries to elementary differential geometry. *Cambridge: Cambridge University Press.*

Ye, F., Zhong, G., Cheng, J., Lu, S., and Zhang, L. (2003). Peas: A robust energy conserving protocol for long-lived sensor networks. *Proceedings of the International Conference on Distributed Computing Systems*, pages 28–37.

Younis, O. and Fahmy, S. (2004). Heed: A hybrid, energy-efficient, distributed clustering approach for ad-hoc sensor networks. *IEEE Transactions on Mobile Computing*, 3(4):366–379.

Zenzo, S. D. (1986). A note on the gradient of a multi-image. *Computer Vision, Graphics, and Image Processing*, 33:116–125.

Zhang, Y. and Fang, Y. (2007). A nmf algorithm for blind separation of uncorrelated signals. *Proceedings of the 2007 International Conference on Wavelet Analysis and Pattern Recognition*, pages 999–1003.

Vita

Yang Bai was born in Zhengzhou, P. R. China. He started his engineering career in Beijing Institute of Technology, Beijing in 2000 where he earned a B.E. degree in Electrical Engineering, 2004 and a M.S. degree in Electrical Engineering, 2006 from the Department of Electrical Engineering. His master thesis titled "Particle Filter and Its Application in Image Restoration" received the Best Thesis Award in Beijing Institute of Technology. To pursue a Ph.D., he enrolled the graduate program in the department of Electrical Engineering and Computer Science at the University of Tennessee, Knoxville in fall, 2006. He worked at the Advanced Imaging & Collaborative Information Processing (AICIP) Lab as a graduate research assistant from fall, 2007. During his Ph.D. study, his research focused on computer vision, digital image processing, and wireless visual sensor networks. He has been involved in the NSF project, "NeTS: Small: Distributed Solutions to Smart Camera Networks". In the summer of 2009, he worked as an engineering intern at the Aldis Inc, a start-up company in the field of Intelligent Transportation Systems (ITS). He will complete his Doctor of Philosophy degree in Computer Engineering in Spring 2011.