Low-Level Image Features and Navigation Systems for Visually Impaired People

Nadia Kanwal

A thesis submitted for the degree of
Doctor of Philosophy

School of Computer Science and Electronic Engineering
University of Essex

September, 2013

If you would not be forgotten
As soon as you are dead and rotten,
Either write things worthy reading,
Or do things worth the writing.

Benjamin Franklin

To

my father (late), my husband, my son

and

my aunt, Moni Khala (late)

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to Lahore College for Women University, Lahore, Pakistan for providing me with the opportunity to undertake research on a topic so close to my heart. I would like also to thank the University of Essex, Colchester, UK at which I was able to carry out this research.

I am indebted to my advisor, Dr. Adrian F. Clark, for his continuous guidance and help. This study could not have been completed without his timely guidance and balanced approach. I am also indebted to Dr. Farhat Saleemi and Dr. Muhammad Abuzar Fahiem who not only encouraged me but always supported me to do a PHD.

A very big thanks goes to my colleagues Erkan Bostanci and Dr. Shoaib Ehsan; I highly appreciate their help, their advice and thought-provoking discussions which led to invaluable improvements to this thesis.

I would not like to forget that none of the experimental work and testing of the navigation system was possible without the technical support of Mr. Robin Dowling. I am so grateful to Mr. Keith Currie, for trying the proposed navigation system without even using his white cane. I am also thankful to Mr. Erkan Bostanci for testing the system as a blindfolded user, a difficult job.

Besides the academic support, I received great support from my family, My deepest thanks go to them all starting with my husband (Mr. Tahseen Muzaffar) for his never-ending support throughout this research. Mrs Muzafar-ul-Haq, my mother-in-law; Mrs. Shazia Ahmed; Mr. Waseem Muzaffar; Mrs. Zoobi Waseem who took very good care of my family during my absence and gave me the peace of mind I needed to complete this work; and my mother who was there when I needed her the most. Special thanks also go to Mr. Naveed Muzaffar and his family for taking care of me during my lonely and difficult times; and finally, my brother, Mr. Mohsin Rafiq, who was always just one call away from me.

I am grateful to my son Hashir Muzaffar for being a very nice baby and a source of energy during the last one and a half years of this research. I owe my every achievement to him and his father.

I wish to extend my gratitute to my friends: Dr. Buthaina Shaheen, Mrs. Betul Bostanci, Mrs. Maria Shoaib, Mrs Beenish Jehanzaib and Mrs. Woroud Melhem, for their lovely companionship and for the cheerful moments I spent with them.

ABSTRACT

This thesis is concerned with the development of a computer-aided autonomous navigation system for a visually-impaired person. The system is intended to work in both indoor and outdoor locations and is based around the use of camera systems and computer vision.

Following a review of the literature to identify previous work in navigation systems for the blind, the location of accurate image features is shown to be a vital importance for a vision based navigation system. There are many operators that identify image features and it is shown that existing methods for identifying which has the best performance are inconsistent. A statistically valid evaluation and comparison methodology is established, centered around the use of McNemar's test and ANOVA.

It is shown that these statistical tests require a larger number of test images than is commonly used in the literature to establish which feature operators perform best. A ranking of feature operators is produced based on this rigorous statistical approach and compared with similar rankings in the literature.

Corner detectors are especially useful for a navigation system because they identify the boundaries of obstacles. However, the results from our testing suggest that the internal angle of a corner is one factor in determining whether a corner is detected correctly. Hence an in-depth study of angular sensitivity of corners is presented. This leads to the development of a pair of descriptors, known as CMIE and AMIE, which describe corners. Experiments show that these descriptors are able to be computed at video rate and are effective at matching corners in successive frames of video sequences.

Finally, a complete navigation system is presented. This makes use of both a conventional colour camera and a depth sensor combined in a device known as the Microsoft Kinect. It is shown that the system performs robustly in both indoor and outdoor environments, giving audio feedback to the user when an obstacle is detected. Audio instructions for obstacle avoidance are also given. Testing of the system by both blindfolded and blind users demonstrates its effectiveness.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

P Precision

R Recall

2D 2-Dimensional

3D 3-Dimensional

AMIE Angle, Mean Intensity and Entropy

ANOVA Analysis of Variance

AUC Area Under the Curve

BRIEF Binary Robust Independent Elementary Features

CMIE Circular Mean Intensity and Entropy

CPU Central Processing Unit

CSS Curvature Scale Space

DoG Difference of Gaussian

ETA Electronic Travelling Aid

FAST Features form Accelerated Segment Test

FN False Negative

FP False Positive

FPR False Positive Rate

FPr	False Positive Ratio
GLC	Global and Local Curvature Points
GLOH	Gradient Location and Orientation Histogram
GPS	Global Positioning System
H&S	Harris & Stephen's
Haraff	Harris Affine
Harlap	Harris Laplace
Hesaff	Hessian Affine
Heslap	Hessian Laplace
KLT	Kanada-Lucas-Tomasi Corner Detector
LoG	Laplacian of Gaussian
PCA	Principal Component Analysis
PETS	Performance Evaluation of Tracking and Surveillance
PR	Precision-Recall
QQ	Quantile-Quantile Plot
RANSAC	Random Sample Consenses
ROC	Reciever Operating Characteristics Curve
S&T	Shi & Tomasi
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SS	Sensitivity-Specificity
SURF	Speeded Up Robust Features
SUSAN	Smallest Univalue Segment Assimilating Nucleus
TN	True Negative
TP	True Positive
TPR	True Positive Rate
TPr	True Positive Ratio
XOR	Exclusive OR

CHAPTER 1

INTRODUCTION

Sight is arguably the most important human sense; certainly a significant proportion of the brain is dedicated to processing visual information. Electromagnetic radiation is converted into electrical signals at the retina, the light sensitive cells on the rear surface of the eye, and a series of regions in the brain convert the electrical signals into what we term “seeing” — understanding the content. Humans sight allows one to differentiate objects through colour, shape, distance and so on. Because of this sense of sight, humans are able to move freely in any environment, indoor or outdoor, recognizing paths, objects, obstacles, animals and other humans around them. Blindness, the lack of visual perception [1], is therefore a serious disability for a person who needs to move around their environment, or find objects within it.

A number of solutions have been developed to ameliorate the effects of blindness, the most common one being guide dogs and white canes. Guide dogs are interactive and responsive, so they are preferable as a visual aid for blind people, but white canes are more widespread because of their low cost, light weight and portability. For a number of years, scientists have been trying to develop aids that can make blind people more independent and aware of their surroundings. Computer-based automatic navigation tools are one example of this, motivated by the increasing miniaturization of electronics and the improvement in processing power and sensing capabilities. Key to this is computer vision, the discipline that attempts to understand the content of visual scenes.

The purpose of the research described in this thesis is to develop a navigation aid for blind people based on computer vision. Vision is particularly attractive for this because it is a passive sensing modality, unlike e.g. radar and sonar, and sensors are low-cost and physically unobtrusive; this is considered in more detail in the next section.

1.1 Research objectives

An autonomous navigation system for a fully or partially sighted person requires enough information from the surroundings to detect and avoid obstacles. It is especially important to identify head-level obstacles such as walls, cupboards etc. To acquire this information, a variety of sensors could be used including sonar, laser stripers and cameras [2]. Sonar and laser stripers are able to provide the distance to objects and therefore have been previously used in developing automated navigation solutions for robots and humans. However, all of these sensors have limitations, such as the poor angular resolution of sonar because of its wide beam-width [3] and the cost of laser stripers. There is no clear winner in the choice of sensor; but the cheapness, small size, and ease of integration of cameras make them attractive if one can overcome the difficulties of segmenting objects etc.

One of the main purposes of this research is to establish a series of processing stages that can implement a navigation system using imagery captured from a camera that is both robust and has good accuracy. In particular, recent advances in local invariant feature detection and matching are attractive for developing navigation systems. However, there are many choices of feature detectors, as a result of which several researchers have carried out performance evaluation studies [4–11]. The results of these studies are not consistent in terms of the ranking of algorithms, which makes selection of a single feature detector for a navigation system difficult. The precise reason for this is unclear, but different datasets and evaluation frameworks are certainly contributing factors. Few of these evaluations are based on statistical methods, so this thesis presents a statistically valid evaluation of the performance of these local feature detectors with a view to establishing whether any feature detector is genuinely better than the others. The methodology used in doing this is, however, quite general.

During the course of this research, Microsoft released the Kinect sensor, principally as an input device for gaming. The Kinect is able to capture both a conventional colour image and corresponding depth values; it does so by projecting an infrared light pattern into the environment and detecting where it appears on objects. The Kinect is useful for tracking motion, especially human motion. A Software Develpoment Kit (SDK) was released in November 2010 (for Windows 7 in June 16, 2011); the research community has adopted the Kinect with some enthusiasm and it has been applied to a wide variety of applications [12–14], including navigation for blind people. Consequently, this research, which initially used only 2-Dimensional (2D) images from cameras, also makes use of the Kinect sensor in a hybrid 2D-plus-depth system — this overcomes some of the shortcomings of a purely depth system, as later chapters will show.

1.2 Contributions

This research has made a number of contributions, highlighted in the following series of bullet points. A number of publications in the open literature have been made and these are listed in the following section.

- Vision research has developed a substantial number of algorithms for tasks such as matching, segmentation, stitching, tracking and navigation. However, not all of these algorithms are equally accurate and reliable. This thesis reviews the strengths and weaknesses of existing performance evaluation measures, including Reciever Operating Characteristics Curve (ROC), Precision-Recall (PR), F-measure etc, and explores the use of statistical tests such as McNemar's test and Analysis of Variance (ANOVA) as a more principled alternative.

- The size and content of datasets can clearly affect algorithms' performances. These factors are explored using two datasets in widespread use in evaluation studies. It is clear that the evaluation studies reported in the literature are affected by both these factors [15].

- Corner points are particularly attractive for a navigation system because they identify the object boundaries. Corner detection has, of course, been well researched and a systematic review of the angular sensitivity of corner detectors has been carried out using the statistical approach established in this research. It is considered appropriate because of the sensitivity of feature detectors explored during their performance evaluation for matching images under geometric transformations [16, 17].

- Although many corner *detectors* exist, there are few corner *descriptors*. Descriptors are important because they allow corresponding corners in successive frames of a motion sequence to be established. Here, two new descriptors are proposed specifically for corner points. They make use of the corner's angle and circular arcs around the corner for matching and are named Circular Mean Intensity and Entropy (CMIE) and Angle, Mean Intensity and Entropy (AMIE). The performances of these descriptors is determined and compared with the well-known Binary Robust Independent Elementary Features (BRIEF) descriptor. The speed of AMIE and CMIE is important for video rate operation in a visual navigation system [18].

- A complete navigation system, based around corners and depth values from the Kinect sensor, has been developed. Obstacles are found in images from a camera using corner detection, while input from the depth sensor provides the corresponding distance. The combination is both efficient and robust. The system not only identifies hurdles but also suggests a safe path (if available) to the left or right side and tells the user to stop, move left or move right. The system has been tested in real time by both blindfolded and blind people at different indoor and outdoor locations, demonstrating that it operates as required [19].

1.3 Publications

1. **Kanwal, N.**, Bostanci, E., and Clark, A. F, "Matching Corners Using the Informative Arc," Revision submitted to IET Computer Vision.
2. Bostanci, E.,Bostanci, B., Clark, A. F. and **Kanwal, N.**," A Fuzzy-adaptive Multiple-motion Model for GPS-IMU Integration", submitted to IET Electronics Letters.
3. Bostanci, E., **Kanwal, N.** and Clark, A. F., "Spatial Statistics of Image Features for Performance Comparison," Revision submitted to IEEE Transactions on Image Processing.

4. Bostanci, E., **Kanwal, N.** and Clark, A. F., “Augmented Reality Applications for Cultural Heritage Using Kinect", submitted to Transactions on Edutainment, Springer.

5. **Kanwal, N.**, Bostanci, E., and Clark, A. F, “ Kinect Aided Navigation System for Visually Impaired People,” Proceedings of the Workshop on Recognition and Action for Scene Understanding (REACTS 2013), York, UK, 30-31 August 2013.

6. Bostanci, E., **Kanwal, N.**, Ehsan, S. and Clark, Adrian F , “User Tracking Methods for Augmented Reality," International Journal of Computer Theory and Engineering, 5 (1). pp. 93-98. ISSN 1793-8201, 2013.

7. Bostanci, E., **Kanwal, N.** and Clark, A. F, “Kinect Derived Augmentation of the Real World for Cultural Heritage,” Proceedings of UKSIM’13, Cambridge, UK.

8. Bostanci, E., **Kanwal, N.** and Clark, A. F., “Extracting Planar Features From Kinect Sensor,” Proceedings of CEEC’12, Colchester, UK

9. Bostanci, E., Clark A. F., **Kanwal, N.**, “Vision-based User Tracking for Outdoor Augmented Reality,” Proceedings of the 17th IEEE Symposium on Computers and Communication, 2012, Cappadocia, Turkey.

10. Ehsan, S., **Kanwal, N.**, Clark, A. F. and McDonald-Maier, K. D., “An Algorithm for the Contextual Adaption of SURF Octave Selection with Good Matching Performance: Best Octaves,” IEEE Transactions on Image Processing, 21(1), 297–304, 2012 (5-Year Impact Factor : 4.139).

11. **Kanwal, N.**, Bostanci, E., and Clark, A. F., “Describing Corners using the Angle, Mean Intensity and Entropy of Informative Arcs” Electronic Letters, 48(4), 209–210, 2012.

12. Bostanci, E., **Kanwal, N.** and Clark, A.F, “Feature Coverage for Better Ho-

mography Estimation: An Application to Image Stitching," Proceedings of IWSSIP 2012, Vienna.

13. **Kanwal, N.**, Ehsan, S., Bostanci, E. and Clark, A. F., "Evaluating the Angular Sensitivity of Corner Detectors," Proceedings of the IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems (VECIMS), Ottawa, Canada, September 2011.

14. Ehsan, S., Clark, A. F., Cheung, W. M., Bais, A. M., Menzat, B. I., **Kanwal, N.** and McDonald-Maier, K. D., "Memory-Efficient Design Strategy for a Parallel Embedded Integral Image Computation Engine," Proceedings of the 15th Irish Machine Vision and Image Processing Conference (IMVIP), Dublin, Ireland, September 2011.

15. **Kanwal, N.**, Ehsan, S., Bostanci, E. and Clark, A. F., "A Statistical Approach for Comparing the Performances of Corner Detectors," Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, B.C., Canada, August 23-26, 2011.

16. **Kanwal, N.**, Ehsan, S. and Clark, A. F., "Are Performance Differences of Interest Operators Statistically Significant?", In Proceedings of Computer Analysis of Images and Patterns, 429–436, Springer, August 2011.

17. Ehsan, S., **Kanwal, N.**, Clark, A. F. and McDonald-Maier, K. D., "Measuring the Coverage of Interest Point Detectors," Proceedings of the 8th International Conference on Image Analysis and Recognition (ICIAR), Burnaby, British Columbia, Canada, June 2011.

18. Ehsan, S., **Kanwal, N.**, Bostanci, E., Clark, A. F. and McDonald-Maier, K. D., "Analysis of Interest Point Distribution in SURF Octaves," Proceedings of the 3rd International Conference on Machine Vision (ICMV), Hong Kong, December 2010.

19. Bostanci, E., **Kanwal, N.**, Ehsan, S. and Clark, A. F., "Tracking Methods for Augmented Reality," Proceedings of the 3rd International Conference on Machine Vision (ICMV), Hong Kong, December 2010.

20. Ehsan, S., **Kanwal, N.**, Clark, A. F. and McDonald-Maier, K. D., "Improved Repeatability Measures for Evaluating Performance of Feature Detectors," Electronics Letters, 46(14), 998–1000, July 2010. (also selected as a featured article)

It is anticipated that a further journal submission, describing the complete navigation system and its assessment will also be submitted.

1.4 Thesis outline

Chapter 2 reviews existing mobility aids and electronic travel aids for visually-impaired people. There is also an in-depth literature review about vision-based navigation systems for blind people. As image matching is fundamental to any vision-based navigation system, state-of the-art feature operators and matching schemes are also reviewed in the chapter. Furthermore, performance evaluation techniques commonly used in vision research are reviewed.

Chapter 3 examines performance characterization methods. A number of existing performance metrics are presented and are shown to yield inconsistent results. A statistically-based approach, using McNemar's test and ANOVA in a null hypothesis framework, is presented and applied.

Chapter 4 explores the discrepancy in evaluation results as a consequence of database content, by comparing the performances on one well-established database with another of similar size. The effects of database size are also explored by employing subsets of a larger database. As a consequence of this analysis some guidelines regarding data size are defined to be used for evaluating vision algorithms.

Chapter 5 presents corner detection algorithms and examines their angular sensitivity. Corner points are image features which are inherently rotation invariant so corner detection algorithms should be able to detect all corner points having the same angle but different orientation. To establish this, synthetic and real image data consisting of geometric shapes were developed and the performances of corner detectors is assessed using McNemar's test. The overall performance of corner detectors was also assessed and inconsistencies with the angular sensitivity experiments identified. Moreover, the complementarity of corner detectors (i.e., combining algorithms with different principles of operations) is explored to ascertain whether this improves overall performance.

Chapter 6 proposes two new descriptors that achieve real-time matching of corners. CMIE describes the entire neighbourhood around a corner point using the mean intensity and entropy of circular arcs around it while AMIE employs the "informative region" enclosed between the intersection of two edges. Matching corner points using AMIE is particularly efficient because the angle between the edges is used to identify only likely close matches.

Chapter 7 describes the complete navigation system. The system scans the area in front of the user for obstacles; if it finds one, it looks for a possible safe path to the left or right, issuing verbal warnings using a speech synthesiser. There are checks in the algorithms to avoid undue warnings and false obstacle detection. The system has been tested in both indoor and outdoor environments and the results demonstrate the effectiveness of the system.

Finally, chapter 8 presents the conclusions drawn from this research and makes suggestions for further work.

CHAPTER 2

FROM WHITE CANE TO AUTOMATED NAVIGATION SYSTEMS

2.1 Introduction

The focus of research into navigation systems for the visually impaired to date has mainly focussed on allowing a person to negotiate their way through an unknown, dynamically-changing environment through obstacle detection, environment mapping, and path, direction and pose estimation. Despite all the efforts, a navigation system for an autonomous robot or for a partially/fully blind person is still an open-ended problem.

Existing mobility aids for visually impaired people, the most common of which are white canes and guide dogs, are discussed in section 2.2. These have been in use for a very long time but do not give full freedom of movement to a blind person. Therefore, research is in progress to develop automated tools for this purpose. A number of active and passive sensors, developed as range finders in robotics, can be used as the basis of travel aids for blind people. Distance measurement of obstacles for safe navigation is what is required and can be achieved using these sensors. Different software and hardware solutions using these active and passive sensors for this purpose have been proposed and are reviewed in section 2.3.

The camera, being a cheap and small vision sensor, is gaining more attention as the basis of autonomous navigation systems for humans and robots. Section 7.3 describes some of the proposed navigation systems using cameras. These systems work by matching captured images, for which local image feature based matching has shown significant progress. Feature detection and matching play a vital role in understanding image content and motion detection. Feature *operators* include the *detection* and *description* of image features. Section 2.4.1 illustrates some well-known scale- and rotation-invariant feature extraction techniques (also called feature operators), and how these have been used in tracking and navigation applications.

In the presence of a large number of feature extraction and matching techniques, performance evaluation becomes pivotal to distinguish which are the more effective. Section 2.5 reviews widely-used methods for evaluating the performance of vision algorithms. These include ROC, PR, Sensitivity-Specificity (SS), Accuracy, and F-measure. The discussion also identifies the inconsistency of results reported in the literature using these performance metrics.

Figure 2.1: White Cane: A travelling aid for visually impaired people

2.2 Mobility aids for a visually-impaired person

A person with partial or no sight can walk or travel from one place to another dependently or independently. In the former situation, the user needs assistance to be able to walk in an unknown environment, while in the latter the user can confidently walk around being familiar with the environment and landscape such as his or her own house, or another familiar place [20]. To navigate in an unknown environment, the conventional aids are white canes and guide dogs. A white cane (shown in Figure 2.1) is designed to find kerbs, steps and obstacles that may be in the way [21]. Although it is a cheap and lightweight device, and hence is very popular, its users have to rely on their own judgment and perception because the cane can only help them sense the obstacle. Similarly, the cane is not very helpful in detecting head-level obstacles [22, 23].

On the other hand, a guide dog is a very handy travelling aid for a visually impaired person, but it needs to be highly trained (Figure 2.2 shows a trained

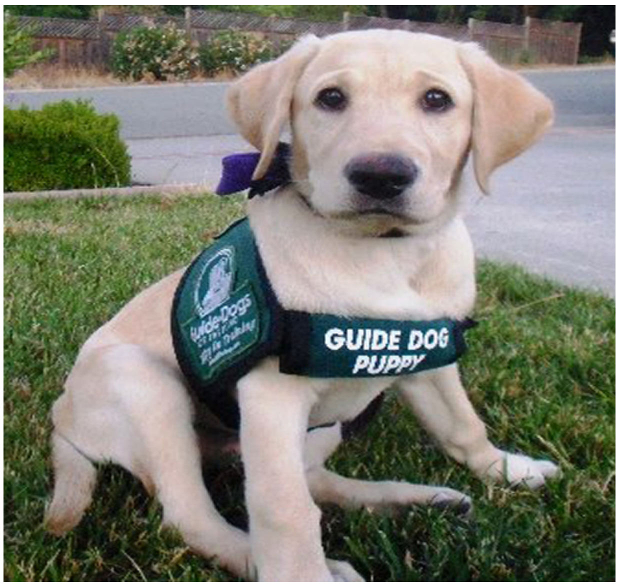


Figure 2.2: A Guide Dog: travelling aid for visually impaired people [24]

guide dog puppy). Dogs are known for their intelligence, efficiency and loyalty, and so are considered reliable as guide dogs for blind people. These dogs usually respond quickly to approaching dangers; however, they are less commonly encountered because of their high cost, which ranges between $12000 and $20000. The dogs themselves need a carer [25]. Despite the support a blind person can get from a white cane or a guide dog, their use can be inconvenient and physically tiring [23]. This helps explain why scientists are exploring less costly automated solutions using high-speed hardware architectures and efficient algorithms.

2.3 Vision-based navigation systems

Autonomous robot navigation and navigation systems for visually impaired people are two closely related fields of research which benefit from each other in terms of both software and hardware developments. There have been quite a few attempts to develop vision based solutions for blind and partially-sighted

Table 2.1: Some range sensors commonly used in robotics

Sensors	Type	Technology	Range
IR sensor	Active	Infrared reflector [26]	3-30cm
Optic sensors	Active	Laser [27]	2cm-2.4m
Sonar	Active	Ultrasonic waves [28]	2cm-1m
Camera	Passive	light waves	1cm-40m depth of field
Stereo pair camera	Passive	light waves	1cm-40m depth of field

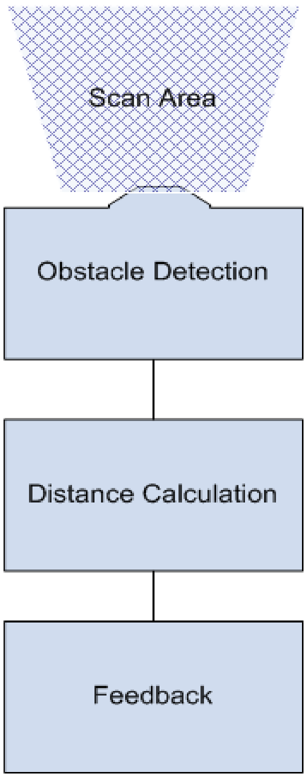


Figure 2.3: A navigation systems using an active sensor scanning frontal area to find obstacles and their distances from user/sensor

people using range sensors, which are also used in robotics [29–31], and which are usually referred as *Electronic Travelling Aid (ETA)*.

There are two types of sensors which are used for autonomous robot navigation that can also be used for developing ETA for blind people. The first type of sensors are termed **Active** sensors, whereas the second type are **Passive** [32]. The distinction between sensors is based on the source of energy used: active sensors project some kind of signal into the environment whereas passive sensors use energy from other sources. Table 2.1 gives a brief overview of some commonly-used sensors and their range for distance estimation.

To develop an automatic navigation system, detecting obstacles and calculating of their distances are the main requirements. Figure 2.3 shows a block diagram of a navigation system based on the input of an active sensor for object detection and its distance estimation. Commonly-used methods for sensing obstacles and finding their distances are time-of-flight, triangulation and phase shift. Because all active sensors use their own energy to illuminate or scan the area in front of them (such as infra-red light, laser, or sound waves), measuring the distance of an obstacle using time-of-flight is the easiest way. This involves measuring the time from the transmission of a pulse to the reception of its echo (the 'round-trip' time) [33]. Conversely, in the triangulation method the distance is measured from the location of the sensor and its angle, which can be measured from the reflection of a laser or ultrasonic wave [34].

The phase shift method is commonly used with high frequency waves such as laser. A laser beam is sent to a target, some reflected light is monitored, and the phase of the power modulation is compared with that of the transmitted light. The phase shift obtained is $2\pi \times$ time-of-flight $\times$ modulation frequency [35]. The output of these systems is reliable and efficient — though they are not popular among the research community, mainly because of cost and hardware in-

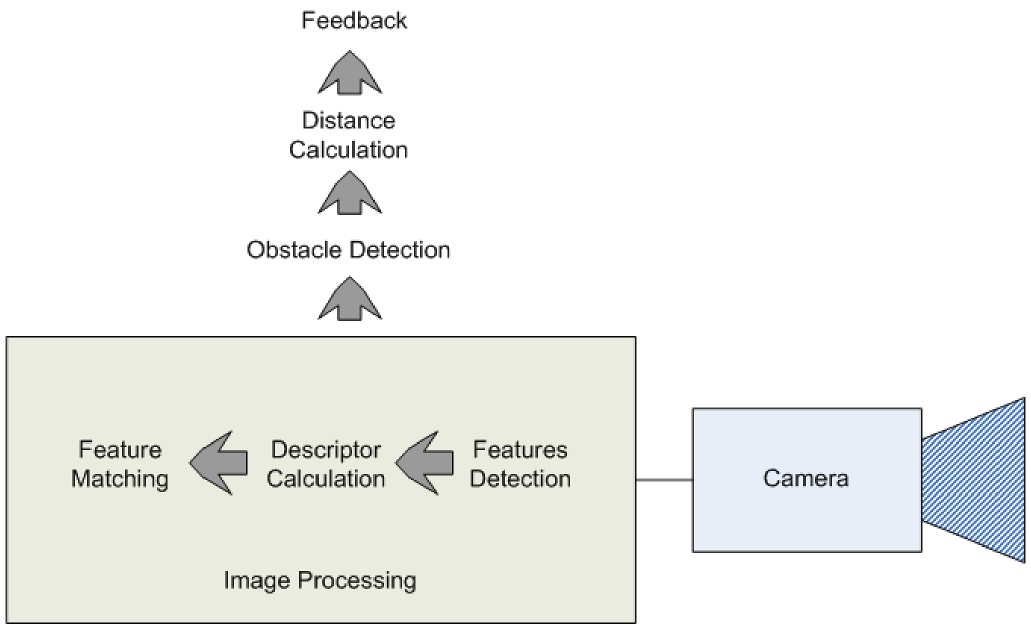


Figure 2.4: A navigation systems using single camera to capture images and process them to find obstacles and their distances from user/sensor

stallation. Some of the previously-developed systems using these sensors are discussed in section 2.3.1.

Cameras are by far the most common passive sensors used for automated navigation systems. They capture light through their shutter function, storing the resulting signal in some storage material, such as a magnetic film or a digital memory. Navigation and tracking systems with cameras are gaining more attention these days because of their small size, low-cost hardware, low power consumption and the availability of cameras in many of the gadgets with on-board processing capabilities, such as smart phones, iPads, and iPhones. The working of a navigation system based on a single digital camera is shown in Figure 2.4.

Vision systems working on images captured from a single camera at different times, or a stereo pair of cameras, need to perform some image processing functions to find obstacles. This processing involves detecting interest points, matching them in consecutive images or in the stereo pair, and calculating distances. If the distance is less than some threshold, feedback is sent to the user, perhaps warning the person of the distance and direction of obstacle when it is in a critical range. Different methods have been used such as audio feedback, vibrations and an audio beep.

2.3.1 Active sensors

The history of developing electronic travel aids for blind people dates back to 1945, when physicist Lawrence Cranberg developed a single-channel optical range detector for obstacle detection for blind people [36]. This relatively successful attempt encouraged researchers to develop sensor-based travel aids for visually impaired people. This led to the development of the first practical laser-cane in 1973 [37]. This was based on optical triangulation, with three laser diodes as transmitters and three photo-diodes as receivers. The cane could detect objects from 1.5 m to 3. m ahead of the user, with the additional function of detecting head-level objects.

The most popular electronic mobility aids in use today are based on sonar sensors that uses sound waves for obstacle detection, such as the sonic guide [38]. However, the sonic guide cannot detect all possible shapes of obstacles because some of the objects absorb the sound waves. *Pathsounder* [39] uses a neck band containing two ultrasonic transducers. The device was able to estimate distance using sound waves, however it gave audio feedback for only three distance levels, using a number of sound clicks.

The *Mowat sensor*,[1] a commercially-available handheld ultrasonic device with tactile output, was able to provide feedback about the distance of detected objects using vibrations. Low vibrations meant objects were at a safe distance, while in-

[1] A commercial product by Wormald International Sensory Aids, 6140 Horseshoe Bar Rd., Loomis, CA 95650, USA

creased vibration indicated that the obstacle was closer. In 1980, the *Nottingham Obstacle Detector* was developed, again a handheld sonar device with auditory feedback, with different musical tones to provide alarms at 8 different distance levels [40].

A system known as MELDOG [41] used landmarks at road junctions and a landmark map to allow safe navigation for a blind person in an unknown environment. In this system, white lines and road edges were used as landmarks, while the position was estimated using an odometer and near landmark map. Obstacles were detected using sonar, and the user alarmed only in the case of danger. The feedback system worked using electro-cutaneous stimulation gloves, which send a pulse train signal to the user.

Drishti [42] is a combination of Differential Global Positioning System (DGPS), a computer and an audio feedback system. It was developed to guide blind users and help them travel in both familiar and unfamiliar environments independently and safely. The system identified the user's location outdoors using its DGPS module and dynamically routed him or her. For indoor navigation, an ultrasound positioning system was used to find the user's location. The outdoor and indoor processing modules could be activated using verbal commands. The system used a small wearable computer which could communicate to a server computer using wireless connection both for data analysis and navigation instructions, but it limits its use in a specific range and makes it highly-dependent on good wireless connectivity.

The system proposed in [43] detected the nearest obstacle via a stereoscopic sonar system, and sent vibro-tactile feedback to inform users about their location.

2.3.2 Passive sensos

All of the navigation systems discussed above used active sensors and different output mechanisms to give users sensing abilities. Solutions using cameras as vision sensors are much cheaper and dynamic than the systems based on sensors such as sonar or laser sensors [2], which need a proper installation base and accurate angle measurements to accumulate the required useful data. A number of vision-based solutions using cameras have presented in the literature, including both those for autonomous vehicle movement and for human mobility.

In [44], stereo camera-based depth recovery was proposed. It used a disparity map and depth information, converted into sound coding blocks which are then transferred using simple sound transmitting devices. Similarly, in [45–47], a stereo pair of cameras was used for obstacle detection, staircase detection and zebra crossing identification.

vOICE [48] was a vision-based navigation system, consisting of a single camera mounted on headgear. The captured image data were converted into frequency tones representing image brightness. Here, the image pixel intensity was used as a distance measure. A similar approach was used in NAVI [49], where the captured image from a single camera was segmented into objects and background areas through an image segmentation technique. The segmented image areas identified as objects were then used to estimate the distance using image pixel intensities, translated into distance. Four distance categories were defined to categorize intensity levels that correspond to four different distance levels. Brighter pixels show closer objects whereas low intensity pixels represent background areas. This intensity representation of distance was converted into stereo sound for feedback.

Image pixel intensities may give distance information but can also fail due to a number of reasons. For example, strongly reflective material, even at a far distance, can cause the system wrongly to identify it as a close object. Similarly, intensity-based object and background segmentation may be affected by uneven light. Hence, precise understanding of image content plays a vital role in the success of any vision-related algorithm. The identification of image points with some characteristics or regions of interest such as edges, corners or blobs detection can bring stability and reliability, as shown by the systems discussed below.

Edges are used to find the depth map and disparity for two cameras in Optophone, an electronic blind aid [50]. This depth map is then converted into sound following the method proposed in vOICE [48]. However, Optophone works intelligently, using edge features for distance calculation, though it cannot be used for obstacle detection due to the complex processing required to find edges corresponding to object boundaries.

In [51], a map-less method for robot navigation using homography estimation from single camera images is shown. Mathematically, a homography is an invertible transformation from a projective space to itself; in the computer vision domain, any two images of the same planar surface are related by a homography matrix [52], a 3×3 matrix representing translation and rotation (collectively known as a transformation) between an image pair. The transformation between two images can be calculated by finding the positions of corresponding image areas. This can be done by identifying distinctive image points (called local image features) that can be matched in both images. A powerful yet efficient method to select image correspondences is called Random Sample Consenses (RANSAC) it randomly selects a few matched points, then finds an image transformation by calculating a homography matrix, and repeats this a number of times. It then gathers votes to identify which homographies are the most consistent.

The method presented in [51] follows the same mechanism and seems to be an improvement. However, the only drawback of the method proposed here is that it focuses on matching lines found in an image, which limits its performance to scenes with dense image content or navigation around objects only. This system has also been integrated into an autonomous wheelchair for handicapped people.

Computer vision systems are witnessing rapid development, with new feature operators being devised that may allow automated navigation systems to have access to better information, such as scale-, rotation- and affine-invariant point detection in images and their distance calculation [53]. The method adopted in [54] for obstacle detection in cross-country environments using stereo cameras is an example of using local invariant image features. The authors defined a method to the categorized detected features to be part of an object called a Projection and Quantification (PQ) method. The system defined a cone-like area around a detected point; if any other feature fell within the volume of the cone, it is considered part of the object. The PQ method projects 3-Dimensional (3D) information from a stereo system into a surface plane, and determines the depth information for a robot.

Tyflos [55] is a device developed for a blind person with two tiny cameras, a microphone and an ear speaker, mounted into a pair of dark glasses and connected to a computer. The system creates a virtual 3D space of surroundings and detects changes using range and image data from a stereo pair of cameras.

2.3.3 Hybrid systems

Navigation systems developed using either active or passive sensors have their own advantages and limitations. Depth recovery with active sensors is more accurate than camera-based methods but the need for a proper installation base and an angle measurement for data acquisition makes their everyday use difficult [22]. Therefore, researchers are trying to develop hybrid technologies by combining these two types of sensors, the hope being that this will result in more stable and reliable navigation systems, some of which are briefly described below.

Mobility Device, a project funded under the European Project *Autonomous system for mobility, orientation, navigation and communication (ASMONC)* [56], examines the mobility of partially-sighted people in outdoor urban environments. The system was a combination of multiple sonar sensors and a stereo pair of cameras. Its vision part used edges to detect objects and edges-based matching for distance calculation, while the tracking part used Kalman filters.

Some examples of the combination of vision with ladar to develop a road follower, and RoadCompass, a navigation project which used road vanishing points for tracking purposes are presented in [57, 58]. Similarly, an autonomous driving robot for rural dessert roads was developed using the same combination of sensors [59].

In [60], the system used optical markers for obstacle detection and distance calculation, while the feedback system consisted of vibrotactile attached to a waist belt. Similarly, an electro-tactile stimulus approach with stereo video cameras and Global Positioning System (GPS) was used to acquire 3D information in [61]. In [62], the obstacle avoidance and navigation in outdoor environments were done with the aid of visual sensors, GPS, and electro-tactile simulations.

The Kinect is a device designed to be used with Microsoft Xbox 360 gaming console. It allows the gamer to interact with games without the need for physical control, tracking the player's movements and position in a three-dimensional space with respect to itself in real time. The device has passive (colour camera) and active (infra-red project pattern, coupled with an infra-red camera) sensors integrated to identify a player's position and distance from the console. This combination of sensors directly benefits vision researchers developing navigation systems for robots or blind people [63]. The Kinect was proposed as an assistant for elderly people (see [64]), wherein a robot assistant uses the Kinect depth sensor for wall detection and self-localization by integrating the map of an environment. In [65], the authors proposed a segmentation scheme using the Kinect to separate humans from background, and to find their distance for different navigation and surveillance applications. Similarly, KinDetect [66], a project to detect objects, uses a depth sensor. It divides the depth image from the Kinect depth sensor into 5×3 grid and looks at depth values in each part. It was developed to work for structured indoor environments — working outdoors may pose some challenge because the infra-red sensor may not get reflections from non-reflective materials and may become 'blind' due to strong energy exposure such as sunlight.

The systems previously proposed using Kinect mostly used its depth sensor input for distance estimation and ignored the camera as sensing device, overlooking the complete capabilities of the device.

2.4 Local image features and obstacle detection

Generally, there are two types of approaches for matching and recognition tasks: the model-driven approach and the data-driven approach. The model-driven approach uses model libraries to find different objects in images [67]. This approach is feasible because most of the objects around us have geometric shapes and therefore can be matched easily using a model-based-matching algorithm. Although this is efficient and robust in finding shapes with a geometrical parameter, it fails to accommodate slight deformations to basic models of shapes. Another problem with the model-driven approach is that a system has to look up the models stored in the library and therefore cannot recognize new, unusual shapes.

The data-driven approach is a non-parametric modelling one. It allows the generation of a model without attributing it to a specific shape category, instead identifying image features and categorizing them to be part of an object or background. The advantage of this method is that deformations and dynamic changes in viewpoint can be identified at run-time. In this way, any type, size and textured shape can be identified, matched and recognized in images [68,69]. Furthermore, they do not need prior knowledge of object shapes [70]. The only drawback for this approach is the complex operations required to find and associate features belonging to an object or shape in images. However, the advantage of local features is that they are invariant to a number of geometric and photometric transformations, making them suitable for the detection and recognition tasks in videos and still images.

With the continuing rapid development in hardware speed and the continuous progress in local feature detection and description techniques, local image feature extraction has become more reliable in changing imaging conditions. That is why they are considered better than other methods for real-time visual applications such as tracking and navigation [70–73]. Different image features can be useful for visual applications such as corners, edges, blobs, ridges etc., but the selection of one of them depends largely on their computation time, repeatability, stability and robustness to geometric and photometric transformations.

Table 2.2: Some of the popular local feature operators for different image features

Features	Features Operators	Features Descriptor	Invariance to Transformations
Blobs	Scale Invariant Feature Transform (SIFT)	SIFT	Scale & Rotation
Blobs	Speeded Up Robust Features (SURF)	SURF	Scale & Rotation
Corners	Harris & Stephen's (H&S)	BRIEF	Scale & Rotation
Corners	Harris Laplace (Harlap)	any	Scale
Corners	Harris Affine (Haraff)	any	Affine
Blobs	Hessian Laplace (Heslap)	any	Scale
Blobs	Hessian Affine (Hesaff)	any	Affine

2.4.1 Interest points and operators

Interest point extraction from an image is a fundamental step in many vision applications, including tracking, navigation, panorama stitching, and mosaicking [11, 74]. Interest point extraction can be sub-divided into two main stages: firstly, the detection of image interest points ('features' or 'keypoints') with high repeatability, robustness and uniqueness under varying imaging conditions and transformations [11]; and secondly the computation of distinctive descriptors for these detected interest points. In combination, these algorithms are called feature operators.

An image is a collection of points which jointly represent a scene. These points can be categorized into different groups based on their individual characteristics or the information they carry, such as edges, corners and blobs [75]. A corner and an edge are image features that lie on the boundaries of image regions, such as the boundary of an object. The difference between the two is that edges represent the whole boundary while the corner is the point that lies at the intersection of two or more than two edges. Contrary to corners and edges, a blob is a dark image pixel surrounded with bright pixels, or a bright pixel surrounded with dark pixels. A blob helps identifying image areas which are very smooth and cannot be detected using corners or edges.

The feature operators that have received recent interest in the literature include H&S [76], Features form Accelerated Segment Test (FAST) [77], Haraff and Hesaff [78] and fast Hessian based detector [79]. State-of-the-art image feature descriptors include the SIFT [80], Principal Component Analysis (PCA)-SIFT [81], Gradient Location and Orientation Histogram (GLOH) [10], SURF [79] and BRIEF [82].

Table 2.2 presents some state-of-the-art detectors and descriptors for extracting different image features [79–90]. A brief description of blob detectors and descriptors is given below while corner detectors are discussed in chapter 5.

SIFT

For local invariant feature detection and matching, SIFT gained huge popularity due to its strong detector and highly distinctive descriptor. Its detector finds blobs at different scales using a scale space pyramid by approximating Laplacian $L_{xx}^2 + L_{yy}^2$ values in the x and y directions. To improve localization accuracy, 3D quadratic interpolation is used to sub-pixel accuracy. The SIFT detector is designed to reject points lying at edges, therefore the Hessian response of detected points is used for this purpose.

After detecting features at different scales, the gradients of these features are calculated for the descriptor and its orientation. This is done by calculating the histogram of gradient orientations and recording the dominant orientation for every feature, which make these descriptors rotation invariant. SIFT uses a 128-element descriptor to store the location, scale and gradient orientation information of detected points. Furthermore, the normalization of this 128-bin descriptor makes it moderately invariant to an illumination change in images.

SURF

SURF is a modification of the predecessor SIFT in terms of efficiency, proposed by Herbert Bay et. al. [79]. The authors reintroduced the use of integral images, that have been used in computer graphics since the 1980s and used by Viola and Jones [91], to speed up the complex computations required to build scale space (a convolution of an image using box filters) and descriptors.

The detection stage works in a similar way to SIFT, by finding interest points at different scales and rejecting points with a strong blob response, which usually lie on the edges, using a determinant of the Hessian matrix. The difference between SIFT and SURF is a reduction in the number of octaves and smaller number of pixels in order to calculate a blob response. During scale space pyramid construction, a number of octaves are calculated per scale to find blobs at different scales. However, in SURF, four octaves per scale are considered sufficient due to the smaller number of responses at higher octaves. Furthermore, to reduce computations, the sampling intervals has also been increased for successive octaves. Although considering 2^{nd}, 4^{th}, 6^{th} and 8^{th} pixels for octaves 1 to 4 affect the overall accuracy of the detector, its authors claimed it to be insignificant. Moreover, the localization accuracy is improved using 3D quadratic interpolation [92].

The SURF descriptor uses the Haar wavelet response around a feature, which is efficiently computed using an integral images — a feature only needs six operations to compute response in the x and y direction, irrespective of size. In [93], further improvements were suggested to efficiently perform calculations over integral images. SURF can store the information in two different descriptor lengths, 64 or 128 bins. Both of these descriptors are claimed to be more efficient and distinctive than SIFT.

GLOH

A 128-bin descriptor proposed in [10] stores the neighbourhood information in the form of a gradient, location and orientation histogram space. It calculates a log-polar location grid with 3 bins in radius and 8 bins for angular directions, leaving the central bin of the detected point. The gradient orientations are quantized in 16 bins, which gives a 272-bin orientation histogram, but the descriptor length is reduced using Principal Component Analysis (PCA) to 128 bins by choosing only the largest eigenvectors for description [10]. The authors have also contributed by developing four different scale-, rotation- and affine-invariant detectors based on the H&S and Hessian detector known as Harlap, Haraff, Heslap and Hesaff. Although GLOH can be used to describe any image feature, in the literature the common detectors used with GLOH are the ones mentioned above and described below.

Harris-Laplace (Harlap)

This is a combination of the Harris detector and Laplacian of Gaussian (LoG) function for scale selection. The use of the Harris detector makes corners and highly-textured points as dominant interest points, as opposed to blobs by the SIFT and SURF detectors. It uses the multi-scale Harris detector to find corners at different scales.

Harris-Affine (Haraff)

For this detector, the shape of a point's neighborhood is used for detection using the second moment matrix proposed in [94]. This makes a detected point stable and repeatable under affine and some perspective transformations of a smooth surface. For a multi-scale Harris detected point, a second moment matrix of automatically-selected integrated and differentiated scales is calculated, which then gives the shape of the neighborhood of a detected point and makes it identifiable in images with a viewpoint change.

Hessian-Laplace (Heslap)

The Hessian-Laplace operator proposed in [78] searches for the local maximum of the Hessian determinant to find blobs in images. It is rather similar to Harris-Laplace. LoG is used for selecting maxima over multiple scales to identify stable features at multiple scales. It claims to have better localization accuracy than Difference of Gaussian (DoG) along with better scale selection accuracy than Harris-Laplace [95].

Hessian-Affine (Hesaff)

The Hessian-Affine detector is similar to the Harris-Affine detector. [85] suggested the use of the trace and determinant of the Hessian matrix to select interest points. The replacement of LoG is done because of its inability to reject points where the signal changes in one direction only, such as on contours and straight edges and being more sensitive to noise. The trace of the Hessian matrix detects points for which the second derivatives change in only one direction and therefore can be filtered.

BRIEF

A descriptor known for its speed in descriptor calculation and matching, it is able to identify distinctive region around both corners and blobs. It is a binary

Table 2.3: SIFT and SURF default parameter values. GLOH is the extension of SIFT descriptor therefore it share similar parameter values as SIFT

process	**SIFT parameters value**	**SURF parameters value**
Read Image	Raw Image	Integral Image
Scale Space calculation	# of octaves=$\frac{log(min(image-width,image-height))}{log(2)-2}$	# of octaves=4
	samples per octave=3	samples per octave = 3
	sampling step = 1	sampling step = 1,2,4
Scale space extremum detection	blob response $\leq$ 300,00	blob response $\leq$ 500,00
Non-maximum suppression	In 8 x 9 x 9 neighbors	In 8 x 9 x 9 neighbor
Orientation Assignment	Gradient orientation histogram	Haar Wavelet response
Descriptor length	128 bins	128 or 64 bins

descriptor that calculates intensity difference of image pixels such as

$$\tau(p;x,y) = \begin{cases} 1 & \text{if } p(x) < p(y); \\ 0 & otherwise, \end{cases}$$

and stores them as an array of 0s and 1s. Because the descriptor contains binary numbers as descriptor information, its matching is efficient, requiring only an exclusive OR operation (a boolean operator that returns TRUE if only one of its operands is TRUE and FALSE otherwise).

2.4.2 Parameter tuning

The algorithms discussed above have a number of free parameters (contrast threshold, sampling rate, number of octaves, etc). Tuning these parameters is an important way to improve an algorithm's performance. On the other hand, tuning may actually compromise performance, as will be illustrated in the fol-

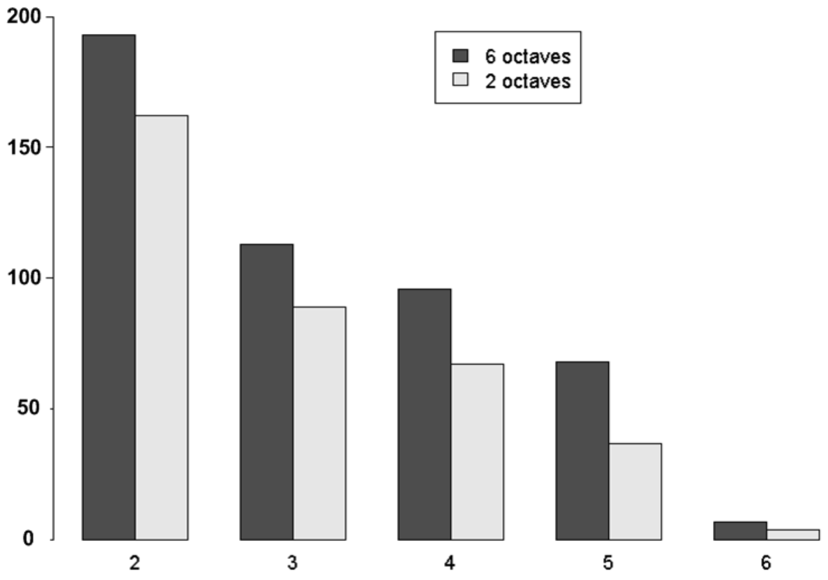


Figure 2.5: Effect of number of octaves on matching performance, matching results of six images from Boat dataset using SURF descriptor

lowing discussion, where SIFT and SURF are used as test cases.

A brief description of the processing stages of SIFT and SURF, along with their default parameter values, are given in Table 2.3. While constructing a scale space, the number of octaves is calculated to find stable features across different scales of an image (using Gaussian filters of different sizes). Originally, both algorithms suggested that the number of octaves depends on the size of the image and therefore, using the equation given in Table 2.3, the number of octaves for an image is decided. For SURF however, authors suggest using only 4 octaves because of the small number of features that can be identified in higher octaves [79]. In order to increase or decrease the number of detected features, the size of the scale space pyramid can be changed by changing the number of octaves or the number of intervals per octave [96,97]. As shown in Figure 2.5, an increase in the number of matches can be achieved by increasing the number of octaves from 2 to 6. For image 1 and 2, there is an 18% increase in matches, and the same per-

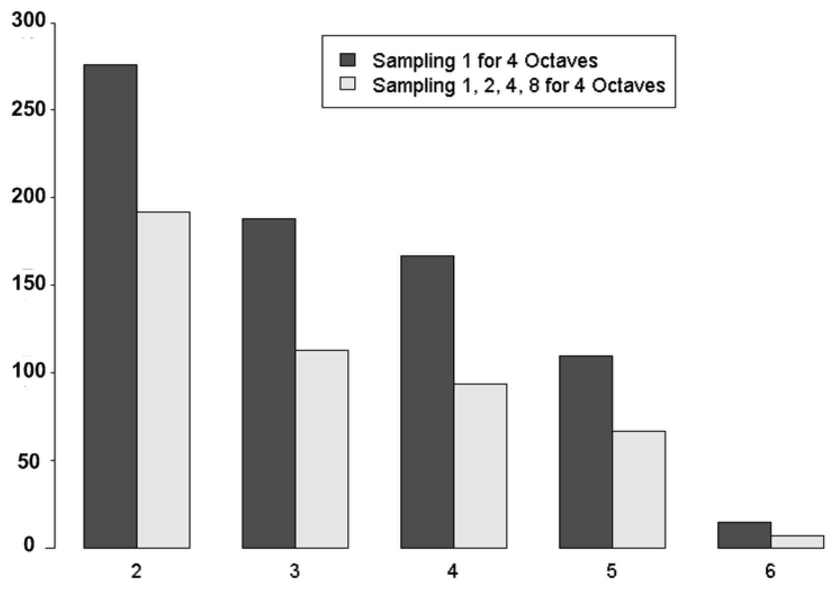


Figure 2.6: Effect of different sampling rate on matching performance, matching results of six images from Boat dataset using SURF descriptor

centage increase can be seen in other images as well. Undoubtedly, the increase in the number of matches does not guarantee the increase in accuracy, as shown in Figure 3.8b.

Another important parameter is the sampling rate, the number of points processed in one sample of an octave. If the sampling rate is kept to unity, the detector considers every image pixel in a scale space to find its blob response. Changing its value from 1 to 2 or 4 means that every 2^{nd} or every 4^{th} pixel will be considered for a blob response calculation, which in turn will affect the number of features detected in an image, as with more features the number of matched points also increases. It has been suggested in [79] that increasing the sampling rate for higher octaves reduces computation with no significant difference in performance. Therefore, the SURF algorithm changes the sampling rate with respect to the number of octaves. For any octave, the sampling rate is determined using $2^{(octavenumber-1)}$ and it will be 2, 4 and 8 for the second to the fourth octaves re-

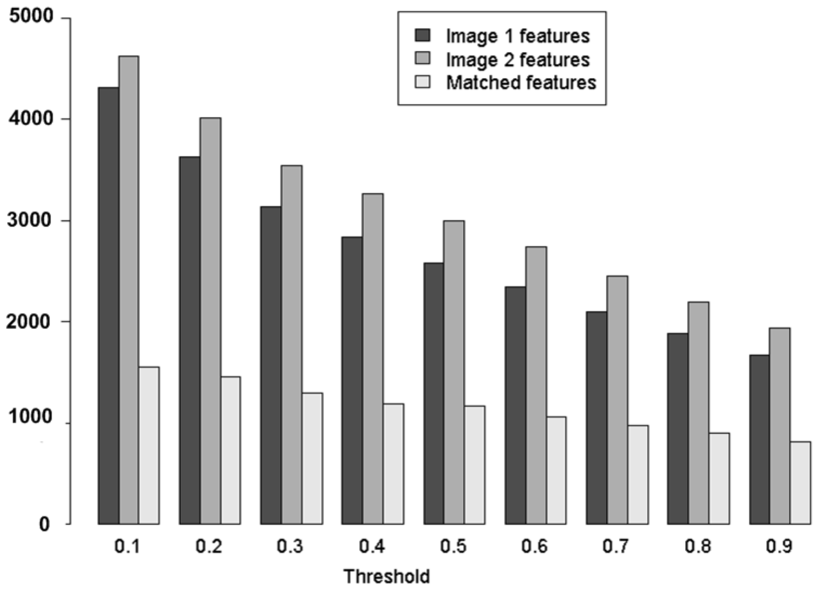


Figure 2.7: Effect of changing blob response threshold on number of features detected and matching performance of descriptor, results of matching image-1 and image-2 from Graffiti dataset using SIFT descriptor

spectively. Figure 2.6 shows a decrease in matched points as the sampling rate for higher octaves is varied while other parameters, such as number of octaves and blob response threshold, are kept constant. This indicates that an increase in the number of interest points by reducing the threshold will cause more noise pixels to be included and hence add more false positive matches.

In a scale space extremum detection stage, the threshold is the most useful tuning parameter. In both SIFT and SURF, an image pixel qualifies as an interest point if it is reasonably different from its surrounding pixels at some scale (i.e.,blob response > threshold). This parameter plays a vital role in increasing or decreasing the number of points detected from an image and has a significant impact on the matching performance of an algorithm. Figure 2.7 shows the increase and decrease in the number of features detected by SIFT for different threshold values: the higher the threshold, the lower the noise will be in the image, but at the same time the low threshold helps identify good features from low-quality

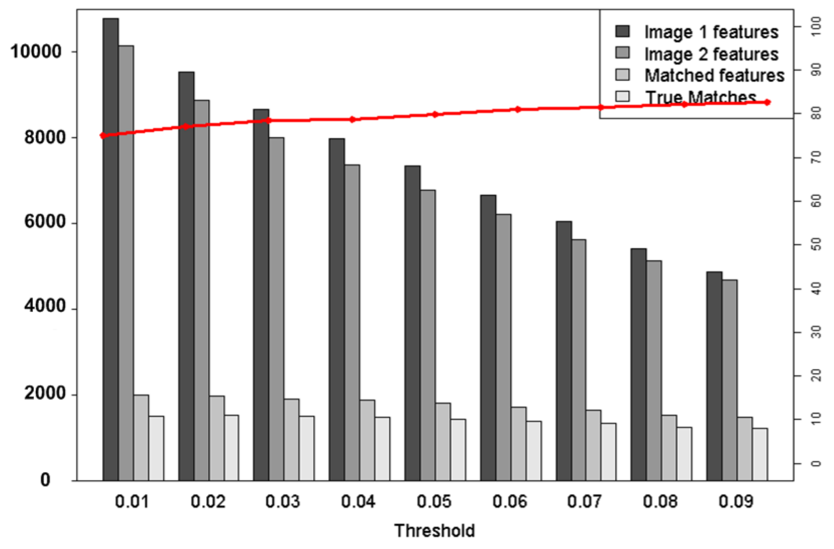


Figure 2.8: Matching performance of SIFT descriptor for boat image-1 and image-2 for different blob response threshold. Increase in % accuracy is shown with red line.

images. Therefore, changing the threshold value may affect the performance of an algorithm.

It is evident from the above discussion that different parameters can be tuned to extract the desired number of features from images, or to alter the matching outcome between a pair of images. However, in this process, the performance of an algorithm may be compromised. As Figure 2.8 shows, the number of matches decreases for a higher threshold, but the percentage accuracy increases. Therefore, parameter tuning is arguably not appropriate while comparing two algorithms and should also be avoided. The original implementations of the SIFT, SURF and GLOH algorithms were employed wherever required using binaries made publicly available by their authors. However, the results of parameter tuning given in Figures 2.5 to 2.8 were taken from open source implementations of

SIFT by Hess[2] and OpenSURF by Evans[3].

2.4.3 Feature matching

Matching is the step required to identify corresponding features in subsequent video frames for tracking or navigation purposes. A number of matching algorithms have been developed and used in the literature. For feature descriptor matching, common techniques are similarity, nearest neighbour, distance ratio nearest neighbour [10], and the correlation matrix [98]. Most of these algorithms have efficient implementations such as using Kd-trees [99] and hashing [100]. In nearest neighbour based matching, two features i and j are matched if the descriptor for feature i (d_i) is the nearest neighbour to the descriptor for feature j (d_j), and if the distance between them is below some threshold. The nearest neighbour distance ratio is similar, except that the threshold is applied to the distance ratio between the first and the second nearest neighbours. Thus, the descriptors are matched if $d_i - d_j / d_i - d_k < \tau$, where d_j is the first and d_k is the second nearest neighbour to d_i. In [10], different descriptor matching approaches are compared and the nearest neighbour matching algorithm is identified to perform better than distance ratio nearest neighbour algorithm in terms of accuracy.

2.4.4 Tracking and navigation

Matching objects in multiple frames and calculating their distances for a moving person (blind/partially sighted) is required for automatic navigation. It can only be achieved if we first detect one or more objects in a frame, and then track them in subsequent video frames. In [101] authors present a new feature called "transition" to detect and track an obstacle in video frames by keeping a history

[2]http://robwhess.github.com/openSIFT/
[3]http://www.chrisevansdev.com/computer-vision-openSURF.html

of the detected features. In [102], Harris corners along with SIFT descriptors are used for object recognition that can be used for tracking. Similarly, a number of algorithms proposed for shape-based object detection, such as a polygonal representation of a curve, are used to develop a non-parametric algorithm for shape identification in [103]. Template matching and optical flow were also used to track objects in real time [104]. Algorithms were developed for MPEG-7 visual description in [84, 105]. These methods are mostly related to shape description techniques. Gradient-based region description [106] and the wavelet-based region description [107] are two other methods used for object detection and matching in video imagery.

Currently the best-known methods in the literature for point tracking are homography-based matching [71], visual odometry, optical flow, particle filtering and Kalman filters [108]. In the robotics research, Simultaneous Localization and Mapping (SLAM) is a technique that can directly or indirectly help building navigation systems for blind people. There are a number of publications to that have implemented vision-based SLAM efficiently and effectively [72, 109, 110]. SLAM also works by matching video frames to build a map of an unknown environment; it is done by keeping a track of the robot's current locations. These tracking methods are also used for indoor and outdoor vision-based augmented reality applications [14, 111, 112]

Similarly, the use of visual odometry is also common in vision research. This is a method of estimating the position and orientation of a moving object (a robot or person) by analyzing the images taken from consecutive poses. This can be done by constructing a pixel-wise optical flow field or matching image projections, as applied in [113] for visual odometry estimation for robot exploration on different types of terrain. Another system used visual odometry on images from an omni-directional camera mounted on top of a car [114] for outdoor robot navigation; their real time ego-motion system combined both optical flow and feature matching approaches, the latter using SIFT features. It used homography to find a column shift, which gives the displacement in feature positions, between two consecutive frames.

In [115], a CENtre SURround Extrema (CenSurE) approach is used to pick out blob-like features and provide a distinctive descriptor, with efficient implementation using integral images and Haar wavelets. The system is claimed to give comparable accuracy to SIFT features but better time efficiency. In the case of a single camera, only three to five points in an image need be used for visual odometry using homography to find a relative pose between frames [116, 117]. Here SIFT descriptors were calculated at FAST corners [118] detected at with a scale space pyramid. To estimate the 3D rotation of ego-motion, a fixed-size window was projected to find landmark correspondences on both images using RANSAC.

In the last decade or so, Kalman filtering is gaining popularity in robotics, in particular for tracking objects and estimating location and position of an object. However, the problem with Kalman filters is that some prior knowledge about the process and the measurement covariance matrices are needed, which are difficult to obtain in most cases [119]. Therefore, particle filters are now preferred to Kalman filters. In [120], an adaptive colour-based particle filtering approach is presented for object tracking.

The review of the literature given above reveals that many algorithms have been proposed for every vision task. Deciding which one of these algorithms is best suited for a particular problem is very difficult. For this purpose, statisticians have defined a number of performance characterization measures. However, the selection of an appropriate performance measure is again vital and demands an in-depth understanding of the domain (such as computer vision) and data (such as images) used for performance evaluation as well as the statistical techniques involved. The discussion that follows aims to explore some performance characterization methods used to evaluate vision algorithms, with a focus on algorithms for image feature extraction.

2.5 Performance evaluation techniques

Performance can be defined as the accomplishment of a given task measured against some known accuracy [121]. In computing, it is the measure of an algorithm's ability to complete the task for which it has been developed and produce an accurate output. One example of an algorithm is counting the number of people passing through a common point from camera images.

Benchmarking is the common method to set standards against which algorithms' relative performances can be measured. However, it requires prior knowledge of data — for example, if we know that in every minute four people pass through a corridor, then the output of a new algorithm can be compared with the previous ones; but if there is no information about human traffic, then comparing one algorithm's performance with others is the only means of performance assessment.

There are a number of performance measures already in use, such as ROC curve, which was developed during World War II to analyze radar accuracy in differentiating signals from noise [122] but has been adopted by other scientific fields, including computer vision, to evaluate the performance of vision algorithms [123–127]. It is a visual form of performance comparison where the curve on a plot is used to represent an algorithm's performance.

Similar to ROC curves, there are other visual performance metrics. These include PR [128] and SS [129] graphs. These measures were introduced to check the validity of ab algorithm's output because calculating the accuracy of algorithms alone was found to be misleading [130] for machine learning algorithms. That is why it has become a convention to present a comparison of a newly-proposed algorithm with existing methods using an appropriate performance evaluation measure. In the context of navigation systems, a good matching algorithm is a kind of a building block as its accuracy directly affects the overall system's performance. As discussed before, current research is more focused on using local image features to match images, identify objects and obstacles. This is because of their invariant behaviour under different geometric and photometric transformations. Due to the presence of a large number of feature detection and description algorithms, their performance characterization has also been performed numerous times in the past.

Repeatability is the most frequently employed evaluation measure for feature detectors [131, 132]. Likewise, ROC and PR curves have been used to evaluate interest point descriptors. For example, in [81] the precision–recall values were calculated to compare the distinctiveness of SIFT and a modified form, PCA-SIFT. Results presented for artificially-generated data proved PCA-SIFT to be more robust than conventional SIFT. In [133] and [134], a comparative study of filters for texture classification was performed using PR criteria. In [135], a phase-based local feature descriptor was presented and its performance compared with other descriptors using ROC curves. For all these studies, a single detector's output was considered using various descriptors. In [136], the evaluation of feature descriptors for a number of correctly-matched video frames is presented using the same PR criterion as in [10].

Most comparative studies of local feature descriptors have ranked the SIFT descriptor [80] and SIFT-like descriptors such as GLOH [10] as the best — though it is claimed that SURF produces better results [79, 136] using the criterion given in [10]. Similarly, a number of corner detectors have been developed in the past, and a number of attempts have been made to assess their performances [76, 118, 137–139]. H&S corner detector turns to be the best in most of these studies.

However, the reliability of these rankings is questionable because, in all of these studies, no more than five image pairs have been used. The impact of the dataset size on the error rate estimation, discussed in [140], can result in spurious estimates of performance and hence incorrect rankings.

This thesis presents a thorough investigation of the use of performance evaluation measures, their statistical significance, reliability and the effect of the data size used for performance characterization. This investigation commences in the next chapter, which examines existing performance metrics in detail.

CHAPTER 3

EVALUATION METHODS: RELIABILITY AND STATISTICAL SIGNIFICANCE

3.1 Introduction

Most vision papers published in reputable journals now have to include some evaluation work in order to demonstrate that the algorithm proposed is an improvement on existing ones. Generally, these evaluation results are presented in tabular or graphical form. Neither of these is ideal because there is no indication as to whether any performance differences are statistically significant. Moreover, the size and nature of the dataset used for evaluation will obviously have a bearing on the results, and neither of these factors is usually discussed. Clearly, there is a need to improve upon this situation.

This chapter explores this evaluation problem and attempts to make some headway by employing statistical tests. Firstly, section 3.2 introduces the way that vision evaluation studies are currently almost invariably performed and describes the measures that represent performance. Section 3.3 goes on to consider a specific performance evaluation in detail and shows that the performance measures imply different results, a finding that calls into question their value.

Having demonstrated the problem with existing performance measures, section 3.4 introduces a null hypothesis testing framework that can be employed to assess the performance differences of vision algorithms. McNemar's test has been used in this framework and it is shown to be able to compare the performances of a pair of algorithm in a statistically sound way. Section 3.4.4 then examines whether using McNemar's test with 'ground truth' data can be used as a figure of merit for assessing performance, and discusses whether this figure of merit indicates statistical significance.

McNemar's test is for the analysis of paired data, and hence to compare more than two samples one needs to perform McNemar's test multiple times. According to statistical theory, multiple two-sample tests increase the probability of Type-I error(false rejection of null hypothesis); corrections can be applied to reduce Type-I errors , the best-known of which is the Bonferroni correction [141]. However, there are concerns regarding these corrections. There are other statistical tests which can reduce Type-I errors while performing multiple comparisons, but they also have limitations. One widely-used such test is ANOVA, which is described in section 4.6. ANOVA is a generalization of the well-known *t*-test for comparing multiple samples simultaneously, i.e. it identifies statistically significant differences in means. The test can only be applied on data which are normally distributed, a significant constraint. Moreover, ANOVA does not identify the sample from the population which has a significantly different mean.

The most important distinction between McNemar's test and ANOVA is that

Table 3.1: Confusion Matrix

	Actual True	Actual False
Predicted True	True Positive (TP)	False Positive (FP)
Predicted False	False Negative (FN)	True Negative (TN)

the former compares outcomes (success/failure) whereas the latter compares a continuous measure. One can convert a continuous measure into a discrete outcome using a threshold but not vice versa. Unlike ANOVA, McNemar's test can answer two questions: one, whether the two samples are statistically different; and second, which one of them is better with a given confidence level.

Section 5.8 concludes this chapter by highlighting the shortcomings of evaluation methods and the rankings of algorithms presented in the literature.

3.2 Performance evaluation measures

To understand the behaviour and characteristics of any performance measures, some numerical data will be used to motivate the discussion. A simple image matching problem is selected (see chapter 4). Figure 3.1 presents a geometric shape and a rotated version of it. If one wants to match these two shapes, the easiest way is to match corresponding corner points. Solid and broken lines are used to indicate the output of a notional matching algorithm. Corners 3 and 4 are correctly matched but corners 1, 2, 5, 6 and 7 are not. Furthermore, corner 2 and 6 are incorrect matches, while corners 1, 5 and 7 are unmatched. In fact, one can identify four possible outcomes of a matching process, namely:

TP: obtained when an algorithm's outcome is correct;

FP: obtained when an algorithm reports a result; that result is incorrect;

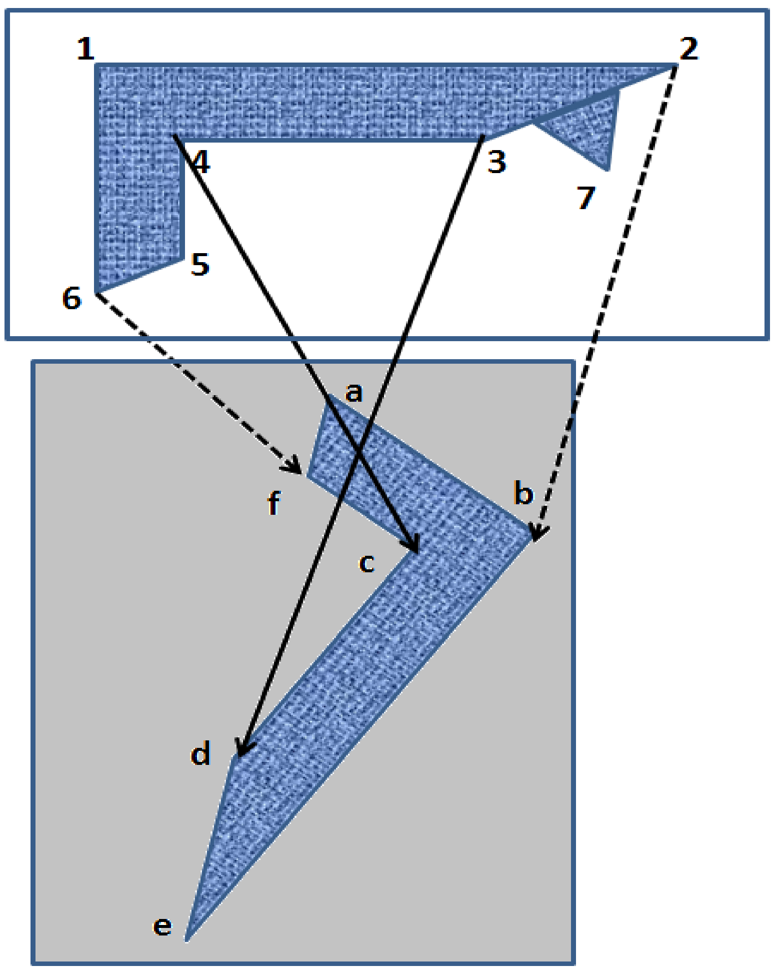


Figure 3.1: A matching example: solid lines show true positive matches and broken lines show false positive matches.

Table 3.2: Confusion Matrix presenting matching example data

	Actual True	Actual False
Predicted True	2	2
Predicted False	2	1

TN: obtained when an algorithm reports a failure when it should do so;

FN: obtained when an algorithm reports a failure when it should not.

These results can be presented in a confusion matrix as shown in Table 3.1.

In Figure 3.1, points 2, 3, 4 and 6 are matched with b, c, d and f of the rotated shape. By visual inspection, one can easily identify points 3 and 4 as true positive matches, points 2 and 6 as false positive matches, points 1 and 5 as false negative matches and point 7 as a true negative match. The resulting confusion matrix is shown in Table 3.2. One can classify this algorithm's performance as poor because it has identified more false matches than true matches. An ideal algorithm should be able to correctly classify TPs and TNs with no FPs and FNs. However, in practice there is always a trade-off between positive and negative results, making it difficult to classify an algorithm's performance using confusion matrix data alone. Consequently, a number of derived metrics has been defined, such as Accuracy, Precision, Recall, sensitivity and specificity which can be calculated from these confusion matrix data. These are discussed later in detail.

Calculating one algorithm's performance in isolation is of limited value; a comparison with other algorithms' performances on the same data is usually more meaningful. The results from each algorithm can be recorded in a confusion matrix and the latter compared. However, it is much more common to derive metrics, such as Precision (P) and Recall (R), from them and compare those metrics. Equations for calculating all these measures from TP etc. are given in

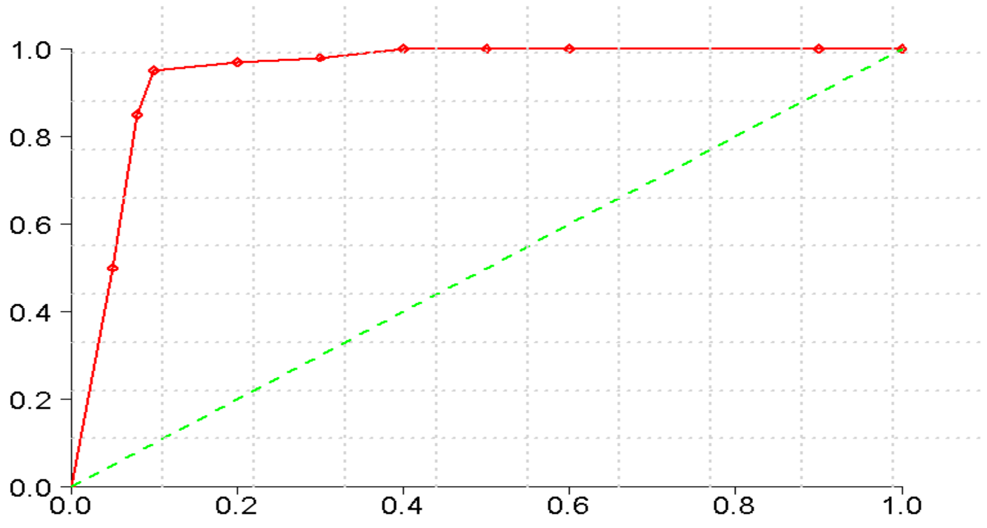


Figure 3.2: ROC space: Dotted line reflects random outputs of an algorithm, while solid curve is the best expected performance of an algorithm and the bottom right area shows high false positive rate and therefore, worst expected performance.

Table 3.4.) Indeed, these metrics are usually plotted for visual evaluation. When a number of algorithms are to be compared, a ranking is often produced from graphs.

Figure 3.2 illustrates an ROC curve. It is important to appreciate what is plotted here: each point on the curve summarizes the performance of an algorithm with a specific set of tunning parameters; hence, the curve shows how the algorithm's performance varies as a tuning parameter changes. An algorithm whose performance is close to the top-left corner of an ROC curve is performing better than one whose curve is further away. In practice, ROC curves often cross as illustrated in Figure 3.3; then one has be careful about the settings of the tuning parameters of algorithms. In an attempt to identify an overall better algorithm when ROC curves cross, several researchers calculate the Area Under the Curve (AUC). However, this is not necessarily reliable [125]. These concerns are also

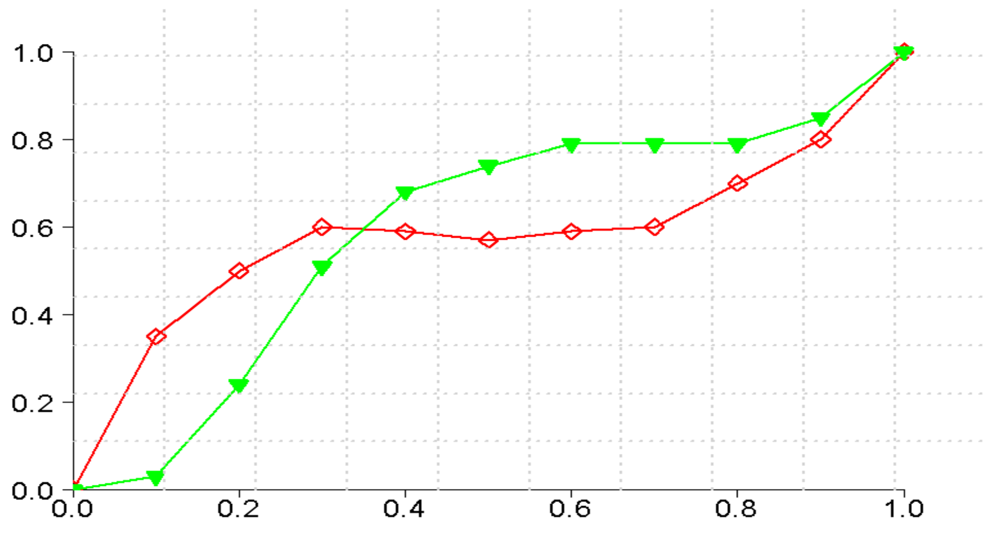


Figure 3.3: Example of crossing curves

applicable to the other performance measures discussed below.

PR curves (plots of P against R) are somewhat analogous to ROC curves, though the top right corner indicates good performance. (Many researchers plot R against 1-P to have similar orientation as ROC curves.) Ideally, $P \approx R \approx 1$ represents good performance; however, R can be easily maximized at the expense of P and vice versa. Hence, for ranking an algorithm, one often combines P and R into the so-called *F-measure*.

Similarly, SS graphs are most commonly used in behavioural sciences but are closely related to ROC curves. Its appearance can be similar to an ROC curve if Sensitivity is plotted against $1 -$ Specificity. It is also interesting to see an algorithm's performance using simple measures such as True Positive Ratio (TPr) against False Positive Ratio (FPr). Lastly, accuracy is probably the most popular method for translating confusion matrix data into a single numeric performance measure.

Table 3.3: Output of a Notional Algorithm; Set A: original output of algorithm, Set B: parameter tunning resulted in down sampling of negative examples, Set C: tunning parameter resulted in a uniform increasing in all examples.

Set	TP	TN	FP	FN
A	3361	2370	1294	375
B	3361	198	101	375
C	3371	2380	1304	385

Table 3.4: Performance metrics based on Data presented in Table 3.3 to show if these are variant to distribution of values in confusion matrix

Performance metric	Description	A	B	C
Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$	0.7744	0.8820	0.7730
Precision	$\frac{TP}{TP+FP}$	0.7220	0.97 08	0.7210
Recall	$\frac{TP}{TP+FN}$	0.8996	0.8996	0.8974
True Positive Rate (TPR)	$\frac{TP}{TP+FP}$	0.7220	0.9708	0.7210
False Positive Rate (FPR)	$\frac{FP}{TN+FP}$	0.3532	0.3378	0.3540
F-measure	$2\frac{PR}{P+R}$	0.8011	0.9339	0.7997
Sensitivity	$\frac{TP}{TP+FN}$	0.8996	0.8996	0.8975
Specificity	$\frac{TN}{TN+FP}$	0.6468	0.6622	0.6460
TPr	$\frac{TP}{N}$	0.4541	0.8330	0.4531
FPr	$\frac{FP}{N}$	0.1749	0.0250	0.1752

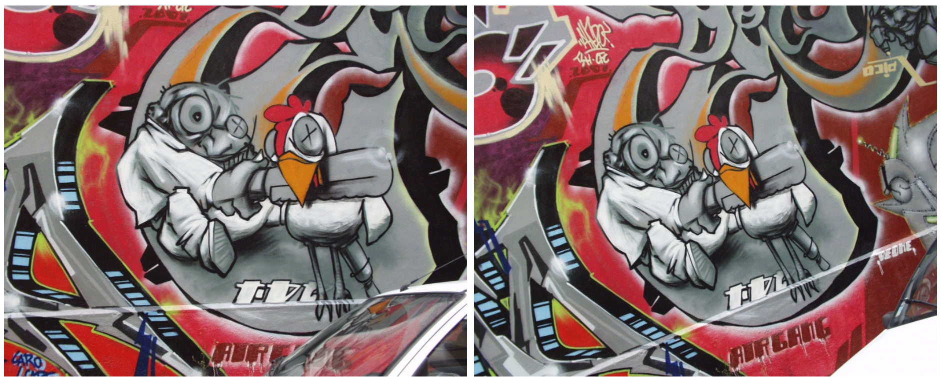
Changing the value of a tuning parameter may, for example, convert a false positive into a true positive, affecting two cells of the confusion matrix. [124] considered these kind of events in detail, identifying for example that if the data in the confusion matrix change proportionally, the ROC curve is unaffected. To illustrate this in more detail, Table 3.3 shows the results from a notional algorithm with different tuning parameter settings A, B and C while Table 3.4 shows the corresponding derived measures.

Set C in Table 3.3 has 40 more points than set A, 10 in each of the four categories. Because there are different counts in each of TP etc, the actual performance is different, yet the TPR in Table 3.4 is unchanged. This is clearly undesirable. Similarly, R is invariant to this type of change, as are both sensitivity and specificity. This means that these curves have some shortcomings for assessing the performances of vision algorithms.

3.3 Comparing performance metrics

With so many performance metrics available, it is illuminating to discover whether they all yield consistent results. If they do not, one could argue that they are actually measuring something other than performance, or are measuring different aspects of performance. The previous section demonstrated that the common performance measures produce different results on a 'thought experiment' but it remains to be established that such performance measures produce inconsistent results in practice. To that end, we now perform an experiment using a pair of images drawn from the well-established Oxford database, described in detail in Chapter 4. The pair of images are shown in Figure 3.4.

The particular problem that we shall explore relates to finding the transformation matrix or *homography* between the two images. As well as being worth-



(a) Graffiti reference image (b) Graffiti test image

Figure 3.4: Images from Graffiti dataset to perform local feature based matching

while for evaluation in general, this is highly relevant to the development a navigation system for the visually impaired, as it gives a way of identifying the motion of objects relative to the observer. To determine the homography one first determines matching features between the two images and uses them effectively to solve a set of simultaneous equations, the result being a transformation matrix. This homography-based approach has to be employed because each feature detection algorithm detects a different number of interest points from images, and at different locations.

Features from the two images are matched; from consistent matches (inliers obtained using RANSAC) [142]), an estimated homography matrix is calculated, which is then compared with the 'true' homography provided with the imagery. (How these 'true' homographies were obtained is described below.)

The simplest way to compare two homographies is to see how closely they project points. Therefore, some points from the reference image were projected using the estimated homography and then compared with the true projection of the point using the homography supplied by the Oxford group.

The spacing of these points to be projected is important: as explained by

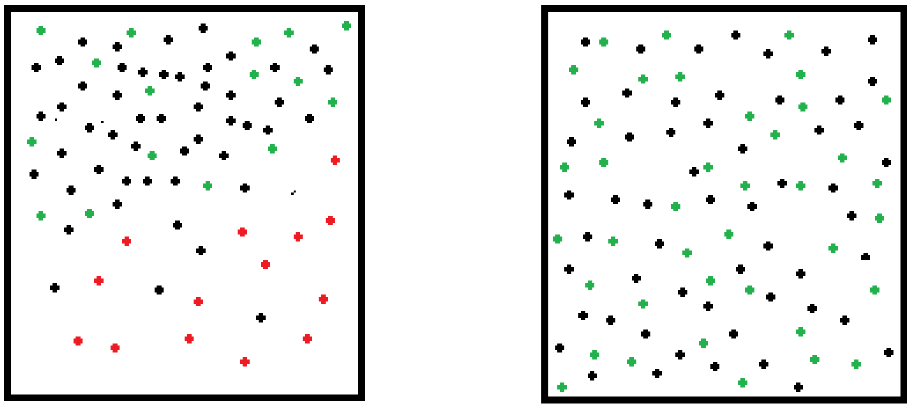


Figure 3.5: If black points are used to calculate homography matrix then red points are those projected wrongly and green points are those correctly projected

Figure 3.5, if the points selected to calculate homography matrix are not evenly distributed over the image, then the homography may represent the transformation of only that part of the image and therefore may well not project all points correctly, as shown by red dots in image [143, 144]. Therefore, to test a homography matrix, equally-spaced points are used to avoid any skew towards the points selected for the calculation of the homoraphy.

For the data used here, the 'true' homography matrix is provided by the authors [10] and is available with the database as 'ground truth.' These ground truth homographies were calculated using hand-selected feature matches, refined by an automatic procedure which apparently used H&S corners. This approach is viable providing any errors due to camera distortions are much smaller than the errors due to inaccurate feature matches. If this were not the case, matched points located towards the edges of the image (where lens aberrations have more effect) would reduce the accuracy of the homography — but this has not been observed in practice.

Similar to homography testing, this ground truth homography matrix can be used to classify matched point (points detected and matched by algorithm itself instead of selecting equally-spaced points) into four categories of confusion matrix discussed above. The matched points are verified by projecting detected points using the ground truth homography to find their actual locations in the second image. So, for each detected point, the actual location is known and compared with the algorithm's estimated matched position. If the Euclidean distance between the two is less than some threshold (such as $\tau = 2$) pixels then it is true match; otherwise it is false one. The detected points which are not matched by the algorithm are checked for being false negatives and true negatives. When the detected point is projected using the ground truth homography and the projection is out of image boundary, it is classified as true negative. Conversely, if the projection results in a pixel position inside the image, showing that the match was present but the algorithm fails to identify it, is classified as a false negative.

More precisely, let $\mathbf{P}_i = (x_i, y_i)$ be the location of a feature in the reference image and $\mathbf{P}_j = (x_j, y_j)$ the location of the corresponding feature in a test image. Let $\mathbf{P}'_i = (x'_i, y'_i)$ be the projection of $\mathbf{P}_i$ onto the test image calculated using (3.1), where $\mathbf{H_{gt}}$ is the ground truth transformation matrix represents the homography between the pair of images:

$$\mathbf{P}'_i = \mathbf{H_{gt}} \times \mathbf{P}_i \tag{3.1}$$

This projection is considered correct if

$$||\mathbf{P}'_i - \mathbf{P}_j|| \leq \tau \tag{3.2}$$

for some threshold τ. This procedure is illustrated in Figure 3.11.

- A matched feature of reference image is true positive (TP) if and only if

$$||\mathbf{P}'_i - \mathbf{P}_j|| \leq \tau \tag{3.3}$$

- A match is considered false positive (FP) if and only if

$$||\mathbf{P}_i' - \mathbf{P}_j|| > \tau \tag{3.4}$$

  To find false negative and true negative features, consider a non-matched features of reference image is represented by P_{ii}, a non-matched feature of test image is represented by P_{jj} and $P_{ii}^{'}$ represents the projection of non-matched reference image feature onto test image.

- A non-matched reference image feature P_{ii} is false negative (FN) match if and only if

$$\forall \mathbf{P_{jj}}\ \exists \mathbf{P_{jj}}\ (||\mathbf{P_{jj}} - \mathbf{P_{ii}^{'}}|| \leq \tau) \tag{3.5}$$

- A non-matched image feature is true negative (TN) match if and only if

$$\forall \mathbf{P_{jj}}(||\mathbf{P_{jj}} - \mathbf{P_{ii}^{'}}|| > \tau) \tag{3.6}$$

For comparison purposes, the images are matched using a sample detector and descriptor (SIFT in this case) and the detected points are matched using a nearest neighbour matching technique (described in chapter 2). Results are collected for three different matching thresholds ($0, 0.7\ and\ 1$). A threshold of zero yields no matched points, while for a threshold of 0.7, all points will be matched for which the descriptor difference is less than 0.7. A threshold of 1 means that all points are matched.

3.3.1 Do all metrics agree?

Table 3.5 presents TP etc for the three thresholds; this is equivalent to three confusion matrices. Table 3.6 shows the corresponding measures calculated from the

Table 3.5: Confusion Matrix values for three different nearest neighbour matching (NN) thresholds, showing how negative outcomes are converted into positive outcomes by increasing threshold value

Matching Threshold	**0.0**	**0.7**	**1.0**
TP	0	375	743
FP	0	114	1048
TN	727	640	0
FN	1064	662	0

data in Table 3.5 for these thresholds. The results are also plotted in Figure 3.6 using the curves commonly encountered in the literature. The shaded area in each graph represents the space where the outcomes are consistent with good performance. Let us consider an algorithm to be good if, in each graph, all three points appear in the shaded region. So, according to this criterion, only two of the plots (the accuracy and SS graphs) classify this algorithm as good. But does this reflect the algorithm's true performance?

The ambiguity in the accuracy graph is obvious because it says that algorithm has same accuracy = 0.4, when all of the outcomes are negatively ($\tau = 0$) or positively ($\tau = 1$) classified! Similarly, the ROC, PR, TPr-FPr and F graphs rate this algorithm at their lowest positions (zero values in each case) at zero threshold when all outcomes fall in negative classes, overlooking a large number of true negative results.

Although some measures do highlight poor performance, they do not do so consistently; for example, the high specificity scores suggests that the algorithm would be good at identifying negative outcomes correctly but completely ignores a large number of false negative outcomes for thresholds of 0 and 0.7. Similarly, the high true positive rate with a lower false positive rate in the ROC curve may rank this an algorithm as having good performance at threshold 0.7, overlooking the large number of false negatives. The same is the case with the PR curve

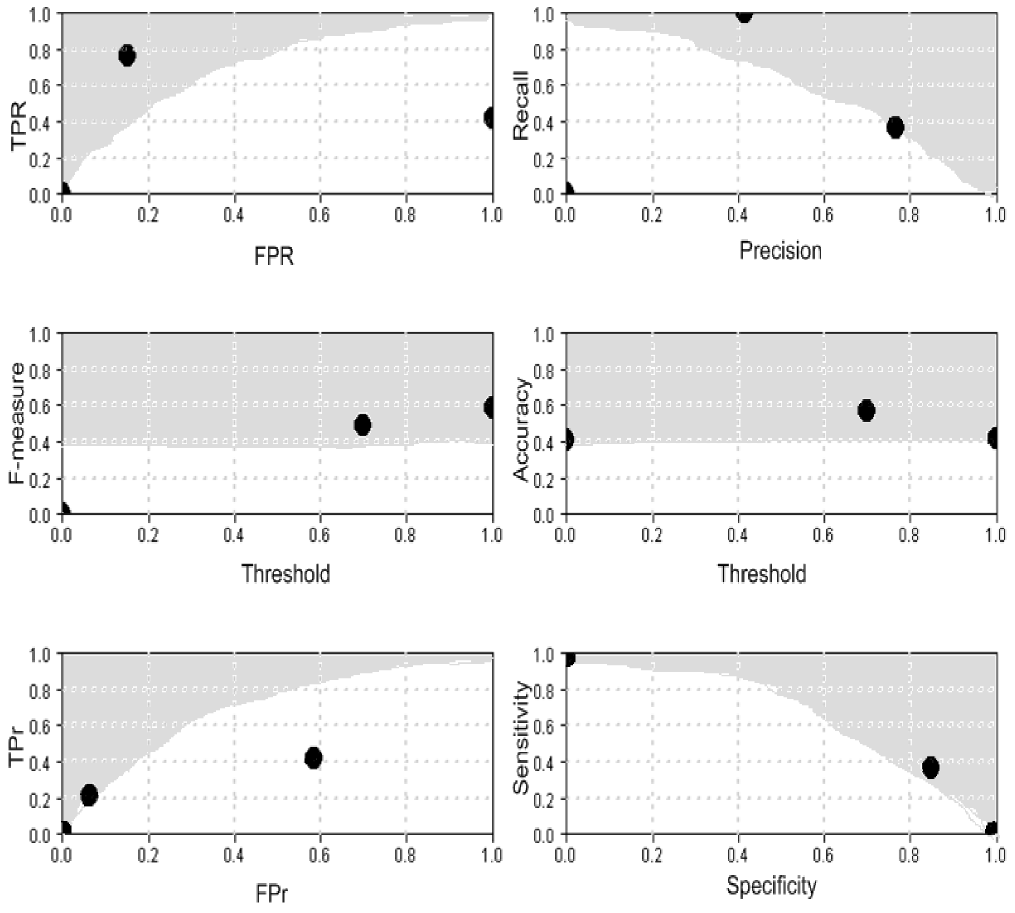


Figure 3.6: 2D plots of performance evaluation measures. The shaded area in each plot shows the area for acceptable performance of an algorithm.

Table 3.6: Different performance metrics calculated from data shown in Table 3.5

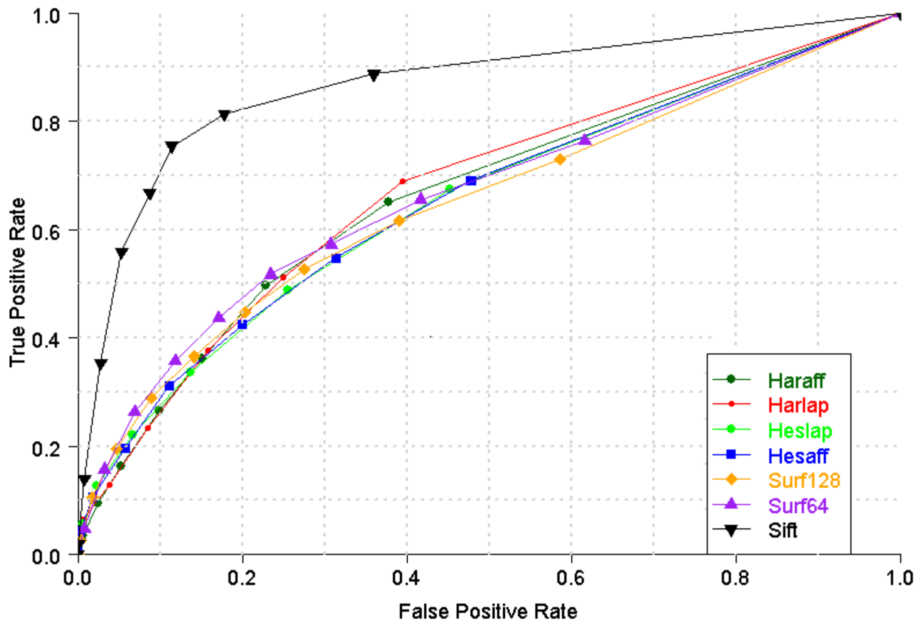
Threshold	**0.0**	**0.7**	**1.0**
FPR	0.0000	0.1512	1.0000
TPR	0.0000	0.7669	0.4149
P	0.0000	0.3616	1.0000
R	0.0000	0.7669	0.4149
Specificity	1.0000	0.8488	0.0000
Sensitivity	0.0000	0.3616	1.0000
Accuracy	0.4059	0.5667	0.4149
F-Measure	0.0000	0.4915	0.5864
FPr	0.0000	0.0637	0.5851
TPr	0.0000	0.2094	0.4149

for thresholds of 0.7 and 1.0. Interestingly, with lower TPr for thresholds 0.7 and 1.0, the TPr-FPr graph seems more representative of the algorithms' actual performances.

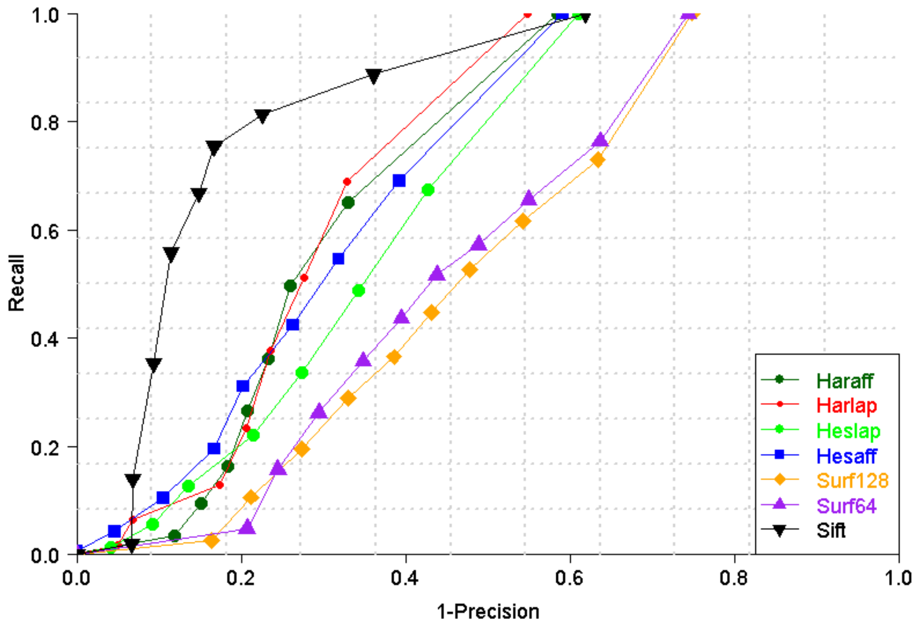
These results highlight some of the weakness of existing performance characterization methods while assessing an algorithm's performance. However, evaluating one algorithm's performance in isolation is usually not required, and hence these methods have been widely used in the literature to compare algorithms performances and produce rankings. To understand their behaviour for comparing multiple algorithms, a number of feature extraction algorithms have been selected for matching features in the same image data and their performances are compared in the next section.

3.3.2 Comparing the performances of multiple algorithms

Table 3.6 summaries a number of well-known algorithms for interest point detection and description; see chapter 2 for their detailed descriptions. Each algo-

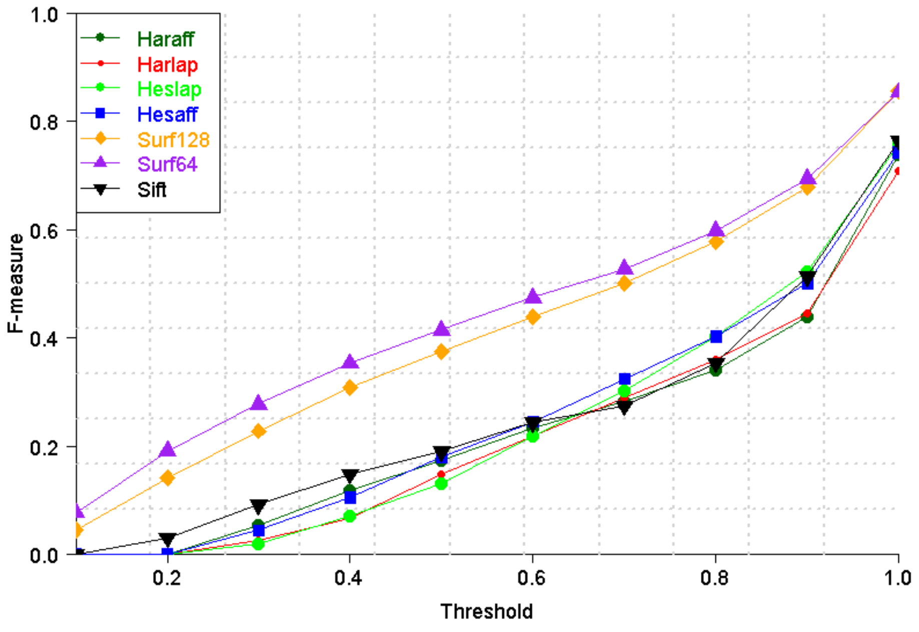


(a)

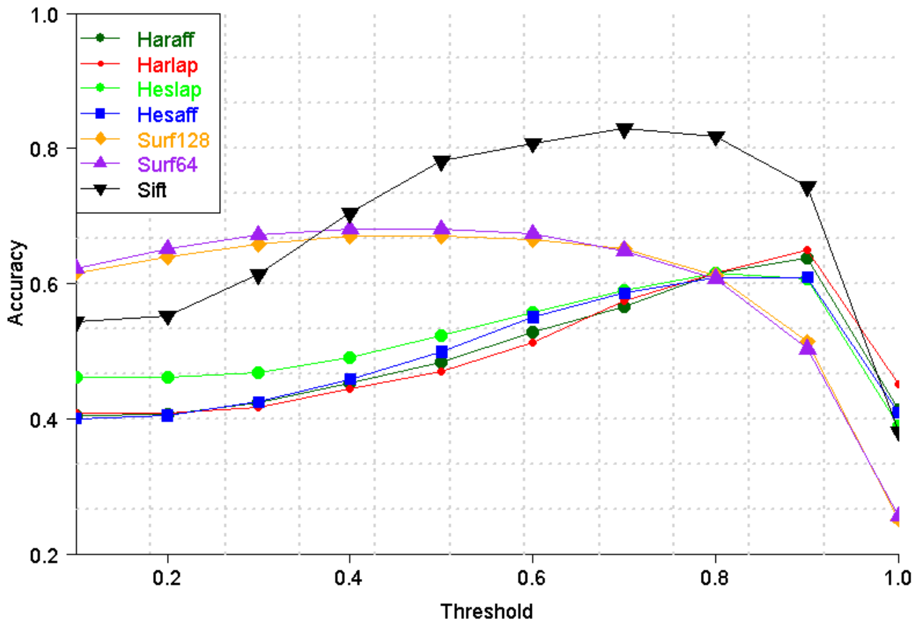


(b)

Figure 3.7: Comparing ROC and PR plot for matching results of Graffiti image 1 and 2.

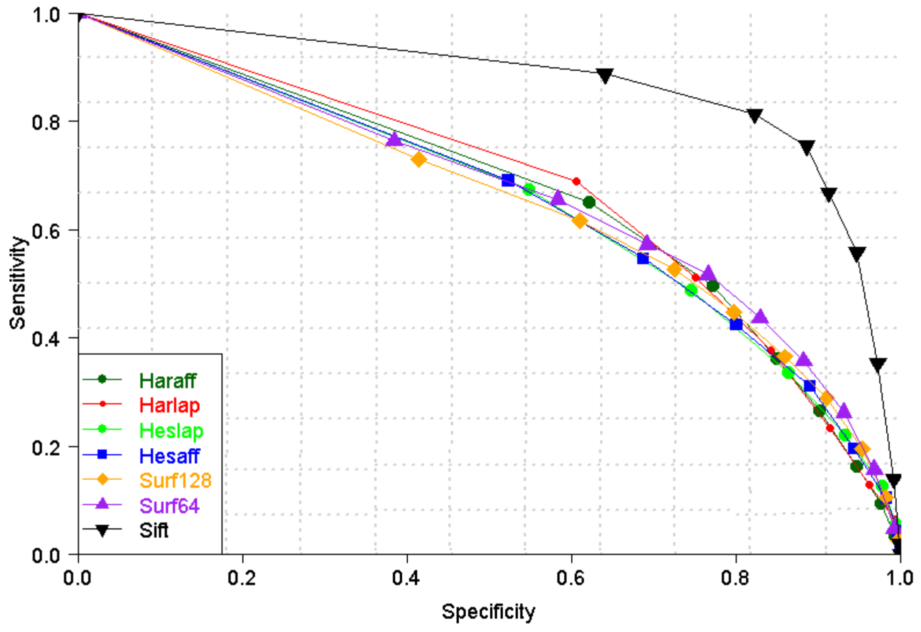


(a)

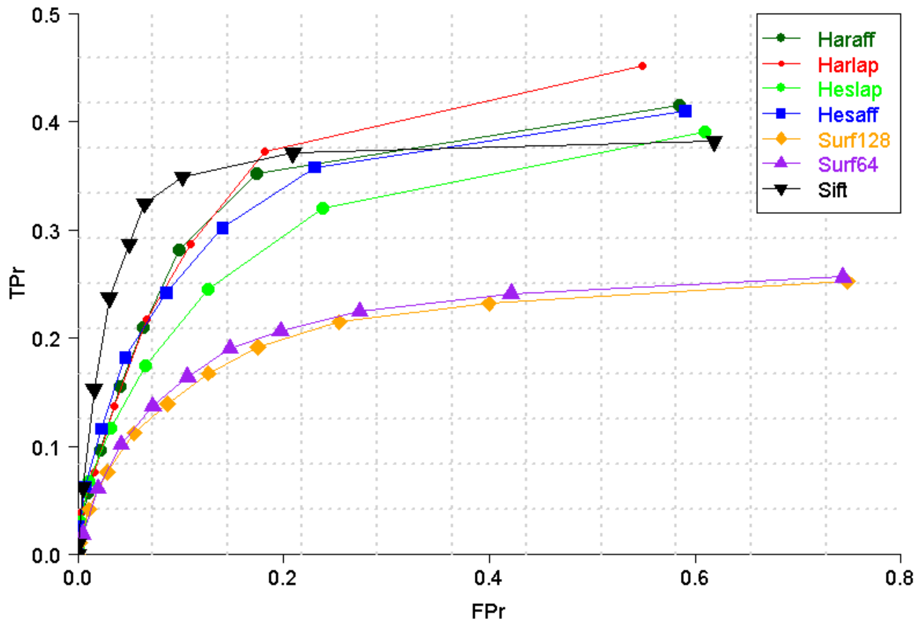


(b)

Figure 3.8: Comparing F-measure and Accuracy plots for matching results of Graffiti image 1 and 2



(a)



(b)

Figure 3.9: Comparing SS and TPr-FPr plots for matching results of Graffiti image 1 and 2.

Table 3.7: Algorithms used to analyse different performance evaluation metrics

Feature operators			
Detector	**Descriptor**	**Descriptor length**	**Matching**
SIFT	Gradient Orientation Histogram	128 bins	Nearest Neighbour
SURF	Haar Wavelet Response	64 bins	″
SURF	Haar Wavelet Response	128 bins	″
Haraff	Gradient Location and Orientation Histogram (GLOH)	128 bins	″
Harlap	″	″	″
Hesaff	″	″	″
Heslap	″	″	″

rithm's output is a list of matched points which are verified and categorized in a confusion matrix using the ground truth homography, as described in section 3.3. This process is repeated for different nearest neighbour matching thresholds and the resultant graphs are shown in Figures 3.7, 3.8 and 3.9. The results presented here are essentially the same as those in Table 3.6 but are produced for number of different matching threshold values (between 0 and 1).

As mentioned before, the hope is that these graphs should show some similarity in the ranking of algorithms based on their performance. The ranking of algorithms can be determined by relationships of the curves in the graphs. Unfortunately, there are obvious discrepancies in the results presented by ROC and ROC curves. According to Figure 3.7b, both versions of SURF show an apparently random behaviour (not good performance); but in Figure 3.7a, all algorithms but SIFT have similar performance. Similarly, when both FN and TN become zero (at a threshold of unity), the ROC curve shows that all algorithms have same TPR and FPR and hence should be considered similar; but this is not the case for the PR and TPr-FPr curves. Figure 3.8b shows a completely different behaviour, where the accuracies of SURF-64 and SURF-128 are significantly better than all other methods except SIFT. This may be happening because, by relaxing the constraint (increasing the threshold), all negative outcomes shift to positive ones, and accuracy and ROC are invariant to this change (a consequence of the combination of confusion matrix values they use). As mentioned earlier, the SS graph in Figure 3.9a is closely related to the ROC curve and hence shares similar properties. Similarly, Figure 3.8a shows SURF-64 and SURF-128 being dominant as they have the highest F-measures, better than SIFT. Although the PR and TPr-FPr ratio graphs mostly agree, the question will still remains valid: do they characterize performance well? Indeed, one might conclude that any algorithm can be presented as performing better than the others by intelligently selecting the most suitable performance measure.

The major drawback of all these graphical methods is that even if they carry any statistical significance, it is not shown. Even error bars do not necessarily indicate that performances are necessarily different in the statistical sense. In any case, these curves may well overlap each other, making it difficult to identify which algorithm is better overall.

Due to these biased performance characterizations of algorithms by different graphical evaluation methods, the author contents that the research community should be looking for other reliable and statistically valid evaluation techniques. The remainder of this chapter explores this.

3.4 Null hypothesis testing

A hypothesis is a way of describing a theory about data. In many cases, a hypothesis can be proven to be right or wrong using evidence. A methodology has been developed over the last few decades that allows for evidence-based decisions to be made about performance, particularly of drug trials. One starts with the so-called *null hypothesis* that (say) a new drug is no better than a current one. The formal definition of this null hypothesis will be

H_o: there is no difference between the two drugs

One can also propose an alternative hypothesis:

H_1: the two drugs have different effects

By gathering the results of a trial employing the two drugs, one can amass evidence as to which hypothesis is better. One assumes that the null hypothesis H_o is correct unless the evidence suggests that it cannot be; hence, null hypothesis testing is inherently conservative. A number of tests can be used for null hypothesis testing, including the χ^2-squared test, t-test, McNemar's test, ANOVA, and so on. An arbitrary level, α, is often used as a cut-off between a statistically significant and a statistically insignificant result.

In this work, two statistical tests are used, McNemar's test and ANOVA. As we shall see, McNemar's test works by exploring where one treatment succeeded and the other failed, ensuring that well-understood binomial statistics apply; indeed, it is sometimes described as a form of the statistical sign test. The test is non-parametric but statistically 'weak' in that it requires a larger amount of evidence to indicate dissimilarity of performance than other, statistically 'stronger' tests. The second test that will be used is ANOVA ("analysis of variance"), which is perhaps best thought of as a generalisation of the t-test to many variates. ANOVA is statistically stronger than McNemar's test but imposes some requirements on the data, including that they are Normally distributed. To be able to employ ANOVA, one needs to ensure that these requirements are met. However, when ANOVA can be employed, less data are required for it to ascertain whether performance differences are significant. Although using less data may reduce the

Table 3.8: Truth table for McNemar's test

	Algorithm A Failed	Algorithm A Succeeded
Algorithm B Failed	N_{ff}	N_{sf}
Algorithm B Succeeded	N_{fs}	N_{ss}

confidence level as P gets affected by the amount of data used for analysis [145], therefore, one has to be careful in evaluating algorithms using statistical tests as well.

3.4.1 McNemar's test

McNemar's test has been widely used in medical research [146–153]; however, it has not been fully explored for vision related algorithms' performance characterization. Therefore, this study explores the use of McNemar's test in null hypothesis testing to ascertain whether it can be used to produce more reliable rankings than the graphical methods presented in the previous section.

McNemar's test is a non-parametric evaluation metric introduced by Quinn McNemar in 1947 [154]. It is used to compare matched pair data using a 2×2 contingency table with a discrete measure to classify the data as similar or dissimilar. The test is used to record the outcomes of two algorithms over multiple tests and therefore it not only counts the number of time an algorithm passed or failed a test but also take into account the total number of tests performed.

For example to compare Algorithm A with Algorithm B, a null hypothesis can be formed by assuming that there is no difference between their performances. One then assesses whether the evidence obtained from testing does or does not support that hypothesis. From the results, one can build up a kind of 'truth table' for a pair of algorithms, which can then be used in McNemar's test.

Table 3.9: Converting Z-scores into confidence limits

Z **value**	**Degree of confidence Two-tailed prediction**	**Degree of confidence One-tailed prediction**
1.645	90%	95%
1.960	95%	97.5%
2.326	98%	99%
2.576	99%	99.5%

Such a table is shown in Table 3.8, where N_{sf} is the number of tests for which algorithm A succeeded and algorithm B failed, and so on. McNemar's test includes calculating the so-called Z-score using

$$Z = \sqrt{\chi^2} \implies \frac{|N_{sf} - N_{fs}| - 1}{\sqrt{N_{sf} + N_{fs}}} \tag{3.7}$$

where the -1 is a continuity correction. This is similar to the χ^2 test:

$$\chi^2 = \frac{(|N_{sf} - N_{fs}| - 1)^2}{N_{sf} + N_{fs}} \tag{3.8}$$

According to central limit theorem, Z should be reliable if $N_{sf} + N_{fs} \gtrsim 20$. If Algorithm A and Algorithm B give similar results, then $Z \approx 0$; as their results diverge, Z increases. It is interesting to note that this expression involves cases where one algorithm succeeds and the other fails, whereas performance evaluation in vision largely focuses on where both algorithms succeed.

McNemar's test involves comparing paired data and therefore has only one degree of freedom. Similar to other statistical tests, one needs to set the level of significance, α, which is used to compare the probability of error for rejecting or accepting the null hypothesis. For $\alpha = 0.05$ (commonly used level of significance: a 95% confidence level, meaning that the results from the algorithms are expected to differ by chance only one time in 20), the critical Z value is 1.96 (refer Table 3.9 for two tailed prediction). The null hypothesis will be rejected if $Z > 1.96$ or

$P < \alpha$ and accepted otherwise. Here, P is probability of Type-I error (incorrect rejection of null hypothesis).

In the physical sciences, it is common to use a more exacting 1-in-1,000 criterion [155]. Values for two-tailed and one-tailed predictions are shown in Table 3.9 as either may be needed, depending on the hypothesis used: if we are assessing whether the performances of two algorithms differ, a two-tailed test should be used; but if we are determining whether one algorithm is better than another, a one-tailed test is needed.

Comparing many algorithms

McNemar's test compares only two algorithms. When several algorithms are to be compared, it is necessary to consider that each pairwise comparison involves a selection from a population of algorithms — the use of multiple comparisons tends to increase the family-wise error rate. In such cases, there are several corrections that one could use, the best-known of which is the Bonferroni correction [156]: for A algorithms, if the significance level for the whole family of tests is to be at most α, then each individual test needs to be performed with a significance level of α/A. The Bonferroni correction is actually a first-order Taylor series expansion of the more general Šidák correction, $1-(1-\alpha)^{1/A}$.

However, there are some concerns over the use of *any* correction [141]. For example, Bonferroni corrections control only the probability of false positives and come at the cost of increasing the probability of false negatives; it may therefore be considered as being too conservative to control the family-wise error rate [157]. There are some other corrections suggested in literature, such as Benjamini & Hochberg (BH) [158] and Benjamini & Yekutieli (BY) [159] corrections which control the expected proportion of false discoveries amongst the rejected

Table 3.10: McNemar's test for evaluating different algorithms for matching Graffiti image 1 and 2 against ground truth using nearest neighbour threshold

	Ground truth Failed	Ground truth Succeeded
Algorithm Failed	0	FP+FN
Algorithm Succeeded	0	TP+TN

hypothesis a less rigid condition than the Bonferroni correction.

These corrections can be applied by adjusting P (calculated probability of error for each test), and the results will be interpreted as: if $P < \alpha$ then reject the null hypothesis and accept the alternate. A comprehensive table of P and the associated corrected value using three different methods (Bonferroni, Benjamini & Hochberg, Benjamini & Yekutieli) is generated for $Z = 0–60$ as Appendix A. The concerns over the application of these corrections are also obvious from Appendix A, where each method can be made to produce different result by adjusting P. If Bonferroni correction rejects null hypothesis ($P < \alpha$), other two corrections accept it ($P > \alpha$), a point of concern. Applying the most conservative correction is the safest approach so in the rest of this thesis, the Bonferroni correction has been applied to every McNemar's test result.

3.4.2 McNemar's test for ROC-like analysis

The purpose of these tests is the same as for other evaluation measures, *i.e.* identifying algorithms which can match two Graffiti images (a reference and a test image) more accurately based on matched features. From these, it is possible to produce a performance-based ranking. McNemar's test is usually applied to a pair of algorithms; but if one algorithm is replaced by the true results ('ground truth') then the result can be compared with the other performance metrics dis-

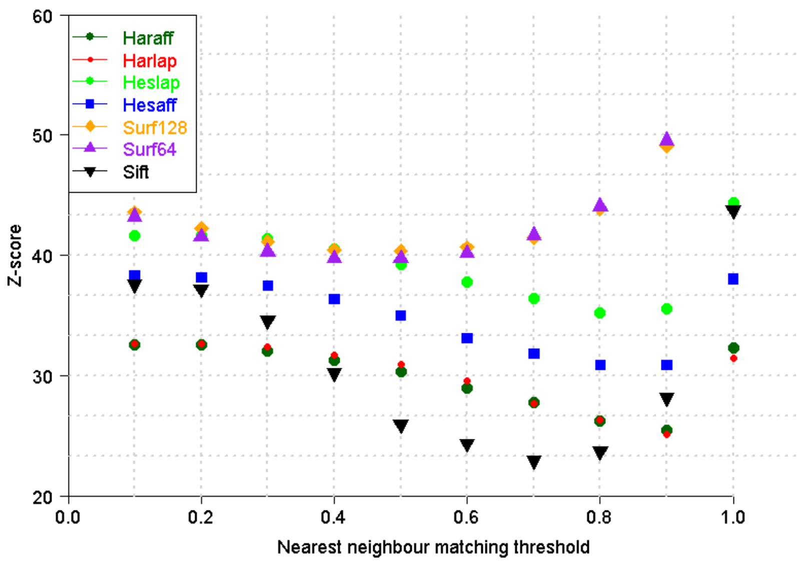


Figure 3.10: McNemar's test results to analyze algorithms' performances against ground truth; Z-score close to zero indicates better performance

cussed earlier. To count an algorithms's success and failure attempts in matching two image, the ground truth homography between the image pair was used to verify each matched point, as described in section 3.3.

The scores for McNemar's test were determined and are shown in Table 3.10. Because the ground truth can never fail, the corresponding columns in the table contain zero. The sum of false positives and false negatives are the cases where ground truth 'succeeded,' because of the correct information about actual matched points' location, but the algorithm did not and therefore produces a wrong result. Similarly, the sum of true positives and true negatives are the cases where the algorithm and ground truth agree.

Here, McNemar's test is used to compare multiple algorithms (seven interest point operators) in pairs; therefore, the Bonferroni correction needs to be applied to reduce the family-wise error rate. There are two ways to apply any correction: one is to adjust the level of significance to obtain similar value as the original significance level (usually 0.05 for a 95% confidence level); and the second method is to adjust the P value for each binary test. Here, the first method, adjusting the significance level α, was adopted for all tests. The number of algorithms under comparison are seven therefore $A = 7$ so to attain 95% confidence, α needs to increase as shown below:

$$1 - (1-\alpha)^{1/A}$$
$$1 - (1-0.32)^{1/7} = 0.05$$

Now, $\alpha_c = 0.32$ is the corrected significance level indicating 95% confidence interval for comparing seven samples. For this α_c, the critical Z-score will be 0.468 and 0.92 for one-tailed and two-tailed tests respectively. Therefore, from here onwards the interpretation of McNemar's test results is that any Z-score less than 0.92 shows no significant performance difference between algorithms.

Z-scores for algorithms' outcomes for each threshold are presented in Figure 3.10. A Z-score closer to zero shows that an algorithm performed more similarly to the ground truth and hence exhibits good performance. The Z-score, PR (Figure 3.7b) and TPr vs FPr graphs (Figure 3.9b) show somewhat similar rankings *i.e.* with SIFT performing best, closely followed by Harlap. However, the TPr-FPr graph has more similarity with the Z-scores.

3.4.3 Homography testing using McNemar's test

Section 3.3 explains how an image pair is matched using a homography matrix. In the previous analysis, the ground truth homography was used to verify matched points, while in homography testing, the matched points obtained from each algorithm is used to calculate an estimated homography (H_e), which is then compared with the ground truth homography (H_{gt}). Figure 3.11 shows a comparison between H_{gt} and H_e. For McNemar's test, some 1000 equally-spaced

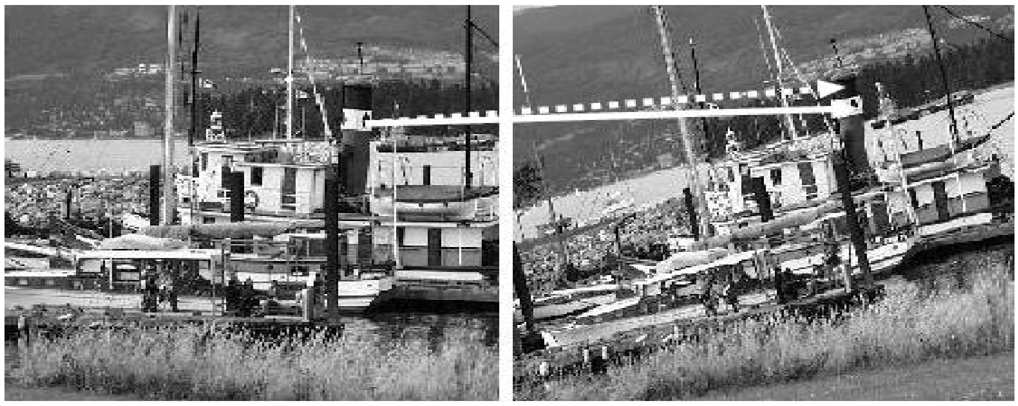


Figure 3.11: Transformation of a point by a homography calculated from feature matches (dotted line) and the ground-truth homography (solid line). This is a failure for $\tau = 2$ pixels.

Table 3.11: Z-scores generated by comparing estimated homographies by algorithms with ground truth homography.

Operators/Threshold	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Haraff	35	35	8.06	22.18	0	17.15	0	15.33	4.48	2
Harlap	35	35	35	35	35	35	12.65	4.48	16.79	2
Hesaff	35	24.45	19.08	18	17	16	15.75	19.72	21.95	10
Heslap	35	35	22	18	15.23	18	18.33	10.34	0	0
SIFT	35	25.38	19.87	12.5	11.3	10	8.89	1.79	1.50	0
SURF-128	35	10.90	10.56	10.29	10.11	10.1	10.17	10.38	10.96	12.27
SURF-64	35	14.41	13.85	13.43	13.25	13.25	13.40	13.90	14.69	16.53

points were selected from a reference image to be projected on a test image using homography matrices estimated through matched points lists from an algorithm under test and the ground truth homography. If the difference between two projections of a point is less than some threshold, then it is a success; otherwise, it is a failure.

Table 3.10 is used to calculate the Z-scores shown in Table 3.11 and plotted in Figure 3.12. There is a limitation of this test *i.e.* if the matched points between two images are less than 4 then homography matrix estimation is not feasible. These cases can be seen in the table with Z-scores of 35. However, this limitation does not affect the overall test results because if there are less than 4 matched points, the algorithm's performance cannot be predicted anyway. To accept the

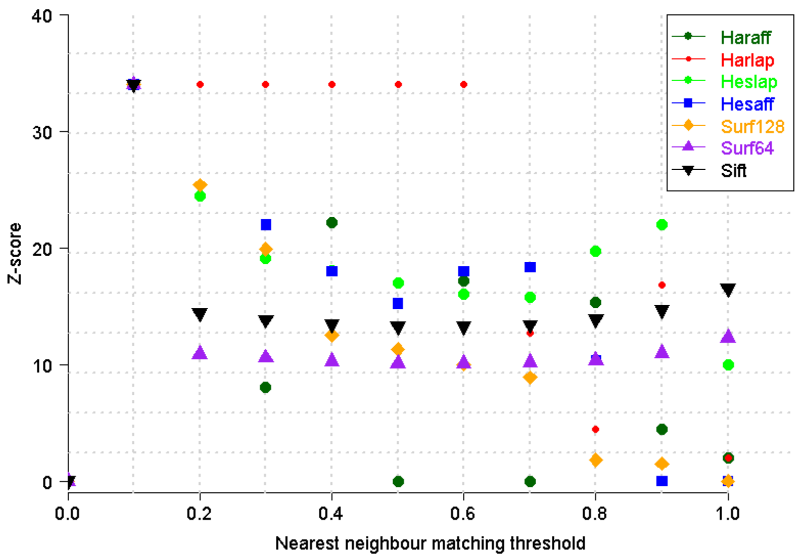


Figure 3.12: Z-scores between algorithms and ground truth homographies for Graffiti image 1 and 2. Z-score of 35 (added manually) denote the cases where homography calculation was not possible due to less than 4 matched points

null hypothesis, one the Z-score should be less than the critical Z value of 1.69; however, all scores are significantly higher than this, making it safe to reject the null hypothesis. For an algorithm to have better performance than others, it needs to show low Z-scores for different thresholds.

Later we will see if this comparison hold any predictive power and helpful in avoiding any pairwise comparisons.

3.4.4 Paired comparisons of algorithms using McNemar's test

Comparing algorithms' outcome with ground truth is a good way to perform performance characterization. However, ground truth data are not available in every case. Therefore, comparing pairs of algorithms is logical and more appropriate. The advantage of doing pairwise comparison is to see which of the algorithms perform better under similar conditions and for the same data, which in

Table 3.12: Z-scores between feature extraction algorithms for matching Graffiti image 1 and 2. To find the better algorithm, follow the arrow-head direction in a pairwise comparison (row-column)

	Haraff	Harlap	Hesaff	Heslap	SURF64	SURF128
SIFT	$\uparrow$ 8.89	$\leftarrow$ 8.89	$\leftarrow$ 12.92	$\leftarrow$ 15.97	$\leftarrow$ 5.40	$\leftarrow$ 2.70
Haraff		$\leftarrow$ 12.65	$\leftarrow$ 15.75	$\leftarrow$ 18.33	$\leftarrow$ 8.60	$\leftarrow$ 5.00
Harlap			$\leftarrow$ 9.27	$\leftarrow$ 13.19	$\leftarrow$ 6.70	0.50
Hesaff				$\leftarrow$ 9.27	1.10	$\leftarrow$ 9.10
Heslap					0	$\uparrow$ 7.20
SURF64						$\uparrow$ 6.90

turn can help finding complementary algorithms, algorithms that exhibit different failure modes. To see how pairwise comparison is different from comparing an algorithm's output with ground truth and producing figure of merit, the following test has been performed.

Homography testing is the criterion used for this comparative study, for the same reasons as in section 3.3, yielding results such as those in Table 3.8. Test scores are calculated for 1000 equally-spaced points per image. For each point, if both algorithms project it within a 2-pixel distance from its correct location (calculated using H_{gt}), both algorithms pass the test, and if projection from both algorithms is more than 2 pixels distance it is counted as a failure. However, if algorithm A projects the point within allowed distance, it is counted for N_{sf}, conversely it is N_{fs}. Calculated Z-scores are presented in Table 3.12. All scores for McNemar's test are calculated from matched points obtained at a matching nearest neighbour threshold of 0.7.

It is interesting to see that the Z-scores given in Figure 3.12 for a threshold of 0.7 correlate to the Z-scores between algorithms given in Table 3.12. In Figure 3.12, Haraff shows a Z-score of zero — which means, at 0.7 threshold, the homography estimated from matches obtained from Haraff is close to the original homography. Note that, in these tables, the arrow-head points towards the algorithm with the higher Z-score. The distance between Harlap and SIFT in Figure 3.12 shows that their difference in the performance is significant, with SIFT showing better performance (lower Z-score). This result is supported by $Z = 8.89$ given in the pairwise comparison. It is evident that if we have ground truth available then comparing algorithms' performances with ground truth using McNemar's test can give a figure of merit and therefore pairwise comparison is not required. However, in the absence of ground truth data, pairwise comparison gives a reliable and statistically-significant ranking. This is not possible using any graphical evaluation method.

3.5 ANOVA: Analysis of variance test

As described in section 3.4.1, performing multiple binary comparisons may increase family-wise error rate (Type-I error), and there remains a point of concern when applying a binary statistical test. ANOVA is a method that is used to perform multiple (more than two) comparisons at the same time without increasing the Type-I error. It is also used for null hypothesis testing for normally distributed data, most commonly in psychological research [160].

ANOVA is used to compare whether more than two groups exhibit a statistically-significant difference in their means (μ)

$$H_o = \mu_1 = \mu_2 = \mu_3 = ... = \mu_n \tag{3.9}$$

where n is the total number of independent groups under comparison. If the test result shows significant difference then there are at least two groups whose means are significantly different from each other. There are different variants of ANOVA test, based on the number of factors that vary. So-called "one-way"

ANOVA is used when one needs to compare data means grouped under a single category; similarly two-way ANOVA is used to compare more than two population means based on two factors or categories, and so on. Before applying ANOVA, there are some conditions for data which need to be checked:

- groups must be independent;
- data in each group must be Normally distributed;
- homogeneity of variance.

The homogeneity of variance criterion means the variances of the groups under analysis should be similar, which can be ascertained using the F_{max} test (Hartley's test) [161]. This test calculates the ratio of the maximum and minimum group variances, called F_{max}, and if this ratio is less than a critical value (obtained from a table), the groups are assumed to have similar variances. However, if the groups' variances do not show homogeneity, then some mathematical treatment is required to prepare the data for ANOVA. This treatment can be calculating natural logarithm of the data or taking its square root. To avoid 0-based arithmetic errors, adding 1 prior to calculation is acceptable. Moreover, the independence and normality of the data can be checked by calculating mean, median and mode of the data for each group. An equivalent mean, median and mode indicates Normally-distributed data.

$$SS = \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{3.10}$$

ANOVA test involves simple algebraical calculations to compare groups by calculating mean square difference between and within groups as shown in Table 3.13, where SS is sum of square differences, calculated using Equation 3.10.

Table 3.13: ANOVA test calculations for k number of groups and n number of data instances per group.

Source of Variation	*SS*	*df*	*MS*	*F*
Between Groups	SS_b	k-1	$MS_b = \frac{SS_b}{k-1}$	$\frac{MS_b}{MS_w}$
Within Groups	SS_w	n-k	$MS_w = \frac{SS_w}{n-k}$	

Table 3.14: Single factor ANOVA test summary

Groups	*Count*	*Sum*	*Mean*	*Variance*
Haraff	1000	576.2359	0.576	0.001
Harlap	1000	1005.323	1.005	0.107
Hesaff	1000	1093.451	1.093	0.161
Heslap	1000	1099.353	1.099	0.402
SIFT	1000	903.7948	0.904	0.123
SURF-64	1000	883.402	0.883	0.072
SURF-128	1000	1109.961	1.110	0.412

To calculate the mean square difference, SS is divided by the number of degrees of freedom (df) for both between and within groups. Commonly, the F-test is used in conjunction with the variance for comparing groups of total deviation using MS between and within groups. F is compared with F_{crit} (based on significance level, α can be determined from $F - table$ [162]). If calculated $F \geq F_{crit}$ or if the probability of error $P \leq \alpha$, then the null hypothesis should be rejected, showing at least two of the groups' mean has statistically significant differences; however, the test does not indicate which mean is different, a limitation of the technique.

To compare the performances of the feature extraction algorithms of Table 3.7 using ANOVA, same image data has been used (Figure 3.4). The test used the same homography matrix testing framework as section 3.4.3 for McNemar's test. As previously discussed, equally spaced points are projected using both ground

Table 3.15: ANOVA test results for comparing group of algorithms shown in Table 3.7 for their matching performance

Source of Variation	*SS*	*df*	*MS*
Between Groups	217.74	6	36.29
Within Groups	1276.86	6993	0.18
	F	P	F_{crit}
	198.75	1.1×10^{-234}	2.10

truth and estimated homography matrices. If a point P_i is projected using both homographies, then $P_e = H_e \times P_i$ and $P_t = H_{gt} \times P_i$ are the projections of that point using the estimated and ground truth homography matrices respectively. Let $d = |P_t - P_e|$ be the difference in their projected positions; this will be close to zero if both homography matrices represent similar transformation and large value otherwise. Hence, this d is used to calculate sum of square difference for ANOVA:

$$SS = \sum_{i=1}^{n} (d_i - \bar{d})^2 \tag{3.11}$$

Before applying ANOVA, the data are checked for basic homogeneity of variances. The distances of false matches makes the data variance too high and non-homogeneous, so some data treatment is required as shown in Figure 3.13. Here, the square roots of data have been used to make variances homogeneous, after which ANOVA is applied and the result shown in Table 3.15.

The result based on $F = 198.75 \gg F_{crit} = 2.10$ and $P = 1.1 \times 10^{-234} \ll \alpha = 0.05$ suggests rejecting the null hypothesis and shows statistically significant differences in the performances of interest point operators, in agreement with the results obtained using McNemar's test. The difference between the two tests (*i.e.* McNemar's test and ANOVA), is that the former used a distance threshold to determine success and failure, but for the latter, d is summed for all points to calculate sum of square differences.

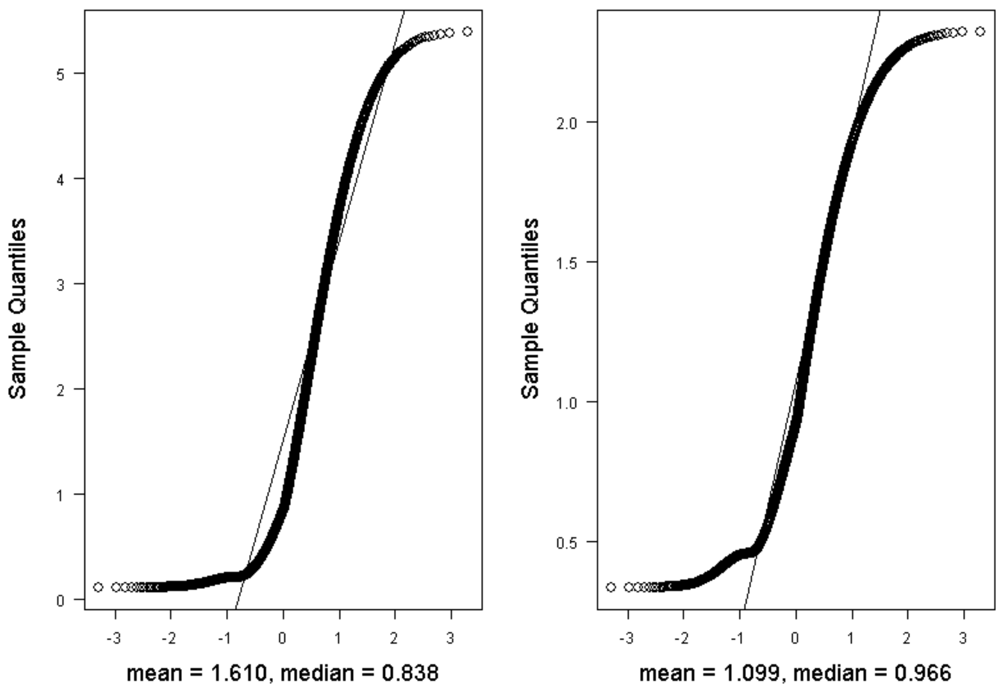


Figure 3.13: Heslap matching results for Graffiti image 1-2, left QQ normal plot shows data which is not Normally distributed, while QQ normal plot on the right shows Normally-distributed data after mathematical transformation (square root of data)

ANOVA is a method that can only highlight the truth about null hypothesis but is unable to tell which group's mean is different. Therefore, some other test is usually applied for this purpose, such as the Student-Newman-Keuls or multiple range test, Tukey-Kramer test or Scheffe test [163].

3.5.1 Multiple range test

This test was developed for multiple comparisons by David B. Duncan [164] using studentized range statistics q_r to compare sets of means. The comparisons involve calculating

$$\sigma_d^2 = \frac{2 \times means\ square(MS)}{n} \tag{3.12}$$

to find the standard deviation of the difference ($s_d = \sqrt{\sigma_d^2}$) between any two means. The comparison starts by listing the samples' means in decreasing order and subtracting lowest mean from the highest one. This difference is compared with Q_{val}, a studentized range value obtained from a Q table for $n-1$ degrees of freedom. If this difference is greater than $Q_{\sigma_d} = Q_{val} * s_d$, then the mean's difference is statistically significant. The process is repeated by calculating the difference of second lowest from highest mean and comparing it with Q_{σ_d}, and so on. The results can be summarized as Table 3.16, where any mean marked with an asterisk has a sample mean whose difference is not statistically significant.

Table 3.16 presents the mean difference of algorithms compared using ANOVA (the algorithms' means are shown in Table 3.14). The results are somewhat similar to McNemar's paired data comparison (Table 3.12). The Haraff detector performs better than the others, exhibiting the lowest mean (0.576) and lowest variance (0.001) in Table 3.14. However, from mean difference analysis using multiple range test, there is no statistically significance performance difference between Haraff, SIFT, SURF 64 and Hessian Laplace. Similarly, SURF 64 and

Table 3.16: Multiple Range Test results for finding algorithms which has statistically significant difference in their means. "*" indicate statistically non-significant difference in mean value.

SURF 128	0.534	0.227	0.206	0.105	0.017	0.011
Heslap	*0.523	0.216	0.196	0.094	0.006	—
Hesaff	0.517	0.210	0.190	0.088	—	
Harlap	0.429	0.122	0.102	—		
SIFT	*0.328	*0.020	—			
SURF 64	*0.307	—				
Operators	Haraff	SURF 64	SIFT	Harlap	Hesaff	Heslap

SIFT's performance are also similar.

Although using McNemar's test these performance similarities do not appear, it is may be because, while preparing data for ANOVA, one has to apply some mathematical treatments to fulfil its requirements — in this case, calculating the square roots of the data to make its variance homogeneous and normally distributed. The nature of this mathematical treatment does affect the overall results, as can be ascertained by changing the mathematical function from square root to natural logarithm. Currently there are no known conditions according to which a mathematical function can be chosen for data normalization.

3.6 Remarks

Performance characterization is a sensitive problem and therefore needs to be dealt with carefully. Graphical methods for evaluation appear to be unreliable and sometimes misleading. The use of statistically reliable methods is more rigorous. To explore this, McNemar's test has been used to carry out a statistically valid performance characterization of vision algorithms relevant to navigation for the blind (and many other applications). It is also important to note that McNemar's test can be used to carry out tasks such as ranking algorithms based on their performance, in which case comparison between algorithms and ground truth is sufficient. Conversely, by carrying out a study of pairs of algorithms allows one to find those that are complementary. As McNemar's test alone can be criticised because it involves assigning an arbitrary threshold to distinguish success from failure, a companion study using ANOVA has been carried out to see whether a similar characterization is obtained. Although the testing procedures for the tests are slightly different, still the results are broadly similar.

Another approach can be the use of multiple performance evaluation methods to rank algorithms. If more than one method yield the same ranking, one has more confidence in the results. However, one has to be careful in the selection of methods because those that are invariant to data distribution will not reflect the actual performance; this is the case with ROC curves, which will not be consistent a non-invariant method such as McNemar's test. This study suggest the use of ANOVA to identify whether there are performance differences between several algorithms, subsequently employing McNemar's test to rank them. Because ANOVA does not tell us which algorithms are significantly different from each other, other tests can be applied for this purpose, the most common being the multiple range test, Scheffe' and Tukey test [165].

CHAPTER 4

SAMPLE SIZE AND VARIABILITY

4.1 Introduction

Performance evaluation studies in other domains regularly use statistical hypothesis tests, such as the χ^2 test, *t*-test, McNemar's test or variance analysis test [146–153, 160, 166–168], though very few of them focus on the amount of data and its variability. Ensuring the dataset is large enough and exhibits enough variability is as important for vision research data as any other mathematical or statistical discipline if the results are to be generalized. Even sophisticated and statistically reliable evaluation techniques may produce misleading results if the sample size is not sufficiently large. In vision research, the algorithms are widely tested on a number of images, though the amount of image data employed is rarely large in the statistical sense.

This chapter uses two widely-used databases of different numbers of images to ascertain whether the performance differences calculated for small number of images reflect the general trends of algorithms. Section 4.2 introduces these databases and their relevant sources. To avoid confusion, the term 'database' in this thesis refers to a collection of images which is intended for evaluation purposes, while 'dataset' is one component of a database, typically images of the same scene.

Firstly, in section 4.3, McNemar's test is used to determine how many points need to be projected from one image to another (using the procedure described in the previous chapter) to produce consistent results. Then section 4.4 goes on to explore how many times this projection procedure must be done in order to identify performance differences consistently. This essentially determines how large a dataset is required. Sections 4.5 and 4.6 present performance analysis of multiple feature operators for standard datasets and explore the inconsistencies appear in results for large and small database of images, using both McNemar's test and ANOVA. The two statistical tests are compared, both in terms of results and ease of use.

Section 4.7 goes on to explore the interplay between image content and dataset size. This is done by dividing datasets from large database into many small subsets and ascertaining whether they produce similar ranking of algorithms as the whole database; as the image content is same in all images, one might expect the results to be consistent.

Finally, section 4.8 concludes the discussion by presenting some rules (or rules of thumb) about selecting an appropriate dataset size and a proper evaluation framework for statistically-valid performance comparisons of multiple algorithms.

Table 4.1: Small database of images

Dataset	**Transformation**	**Number of images**
Bikes	blur	6
Trees		6
Graffiti	viewpoint change	6
Wall		6
Bark	zoom + rotation	6
Boat		6
UBC	JPEG compression	6
Leuven	illumination	6

Table 4.2: Large database of Images

Dataset	**Transformation**	**Number of images**
Asterix	zoom	16
BIP		8
Crolle		7
East-Park	zoom + rotation	10
East-South		9
Ensimag		10
Laptop		21
Resid		10
Laptop_rs	rotation	13
Mars		18
Monet		18
NewYork		35
VanGogh		16

4.2 Datasets

In order to assess the matching performance of an algorithm using the approach described in the previous chapter, an image pair is required. This image pair should belong to same scene or have some overlap which can be matched. There can be different geometric transformations applied to images, such as rotation and translation, obtained by rotating or moving the camera position. Similarly, different photometric transformations may be present due illumination changes, compression and blur can occur in images, due to the lighting conditions, image

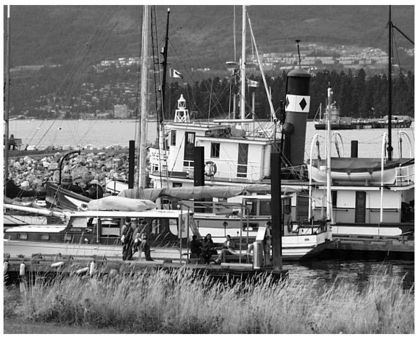
(a) Bikes

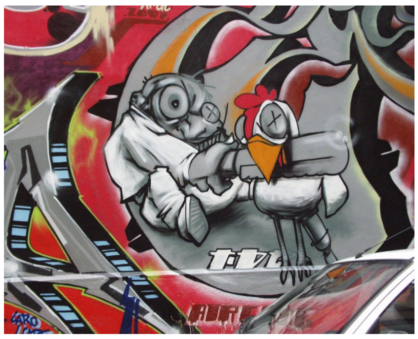
(b) Trees

(c) UBC

(d) Leuven

(e) Bark


(f) Boat


(g) Graffiti

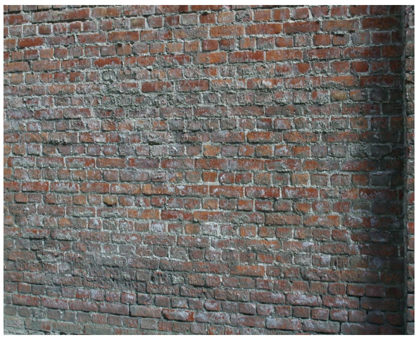
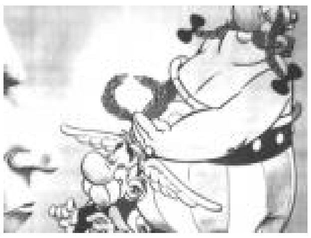
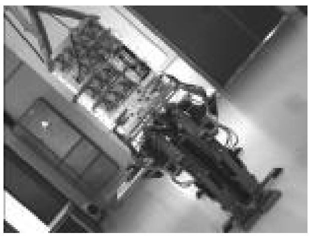

(h) Wall

Figure 4.1: Database 1: Small database of eight datasets


(a) Asterix


(b) BIP

(c) Crolle

(d) East-Park

(e) East-South

(f) Ensimag

(g) Laptop

(h) Resid

(i) Laptop_rs

(j) Mars

(k) Monet

(l) New York

(m) Van Gogh

Figure 4.2: Database 2: Large database of thirteen datasets

storing format and changes in the camera's optical system respectively.

For local image feature matching, the most widely used database of images was introduced in [10]. This database[1] comprises several datasets of real images with different geometric and photometric transformations; this was made publicly-available and several subsequent studies have also used it [11,79,82,86–90,132,136,169–184]. This work employs it too; and it is summarized in Table 4.1.

To assess whether the amount of data affects the relative performances of interest operators, this work also employs a second database.[2] This was collected by the researchers who devised the smaller database discussed above, perhaps implicitly indicating that they believe the small database is really too small. This larger database contains 191 images in 13 datasets (Table 4.2). Different datasets within each database encompass geometric and photometric transformations that include zoom, rotation, viewpoint change, blurring, change in illumination and JPEG compression. All images in each dataset are planar scenes or taken with a fixed camera position, so each image pair is related by a homography (transformation matrix), which is supplied along with the imagery as 'ground truth.' The method of calculating ground truth homography matrices between an image pair was discussed in chapter 3.

4.3 How many points are required to produce consistent results?

The objective here is to establish the number of points that need to be projected from one image to another using a calculated homgraphy in order for it to produce consistent results. This is done by starting with a small number of regularly-

[1] http://www.robots.ox.ac.uk/~vgg/research/affine/
[2] http://www.featurespace.org/

Table 4.3: Z-scores calculated between SIFT and SURF128 for different number of points selected for homography testing between an image pair showed in first column. Z-score of 0 means similar performance of both algorithms mostly occurred.

Number of Points	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
Bark 1-2	3.75	3.75	3.75	4.25	4.25	4.25	4.25	4.25	4.25	4.25
Bark 1-3	0	0	0	0	4.8	4.8	4.8	4.8	4.8	4.8
Bark 1-4	0	0	0	1.15	1.5	1.5	1.5	1.5	1.5	1.5
Bark 1-5	0	0	0	0	0	0	4.59	11	11.53	11.53
Graffiti 1-2	0	0	0	0	0	0	6.56	11.96	15.59	16.19
Graffiti 1-3	8.89	8.89	8.89	8.89	11.66	15.36	18.33	20.88	23.07	23.07
Graffiti 1-4	9.9	14.07	17.26	19.95	18.85	18.85	15.37	15.37	15.37	15.37
Graffiti 1-5	0	0	0	0	0	0	0	0	0	0

spaced points for projection, then increasing the number of points. When enough points are projected, the results become consistent; if consistency is not reached then clearly this method would be inappropriate for assessing performance.

Table 4.3 presents results for the well-known Graffiti and Bark datasets from the small database of images. Both of these datasets contain images with complex transformations: the Graffiti images have been taken from different viewpoints while the Bark images are zoomed and rotated. McNemar's test is applied to find the number of correct and false projections for two algorithms, SIFT and SURF-128, and the Z-scores between them are presented in Table 4.3. Shaded cells in the table indicate the point at which a significant result is obtained, highlighting the number of points required to obtain consistent results (for the particular image data). It is also true that, for some image data, 100 points are enough to establish the performance differences between operators, such as for Bark images 1 and 4, where the result remained insignificant even for 1000 points ($Z < 1.96$).

Similarly, for Graffiti image pairs 1-3, 1-4 and 1-5, the evaluation shows a similar trend for different numbers of points. However, for the rest of the images, at least 700 points are required to obtain consistent results. From this, we are able to conclude that, for homography testing, the number of points should be greater than 700.

In principle, it would be helpful if the conclusion obtained using McNemar's test were confirmed by some other statistical test, perhaps ANOVA or the *t*-test. Both ANOVA and the *t*-test requires the data to be normally distributed — which is not in this case, as shown in Figures 4.3 and 4.4.

4.4 How many image pairs are required to produce consistent results?

Having established that more than 700 points need to projected between a pair of images, we are now able to ask how many image pairs are required to produce consistent results. The same general approach as in previous section is adopted, i.e. starting with a small number of image pairs and increasing the number; again the aim is to find a point at which results become consistent. This establishes the minimum number of images that is required in a dataset. Again, this is done using McNemar's test for two algorithms, SIFT and SURF-128.

Five images from the each of the New York, Laptop, Mars and Asterix datasets of the large database were selected as these contain the largest number of images (35, 21, 18 and 16 respectively). A subset of 5 image pairs is used as starting point because this is the size of the datasets in the small database. If the number of images do not affect the performance evaluation results, than Z-scores of small subsets should be similar to the result from the whole dataset. However, the re-

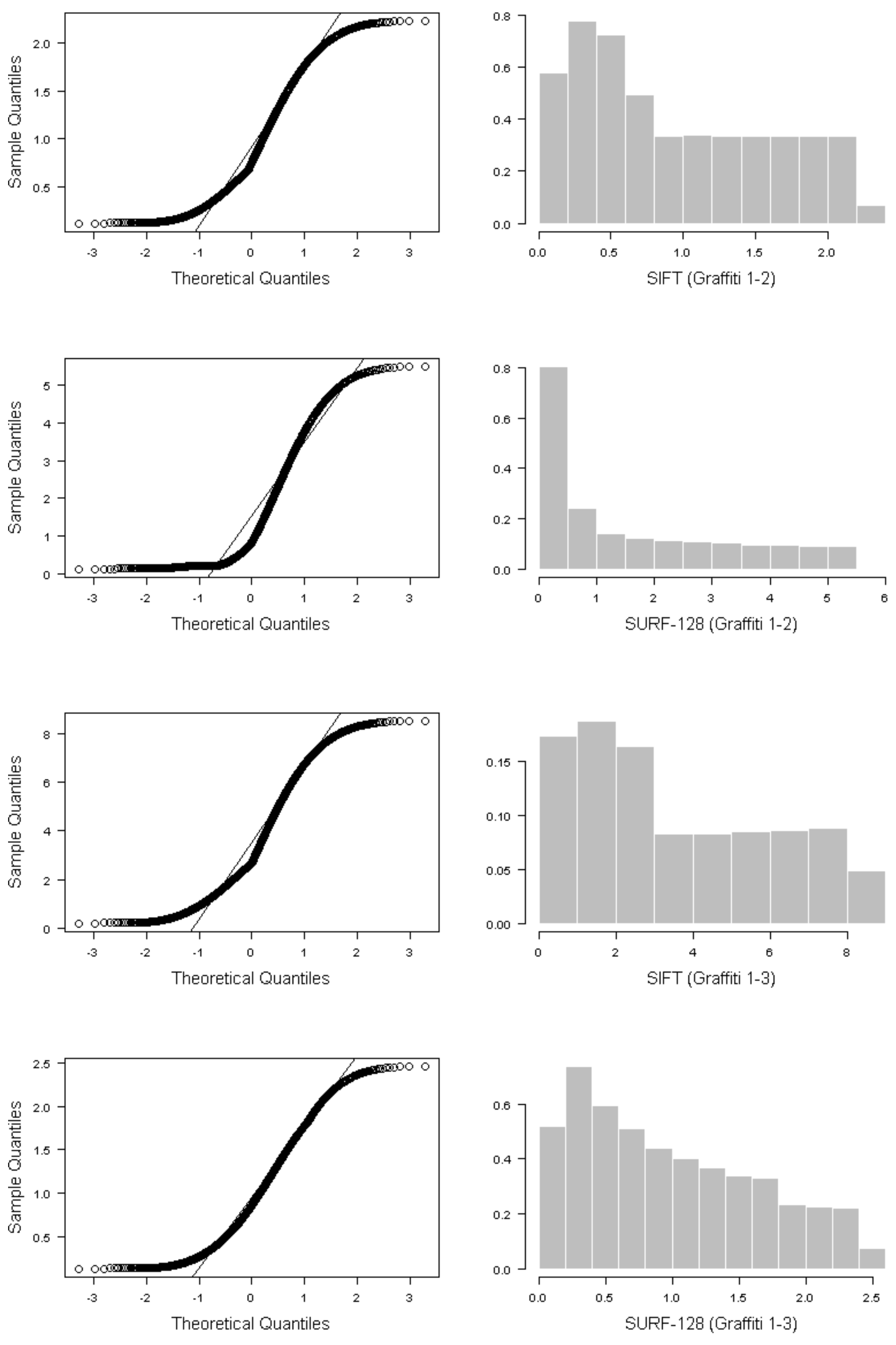


Figure 4.3: Data (Projection error of 1000 image points by estimated homograpy matrix) distribution generated by SIFT and SURF-128 for Graffiti images. First column shows Q-Q normal plots where the data should be aligned to theoritical line, and if not, shows non-normal distribution of data which can also be seen from probability density distribution plots in the second column.

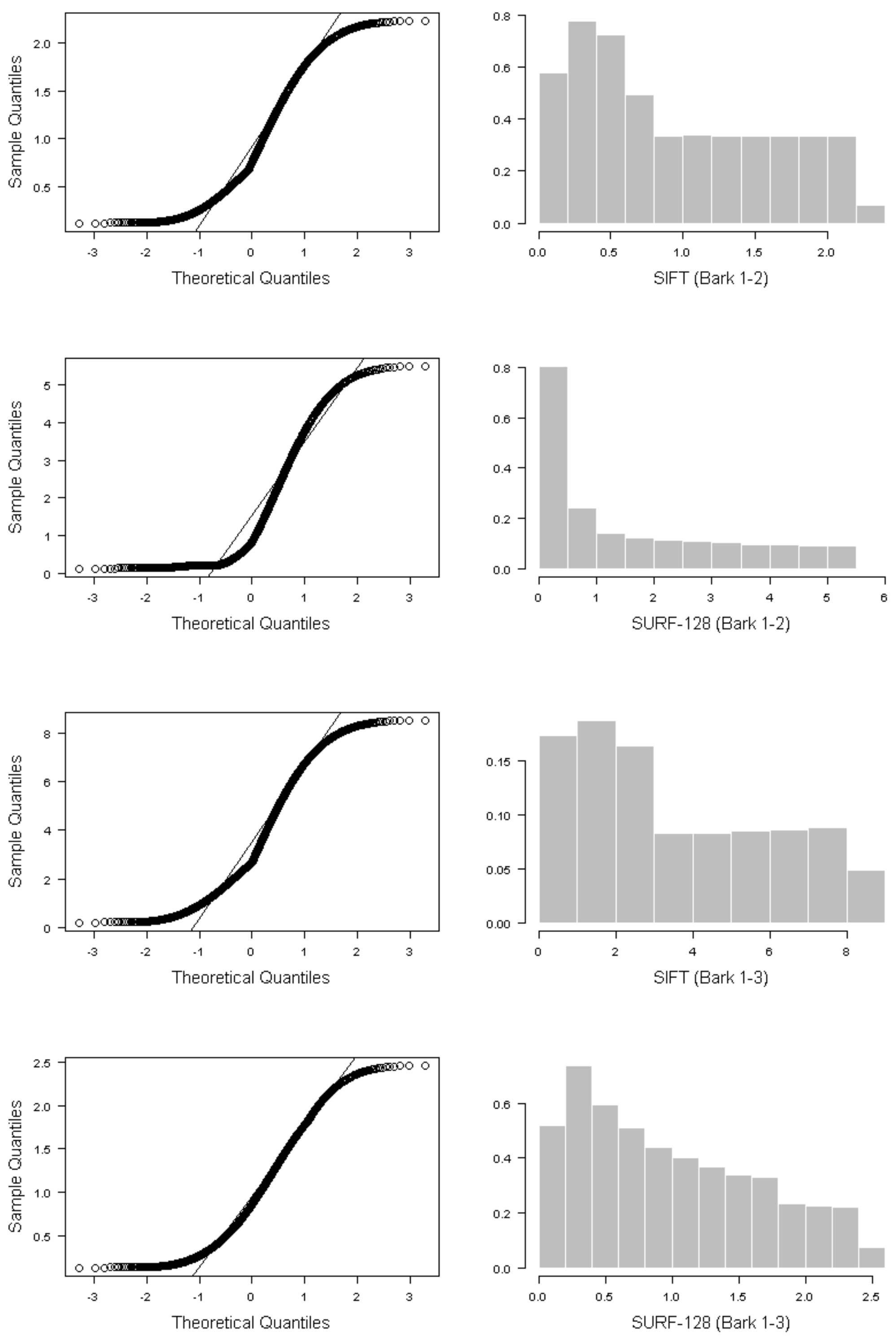


Figure 4.4: Data (Projection error of 1000 image points by estimated homograpy matrix) distribution generated by SIFT and SURF-128 for Bark images. First column shows Q-Q normal plots where the data should be aligned to theoretical line, and if not, shows non-normal distribution of data which can also be seen from probability density distribution plots in the second column.

Table 4.4: Z-scores calculated between SIFT and SURF-128 to identify the minimum number of image pairs required for performance evaluation. The Z-scores less than 1.96 (critical Z-score) is considered non-significant. Z-score with "*" sign represent a case where SURF-128 performed better than SIFT, in all other cases SIFT outperformed SURF-128.

New York dataset (35 images)

Number of Image Pairs	5	10	15	17	34
Set 1	12.62	7.00	7.52	4.92	18.84
Set 2	*2.89	3.69	15.83	21.42	
Set 3	3.41	18.29			
Set 4	1.67				
Set 5	12.17				

Laptop dataset (21 images)

Number of Image Pairs	5	10	15	20
Set 1	0.00	0.00	11.40	11.40
Set 2	0.00	11.40		
Set 3	0.00			
Set 4	11.36			

Monet dataset (18 images)

Number of Image Pairs	5	10	15	17
Set 1	26.27	26.53	36.91	40.45
Set 2	3.47	30.50		
Set 3	25.61			

Asterix dataset (16 images)

Number of Image Pairs	5	10	15
Set 1	3.34	1.66	7.48
Set 2	0.99	6.89	
Set 3	12.90		

sults presented in Table 4.4 show that if the performance difference between two algorithms is insignificant for 5 pairs of images (set 4 of New York dataset in the Table 4.4), it subsequently becomes significant when the number of image pairs are increased. Similarly, set 2 of the same dataset shows that SURF-128 is significantly better than SIFT; however, this result is not occur when the number of image pairs were increased in the other sets. Similarly, there is no performance difference between SIFT and SURF-128 for three sets of the Laptop dataset with 5 image pairs, but this changes for the sets containing 15 or more image pairs. The same evaluation differences can be seen in the Monet and Asterix datasets. From these results, a rule of thumb can be defined that at least 15 image pairs are required to obtain consistent performance evaluation results.

4.5 Exploring the performances of feature operators

So far, only SIFT and SURF have been used in the results presented. These were selected because they are widely used and because they have been found in preliminary experiments to be representative of a wide range of feature operators. However, there are many feature detectors and descriptors apart from SIFT and SURF, and the framework of null hypothesis testing using McNemar's test or ANOVA allows us to explore whether these operators have statistically significant differences in performance. Hence, the set of feature operators described in chapter 2 are used to evaluate their performances for large and small databases of images.

As before, some 1000 uniformly-spaced points were selected from a reference image and projected using both the ground-truth homography and estimated homographies from all seven algorithms under study. Algorithms are compared with each other using McNemar's test and ANOVA in the null hypothesis framework discussed in chapter 3. Although it has been established that 5 image pairs are not sufficient to draw statistically valid conclusions by comparing SIFT and SURF-128 for small sets of image pairs, this database is used here because it has been so widely used in the literature for evaluations [11, 79, 82, 86–90, 132, 136, 169–184]. These results will give a way to ascertain any inconsistencies present in the literature.

As discussed before, performing multiple binary comparisons tends to increase the family-wise error rate, and this bias can be reduced using corrections. Here, as seven algorithms are involved, the Bonferroni correction [141] is used as described in chapter 3 to adjust the significance level. This results in the significance level being $\alpha = 0.32$, for which the critical Z-score is 0.92 for two-tailed test. Hence, any Z-score below 0.92 is considered statistically insignificant.

4.5.1 Results using the small database

Results for all eight datasets of the small database (shown in Figure 4.1) are recorded in Table 4.5, grouped according to the image transformations involved. An easy inspection method is to follow the ranks generated for each algorithm in the last column for each dataset. None of the algorithms appears to be best for matching images with all kind of transformations. Of course, we need to bear in mind that the number of image pairs — 10 for first three sets and 5 for the last two — are not sufficient to draw any statistically valid conclusion according to the rule of thumb established earlier. According to these results, SIFT is robust for matching all images except for viewpoint change. Similarly, Harris-affine with GLOH descriptor appears to be a strong combination of detector and descriptor for matching images under view-point change and change in illumination. SURF-128 showed, unexpectedly, to be the worst algorithms for matching zoomed and rotated images, a contradiction of [79].

Table 4.5: Z-scores of feature operators for datasets with different transformations in small image database. To find the better algorithm, follow the arrow-head direction in a pairwise comparison (row-column). Last column presents the overall score of the algorithm equal to the sum of arrows pointing towards the algorithm in a row and a column (indicating its superiority over others). The Z-score less than 0.92 shows similarity in performance therefor, does not contribute towards score calculation. An algorithm's rank is the sum of arrows pointing towards it in a row or a column.

Operators	SURF-64	SURF-128	Haraff	Harlap	Hesaff	Heslap	Score
			Blurring (Bikes + Trees)				
SIFT	↑ 3.31	← 10.25	← 40.32	← 28.87	← 38.55	← 9.46	5
SURF-64		↑ 3.99	← 41.30	← 30.51	← 35.98	← 13.84	5
SURF-128			← 41.72	← 33.75	← 43.18	← 15.04	5
Haraff				↑ 13.50	↑ 1.09	↑ 33.86	0
Harlap					← 10.84	↑ 18.84	2
Hesaff						↑ 32.64	1
Heslap							3
			Viewpoint change (Wall +Graffiti)				
SIFT	↑ 4.20	0.79	↑ 26.98	← 5.08	↑ 14.58	← 18.58	2
SURF-64		↑ 2.83	↑ 19.78	← 9.32	↑ 10.61	← 22.66	3
SURF-128			↑ 19.11	← 2.39	↑ 12.48	← 19.84	4
Haraff				← 17.20	← 9.50	← 29.69	6
Harlap					↑ 3.47	← 12.53	1
Hesaff						← 26.67	5
Heslap							0
			Zoom+Rotation (Bark + Boat)				
SIFT	← 29.88	← 30.13	← 22.08	↑ 1.65	← 31.10	← 23.85	5
SURF-64		← 4.07	↑ 6.18	↑ 24.02	← 1.63	↑ 6.05	2
SURF-128			↑ 15.00	↑ 28.33	↑ 2.23	↑ 13.29	0
Haraff				↑ 22.63	← 7.35	0.45	3
Harlap					← 26.56	← 24.56	6
Hesaff						↑ 8.18	1
Heslap							3
			JPEG Compression (UBC)				
SIFT	← 14.73	↑ 9.22	← 8.74	↑ 8.72	← 9.33	← 2.47	4
SURF-64		↑ 18.65	↑ 4.67	↑ 17.18	↑ 11.31	↑ 14.46	0
SURF-128			← 14.10	← 4.73	← 12.90	← 9.59	6
Haraff				↑ 14.11	↑ 1.96	↑ 8.00	1
Harlap					← 12.85	← 9.17	5
Hesaff						↑ 8.89	2
Heslap							3
			Change in Illumination (Leuven)				
SIFT	← 5.62	← 8.05	↑ 11.27	← 32.36	← 19.60	← 16.87	5
SURF-64		← 3.69	↑ 14.38	← 31.97	← 17.76	← 9.36	4
SURF-128			↑ 15.25	← 31.26	← 16.85	← 7.11	3
Haraff				← 36.15	← 25.01	← 22.60	6
Harlap					↑ 21.80	↑ 23.82	0
Hesaff						↑ 8.29	1
Heslap							2

Table 4.6: Rankings of feature operators for small database, generated by the counting number of arrow-heads pointing towards each algorithm in rows and columns of Table 4.5. The table is sorted on total score of algorithms, hence the top most is the one with highest rank in the pool.

Operators	**Blur**	**View Point Change**	**Zoom + Rotation**	**JPEG compression**	**Change in illumination**	**Overall Score**
SIFT	5	2	5	4	5	21
SURF-128	5	4	0	6	3	18
Haraff	0	6	3	1	6	16
SURF-64	5	4	2	0	4	15
Harlap	2	1	6	5	0	14
Hesaff	2	5	1	2	1	11
Heslap	3	0	3	3	2	11

The overall ranking is shown by the order of algorithms in Table 4.6, according to which SIFT, SURF-128 and Harris-affine show statistically better performance when compared with other algorithms. Hesaff performed better only for one type of images, i.e. matching images with different view-points.

4.5.2 Results using the large database

These experiments allow us to ascertain whether the small database contains enough images to characterize the performances of algorithms: if differences are obtained using a larger database, we should be concerned that there are not enough. The larger database comprises 191 images in 13 datasets (Table 4.2), and it was used in exactly the same way as described in the previous section.

In order to demonstrate an algorithm's behaviour for a particular transformation between image pairs, a summary of these results is presented in Table 4.7. This table identifies performance differences more clearly than ROC or Precision-Recall curves and has the advantage of associating a statistical confidence with each comparison. The Z-scores and directions of the arrows show that SIFT's detector and descriptor are effective in identifying stable features under geometric transformations. The performance of SURF-128 closely follows that of SIFT but

Table 4.7: Z-scores of feature operators for three different transformations. To find better algorithm follow the arrow-head direction in pairwise comparison (row-column). Last column presents the overall score of the algorithm equal to the sum of arrows pointing towards the algorithm in a row and a column (indicating its superiority over others). The Z-score less than 0.92 shows similarity in performance therefor, does not contribute towards score calculation.

Operators	SURF-64	SURF-128	Haraff	Harlap	Hesaff	Heslap	Score
Zoom (Asterix, BIP, Crolle)							
SIFT	←5.99	←3.42	←26.63	←25.961	←50.05	←48.61	6
SURF-64		↑5.23	←20.48	←20.25	←45.18	←45.03	4
SURF-128			←24.78	←22.59	←45.20	←42.56	5
Haraff				←2.02	←35.24	←44.54	3
Harlap					←35.85	←42.13	2
Hesaff						←26.68	1
Heslap						-	0
Rotation (East Park, East South, Ensimag, Laptop, Resid)							
SIFT	←36.19	←46.09	←5.36	←8.15	←52.08	←55.38	6
SURF-64		←17.68	↑29.35	↑25.08	←28.22	←28.79	3
SURF-128			↑41.77	↑34.80	←18.95	←21.10	2
Haraff				←4.69	←52.87	←52.41	5
Harlap					←49.51	←47.73	4
Hesaff						←3.24	1
Heslap						-	0
Zoom + Rotation (Laptop_rs, Mars, Monet, New York, VanGogh)							
SIFT	←10.66	←11.68	←67.07	←58.86	←64.57	←88.36	6
SURF-64		↑7.99	←55.40	←47.89	←51.72	←78.24	4
SURF-128			←61.58	←53.12	←56.35	←81.95	5
Haraff				↑11.21	↑3.68	←38.77	1
Harlap					←5.58	←44.29	3
Hesaff						←40.28	2
Heslap						-	0

Table 4.8: Rankings of feature operators for larger database, generated by counting the number of arrow-heads pointing towards each algorithm in rows and columns of Table 4.7. The table is sorted by score of algorithms, hence the top most is the one with highest rank in the pool.

	Zoom	**Zoom + Rotation**	**Rotation**	**Overall Score**
SIFT	6	6	6	**18**
SURF-128	5	2	5	**12**
SURF-64	4	3	4	**11**
Haraff	3	5	1	**9**
Harlap	2	4	3	**9**
Hesaff	1	1	2	**4**
Heslap	0	0	0	**0**

Table 4.9: Ranking of algorithms based on sample size (number of images with zoom + rotation) from Table 4.6 and 4.8.

Ranking based on 10 image pairs	**Score**	**Ranking based on 59 image pairs**	**Score**
Harlap	6	**SIFT**	6
SIFT	5	**Haraff**	5
Haraff	3	**Harlap**	4
Heslap	3	**SURF-64**	3
SURF-64	2	**SURF-128**	2
Hesaff	1	**Hesaff**	1
SURF-128	0	**Heslap**	0

there is a significant difference between their performances, evident by Z-scores such as 3.42, 11.68 and 46.09 for zoomed images, zoomed + rotated images, and images with only rotation respectively.

Table 4.8 gives a summarized characterization of the performances of all algorithms by collecting their scores from Table 4.7. A comparison of the rankings produced for large and small databases shows broadly similar characterization, because only SURF-64 and Haraff operators have changed their positions in the table. Of course, one needs to keep in mind the difference in the image transformation in both databases, which can be a critique to the comparison of these results as being an unfair.

To see a comparison between similar transformations, Table 4.9 presents a ranking of algorithms for those image datasets featuring both zoom and rotation. Again the order of algorithms highlights the performance on different amount of data. Unfortunately none of the algorithms share similar positions in the tables. Interestingly, Harlap with GLOH is at the top when there are smaller numbers of image pairs, while SIFT secured second position; this is not the case for the larger dataset. Having this level of dissimilarity in results suggests that the size of the database used has a significant effect on the ranking — and this means that existing evaluations based around the small database need to be treated with some suspicion.

To confirm these results are not an artefact of the use of McNemar's test, the same general procedure has also been carried out with ANOVA, and the mean performances of these operators are compared using it in the next section.

4.6 ANOVA: Variance analysis of matching results for different sample sizes

ANOVA is applied to determine whether the average error in matching image features by algorithms is same for all algorithms; a null hypothesis for ANOVA. The experimental procedure adopted is the same as described in chapter 3 i.e. homography testing, but instead of examining an algorithm's output for pass or fail, as was with McNemar's test, here the Euclidean distance of projected point using estimated and original homographies is summed, to give an overall error. Hence, a lower mean error should depict better performance. As per the conditions for using ANOVA, the data should be Normally distributed and there should be homogeneity of variances. Therefore, before applying ANOVA, the data are checked for this homogeneity of variances.

The distances of false matches makes the data variance too high and inhomogeneous (using the F-max test [185]), so some data treatment is required. Here, the square root of the data were calculated in an attempt to make them Normally distributed and its variances homogeneous.

4.6.1 Result for small database

Table 4.10 presents ab analysis of interest point operators for small database of images, grouped according to the type of transformation in datasets. To interpret the results, the F and P are compared with F_{crit} and $\alpha(0.05)$ respectively: if $F > F_{crit}$ or $P < \alpha$ one can reject the null hypothesis and accept the alternative. ANOVA is a test that handles type-I errors, so the α adjustment is not required for this test.

For datasets containing images with two geometric transformations (zoom and rotation) i.e. bark and boat, the algorithms show statistically significant performance difference as $F = 75.91 >> F_{crit} = 2.10$. One can be confident in this result as P is close to zero, indicating that the probability of these results occurring by chance is low. The algorithms are sorted based on minimum mean error and then for variance, and it is found that SIFT performs better than all other algorithms; similarly, both versions of SURF and Haraff also exhibit good performance.

For images with blur (Bikes and Trees datasets), SURF performed better than the other algorithms. Although SIFT has low variance, its mean error is higher than that of SURF. Datasets presenting change in viewpoint transformation, such as Graffiti and Wall, are the most difficult images for these detectors because the overlapped area in the image pair decreases as the viewing angle changes. Hence, algorithms show high variance on these datasets. As expected, Harris

Table 4.10: Performance analysis using ANOVA for small database of images

Source of Variation	**Between Groups**	**Within Groups**	**Groups**	**Mean**	**Variance**
Zoom + Rotation (Bark and Boat)			SIFT	1.50	1.14
Sum of Square (SS)	25973.8	3991675	SURF-64	2.40	13.62
Degree of Freedom (df)	6	69993	Haraff	2.49	14.74
Mean Square (MS)	4328.97	57.03	SURF-128	2.56	5.19
F	75.91		Heslap	2.62	23.15
P	6.8×10^{-95}		Harlap	2.87	48.50
F_{crit}	2.10		Hesaff	3.72	292.87
Blur (Bikes and Trees)			SURF-128	1.25	0.95
Sum of Square (SS)	4183.34	123537	SURF-64	1.28	1.04
Degree of Freedom (df)	6	69993	Heslap	1.42	1.59
Mean Square (MS)	697.22	1.76	SIFT	1.43	1.18
F	395.03		Harlap	1.64	1.89
P	0		Haraff	1.82	2.67
F_{crit}	2.10		Hesaff	1.92	3.03
Change in view point (Graffiti and wall)			Harlap	1.88	8.07
Sum of Square (SS)	118817	1826445	Hesaff	2.52	8.75
Degree of Freedom (df)	6	69994	Haraff	2.60	8.84
Mean Square (MS)	19802.8	26.09	SURF-64	3.40	17.12
F	758.89		Heslap	4.34	36.20
P	0		SURF-128	4.85	52.70
F_{crit}	2.10		SIFT	5.75	50.99
Change in illumination (Leuven)			Haraff	1.84	3.70
Sum of Square (SS)	49404	436862	SURF-64	1.92	3.48
Degree of Freedom (df)	6	35063	SURF-128	2.03	3.70
Mean Square (MS)	8234	12.46	SIFT	2.11	4.45
F	660.87		Hesaff	2.27	3.97
P	0		Heslap	2.28	4.24
F_{crit}	2.10		Harlap	5.44	63.67
JPEG compression (UBC)			Haraff	1.84	3.70
Sum of Square (SS)	49404	436862	SURF-64	1.92	3.48
Degree of Freedom (df)	6	35063	SURF-128	2.03	3.70
Mean Square (MS)	8234	12.46	SIFT	2.11	4.45
F	660.87		Hesaff	2.27	3.97
P	0		Heslap	2.28	4.24
F_{crit}	2.10		Harlap	5.44	63.67

Table 4.11: ANOVA test results for datasets grouped according to images with same transformation. Large database

Zoom			**Mean based groups' ranking**		
Source of Variation	*Between Groups*	*Within Groups*	*Groups*	*Mean*	*Variance*
			SIFT	6.32	88.88
Sum of Square (SS)	1704064	9.67^{+08}	SURF-64	10.10	461.51
Degree of Freedom (df)	6	209991	Haraff	12.22	474.49
Mean Square (MS)	284010.69	4605.29	Heslap	12.55	23715.91
F	61.67		SURF-128	12.58	1983.18
P	9.07×10^{-77}		Hesaff	14.78	795.56
F_{crit}	2.10		Harlap	15.53	4718.77
Rotation					
			SIFT	3.11	52.95
Sum of Square (SS)	198418.2	97346697	Haraff	3.27	63.67
Degree of Freedom (df)	6	671996	SURF-64	3.52	154.15
Mean Square (MS)	33069.70	144.86	SURF-128	3.70	147.36
F	228.28		Heslap	4.11	104.45
P	1.8×10^{-292}		Harlap	4.43	141.58
F_{crit}	2.10		Hesaff	4.66	349.89
Zoom + Rotation					
			SIFT	1.56	1.28
Sum of Square (SS)	120086	12789590	SURF-128	1.75	1.92
Degree of Freedom (df)	6	412993	SURF-64	1.82	4.22
Mean Square (MS)	20014.34	30.97	Harlap	2.40	13.36
F	646.29		Haraff	2.42	92.74
P	0		Hesaff	2.93	73.47
F_{crit}	2.10		Heslap	3.04	29.78

and the Hessian-based detectors with GLOH descriptor, which are considered specialist for detecting this kind of transformation, performed better than SIFT and SURF for 10 image pairs.

For photometric transformations, two datasets are available, Leuven and UBC. For both datasets, Haraff gave better results than SIFT and SURF. ANOVA statistics in Table 4.10 show that there is significant difference in the performance of algorithms for all datasets and $P \approx 0$ strengthens confidence over these results.

4.6.2 Results for large database

ANOVA results for the large database of images are presented in Table 4.11. Large database images can be divided into three categories based on image transformations: datasets with zoom, datasets with rotations, and datasets with both zoom and rotation. Results for all categories show high confidence levels because of the low P, especially in last category (zoom and rotation). In every category, ANOVA rejects the null hypothesis, showing that there are significant performance differences in feature operators' performances.

An interesting question is whether these results are different from the ones for the small database. To answer this, let us compare the last category results (zoom and rotation) with the same category in the small database results (the Bark and Boat datasets in Table 4.10). Both results reject the null hypothesis with $F > F_{crit}$. However, the ANOVA analysis shows some differences. Apart from SIFT, all other algorithms change their positions in the ranking table. Haraff performed much better when the number of image pairs is only 10 but when these pairs are increased to 59, its performance worsens compared to the other algorithms.

Generally, ANOVA test results are in agreement with McNemar's test, that small sample size cannot be used to predict general performance trends of algorithms. Therefore, the large number of evaluation studies using the small database reported in the literature are questionable and may need to be repeated using more sophisticated performance evaluation and with sufficiently large volumes of data. However, before delivering a final verdict, it is also important to ascertain whether the lack of agreement in the results produced for large or small databases is due to image content. The following discussion will explore this.

4.7 Does image content affects performance analysis?

Use of different sample sizes for performance analysis revealed statistically significant performance differences of algorithms. However, it does not show if these results will be different if images are changed; in other words, does image content play a role in favour of any operator? To explore this effect, the larger datasets of images are divided into smaller subsets of fifteen image pairs each and McNemar's test and ANOVA have both been used to study the behaviour of the feature operators.

However, one need to keep in mind that a different amount of transformation has applied to each image in a dataset; the images in the Mars, Monet and New York datasets are rotated at different angles compared to the first image which is used to used to match with them, as shown in Table 4.12. All of these operators are sensitive to these geometric transformations and may perform differently for different amount of transformation. Theoretically speaking, a sufficiently large sample size should overcome this problem and one should be able to observe the general behaviour of algorithms.

Let us examine the results generated for different subsets of the four datasets Laptop, Mars, Monet and New York. All of these sets have more than 15 images and allow subsets of 15 image pairs to be selected. The performances of all operators are compared for these datasets. McNemar's test results are presented in Tables 4.13 and 4.14 and show Z-scores between pairs of feature operators for each subset and for the whole dataset (at the bottom of each set in bold).

The results for the subsets from Laptop, Mars and Monet are consistent with the whole dataset, showing that the appropriate evaluation framework with sufficient dataset size can predict the behaviour of the algorithms. However, for the New York dataset, the better performing operator changes for different subsets.

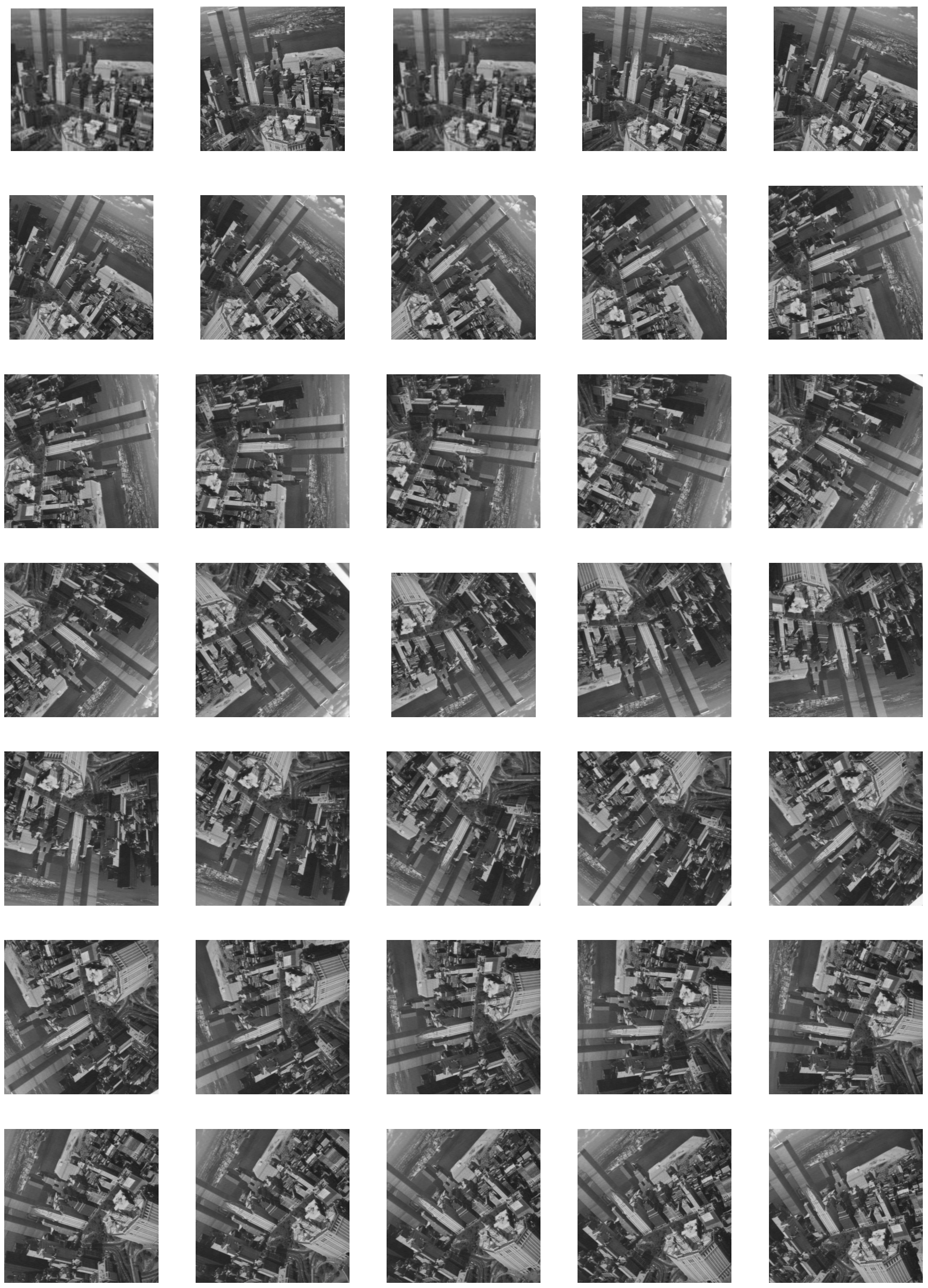


Figure 4.5: New York dataset

Table 4.12: Geometric transformation (scale and angle) between each image pair of four datasets calculated from original homography matrices

Image pairs	New York		Laptop		Mars		Monet	
	Scale	Rotation	Scale	Rotation	Scale	Rotation	Scale	Rotation
Image 1-2	1.01	9.83	0.89	0.01	1.00	5.80	1.00	5.80
Image 1-3	1.00	19.71	0.83	0.10	1.00	13.30	1.00	13.30
Image 1-4	1.00	29.73	0.77	0.02	1.00	23.33	1.00	23.33
Image 1-5	0.99	39.75	0.72	0.11	1.00	31.86	1.00	31.86
Image 1-6	1.00	49.83	0.66	0.11	1.00	40.12	1.00	40.12
Image 1-7	1.00	59.80	0.60	0.01	1.00	46.59	1.00	46.59
Image 1-8	0.99	69.94	0.55	0.31	0.99	55.39	0.99	55.39
Image 1-9	0.99	80.12	0.48	0.13	0.99	64.03	0.99	64.03
Image 1-10	0.99	90.09	0.45	0.18	0.99	75.41	0.99	75.41
Image 1-11	0.99	100.04	0.42	0.08	1.01	-45.93	1.01	-45.93
Image 1-12	0.99	109.74	0.39	0.04	1.01	-39.52	1.01	-39.52
Image 1-13	1.00	120.38	0.36	0.02	1.01	-35.36	1.01	-35.36
Image 1-14	1.00	130.15	0.34	0.39	1.01	-31.41	1.01	-31.41
Image 1-15	1.01	139.90	0.31	0.61	1.00	-27.63	1.00	-27.63
Image 1-16	1.01	149.74	0.28	0.15	1.00	-21.42	1.00	-21.42
Image 1-17	1.00	159.79	0.26	1.25	1.00	-16.64	1.00	-16.64
Image 1-18	1.00	169.71	0.23	1.28	1.00	-12.55	1.00	-12.55
Image 1-19	1.00	169.91	0.20	0.27				
Image 1-20	1.01	-160.11	0.19	2.71				
Image 1-21	1.01	-149.92	0.16	0.54				
Image 1-22	1.01	-140.06						
Image 1-23	1.01	-130.03						
Image 1-24	1.01	-120.04						
Image 1-25	1.01	-110.21						
Image 1-26	1.00	-100.20						
Image 1-27	1.00	-90.36						
Image 1-28	1.01	-80.12						
Image 1-29	1.00	-70.37						
Image 1-30	1.00	-60.44						
Image 1-31	1.00	-50.47						
Image 1-32	1.01	-40.34						
Image 1-33	1.00	-30.40						
Image 1-34	1.01	-20.19						
Image 1-35	1.00	-10.51						

Table 4.13: Performance comparison of SIFT and SURF with other operators for subsets of large dataset where each subset contains 15 image pairs. The result of each subset can be compared with the whole dataset result given at the bottom of each set in bold fonts. Arrow-heads point towards the operator with better performance on particular set of images. Z-scores less than 0.92 are non-significant and hence do not point in any direction.

Subsets		**SURF-64**	**SURF-128**	**Haraff**	**Harlap**	**Hesaff**	**Heslap**
laptop		0	0	← 12.962	← 3.75	← 5.66	← 22.98
laptop	**SIFT**	← 11.18	← 11.40	← 59.254	← 46.658	← 19.053	← 62.282
laptop		**← 11.18**	**← 11.40**	**← 59.25**	**← 46.66**	**← 19.08**	**← 62.28**
Mars		0	0	0	0	0	← 2.67
Mars	**SIFT**	0	0	0	0	0	← 1.789
Mars		**0**	**0**	**0**	**0**	**0**	**← 2.67**
Monet		← 13.47	← 36.91	1.15	← 27.695	← 61.68	← 55.60
Monet	**SIFT**	← 18.71	← 37.336	1.15	← 27.166	← 58.489	← 59.523
Monet		**← 20.27**	**← 40.45**	**1.154**	**← 27.69**	**← 61.75**	**← 60.93**
New york		← 13.526	← 7.5202	↑ 13.55	↑ 22.875	↑ 9.0387	↑ 11.172
New york		0.8157	← 7.5884	↑ 10.484	↑ 14.404	↑ 7.603	↑ 10.929
New york	**SIFT**	← 13.599	← 22.101	← 3.7275	1.072	← 10.291	← 16.614
New york		**← 16.88**	**← 18.83**	**↑ 12.30**	**↑ 22.87**	**↑ 5.59**	**0.017**
laptop			0	← 13.417	← 5.0709	← 5.4801	← 21.52
laptop	**SURF-64**		0.47	← 56.903	← 44.812	← 11.386	← 60.202
laptop			**0.468**	**← 56.90**	**← 44.81**	**← 11.49**	**← 60.20**
Mars			0	0	0	0	1.79
Mars	**SURF-64**		0	0	0	0	0.71
Mars			**0**	**0**	**0**	**0**	**1.789**
Monet			← 33.211	↑ 12.123	← 17.124	← 60.22	← 53.359
Monet	**SURF-64**		← 29.519	↑ 18.6	← 9.5549	← 52.601	← 53.371
Monet			**← 31.43**	**↑ 20.19**	**← 7.76**	**← 54.77**	**← 54.18**
New york			↑ 9.79	↑ 22.92	↑ 29.50	↑ 19.13	↑ 20.62
New york			← 10.332	↑ 5.9465	↑ 12.491	↑ 6.7651	↑ 8.7899
New york	**SURF-64**		← 11.77	↑ 9.19	↑ 12.61	↑ 2.17	← 4.55
New york			**1.36**	**↑ 24.23**	**↑ 33.61**	**↑ 18.07**	**↑ 12.17**
laptop				← 10.536	← 4.7246	← 4.5873	← 22.383
laptop	**SURF-128**			← 57.318	← 45.35	← 10.90	← 60.856
laptop				**← 57.31**	**← 45.35**	**← 10.94**	**← 59.19**
Mars				0	0	0	← 1.79
Mars	**SURF-128**			0	0	0	0
Mars				**0**	**0**	**0**	**← 1.78**
Monet				↑ 36.824	↑ 15.531	← 47.605	← 31.688
Monet	**SURF-128**			↑ 36.715	↑ 16.671	← 40.042	← 38.258
Monet				**↑ 40.01**	**↑ 20.70**	**← 41.59**	**← 36.02**
New york				↑ 20.193	↑ 23.563	↑ 12.863	↑ 15.231
New york				↑ 15.602	↑ 18.242	↑ 14.504	↑ 13.707
New york	**SURF-128**			↑ 19.015	↑ 18.776	↑ 8.9711	0.23
New york				**↑ 29.78**	**↑ 33.30**	**↑ 19.67**	**↑ 11.67**

Table 4.14: Performance comparison of Harris and Hessian based operators for subsets of large dataset where each subset contains 15 image pairs. The result of each subset can be compared with the whole dataset result given at the bottom of each set in bold fonts. Arrow-heads point towards the operator with better performance on particular set of images. Z-scores less than 0.92 are non-significant and hence do not point in any direction.

Subsets		**Harlap**	**Hesaff**	**Heslap**
laptop		↑ 9.3991	↑ 8.9745	← 14.425
laptop	**Haraff**	↑ 25.017	↑ 53.802	← 7.9418
laptop		**↑ 25.02**	**↑ 53.75**	**← 7.94**
Mars		0	0	1.1547
Mars	**Haraff**	0	0	0.7071
Mars		**0**	**0**	**1.15**
Monet		← 26.42	← 62.081	← 55.426
Monet	**Haraff**	← 26.42	← 62.177	← 60.133
Monet		**← 26.42**	**← 62.17**	**← 60.75**
New york		↑ 12.556	← 2.6052	0.4138
New york		↑ 7.9206	1.1107	↑ 3.8999
New york	**Haraff**	↑ 4.9357	← 8.9805	← 13.802
New york		**↑ 14.81**	**← 4.67**	**← 8.92**
laptop			0.8571	← 19.372
laptop	**Harlap**		↑ 39.068	← 32.098
laptop			**↑ 39.03**	**← 32.09**
Mars			0	← 3.6148
Mars	**Harlap**		0	← 2.6667
Mars			**0**	**← 3.61**
Monet			← 54.483	← 43.723
Monet	**Harlap**		← 50.938	← 48.516
Monet			**← 54.50**	**← 49.90**
New york			← 11.514	← 8.3461
New york			← 4.0682	0.596
New york	**Harlap**		← 10.99	← 15.37
New york			**← 14.42**	**← 17.93**
laptop				← 22.38
laptop	**Hesaff**			← 59.425
laptop				**← 59.40**
Mars				1.5
Mars	**Hesaff**			0
Mars				**1.5**
Monet				↑ 14.051
Monet	**Hesaff**			↑ 4.1503
Monet				**↑ 2.99**
New york				0.9843
New york	**Hesaff**			↑ 4.6988
New york				← 4.2632
New york				**← 5.39**

Table 4.15: ANOVA test for New York dataset as a whole.

Groups	*Mean*	*Variance*
SURF64	2.53	2.28
Harlap	2.56	2.57
SURF128	2.60	2.43
Hesaff	2.61	2.51
Haraff	2.63	2.56
Heslap	2.64	2.81
SIFT	2.64	2.61
Source of Variation	*Between Groups*	*Within Groups*
Sum of Squares (SS)	474.40	791252.1
Degree of Freedom (df)	6	311619
Mean Square (MS)	79.07	2.54
F	31.14	
P	1.23×10^{-37}	
F_{crit}	2.10	

Table 4.16: ANOVA test for subsets from New York dataset. Each subset contains fifteen image pairs

New York Dataset	Subset (Image1-16)		Subset Image13-27)			Subset (Image21-35)		
Groups	*Mean*	*Variance*	*Groups*	*Mean*	*Variance*	*Groups*	*Mean*	*Variance*
Harlap	2.34	2.16	SURF64	2.78	2.63	SURF64	2.40	1.96
SURF64	2.41	2.13	Heslap	2.79	3.02	Harlap	2.42	2.09
Haraff	2.44	2.26	Hesaff	2.83	2.85	SURF128	2.46	2.02
Hesaff	2.44	2.24	SURF128	2.85	2.74	Haraff	2.51	2.25
SIFT	2.48	2.33	Harlap	2.91	3.27	SIFT	2.52	2.36
Heslap	2.48	2.68	SIFT	2.92	3.01	Hesaff	2.56	2.36
SURF128	2.48	2.41	Haraff	2.93	3.03	Heslap	2.64	2.69
Source of Variation	*Between Groups*	*Within Groups*		*Between Groups*	*Within Groups*		*Between Groups*	*Within Groups*
Sum of Squares (SS)	256.17	243110.6		360.13	308371.8		617.53	228337.8
Degree of Freedom (df)	6	104993		6	104993		6	101619
Mean Square (MS)	42.69	2.32		60.02	2.94		102.92	2.25
F	18.44			20.44			45.80	
P	1.54×10^{-21}			4.75×10^{-24}			2.41×10^{-56}	
F_{crit}	2.099			2.099			2.0989	

This appears to be principally because of the varying amounts of transformation alluded to above: the first subset contains images with less rotation, while the last subset has images with rotation up to $360°$, as shown in Figure 4.5 and Table 4.12. For these kind of data, the subset size needs to be large to accommodate the maximum variation in transformation.

ANOVA results for whole New York dataset, given in Table 4.15, mostly agree with subsets results shown in Table 4.16 in showing statistically significant performance differences of feature operators ($F >> F_{crit}$) but do not yield similar rankings based on low means and variances. The major problem with ANOVA is that the data are required to be Normally distributed and the variances need to be homogeneous; the data under analysis do not obey these rules and, even though they have been transformed by calculating its square root — the best of the standard transformations for these data — they do not fit a Normal distribution well. This transformation makes these rankings unreliable and so the rankings produced by McNemar's test are considered more trustworthy.

Apart from image content, the amount of transformation appears to be a factor that affects a feature operator's performance, as the results for the New York dataset highlighted the sensitivity of feature operators to rotation in images. The feature operators evaluated in this chapter are blob detectors, for which the descriptors are designed to be rotation invariant. However, corner points are image features which are inherently rotation invariant, so it would be interesting to ascertain the ability of corner detectors to identify corners at different angles and orientations. Hence, the next chapter explores the angular sensitivity of corner detectors in both synthetic and real images.

4.8 Remarks

Appropriate evaluation method and sample size are vital foundations for any evaluation study. Both of these factors need to be selected with good understanding of the data and evaluation framework.

Unlike previous studies, this research takes account of the size of the dataset used in making comparisons. Contrary to previous evaluation studies [10, 131], in which overlapping PR curves made it difficult to determine which algorithms out-performed others, the results presented here are not only statistically reliable but also clearly indicative of differences in the performances of algorithms for the same set of data. Table 4.9 reflects the changes in ranking when the evaluations were performed on datasets with different sample sizes. Therefore, one needs to be very careful in drawing conclusions when the amount of data is not sufficiently large.

The chapter laid out some valuable rules of thumb regarding data size for performance evaluation of vision algorithms. The homography testing framework proposed and used for the evaluation of feature operators should use a minimum of 700 points. Similarly, it has been established that 5 images pairs are not sufficient and should be increased to at least 15 image pairs for statistically valid performance evaluation.

Using these rules, a number of feature operators are characterized based on their performance and the results are compared with the standard dataset of 5 to 10 image pairs widely used for this purpose. The results show that the SIFT detector and descriptor are more distinctive and robust for matching under different image transformations and give consistent performance regardless of the type of images. Conversely, the Harris-based detector combined with the GLOH descriptor (which is an extended form of SIFT descriptor) gives good performance only when there is a significant viewpoint change in images. It should therefore be most useful when used in conjunction with another good feature descriptor, such as SIFT or SURF.

Although McNemar's test is valuable in identifying performance differences between algorithms, and thus has a place in evaluation studies, the fact that it identifies cases where one algorithm succeeds while another fails also has an important place in algorithm development: when this is so, the algorithms are operating in significantly different ways and a hybrid algorithm that incorporates elements of both these intelligently is likely to achieve a much better performance than either in isolation.

CHAPTER 5

CORNER POINTS

5.1 Introduction

A corner may be defined as a point of intersection between two or more edges. Just like blobs, edges and regions, these are interest points in images. However, they have some inherent properties which make them more distinctive and informative, for example an associated angle that can be useful for a number of matching-based applications. Corners can be grouped into three categories based on their internal angle, *i.e.* convex, concave and plain corners with intersecting edges making acute, obtuse or right angles respectively as shown in Figure 5.1.

In digital image processing, a corner is an image region where there is a change in gradient (directional change in image intensity) in both x and y directions. A number of detectors have been designed to find corners, some well

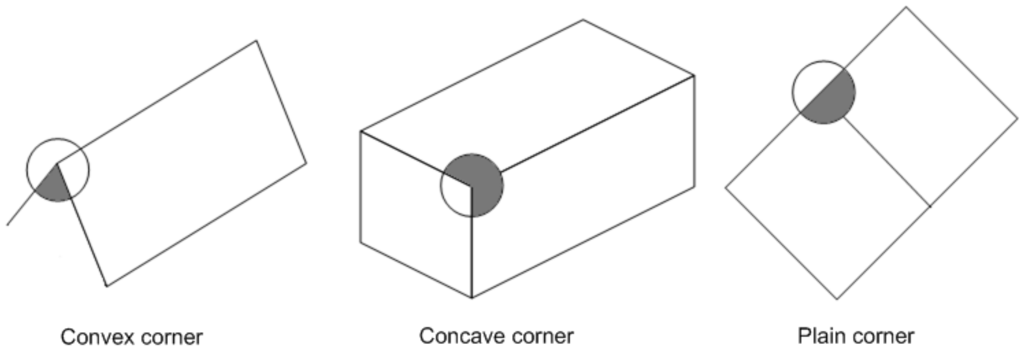


Figure 5.1: Corner types based on their internal angle

known ones being described in Section 5.2. As the previous chapters have established, statistically valid performance evaluation is critical for choosing the best detector for a vision application; hence, previous studies of evaluating corner detectors also need reviewing critically.

Previously, corner detectors have been evaluated using qualities such as repeatability [11,118,131,132], detection accuracy, consistency, stability, and localization accuracy [5,8,186–190]. The most important of these is the localization of true corner points. The remaining criteria are application-based: for example, repeatability measures the usefulness of detectors for matching applications, while stability refers to the appearance of corners in multiple images of the same visual scene, good for stereo and tracking applications. This chapter evaluates some state-of-the-art corner detectors for detecting corner points at different angles and different orientations.

In previous evaluation studies, the amount of data used has been low, often 4 to 10 synthetic or real images with ground truth (*i.e.*, knowledge of the true corner locations) [5,8,187,189,191–193] which is not sufficient. This evaluation uses a significantly larger amount of data and a combination of evaluation methods. Section 5.3 introduces the image data developed for this purpose, comprising synthetically-generated digital images of geometric shapes and real images of them captured using a camera. In order to compare detectors' performance for locating corner points at correct locations, the knowledge of the ground truth is important. Therefore, methods for identifying corners' true locations are also described in this section.

In section 5.4, the conventional detection rate and F-measure have been used for performance comparison over synthetic and real data. Moreover, angular sensitivity of corner detectors is analyzed using McNemar's test in section 5.5, to ascertain whether or not results are consistent with conventional methods.

As mentioned before, most of the evaluation frameworks previously used perform application-based performance analysis. Computing detection rate or F-measure for the number of corner points in each image cannot characterize an algorithm's general behaviour as it overlooks the identification of non-corner points as corners. To fill this gap, so-called 'technology evaluation' is still required to understand detectors' general performance trends. Section 5.6 presents a technology evaluation framework for corner detectors for both synthetic and real images. The discrepancies between general and application-based performance analyses are also highlighted and discussed in this section.

Both application-based evaluation and technology evaluations suggests that some detectors work well on one type of data while others work better on others; for example, some detectors are able to find acute angles better than obtuse angles and vice versa. Therefore, it may be wise to combine the detectors to get optimal results. To study the best combination, section 5.7 presents combined results of detectors for synthetic and real data. Finally, section 5.8 concludes the discussion and pinpoints some future directions in terms of hybrid corner detectors.

5.2 Corner detectors

As a corner is characterized by a region with intensity change in two different directions, a simple way of finding a corner point is to select a square image patch as a lookup window and the calculate sum squared difference between this and a second image patch, selected by moving the lookup window:

$$S(x,y) = \sum_u \sum_v w(u,v)(I(u,v) - I(u-x,v-y))^2 \tag{5.1}$$

If $S(x,y)$ is high in all 8 directions, it will be classified as corner point. The problem with a square image patch is that it is not isotropic (non-uniform at different orientations). However, a circular window's response is normally isotropic, so a weighted circular window is used in a Gaussian-profile mask in some detectors.

A number of corner detectors have been proposed during the last few decades, so vision researchers understandably would like to know which one is the best to use. Corner detectors are usually classified into two broad categories, *template-based* and *geometry-based*. Methods based on the former find the similarity between a template and a sub-window of the same size in an image, whereas geometry-based methods measure the differential geometric features such as finding edges, calculating topology or using autocorrelation [194]. A number of corner detectors can be found in surveys of corner detection methods [137, 194]; in this work, a representative subset was employed: well-established detectors (H&S [76], Smallest Univalue Segment Assimilating Nucleus (SUSAN) [137], the detector used in the Kanada-Lucas-Tomasi Corner Detector (KLT) tracking algorithm [138] and Shi & Tomasi (S&T) [139]); and more recent ones: FAST [118] and Global and Local Curvature Points (GLC) [195]). The following section provides a brief summary of these detectors and their principles of operation.

5.2.1 Harris & Stephens (H&S)

H&S [76] used the principle of similarity between template and a sub-window described at the start of this chapter; however, instead of using a moving window, it calculates the differential of the corner score with respect to direction for each image point using autocorrelation. For further simplification, the eigenvalues of first-order derivatives of image pixels are calculated:

$$A = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} \tag{5.2}$$

Using this, the corner measure M can be calculated using

$$M = \lambda_1 \lambda_2 - K(\lambda_1 + \lambda_2)^2 = det(A) - Ktrace^2(A) \tag{5.3}$$

Depending on the magnitude of the eigenvalues, the following inferences can be made

- if both λ_1 and $\lambda_2 \approx 0$ then the pixel is neither a corner nor edge;
- if $\lambda_1 \approx 0$ and λ_2 has some large positive value, the pixel is on an edge;
- if λ_1 and λ_2 both have large positive values, it is a corner point.

So instead of computing a sliding window response, calculating a determinant and trace of matrix A is sufficient to find corner points.

5.2.2 SUSAN

SUSAN is a mask-based method of finding corner points [137]. It defines a circular mask (M) of radius r over image pixels and compares every image pixel (m)

covered by the mask with its central pixel (nucleus, m_0) using the function

$$n(M) = \sum_{m \in M} e^{-\left(\frac{I(m)-I(m_0)}{r}\right)^6} \tag{5.4}$$

where the exponent value is determined empirically. SUSAN uses a geometric threshold to categorize pixels as corners or parts of an edge. Greater g helps finding edge points, whereas small g values let corner points be selected. The following recurrence function defines how SUSAN score is calculated:

$$R(M) = \begin{cases} g - n(M) & \text{if } n(M) < g; \\ 0 & otherwise, \end{cases}$$

For a corner, n must be less than half of its maximum possible value [137].

5.2.3 The KLT corner detector

Described in [138], this is a similar detector to Harris & Stephens but uses a greedy approach for corner selection, by selecting points with the smallest eigenvalues. From the eigenvalues of all image pixels, it selects points for which $\lambda_1 > \tau$, where τ is some threshold. Then the point with maximum λ_1 from the list is selected as a corner point and all other points which are in its neighbouhood (selected empirically) are removed from the list. The process is repeated until the list is completely explored for new corner points. This operator does not consider the difference between λ_1 and λ_2 and hence a large number of image points can be determined to be corner points, unlike the H&S detector.

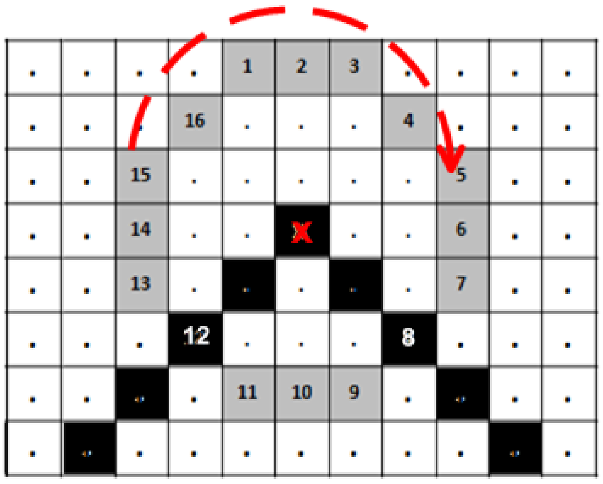


Figure 5.2: FAST corner detection, showing radius of 3 and clockwise direction of testing image pixels for corner detection

5.2.4 Shi & Tomasi (S&T)

S&T [139] found that the minimum of eigenvalue λ_1 and λ_2 should be greater than some threshold to identify a corner point. These eigenvalues are calculated from image derivatives shown in Equation 5.2. The S&T corner points are said to be more stable for tracking applications, and the technique is hence widely used (it is implemented in OpenCV).

5.2.5 Features from Accelerated Segment Test (FAST)

FAST: a corner detector described in [77]. Like SUSAN, it uses a circular mask of radius r; however, instead of comparing all pixels in the circular mask area, it considers the pixels on circle's boundary for comparing with the nucleus, as shown in Figure 5.2. If n contiguous pixels are all brighter than or darker than the nucleus, then the nucleus pixel is a corner point. It uses a machine learning approach for pixel selection, with the order of pixels in a circular arc selected using the ID3 (decision tree learning) algorithm from a training set of images.

FAST uses circular arcs of radius 3, 6, 9 and 12. By comparing results for different radii, the authors found 9 and 12 to be the most effective in finding corner points in a variety of images [77].

5.2.6 Global and local Curvature Points (GLC)

This detector uses a classical Curvature Scale Space (CSS) technique [196] to find corners and is called the Global and Local Curvature method [195]. Conventional CSS-based methods find contours in an image from its edge map, created using any edge detector; T-junctions are identified from these contours for which curvature is calculated at a high scale. The algorithm selects the points where the curvature value is higher than some threshold τ and not very close to some other corner point. Lastly, selected points are tracked to the lowest scale for each contour, the aim being to improve the localization accuracy.

For the GLC detector, corner points are identified as maximum local curvature points, similar to CSS, but the difference lies in the use of an initial local threshold for selecting candidate corner points. It then discards false corners using an adaptive local threshold and a region of support for selecting points with only sharp angle. Consequently, this method should produce more accurate and stable corner points than simple CSS-based methods.

5.3 Datasets

The principal characterizing feature of a corner is its internal angle. Hence, it makes sense to ascertain whether the performances of corner detectors change as the angle and orientation of the corner changes. A convenient way of establishing this angular sensitivity is to use regular geometric shapes, as a variety

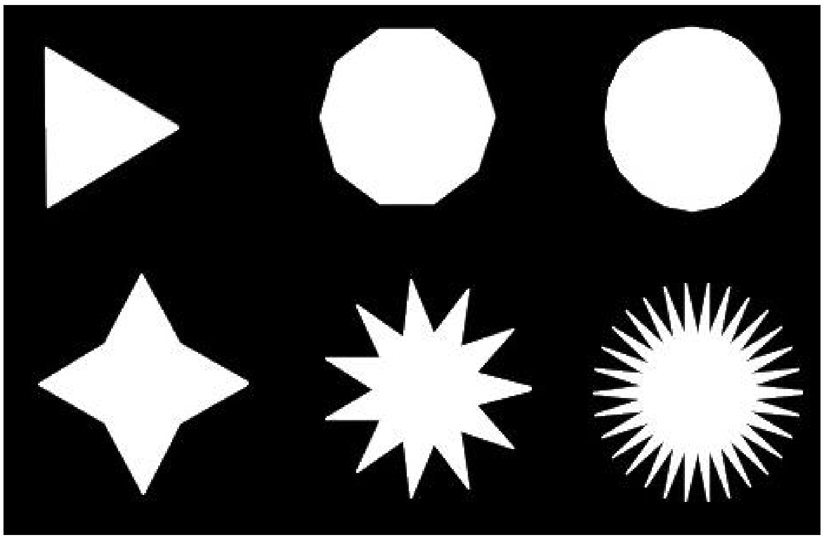


Figure 5.3: Some of the geometric shapes used for evaluation. The first row shows 3-, 10- and 20-sided polygons, while the second row shows star-polygons with 4, 11 and 30 arms.

of interior angles and orientations can be generated at known angles; *e.g.*, the corners of an equilateral triangle all have 60° internal angles but different orientations. Of course, the vertices of polygons are frequently used in their own right for object representation, recognition [197], 3D object pose estimation [198] and 3D shape retrieval [199].

Both synthetic and real image data have been developed and used for performance evaluation of corner detectors. The primary consideration is that the amount of data is large enough for statistically reliable results to be collected. The following sections illustrate the method adopted to generate data and corresponding ground truth (actual corner locations) in both synthetic and real images. As discussed above, the algorithms compared cover a range of approaches and encompass well-established and comparatively modern techniques. In all cases, original implementations by their authors have been used where available, or the OpenCV implementations with their default parameter settings.

Table 5.1: Distribution of corners in datasets, based on their angles

set	angle range	Number of corners
A	11–45°	766
B	45–90°	733
C	90–135°	417
D	135–164°	318

5.3.1 Synthetic data

To characterize the angular sensitivity of corner detectors, a large number of synthetic images of geometric shapes were used. Geometric shapes were used for several reasons: firstly, a corner detector should be able to produce 100% accurate results on synthetic images; a large volume of data with exactly the same parameters (such as number of corners and angle between edges corresponding to corner points) can be reproduced easily; and human judgement is not involved in determining the actual locations of corner points ("ground truth"). Both conventional polygons and star-shaped objects were used. These have the desirable property for evaluation of having the same angle at different orientations. The corner locations were determined by first calculating the internal angle, θ. For a polygon, this is $\theta = \frac{2\pi}{S}$ where S is the number of sides, while for a star it is $\theta = \frac{\pi}{S}$. The locations of the corners in a shape subscribed in a circle of radius r are then given by $x = x_0 + r \cos n\theta$ and $y = y_0 + r \sin n\theta$ where $n = \{1, 2, \ldots, S\}$ and (x_0, y_0) is the centre of the shape.

These equations were used to draw polygons and the exact pixel positions of their corner points, using OpenCV's anti-aliased line drawing routine to join these points together and its `FillPoly` function to fill them. Anti-aliased lines and filled shapes were used to represent more closely what one would obtain from a real-world camera.

Corners were produced from a minimum angle of 11° (star polygon with 60

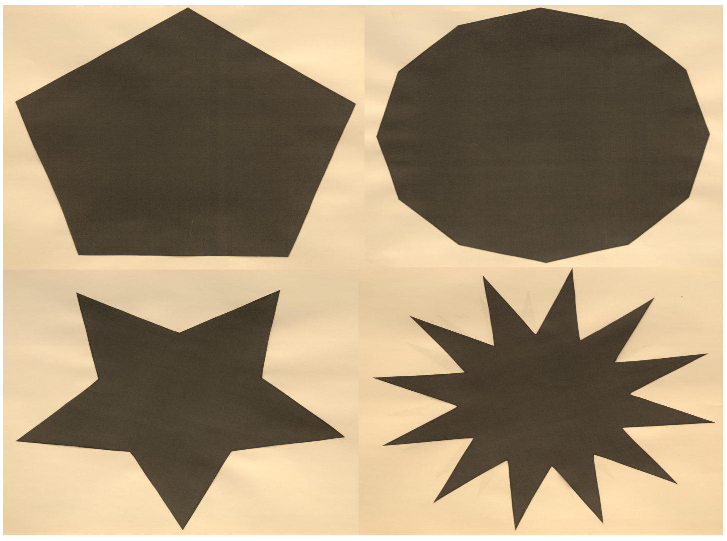


Figure 5.4: Some of the polygon images used for evaluation. Images size are approximately 2800-4000 x 2800-4000; Not all images are of the same size.

points) and a maximum of 164° (20-sided polygon). It is impossible to achieve exact similarity for all sides of a polygon due to the rasterization of pixels' positions in a digital image; all the angles in a polygon differ by about $\pm 1°$. The angle at a corner ϕ is measured by taking the scalar product of the vectors between a corner point V_2 and its neighbouring corner points V_1 and V_3: $cos\phi = \frac{L_1.L_2}{|L_1||L_2|}$ where $L_1 = V_3 - V_2$ and $L_2 = V_1 - V_2$. The amount of data produced is shown in Table 5.1.

5.3.2 Real image data and ground truth

Polygons and stars were generated on a computer and printed. These were photographed using a Nikon D300 camera equipped with a Nikkor 18–200 mm lens in 16-bit RAW format and converted into 8-bit PNG using `dcraw`; typical images are shown in Figure 5.4. It was then necessary to establish the locations of the corners as accurately as possible. In previous studies, the most commonly-used method to identify corners' actual locations is human judgement: for ex-

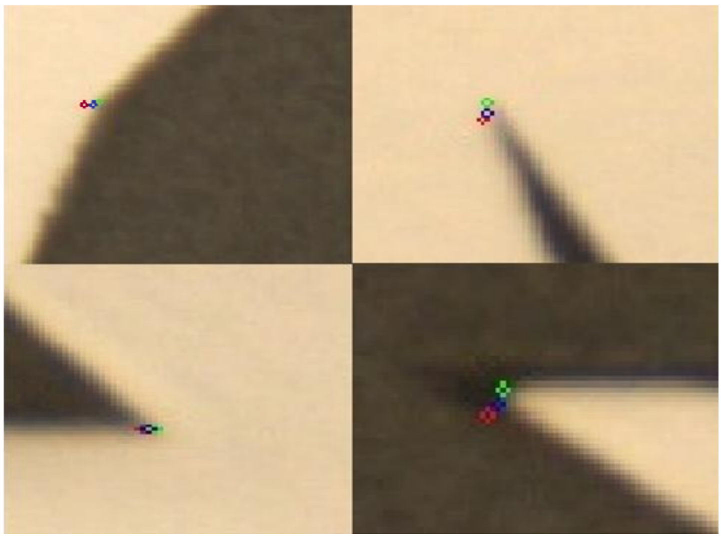


Figure 5.5: Estimations of corner locations on real images — red circle: human guess; green circle: automatically detected; blue circle: mean of human- and machine-identified locations.

ample, in [5] a corner location is taken as the mean of ten human-judged locations. This approach is acceptable for limited amounts of data but rapidly becomes too time-consuming as the volume of data increases, and is certainly not feasible for the 2,234 corners in this study (which the authors consider is the smallest number from which statistically meaningful conclusions can be drawn with any confidence). An automatic mean is provided by OpenCV's routine `cvFindCornerSubPix`[1] which purports to find the location of the intersecting vectors to an accuracy of 0.1 pixel [200] in the chess-board patterns of calibration targets. However, sometimes it produces a systematic offset of the order of 0.1 pixel towards the center of the gradient for intersecting vectors at angles other than 90°. In the case of chess-boards, this error is cancelled out between two touching squares; but this is not the case for the images employed here.

To establish whether this automatic method could be used instead of human estimation, at least ten people were asked to identify 24 corner locations

[1] http://opencv.willowgarage.com/documentation/feature_detection.html

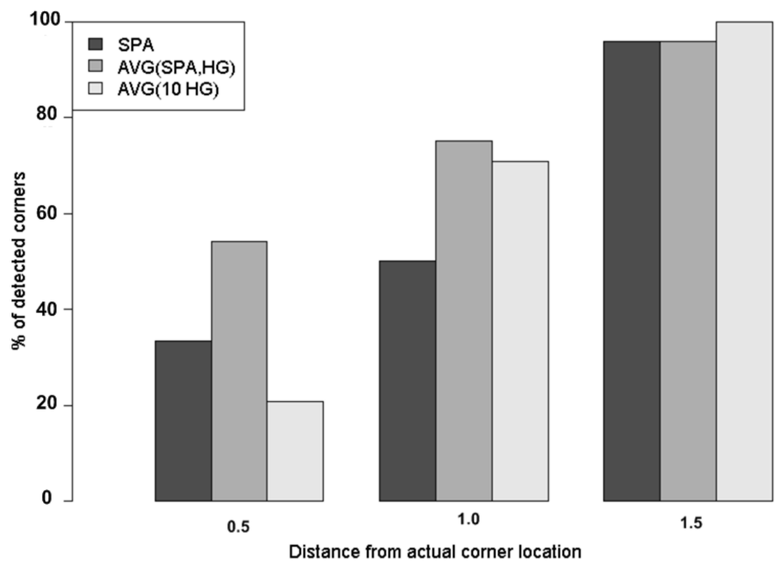


Figure 5.6: Error analysis of three methods to find actual corner locations in two synthetic images; Human Guess (HG), Corner Sub Pixel Accuracy (SPA) and Average of (SPA and HG)

manually in selected synthetic and real images of polygons and their means were compared with those generated using the above-mentioned technique; Figure 5.5 shows typical results. Red circles indicate the corner locations marked by humans and used to initialize sub-pixel accuracy function; green circles are the outputs of the function; and blue circles are the means of the two. The synthetic images allowed the error to be determined for human and machine approaches, illustrated in Figure 5.6, while the real images were used to ensure that the algorithm is sufficiently robust. This procedure determined the mean of human and machine results to be the best estimate of the actual corner locations, and that the mean machine-identified locations were within one pixel of them, better than that of a human. Consequently, to find the ground truth corner locations in the dataset, their positions were marked approximately by a human and then refined using `cvFindCornerSubPix`. Visual inspection confirmed that the results were accurate enough. Images were then grouped based on the internal angles of the corners contained; see Table 5.1.

Having obtained a dataset of suitable images and ground truth data of corner locations, it only remains to establish how close a detected corner has to be to its true position to be deemed a success. The criterion that has been used here is that the detected corner must lie within a Euclidean distance of two pixels from the actual corner location for it to be deemed a success.

5.4 Test 1: Detection Rate

The correctness of the detector is based on its localization accuracy and the number of corners detected. Localization accuracy was used in preference to other, more complex, measures such as repeatability because the actual locations of corner points are what are important for a corner detector. The number of false corners detected by a detector based on this localization criterion yields a false-positive rate, R_f:

$$R_f = \frac{\text{number of FP corners}}{\text{actual number of corners}} \tag{5.5}$$

5.4.1 Detectors' response on synthetic data

The test data comprises synthetic images with no noise, so one might expect that no corner detector would mark a false positive corner. In Figures 5.7 and 5.8, an R_f of zero indicates the best achievable performance of a corner detector. Unfortunately, none of the detectors achieves this for all images, though H&S, SUSAN and FAST-9 do so for polygons in which the corners are formed at angles greater than 60°. The underlying principle of the H&S and SUSAN detectors is to find changes in gradient in two directions simultaneously, so when the angle is less

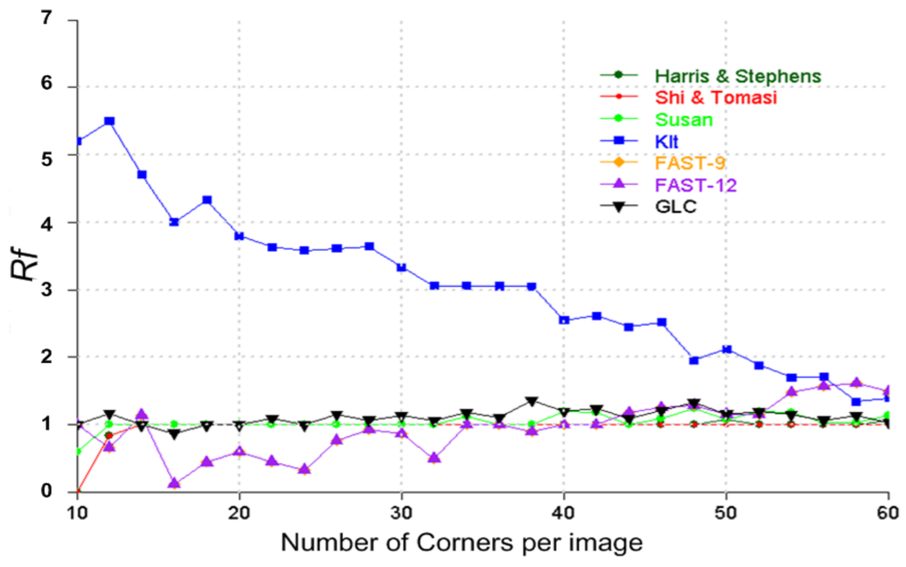


Figure 5.7: False positive rate of detectors for images containing stars with angles of 11° to 56°.

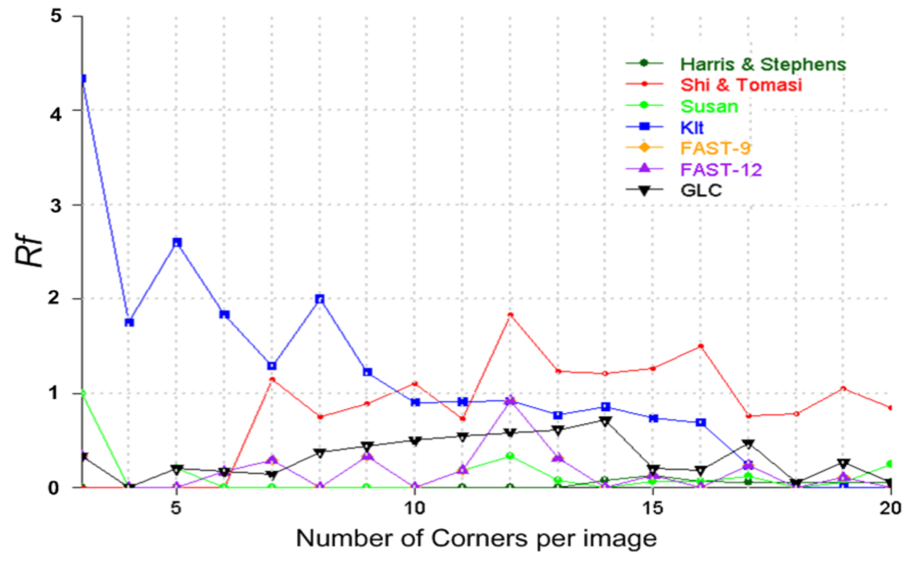


Figure 5.8: False positive rate of detectors for images containing polygons with angles greater than 60°.

than 45° they seem to accept more points near actual corners due to neighbouring edge pixels. Though S&T corner (1994) detector is a modified form of the detector proposed by KLT (1991), its R_f value shows that its modification helps reduce the false positive rate — but, considering the data are synthetic, even a small number of false positives is not acceptable. The behaviour of all these detectors with real imagery, which inevitably contains noise, is likely to be significantly poorer.

5.4.2 Detectors' response on real data

When a corner detector is run on an image, there are several possible outcomes for each corner in it:

TP: the corner is detected;

FN: the corner is not detected;

FP: a corner is identified where there is no corner in the image;

TN: all remaining pixels in the image.

Let us write N_{TP} as the number of true positives found *etc*. Both false negatives and false positives reduce a detector's performance, so two measures which can encapsulate them are P and R for ranking an algorithm, often combined into the so-called *F-measure*, the harmonic mean of P and R. For characterizing the overall performance of the corner detectors, F values are calculated across the entire dataset. One can argue about the use of F to assess a detector's performance instead of detection rate as used for synthetic data. The logic behind this is the nature real imagery (which contain noise not present in synthetic images) and it become important to consider all negative outcomes along with positive results.

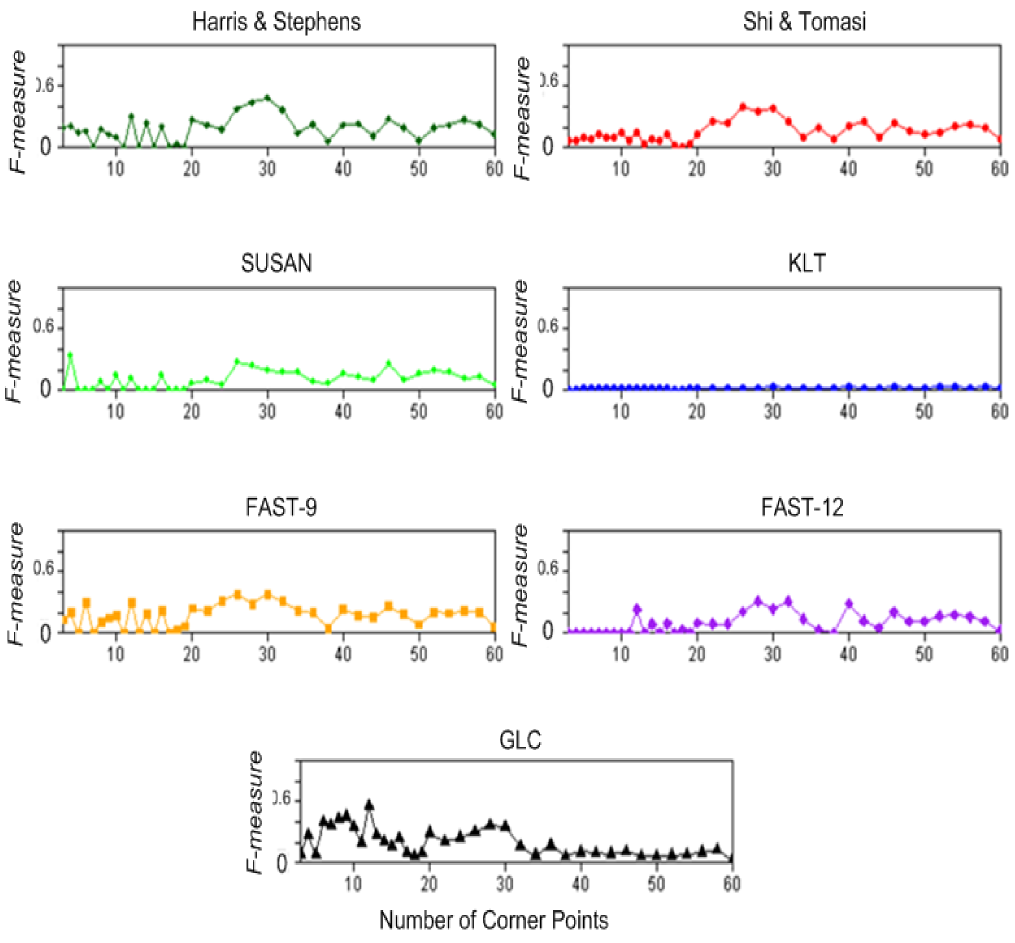
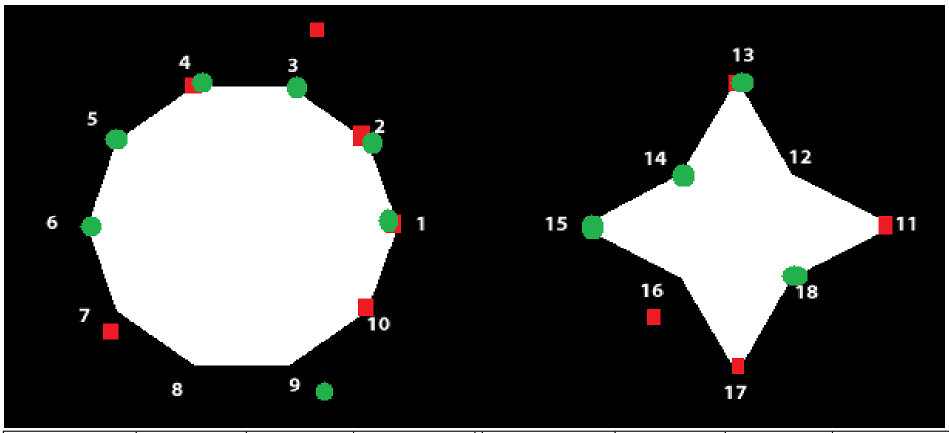


Figure 5.9: Variation of F across images, where each image contains 3–60 corners



corner	A (■)	B (●)	result	corner	A (■)	B (●)	result
1	✔	✔	SS	10	✔	×	SF
2	✔	×	SF	11	✔	×	SF
3	×	✔	FS	12	×	×	FF
4	✔	✔	SS	13	✔	✔	SS
5	×	✔	FS	14	×	✔	FS
6	×	✔	FS	15	×	✔	FS
7	×	×	FF	16	×	×	FF
8	×	×	FF	17	✔	×	SF
9	×	×	FF	18	×	✔	FS

Figure 5.10: Success criterion for McNemar's test for synthetic images

Conventionally, one distinguishes algorithms simply on the basis of P, R, F, or a plot of P and R. Figure 5.9 shows how F varies from image to image. H&S, S&T, FAST-9 and GLC detectors achieve somewhat better F values than the others. It is surprising that all detectors achieved $F < 0.5$: this is because all of them have high false positive rates — and this is despite the images having excellent contrast and essentially consisting of straight edges and corners. To back up the points made in chapter 3, all graphs in Figure 5.9 result in overlapping and crossed curves, making it impossible to say that one detector is better than the others. Hence, the value of F can really be used only to distinguish success from failure: an algorithm fails if its F is less than some threshold.

Table 5.2: Successes and failures for 417 corners at angles 90–135°

	H&S Pass	H&S Fail
SUSAN Pass	187	36
SUSAN Fail	87	107

However, also as discussed in Chapter 3, consideration of F, P and R do not take account of the number of corners processed. This can be remedied by employing McNemar's test. As in previous chapters, algorithms were assessed in pairs; for each corner in the dataset, the algorithms' results were categorized as one of *SS*, *SF*, *FS* and *FF*, as shown in Figure 5.10; the number of each of these determine the Z-score.

Table 5.2 shows a sample comparison, that between the Harris & Stephens and SUSAN detectors for corners belonging to Set C in Table 5.1. From the 417 corners, both detectors found 187 corners accurately and misidentified 107 corners. However, the remaining 123 corners are what determine the Z-score and hence the relative performances of the algorithms. Z-score $4.50 > Z_{\text{crit}}$, giving 99.9% confidence that H&S out-performs SUSAN.

For comparing multiple algorithms, McNemar's test tends to increase Type-1 error. To reduce this Type-1 error for comparing the six corner detection algorithms here, α is adjusted as follows:

$$1 - (1 - \alpha)^{1/A}$$
$$1 - (1 - 0.28)^{1/6} = 0.05$$

For $\alpha = 0.28$, $Z_{\text{crit}} = 1.08$; hence, a Z-score less than this value is considered insignificant; for example, $Z = 0.41$ in Table 5.3 shows similar performance of SUSAN and FAST-12. All six algorithms are compared in a similar way; the Z-scores explaining overall performances are given in Table 5.3.

Table 5.3: Z-score based on F-measure

	SUSAN	S&T	KLT	FAST-9	FAST-12	GLC
H&S	$\leftarrow$ 2.91	$\leftarrow$ 1.44	$\leftarrow$ 4.90	0.41	$\leftarrow$ 3.61	0.97
SUSAN		$\uparrow$ 1.66	$\leftarrow$ 3.33	$\uparrow$ 3.61	0.41	$\uparrow$ 1.57
S&T			$\leftarrow$ 4.25	$\leftarrow$ 1.58	$\leftarrow$ 2.41	0.00
KLT				$\uparrow$ 4.90	$\uparrow$ 3.02	$\uparrow$ 4.36
FAST-9					$\leftarrow$ 3.61	1.03
FAST-12						$\uparrow$ 1.92

The results presented in Table 5.3 illustrate performance differences much more clearly than the plots of Figure 5.9. In terms of overall performance, FAST-9 and H&S and GLC are roughly equal (with $Z < 1.08$). To explore whether this is the case for all types of corners, these detectors need to be compared separately on each of the sets A–D (Table 5.1), and these results are presented in the next section.

5.5 Test 2: Angular sensitivity

The second test was to analyze the angular sensitivity of corner detectors. Actual corners were divided into four groups based on their corresponding angles, as shown in Table 5.1. McNemar's test was employed to investigate the angular sensitivity of corner detectors based on detecting multiple corners with same angle but different orientation. The criterion for ascertaining success is shown in Figure 5.10.

5.5.1 Synthetic images' results

All six algorithms have been compared, with their Z-scores provided in Table 5.4. To read the score between two detectors, select the row of detector A and the column of detector B in the table; the direction of arrowhead indicates which corner detector performed better and the corresponding number gives the Z-score of the result. High Z-scores indicate more confidence in identifying the dominance of one detector over the other, while a Z-score less than Z_{crit} indicates that the two detectors performed similarly.

According to Table 5.4, H&S, S&T and GLC are the best detectors in detecting corners at acute angles ($< 90°$). However, in the case of obtuse-angled corners ($> 90°$), GLC maintained its ability to detect corners along with S&T, but H&S and SUSAN became unsuitable. Corners at angles greater than 135° are best-detected by KLT and FAST-9. It is worth noting that, for acute-angled corners, KLT appeared to be more sensitive (less Z-score), but for obtuse corners its high false positive rate cannot be ignored. Therefore, combining the two test results, KLT cannot be considered as being better than other detectors. Nevertheless, overall the best results are produced by the S&T and GLC — though both failed (along with other algorithms) in terms of detecting false corner, they perform similarly regarding angular sensitivity.

5.5.2 Results for real images

Similar to the procedure with synthetic images, Z-scores were calculated for corners with internal angles in specific ranges in real images and results are presented in Table 5.5. S&T out-performed all other detectors in all cases *i.e.,* the detection rate is better irrespective of corner angle. Similarly, H&S scores were out-performed by S&T due to the former's well-known systematic error in determining the locations of corners [201]. GLC with good scores for set B and set C shows that the corners at angles 45° to 135° are accurately detected by this detector.

FAST-9 and H&S appear to perform similarly, with $Z \approx 0$ for corners with

Table 5.4: Pairwise comparisons of detectors for corner detection at different angles in synthetic images. The direction of arrow-head points the better operator in pairwise comparisons. Z less than 1.08 presents a statistically non-significant performance difference.

	SUSAN	S&T	KLT	FAST-9	FAST-12	GLC	Score
Z-scores for corners at angles 11° *to* 45° (SET A)							
H&S	←1.5	0.0	← 9.86	←19.29	←19.29	←4.25	5
SUSAN		↑1.5	←9.54	←19.03	←19.03	←3.06	4
S&T			←9.86	←19.29	←19.29	←5.20	5
KLT				←13.87	←13.87	↑8.27	2
FAST-9					0.0	↑18.71	0
FAST-12						↑18.71	0
GLC						-	3
Z-scores for corners at angles 45° *to* 90° (SET B)							
H&S	←3.33	0.0	←14.59	←14.93	0.62	↑1.51	3
SUSAN		↑3.33	←14.14	←14.49	0.22	↑2.54	2
S&T			←14.59	←14.93	0.62	↑1.51	3
KLT				←2.85	↑13.64	↑15.20	1
FAST9					↑14.63	↑15.52	0
FAST-12						↑1.76	2
GLC						-	6
Z-scores for corners at angles 90° *to* 135° (SET C)							
H&S	←4.51	0.0	←6.76	←5.60	←12.93	0.0	4
SUSAN		↑4.44	←5.10	←1.39	←11.04	↑4.70	3
S&T			←6.70	←5.53	←12.89	0.0	4
KLT				1.06	←9.78	↑7.03	1
FAST-9					←10.45	↑5.05	1
FAST-12						↑13.59	0
GLC						-	4
Z-scores for corners at angles 135° *to* 164° (SET D)							
H&S	↑8.78	↑12.25	↑13.08	↑8.83	←1.08	←1.12	2
SUSAN		↑8.31	↑9.59	↑2.46	←6.46	←8.23	2
S&T			↑4.36	←5.13	←11.73	←12.29	5
KLT				←8.19	←13.71	←13.53	6
FAST-9					←10.91	←9.58	4
FAST-12						0.30	0
GLC							0

Table 5.5: Angular sensitivity test results for real images. The direction of arrowheads point the better operator in pairwise comparisons. Z less than 1.08 presents a statistically non-significant performance difference.

	SUSAN	S&T	KLT	FAST-9	FAST-12	GLC	Score
Z-scores for all pairs of detectors for corners at angles 10° *to* 45° (SET A)							
H&S	← 3.43	↑ 3.69	↑ 3.18	↑ 2.12	← 4.99	← 3.41	3
SUSAN		↑ 6.11	↑ 5.84	↑ 5.20	0.81	← 6.23	1
S&T			0.17	← 1.18	← 7.58	← 6.23	5
KLT				← 1.19	← 7.07	← 6.23	5
FAST-9					← 7.21	← 4.88	4
FAST-12						0.62	0
GLC						-	3
Z-scores for all pairs of detectors for corners at angles 45° *to* 90° (SET B)							
H&S	← 4.38	0.24	← 1.69	↑ 1.08	← 5.58	0.99	3
SUSAN		↑ 4.41	↑ 3.20	↑ 5.32	← 1.12	↑2.68	1
S&T			← 2.03	0.72	← 5.88	0.70	3
KLT				↑ 3.01	← 4.53	↑ 2.56	2
FAST-9					← 6.42	0.16	4
FAST-12						↑ 6.16	0
GLC						-	6
Z-scores for all pairs of detectors for corners at angles 90° *to* 135° (SET C)							
H&S	← 2.21	↑ 3.93	↑4.53	0.87	← 2.67	↑ 2.55	2
SUSAN		↑ 4.83	↑ 5.59	↑ 2.94	0.0	↑ 0.74	0
S&T			↑ 1.65	← 2.80	← 5.10	0.83	4
KLT				← 3.83	← 5.83	← 2.50	6
FAST-9					← 3.33	↑ 1.77	2
FAST-12						↑ 4.59	0
GLC						-	4
Z-scores for all pairs of detectors for corners at angles 135° *to* 164° (SET D)							
H&S	↑ 1.15	↑ 5.75	↑ 9.75	0.0	← 1.15	↑7.42	1
SUSAN		↑ 6.0	↑ 9.90	↑ 1.50	0.0	↑ 7.62	1
S&T			↑ 7.51	← 5.66	← 6.0	↑ 3.10	4
KLT				← 9.70	← 9.90	← 4.80	6
FAST-9					← 1.50	↑ 7.22	2
FAST-12						↑ 7.62	0
GLC						-	1

obtuse internal angles. For smaller angles, FAST-9's performance is significantly better than that of H&S. One might speculate that this is because corners at angles $> 135^\circ$ start to appear rounded due to shaded and noisy pixels and therefore identifying the precise location of the corner is difficult. Although KLT shows excellent performance for this set of corners, from the detector's F-measure curve in Figure 5.9 it can be seen that its exhibits a high false positive response: these apparently good results could be due to false positives falling at true corner locations. In other words, when choosing a detector, one must consider both the overall performance and the sensitivity to angle.

5.6 Is finding corners enough?

The results calculated in section 5.4.2 using McNemar's test are based on the F-measure; some might consider them as transforming F-measure plots into numerical comparisons. Similarly, the criteria developed in sections 5.4 and 5.5 for angular sensitivity focuses on identifying corners at the correct locations, ignoring the detection of non-corner pixels of images. True negative outcomes, although not very useful while evaluating detectors for the data used in previous sections, is also important but ignored in the calculation of the F-measure. Therefore, the evaluation framework used for these results cannot describe the complete behaviour of algorithms.

To assess the behaviour of corner detectors comprehensively for classifying *all* image points correctly, an evaluation framework is presented here where all pixels in an image need to be detected according to their class. For this purpose, instead of finding only corner pixels' locations, all pixels in an image are defined to be either corner or non-corner. The image that represents this classification, a 'ground truth image' has each pixel set to one of the following values:

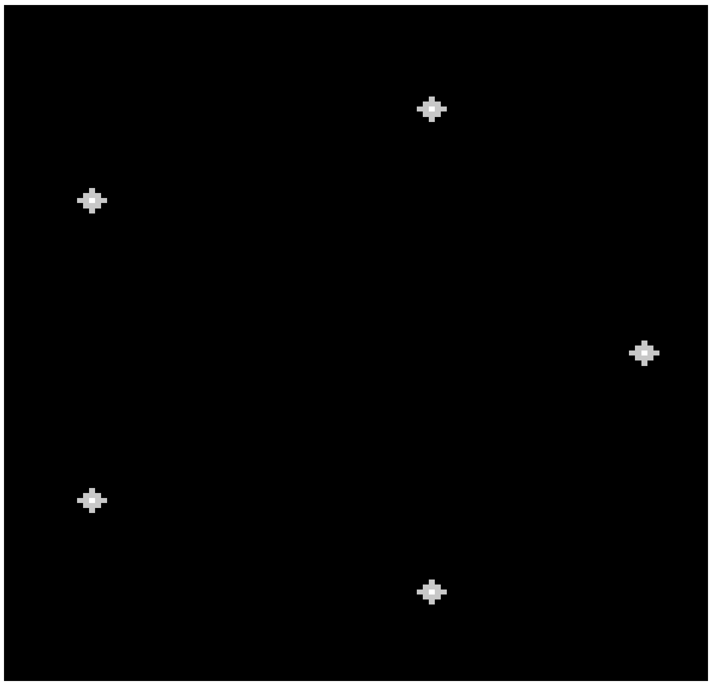


Figure 5.11: Ground truth image of a Pentagon

- 0 → background
- 255 → corner
- 200 → corner neighbourhood

A sample ground truth image is shown in Figure 5.11. This kind of ground truth image was generated for all synthetic and real images. The actual corner locations are known in case of synthetic images but for real images, the corner locations were determined using a combination of automatic and manual method as described in section 5.3.2.

In order to access individual detector's performances, they were compared using these synthetic and real images. To count pass and fail cases for McNemar's test, the following method was adopted: if a corner detector detects a corner pixel in original image, it is matched against three values in the ground truth image. If the detected location has value 0 in ground truth image, the de-

Table 5.6: Comparison of corner detectors with ground truth synthetic images of size 200 x 200. Algorithms are sorted on minimum Z-score and high % of TP corners

Detector	PP	FF	PF	FP	Z-score	%TP Corners
H&S	1793702	0	452	0	21.21	46.76%
FAST-12	1791102	0	496	0	22.23	14.76%
SUSAN	1788947	0	504	0	22.41	51.20%
FAST-9	1791096	0	517	0	22.69	19.73%
GLC	1789622	0	556	0	23.54	48.89%
S&T	1792747	0	629	0	25.04	59.91%
KLT	1765464	0	1309	0	36.15	48.44%

Table 5.7: Comparison of corner detectors with ground truth real images. Algorithms are sorted on minimum Z-score and high percentage of TP corners

Detector	PP	FF	PF	FP	Z-score	%TP Corners
FAST-12	401214659	0	917	0	30.25	7.02%
SUSAN	401210830	0	974	0	31.18	7.56%
H&S	401211691	0	1122	0	33.47	11.29%
FAST-9	401172872	0	1740	0	41.69	14.04%
GLC	401169192	0	2047	0	45.22	9.51%
S&T	401091955	0	3285	0	57.30	16.36%
KLT	378926953	0	44780	0	211.61	19.56%

tector failed and ground truth passed. If the detected location has value 255 or 200, both detector and ground truth passed. For rest of the pixels, both detector and ground truth are considered as passes as the detector did not detect these background pixels as corner pixels. The scores are presented recorded in Tables 5.6 and 5.7. In the table, PP means a detector's pixel classification is correct, FF means the detector and GT image both show wrong classification (which can never be the case), PF shows detector's failure, and FP show ground truth failure (which again can never occur). Low Z-scores highlight a good corner detector, showing similar behaviour to the ground truth.

This test focuses on correctly classified image pixels but does take into account the total number of corners correctly identified in the imagery. However, the best detector should identify maximum number of corners in an image, so

Table 5.8: Pairwise comparison of corner detectors for synthetic images using ground truth images. Z-score less than 1.08 is considered non-significant.

	SUSAN	S&T	KLT	FAST-9	FAST-12	GLC	Score
H&S	← 7.04	←6.90	← 15.86	← 14.96	← 18.74	← 2.65	6
SUSAN		↑ 4.46	← 16.45	← 14.77	← 17.52	← 2.61	4
S&T i			← 14.88	← 11.94	← 14.80	← 1.71	5
KLT				↑ 8.96	↑ 7.21	↑ 13.82	0
FAST-9					←18.89	↑ 5.64	2
FAST-12						↑ 8.04	1
GLC						—	3

Table 5.9: Pair wise comparison of corner detectors for real images using ground truth images. Z-score less than 1.08 is considered non-significant.

	SUSAN	S&T	KLT	FAST-9	FAST-12	GLC	Score
H&S	↑ 1.51	←42.85	← 203.72	← 17.48	0.74	← 17.11	4
SUSAN		← 33.85	← 203.59	← 15.68	← 2.21	← 18.67	6
S&T			← 192.28	↑ 14.85	↑ 32.35	↑ 14.81	1
KLT				↑ 197.93	↑ 203.72	↑ 196.12	0
FAST-9					↑4.99	← 6.44	3
FAST-12						← 19.81	4
GLC						—	2

the last column shows the percentage of correct corners identified by each detector. Hence, a low Z-score and high percentage of TP corners identifies the best detector. From Tables 5.6 and 5.7, H&S worked very well for synthetic images but for real, noisy images SUSAN and FAST-12 performed better than Harris & Stephens.

Similarly, the same principle can applied to pairwise comparisons of detectors. If both detectors detect a pixel as corner for which the ground truth image pixel value is 200 or 255, then both detectors pass, otherwise only one of them can pass; or both can fail if the ground truth image pixel value is zero.

These pairwise comparisons among detectors reveal similar results as generated while comparing detectors with ground truth, that H&S worked very well for synthetic images but for real, noisy images SUSAN and FAST-12 performed better than H&S as indicated by the rankings shown in Tables 5.8 and 5.9 con-

Table 5.10: Combining H&S with FAST, SUSAN and GLC; Results are sorted according to minimum Z-score and high percentage of corners detected

Detectors	**Z-score for Synthetic Images vs GT**	**% of Corners Detected (Synthetic Images)**	**Z-score for Real Images vs GT**	**% of Corners Detected (Real Images)**
H&S + GLC	21.19	59.91%	30.53	16.98%
H&S + FAST-9	22.16	56.18%	30.71	16%
H&S + SUSAN	23.11	52.36%	30.66	16.27%
H&S + FAST-12	24.12	48.09%	31.03	14.22%

trary to the results reported in sections 5.5.1 and 5.5.2 shows S&T to be better than H&S for synthetic images and FAST-9 appeared to be better for real images. Both algorithms (H&S and S&T) share same working principle to find corner pixels in an image, and hence can be considered as similar.

These results suggest that some detectors are better than others for different corner angles; so combining them may give better results. The following section explores this.

5.7 Complementarity of corner detectors

Different detectors have different principles of operation, and hence perform differently on the same images. H&S and S&T use eigenvalues to find corner and edge pixels, which appears to be a more effective approach than mask-based methods like SUSAN and FAST. However, for real images, SUSAN and FAST give better performance, so it is interesting to combine two detectors that work differently to see whether overall performance on all kind of images is improved.

Table 5.11: Combining S&T with FAST, SUSAN and GLC; Results are sorted on minimum Z-score and high percentage of corners detected

Detectors	**Z-score for Synthetic Images vs GT**	**% of Corners Detected (Synthetic Images)**	**Z-score for Real Images vs GT**	**% of Corners Detected (Real Images)**
S&T + GLC	20.74	61.6%	29.73	21.24%
S&T + FAST-12	20.86	61.12%	30.22	18.67%
S&T + FAST-9	20.98	60.71%	29.93	20.18%
S&T + SUSAN	21.09	60.27%	29.79	20.98%

Based on individual performances on synthetic and real images, H&S and S&T are combined with other detectors and their combined performances evaluated. The results presented in Tables 5.10 and 5.11 indicate that combining either of these two detectors with GLC leads the detection of more corner points with fewer negative results, in both noisy real and noise-free synthetic images. These results are obtained using the ground truth images described in section 5.6 to predict general performance trends of combination of detectors.

5.8 Remarks

This chapter presented statistical approach for evaluating corner detectors using the non-parametric McNemar's test, which proved to be an improvement on classical performance evaluation methods. More importantly, it has been shown that statistical approaches to comparing performances can be used easily in the vision domain. Statistically significant performance differences among detection algorithms have been observed, both in terms of overall performance and with regard to performance in specific ranges of angle. With high Z-scores and low false positive rates, H&S, S&T and SUSAN proved to be the best detectors of those tested. The FAST detector, though time-efficient, was found to be significantly more sensitive to the corner angle and therefore needs to be used in roles where this is not important. Likewise, combined detectors' results can be used to direct research into hybrid corner detectors for more stable and reliable detection of corner points at all angles.

The detection of corner points is important — but detecting corners is only half of the story, as many applications require that they can be matched accurately across multiple images. Recent research has produced a number of feature descriptors [79–90] that attempt to identify distinctive information around an image feature, helping match it in other images. However, up till now, no descriptor has been defined to explore a corner's most distinctive property, its internal angle; and furthermore, the time take to calculate the above descriptors is too long for them to be used in real-time applications. Although it is true that all other feature descriptors can be used for describing corner points, it would be interesting to explore matching corners on the basis of their characteristic angle. If this is possible, it provides an important foundation for a navigation system for the blind. To this end, the next chapter presents two descriptors specifically designed to match corner points in real time for vision-based tracking and navigation problems.

CHAPTER 6

DESCRIPTORS FOR CORNER POINTS

6.1 Introduction

Corners are important features in images because they typically delimit the boundaries of regions or objects. Moreover, they also identify reproducible locations on the boundaries of objects, making them important for identifying equivalent locations in stereo pairs or successive frames of video sequences. For real-time applications such as the subject of this thesis, it is essential that corners are detected and matched reliably and rapidly.

A *descriptor* collects information around a point of interest such as a corner with the aim of being describing it sufficiently well for it to be matched uniquely with the same corner of the same object in a different image. Detailed neighbourhood information accommodates matching points in images under different transformations. A number of descriptors have been presented in the literature, amongst which SIFT [80], SURF [79], GLOH [10] and BRIEF [82] are considered state-of-the-art. SIFT and GLOH use the gradient direction of pixels in a specified region around an interest point, while SURF calculates the Haar wavelet response of neighbouring image pixels for fast descriptor calculation and matching. BRIEF is different from others in that it uses simple intensity comparisons of neighbourhood pixels to generate a binary descriptor which can be matched using Exclusive OR (XOR) binary operator, a very fast matching technique. Although all of these descriptors can be used to describe corner points, none of them exploit a corner's inherent property, its internal angle. Of those listed above, only BRIEF descriptor can be calculated in real time.

Two related descriptors are presented in this chapter which are compatible with standard corner detectors: one encodes the entire circular region within a corner and is called **C**ircular **M**ean **I**ntensity and **E**ntropy (CMIE), while the other describes only the region within an object and is called **A**ngle, **M**ean **I**ntensity and **E**ntropy (AMIE). Section 6.2 describes the informative region around a corner, which motivates the development of the CMIE and AMIE descriptors.

The method of selecting and describing whole circular region around corner for CMIE is detailed in section 6.3. Similarly, the method of selecting only the internal part of an object by calculating corner's internal angle for AMIE is given in section 6.4. As mentioned before, real-time matching of corner points is essential for a corner descriptor used in a navigation system, so the speed of calculating and matching descriptors is examined in section 6.5.

The robustness to noise of AMIE and CMIE is explored in section 6.6. The advantage of encoding only the region within an object is demonstrated and it is shown that the accuracies of both descriptors are comparable to that of BRIEF for vision application like template matching, tracking and surveillance.

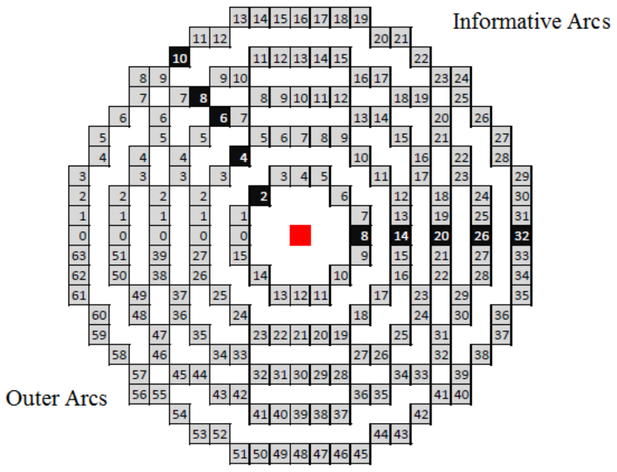


Figure 6.1: Circles of different radii around a corner point. The central red cell represents the corner and black cells edge pixels. The values in the grey cells represent the order in which the values around the arcs of radii 3, 5, 7, 9, and 11 pixels are processed.

6.2 Describing the region around a corner

In devising a descriptor, one needs to bear in mind that matching descriptors needs to be rapid if a system that uses them is to execute in real time; this mitigates towards a short descriptor. Conversely, the descriptor must summarizes the region around a corner sufficiently well that the number of incorrectly-matched corners remains low. The approach adopted here, inspired by the simplicity and effectiveness of FAST [118], is to encapsulate information gathered from several circular arcs around a corner into a descriptor as these summarize the corner concisely.

At each corner point, circles of different radii are used and the pixels on each circle contribute towards the descriptor. There is a trade-off between the number and spacing of circles, their radii, and the speed of matching; radii of 3, 5, 7, 9 and 11 have been found to yield a descriptor that is able to distinguish different corners and still be calculated rapidly. Bresenham's circle drawing algorithm [202] is employed to identify those pixels that lie on the required circles as shown in Figure 6.1.

The simplest way to describe an image region around any circular arc of radius r is the mean intensity of the n pixels p_i that describe the arc

$$\mu_r = \frac{\sum_{i=1}^{n} p_i}{n} \tag{6.1}$$

However, many different sets of pixel values may yield the same mean, so further information is required to make a descriptor unique. Entropy measures the information content and has been used for characterizing texture [203] and for image segmentation [204]; hence, it should be complementary to the mean and the combination of the two should have much more discriminatory power than either mean or entropy in isolation. The entropy E_r of the pixels lying on the circular arc is

$$E_r = -\sum_i p_i \log_2 p_i \tag{6.2}$$

where p_i is the probability that the difference between two adjacent pixels is equal to i. It can be calculated quickly using histograms and look-up tables. Entropy is the measure of amount of image information, therefore, becomes zero for flat areas of the image. The solution for this is to calculate entropy values for small enough areas. Here the use of circular arcs for entropy calculation compensates for the flat area issue.

A circle around a corner can be divided into two distinguishable arcs separated by edge pixels. The arc whose mean is closer to the value of the corner is denoted here as the “informative arc” while the remaining arc is the “outer arc”, as illustrated in Figure 6.2. Figure 6.3 plots the entropies of the corners points detected in Figure 6.2; it is evident that the entropies of informative arcs are higher than those of outer arcs and whole circles.

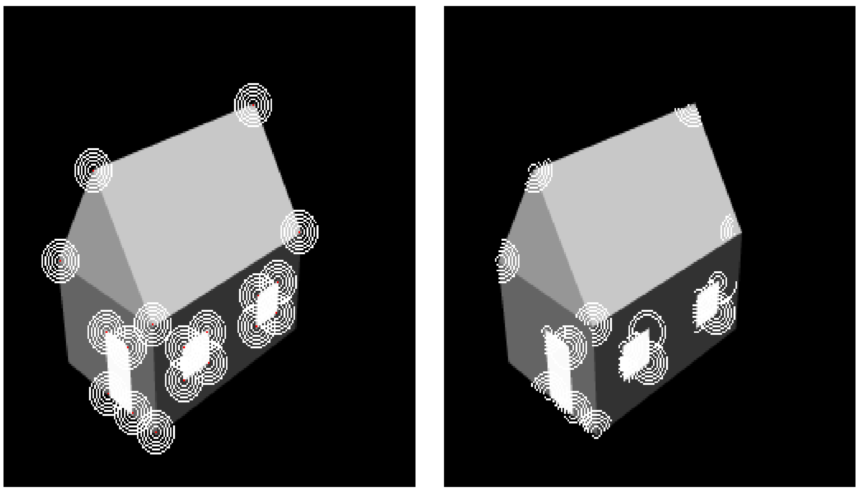

(a) Informative Circles (b) Informative Arcs

Figure 6.2: Circular arcs around corner points

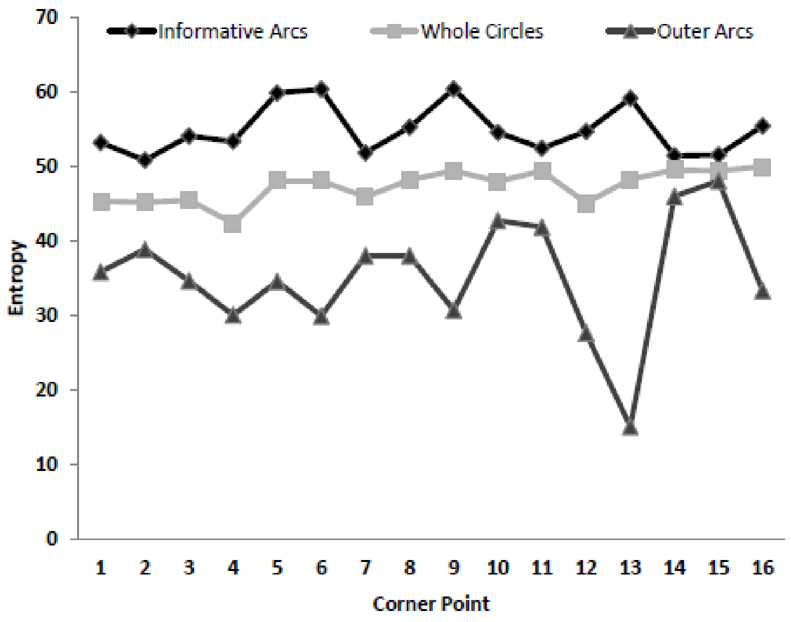


Figure 6.3: Entropy distribution around the corners in Figure 6.2

The entropy values for whole circles around corner points are reasonably close to those of informative arcs. Hence, employing complete circles in a descriptor should be effective if the background outside a corner does not change substantially, as one would usually find from frame to frame of a video, say. However, if the background is not consistent, using a complete circle is likely to cause mismatches and a descriptor employing only the informative arc should perform better. Hence, two related descriptors are discussed and compared here, one using complete circles and the other only informative arcs.

6.3 Describing complete circles: CMIE

The CMIE (**C**ircular **M**ean **I**ntensity and **E**ntropy) descriptor encapsulates information around the complete circles of Figure 6.2(a). The mean intensity and entropy around each circle are calculated and stored in a descriptor (Figure 6.4): the first five locations contain the mean intensities μ_r around the circles and the remaining five contain entropies E_r.

To find corresponding corner points in different images or video frames, the Euclidean distance between the mean intensity and entropy parts of two CMIE descriptors d_1 and d_2 is used

$$D_\mu = \sqrt{\frac{1}{5}\sum_{i=1}^{5}(d_{1_i} - d_{2_i})} \tag{6.3}$$

$$D_E = \sqrt{\frac{1}{5}\sum_{i=6}^{10}(d_{1_i} - d_{2_i})} \tag{6.4}$$

where the subscript i relates to Figure 6.4. Two different thresholds are required, one for the mean intensity part of the descriptor and the other for the entropy

CMIE :	μ_3	μ_5	μ_7	μ_9	μ_{11}	E_3	E_5	E_7	E_9	E_{11}						
AMIE :	μ_3	μ_5	μ_7	μ_9	μ_{11}	E_3	E_5	E_7	E_9	E_{11}	θ_3	θ_5	θ_7	θ_9	θ_{11}	ϕ
index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Figure 6.4: The content of the CMIE and AMIE descriptors (μ_r is given by 6.1, E_r by 6.2, θ_r by 6.5 and ϕ is the orientation in radians)

part. Comparing two vectors containing two sets of five values is fast and, even using exhaustive search, the time to find matches is rapid; this is discussed in detail in section 6.5.

6.4 Describing informative arcs: AMIE

While CMIE describes the circular region around a corner, AMIE (**A**ngle, **M**ean **I**ntensity and **E**ntropy) contains information about only the informative arc. The following two-step algorithm allows the informative arc of a corner to be determined.

1. Edge pixel identification. Edge pixels around a corner are first identified on circles of radii of 3, 5, 7, 9 and 11 pixels, as with CMIE . While scanning the circles, the eigenvalues (λ_1 and λ_2) at each pixel are used to find corner and edge pixels as described in [76]: if $\lambda_1 \sim \lambda_2 \gg 0$ then the point is a corner but if $\lambda_1 \not\sim \lambda_2$ then the point lies on an edge. The result is two candidate circular arcs separated by edge pixels on each circle (6.2). Noise and blurring effects lead to ambiguities in the intersecting edges, so edge pixels are chosen to maximize the corner angle. Arcs at different radii are allowed to have different angles as this accommodates irregular boundaries.

2. Corner orientation detection. The second step is to identify which arc is the informative arc. For this, the orientation of the corner point is determined using a voting mechanism in which each circle can vote for either of the candidate arcs or the entire circle; the latter is chosen when the number of edges found is less than two — this accommodates incorrect responses from the detector due to noise *etc.* The mean value of pixels in each candidate arc is compared with the corner point's value: if the difference between them is less than some threshold (a normalized intensity threshold of 0.12 has been found empirically to be effective), then the orientation is towards the first candidate arc; otherwise, it is towards the second candidate arc. This calculation is carried out for each circle independently, the final orientation being chosen as that receiving the largest number of votes. Once the orientation has been determined, the selected candidate arc is taken to be the informative arc. The angle of a circular arc of radius r can easily be estimated as:

$$\theta_r = 2\pi \left(\frac{\text{no. of pixels in arc of radius } r}{\text{no. of pixels in circle of radius } r} \right) \tag{6.5}$$

The θ_r values for the five informative arcs are calculated and stored in the AMIE descriptor along with the orientation ϕ of the informative arc, represented by 1, 2 or 3 for the inner arc, outer arc or whole circle respectively; see Figure 6.4. Arcs at different radii are allowed to have different angles as this accommodates irregular boundaries.

The AMIE descriptor is computationally inexpensive and memory efficient, involving only 16 values. Although the descriptor components are only simple summary statistics of pixels, it will be shown in section 6.6 that the descriptor is sufficiently discriminatory for use in applications requiring the matching of corners.

Matching AMIE descriptors involves two steps. Given a corner to be matched from one image in another, the first step is to find corners in the second image whose descriptors have similar orientation and angle to the corner being

Table 6.1: Time required for the steps involved in calculating the CMIE and AMIE descriptors for 100 corner points

Processing step	CMIE	AMIE
Circular templates	0.08 ms	0.08 ms
Descriptor components	1.62 ms	1.46 ms
Total time	1.70 ms	1.54 ms

matched, by calculating the Euclidean distance (D_θ) of the angular part of two descriptors d_1 and d_2:

$$D_\theta = \sqrt{\frac{1}{5}\sum_{i=11}^{15}(d_{1_i} - d_{2_i})} \tag{6.6}$$

If the corners are within $\approx 10^\circ$, the point qualifies as a candidate corner. Then, from the set of candidate corners, the second step is to find the best match based on the Euclidean distances of the mean intensity and entropy values of the informative arcs; this is done in a similar way to CMIE , using (6.3) and (6.4).

CMIE encapsulates the information around complete circles, so is inherently rotation-independent. AMIE has also been designed to be rotation-independent because it summarizes the region within the informative arc, which is determined by the arc's enclosing edges. However, the selection of the correct orientation depends crucially on the localization accuracy of the corner detector: if a corner is located inside the boundary, the orientation is towards the inner part of the boundary, and conversely if located outside. To ensure that noise and small systematic errors in locations do not cause problems, in practice a 3×3 region around each corner point is used for calculating its mean intensity.

Although CMIE and AMIE are rotation-invariant, they are not scale-invariant: they are intended for use in applications involving matching frames in real-time video, where scale changes of moving objects are typically small.

6.5 Speed of computation

As mentioned above, the speed of calculating and matching descriptors is paramount in real-time applications. The processing times required to calculate CMIE and AMIE descriptor components are given in Table 6.1; note that AMIE is typically faster to calculate than CMIE because fewer pixels are involved. There are two mains steps: the first is to identify the locations of pixels on circles around a corner, while the second is to calculate the mean intensities and entropy values of the pixels on the circles. The mean overall time to calculate the CMIE descriptors of 100 corners comes to 1.70 ms on a desktop PC with Core 2 Quad CPU Q9550 2.83 GHz processors.

AMIE requires one more computational step than CMIE , that of calculating an eigen decomposition for pixels around a corner point. If the detector is H&S or S&T, this calculation is part of the corner detection process and the result of that calculation can be used. However, to use AMIE with another corner detector, the eigen decomposition for a patch size of 11×11 around the corner point is calculated for edge pixel identification. This additional step takes 5.48 ms for 100 corners; the time for the remaining calculations is given in the last column of Table 6.1.

To assess performances, AMIE and CMIE are compared below with the well-established BRIEF descriptor, known for its fast computation and matching. BRIEF is matched using the originally proposed Hamming distance [205] calculation using OpenCV library.

As AMIE calculates the informative region around a feature, its computation time varies with image content: corners with large angles take more time than those with small angles. To achieve a fair analysis, a set of 28 images was selected: video frames (shown in Figure 6.9), the UBC and Graffiti images from the

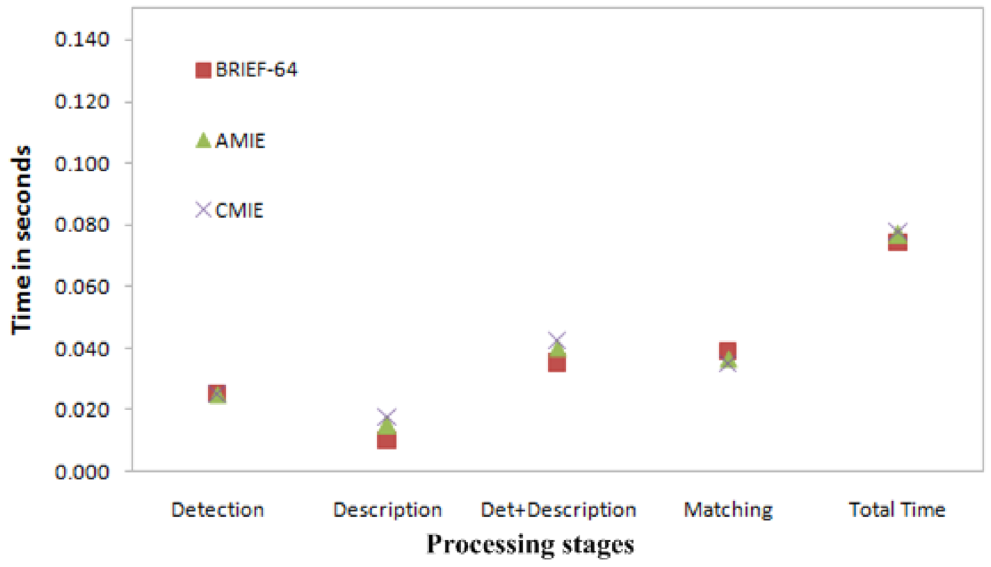


Figure 6.5: Mean time per image required to calculate descriptors for ≈ 800 corner points using AMIE , CMIE and BRIEF for 28 images of 640 × 480 pixels

well-known Oxford database, and some frames from Performance Evaluation of Tracking and Surveillance (PETS)-2012[1] video data. The mean times taken for detecting corner points, calculating descriptors and then matching them are shown in Figure 6.5. Although the AMIE descriptor is longer than the CMIE one, its two-step matching algorithm takes on average somewhat less time. The mean time of 0.052 seconds per frame for descriptor calculation and matching makes it suitable for real-time video applications. In fact, the time required to calculate all three descriptors (AMIE , CMIE and BRIEF) is similar.

6.6 Performance Characterization

This section compares the performances of CMIE and AMIE with BRIEF in two ways: the first assesses their ability to accommodate image noise, as video frames tend to be rather noisy, while the second involves the type of matching one

[1]http://www.cvg.rdg.ac.uk/PETS2012

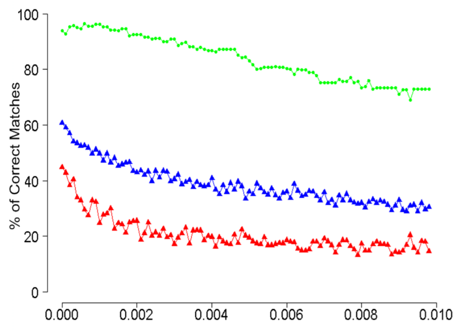
would expect to encounter in real applications. Indeed, three such applications are considered, matching for tracking or navigation, video surveillance, and template matching.

6.6.1 Performance on noisy images

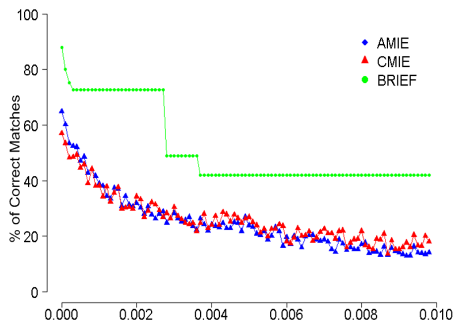
Applications aspiring to match corners in video frames require the descriptions of corners to be robust to noise. To assess the degradation of the descriptors with noise, the S&T corner detector was first used to locate the corners in a database of 500 images [10] and descriptors of them were obtained using CMIE , AMIE and BRIEF. Some 100 versions of each image were then produced with additive Gaussian noise, the noise being added using Matlab's `imnoise`[2] function. Descriptors were then calculated for each corner position in each noisy image; only if the descriptor at the correct corner position was the best match for the noise-free corner can one say that the corner has been matched correctly. This process was repeated for noise variances ranging from 0.0001 to 0.01; the latter is equivalent to 10% noise. These results are presented in Figure 6.6. BRIEF is clearly more robust to noise than both CMIE and AMIE , particularly when the amount of noise is low. However, there are significant discontinuities in the performance of BRIEF for most of the standard image sets in Figure 6.6, an undesirable characteristic. One might expect AMIE to be more susceptible to noise than CMIE because its calculation involves edge pixels around the corner point, yet that is not the case: AMIE generally out-performs CMIE.

Although the greater robustness of BRIEF is desirable, practical applications rarely need exact corner-to-corner matching; instead, one usually wants a rough outline of an object or an accurate estimation of a homography; providing there is no systematic error in positioning, achieving many matches between features

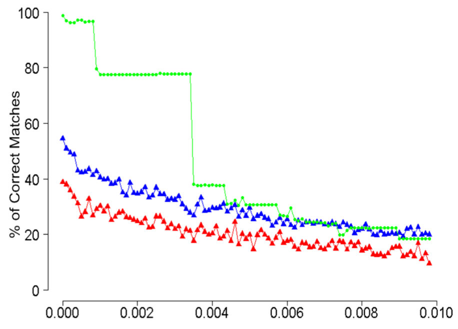
[2] http://www.mathworks.co.uk/help/images/ref/imnoise.html



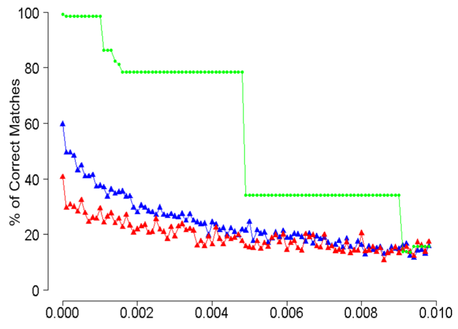
(a) Graffiti



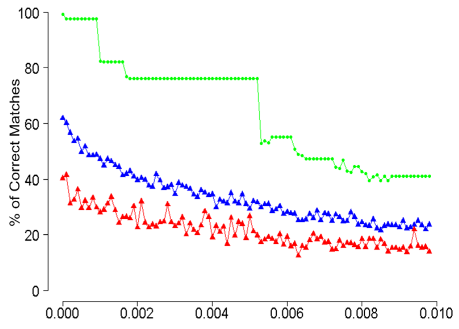
(b) Coin



(c) UBC



(d) Snow



(e) Leuven

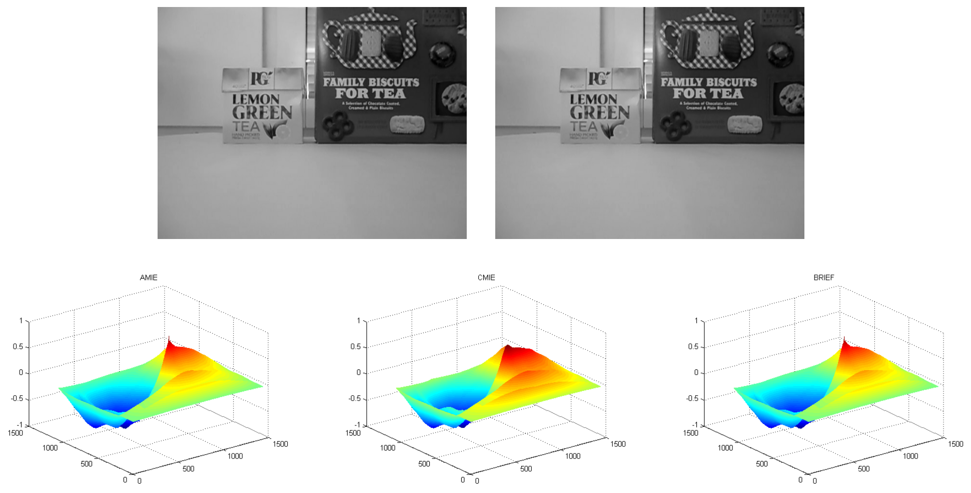
Figure 6.6: Percentage of correct matches between noise-free and noisy images (the x-axis shows noise variance and the y-axis gives the percentage of true corners)

will ameliorate the inaccuracy of the individual matches — this will be considered in detail in section 6.6.2.

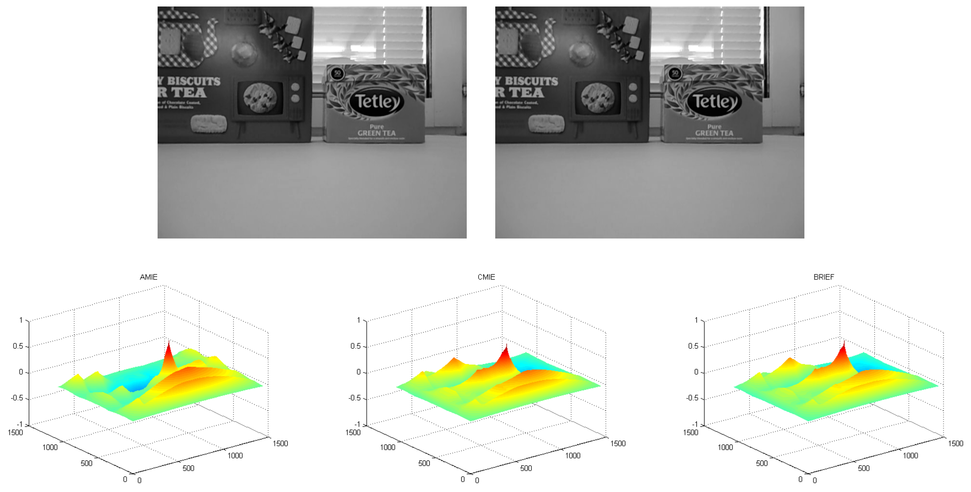
6.6.2 Matching video frames (captured using hand-held camera)

Robust and fast matching is vital for tracking and navigation applications. Evaluating the performance of feature detectors on real-world video sequences in the way used above for noise is notoriously difficult due to the need for 'ground truth.' To circumvent the need for knowledge of the locations of all corners in all frames, the homography method described in chapter 3, based on that of [10] and [14], is used: matches from a pair of frames is used to estimate the homography between the frames; one image is then projected onto the other and image subtraction or cross-correlation used to assess the accuracy of that projection. If the transformation is accurately determined, then the subtraction of the projected image from the reference should result in a completely black image, while cross-correlation should yield a single sharp peak. This is a better estimate of real-world performance because it uses real video sequences rather than images with artificially-added noise, and because the estimation of a transformation matrix is similar to what is done in many practical applications involving matching corners. Here, the video is captured using a moving, hand-held camera.

Figure 6.7 shows some of the matching results between frames and the corresponding cross-correlation peaks. Two pairs of frames are considered, frames 50 and 51, and frames 350 and 351. In each case, the two frames are first shown, then the normalized cross-correlation surfaces between frames as 3D plots using AMIE , CMIE and BRIEF respectively. In these plots x and $y - axis$ are the image sizes while $z - axis$ shows the cross-correlation value. These surfaces all exhibit a single, reasonably narrow peak, indicating that the matches are accurate.



(a) Frames 50 and 51



(b) Frames 350 and 351

Figure 6.7: Matching video frames using CMIE , AMIE and BRIEF (see text for details)

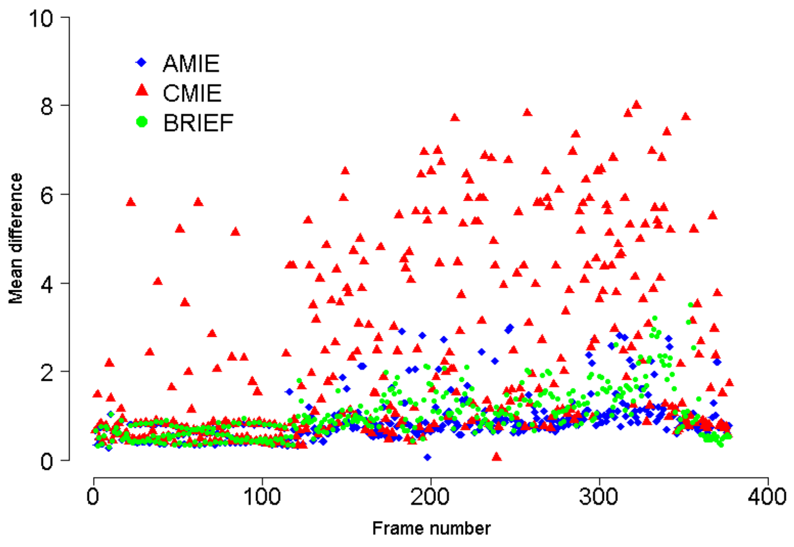
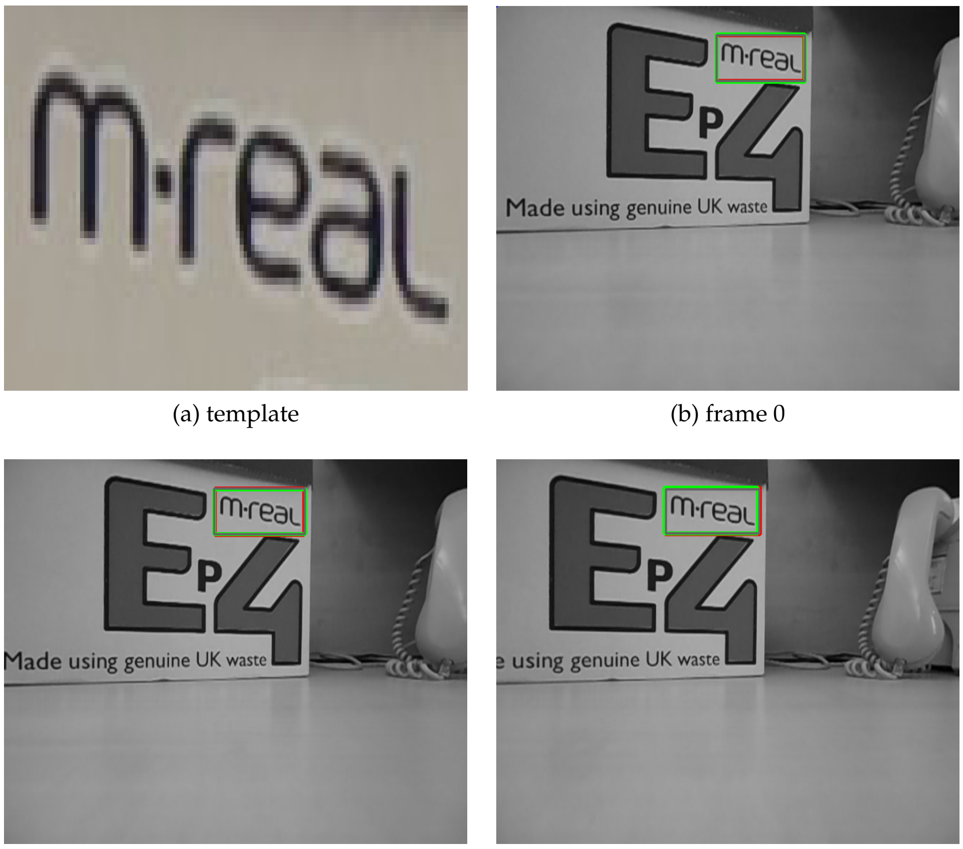


Figure 6.8: Mean difference between warped and original frame in the whole video

The mean difference between projected and reference images across all 380 frames in the video sequence is shown in Figure 6.8. As expected, CMIE gives the worst performance because it experiences a larger number of false matches, essentially because it encodes the entire circular region around each corner. The requirement for a pair of corners to have similar angles means that AMIE generates much fewer false matches and consequently its performance is visually indistinguishable (and statistically insignificant) from that of BRIEF.

6.6.3 Template matching in a video sequence

To explore the effectiveness of the descriptors for template matching, a video was captured using a hand-held camera. A template was cropped from the first frame and matched in all subsequent frames of the video using CMIE , AMIE , BRIEF, and SIFT. In Figure 6.9, the positions of the template obtained using the different descriptors is drawn on several frames of the sequence. In this case,



(a) template (b) frame 0

(c) frame 200 (d) frame 400

Figure 6.9: A template from first frame of video sequence is matched in every frame. Estimated position of template from matched points is shown with different colour boxes: red represent SIFT, green CMIE , grey AMIE and blue BRIEF.

CMIE performed either similarly or a little better than AMIE , due to the similar background of the template and the frames.

The homography for projecting the template into each frame was then determined using matched points produced by all four descriptors; that obtained using SIFT was taken as being the most accurate. Points in the template were then projected into subsequent frames using homographies calculated from each descriptors' matches. These homography matrices were tested using the homography framework described in chapter 3: the estimated homography matrices calculated using AMIE, CMIE and BRIEF were compared with that of SIFT for 1000 equally-spaced points and Z-scores were calculated for each frame. The Z-scores for 1000 frames of the sequence are shown in the Figure 6.10. There are three algorithms under comparison and hence, applying the Bonferroni correction to McNemar's test, $Z_{\text{crit}} = 1.43$ for $\alpha = 0.15$. It is obvious from the graph that for almost 900 frames the performances of all three algorithms are similar to that of SIFT, with $Z < Z_{\text{crit}}$. However, in the last frames there is a considerable amount of scale change as compared to first frame and therefore all three descriptors' performance decreased significantly, with $Z > Z_{\text{crit}}$.

Those cases where the performances of all three descriptors dipped sharply (around frames 240, 290 and 315) correspond to blurred imagery due to rapid camera motion or auto-focus hunting; AMIE and CMIE appear to be less affected by this than BRIEF but in no case did performance become unacceptable for any descriptor.

6.6.4 Video surveillance (Performance on PETS-2012 videos)

The descriptors have also been tested on widely-distributed datasets for surveillance, the PETS-2012 video data. These differ a little from the results of Fig-

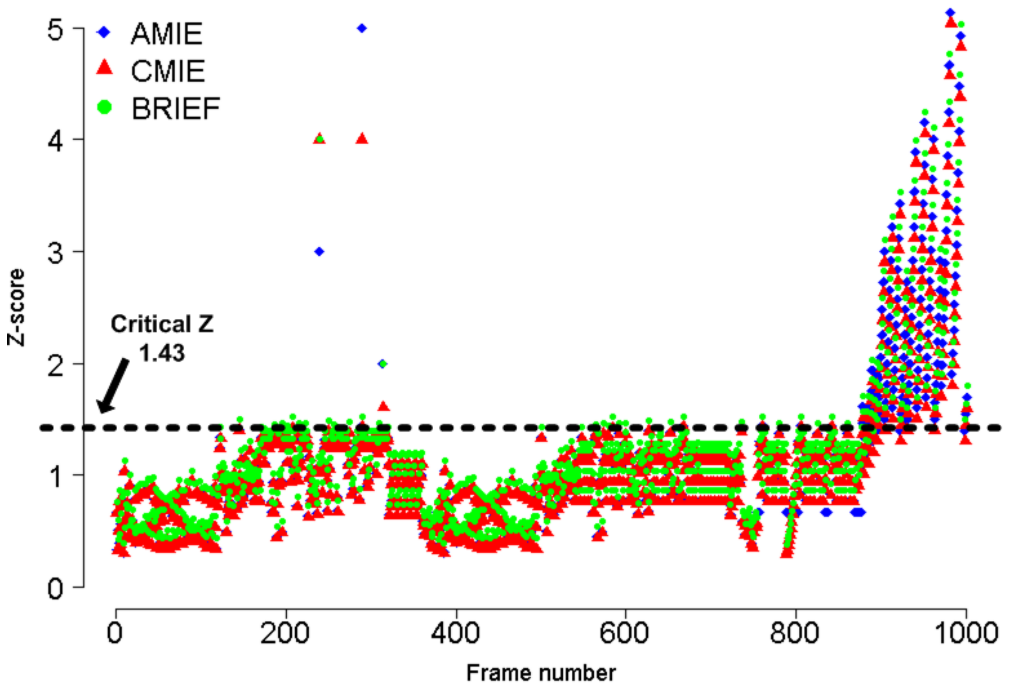


Figure 6.10: McNemar's test between SIFT and other descriptors. Comparing three algorithms using McNemar's test needs α adjustment. According to Bonferroni correction the α should be 0.15 for which the critical Z-score is 1.43. Hence a Z-score less than 1.43 represents a non-significant performance difference between SIFT and other descriptor.

Table 6.2: The mean warping difference of video frames and corresponding variances in sequences from the PETS-2012 dataset. The first column gives video name, where BG = Back Ground, CC = City Center, RF = Regular Flow, and the numbers 01, 02 *etc.* represent views. The time of the video is given in parenthesis.

	BRIEF		AMIE		CMIE	
Video	$\bar{d}$	σ^2	$\bar{d}$	σ^2	$\bar{d}$	σ^2
BG-01(13-38)	2.103	0.004	2.125	0.010	2.181	0.030
BG-02(13-06)	2.033	0.064	2.059	0.067	2.122	0.075
BG-02(13-32)	1.755	0.003	1.774	0.005	1.860	0.019
BG-02(13-38)	1.864	0.018	1.886	0.020	1.963	0.028
BG-03(13-06)	3.121	0.354	3.328	0.020	3.114	0.169
BG-03(13-19)	3.701	0.277	3.898	5.318	3.757	0.133
BG-03(13-32)	2.560	0.014	2.636	0.031	2.731	0.151
BG-03(13-38)	2.603	0.112	2.619	0.079	2.685	0.126
BG-005	1.959	0.001	1.987	0.001	5.768	75.846
BG-006	1.858	0.004	1.881	0.002	2.622	11.319
BG-007	2.219	0.003	2.256	0.005	5.029	53.828
BG-008	2.184	0.002	2.253	0.055	4.134	13.060
CC(12-34)-01	3.191	0.219	3.213	0.219	3.256	0.225
CC(14-55)-01	6.588	5.028	9.725	5.030	12.394	25.040
CC(14-55)-02	4.421	2.443	4.924	17.134	5.911	38.123
CC(12-34)-02	2.331	0.390	2.346	0.393	2.448	0.621
CC(12-34)-03	2.214	0.232	2.239	0.229	2.363	0.261
CC(14-55)-03	2.029	0.058	2.055	0.062	2.124	0.074
L1(13-59)-02	5.299	2.764	6.179	6.505	7.732	35.442
RF(13-57)-01	6.429	3.420	7.041	4.665	9.296	42.128
RF(13-57)-03	5.732	0.434	7.251	0.031	6.602	1.083
RF(13-57)-04	4.516	1.797	6.106	1.650	7.837	48.986
RF(13-59)-01	5.289	2.778	6.158	5.160	7.711	35.398
RF(13-59)-02	2.461	1.969	2.731	2.759	3.177	6.856
RF(13-59)-03	5.611	1.652	7.009	1.620	6.517	5.752
RF(14-03)-01	5.118	1.041	5.350	1.485	6.472	14.523
RF(14-03)-02	2.728	0.751	2.891	0.950	3.266	2.047
RF(14-03)-03	5.377	0.254	6.170	4.968	5.968	1.212
RF(14-06)-01	5.859	7.216	7.404	15.717	9.349	43.279
RF(14-06)-02	2.960	1.814	3.412	3.241	4.153	8.561
RF(14-06)-03	4.969	0.760	6.015	1.467	5.871	1.299
RF(14-29)-01	4.835	0.937	5.392	3.106	7.303	31.851
RF(14-29)-02	3.607	2.518	3.704	2.696	4.327	5.381
RF(14-29)-03	4.572	0.427	4.790	0.721	4.615	0.481

Table 6.3: Warping followed by subtraction of matched video frames from the glspets-2012 video database, using homographies estimated from matched points from descriptors under study. The frames along each row were obtained using AMIE , BRIEF and CMIE respectively.

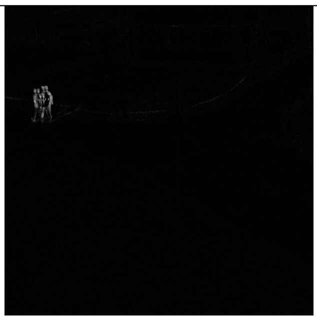
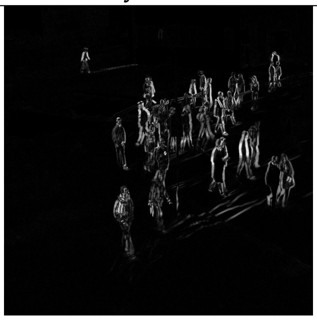
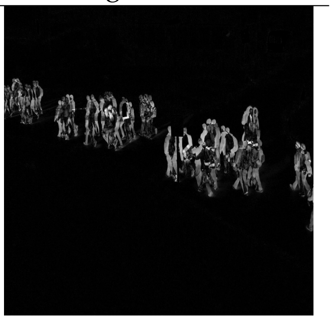
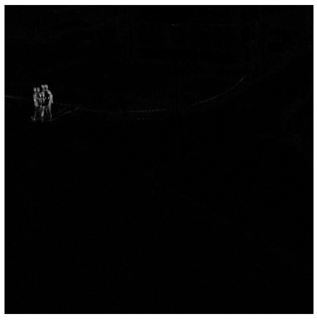
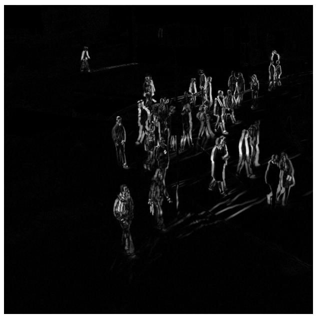
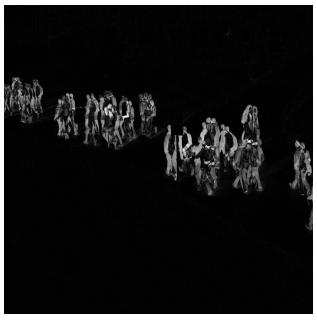
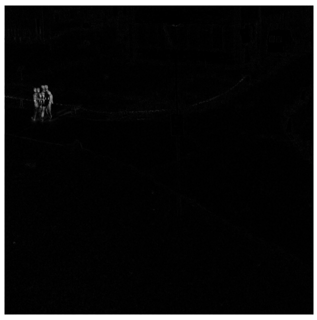
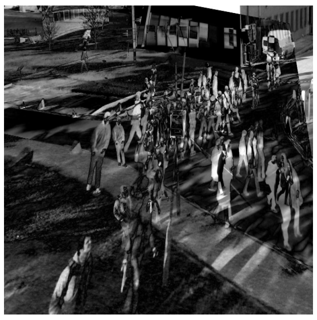
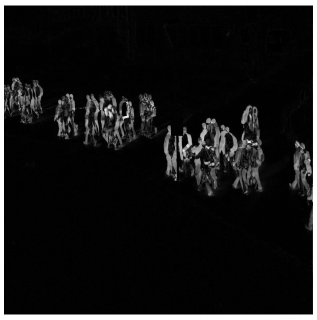
Dataset	Back Ground	City Center	Regular Flow
AMIE			
BRIEF			
CMIE			

Table 6.4: Single Factor ANOVA test to compare BRIEF, AMIE and CMIE for average matching difference of 34 videos given in Table 6.2.Here α is taken as 0.05

	Mean	*Variance*	*F*	P	F_{crit}
BRIEF	3.605	2.400	2.911	0.05	3.085
AMIE	4.083	4.513			
CMIE	4.823	6.675			

ure 6.6.2 in that the camera is stationary and the content moving. The same approach of determining the mean difference between warped and original frames using an estimated homography from matched points was employed. To encapsulate the results for a number of videos, the overall mean difference per video is reported along with its variance — the variance allows one to estimate easily the spread of frame differences. A low mean difference value depicts good matching performance of a descriptor, while a low variance shows that it produces similar results for a whole sequence of frames. Examination of Tables 6.2 and 6.3 shows that AMIE and BRIEF descriptors perform similarly, while CMIE give some unstable results with high variance for video sequences such as CC(14-55)01 and BG-005. To ascertain whether the performance differences between AMIE and BRIEF are statistically significant, ANOVA was applied to the mean differences obtained for the 34 videos in Table 6.2 and the resulting statistics are shown in Table 6.4. These show that the differences are not significant for $\alpha = 0.05$; in other words, we can be 95% confident in accepting the null hypothesis that there is no difference in the performance of AMIE and BRIEF.

6.7 Remarks

AMIE and CMIE are easily-calculated descriptors that encapsulate information around corners and integrate well with mainstream corner detection algorithms. AMIE performs significantly better than CMIE when the background is subject to change because of its use of the informative arc rather than a complete circle around the point, an approach that could be used with any other descriptor that describes a circular region around a corner. Furthermore, the use of the informative arc in AMIE means it can be matched more quickly than CMIE. The matching time of AMIE is comparable to that of BRIEF, a descriptor known for its fast computation.

Assessment of the stability of corner descriptors shows that the per-corner noise performances of AMIE and CMIE is a little poorer than those of BRIEF; however, in practical applications where the number of corners matched is reasonable, examination of datasets spanning applications such as template matching and surveillance shows that the overall performance of all three descriptors is similar.

The development of AMIE provides the key facility for developing a navigation facility for blind people: it is quick to calculate and match, should work in real time on even embedded processors, and robust to noise. The original intention was that it would form the basis for the navigation system described in the next chapter; however, the advent of the Microsoft Kinect sensor and associated software meant that a purely vision-based system could be improved upon. Hence, the following chapter describes the final navigation system, which is principally depth-based but uses corner detection and processing to make it more robust. Regrettably, this system need not employ the descriptors described and assessed in this chapter.

CHAPTER 7

KINECT AIDED VISUALLY GUIDED NAVIGATION SYSTEM

7.1 Introduction

A navigation system for a person with visual impairment involves identifying the layout of the 3D space around them and then helping them negotiate their way around obstacles *en route* to their destination. The traditional aid for this is a white cane, swept from side to side in front of the person; however, computer technology has the potential to provide less obtrusive and longer-range aids. Arguably, the most appealing way to produce such a system is to use a body-mounted video camera combined with computer vision. As a single camera is unable to detect distance, a pair of cameras is normally used as they allow computational stereo to determine the distance to obstacles.

An even more attractive solution would be to detect distance directly, a capability offered by the Microsoft Kinect and similar devices. Section 7.2 introduces this device. Although developed initially as a gaming input device for the Xbox 360, the Kinect has become popular among vision researchers because of both its low cost and the availability of software to acquire and work with data from it [12]. The Kinect features both a conventional colour camera and a depth sensor, the latter operating by projecting an infra-red structured light pattern and using a camera sensitive in the infra-red to capture where it falls on objects, then comparing the captured pattern with a reference one to determine disparity and hence depth. In principle, one should be able to determine the placements of obstacles using only the depth sensor; however, in practice, there are distance estimation, structural and noise problems [206] which make it difficult to obtain distance information from some objects because they are not reflective at infra-red wavelengths. For example, occlusion results in shadows, creating blind spots which cannot be used to obtain distances. Nevertheless, researchers have already employed the Kinect to produce navigation systems. In [63], the Kinect was integrated with an array of vibrotactile elements to build an assistant for blind navigation. Similarly, [66] used the Kinect to identify the distances to objects, though in that work the sensor is static and the system uses only the distance sensor's data to determine an obstacle's distance from user.

This research also uses a Kinect sensor but processes data acquired by both its depth sensor and its camera. Section 7.3 provides context concerning the computer vision processing and reviews briefly some local image features that can be used for vision-based navigation. The nature of this processing, and the construction of a complete navigation system that can be used by a visually impaired person while walking, form the focus of this chapter. In particular, an attempt is made to overcome the limitations of vision algorithms in detecting and matching

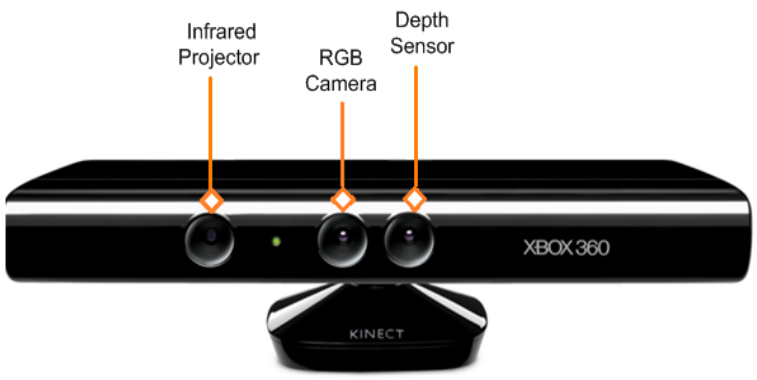


Figure 7.1: Microsoft Kinect for Xbox-360

features subject to geometric and photometric transformations [15]. Section 7.4 presents the complete, working navigation system, results from which are presented in section 7.5.

The accuracy of this kind of systems is more important than its speed; hence, the system is tested on two people, one blindfolded while and other visually impaired. Section 7.6 describes how a blindfolded person responded to the system's guidance and then how well the system guided a person with visual impairment. The feedback from these users is also analysed in this section. Based on this feedback, section 7.7 examines the current limitations of the system and how these can be addressed in the future. Finally, section 7.8 gives some concluding remarks.

7.2 Microsoft Kinect for Xbox 360

The Kinect was sensor developed by Microsoft to capture human motion for the Xbox-360 gaming console, shown in Figure 7.1. It has two sensors, a camera and an infra-red sensor designed to estimate a user's motion while playing games. Soon after its SDK release, researchers have not only used it to develop 3D games but also for a number of other interesting applications, including face tracking, picture frame, and digital mirror [12].

However, one has to be careful while using Kinect to develop navigation systems as the sensor is not flawless and has some limitations. For example, it uses it projects infra-red light and uses its reflection to calculate the depth image. These reflections may sometimes be sensed incorrectly or may be absent due to non-reflective or irregular surfaces. Similarly, the infra-red light can be swamped by strong light sources such as sunlight; and, most importantly, depth values from the sensors are not stable even if it is stationary [206]. These inaccurate depth data reduce the accuracy of the system. Furthermore, reflections from the complete field of view e.g. floor are also obtained, so obstacle detection becomes difficult without understanding the image content.

7.2.1 Calibrating the Kinect sensor

The Kinect sensors are located a short distance apart, so the first step in using the Kinect in a vision system is to calibrate them individually and to determine their separation; the latter is particularly important for this application as poor measurement of the separation is manifested as disparity between the colour and depth sensors.

To calibrate the Kinect, one can either use a dedicated Matlab toolbox [207], or identify corners in both the colour and depth images as described in [208] and [209]. The latter method was used here as it is similar to the algorithms used in the navigation system, allowing it to be integrated into its initialisation phase rather than as a once-only, offline calibration. To perform the calibration, one captures images of a calibration target using both Kinect sensors, then identifies the same corner points in the colour and distance images manually. From

Table 7.1: Calibration parameter values of the Kinect sensor used in this work. f_x, f_y and f_{xd}, f_{yd} are the focal lengths in x and y (to accommodate astigmatism of the lens systems) of the colour camera and depth sensor respectively. p_1 and p_2 are radial distortion parameters, while k_1, k_2 and k_3 are tangential distortion parameters.

$f_x = 5.2922 \times 10^2$	$t_x = 3.2895 \times 10^2$
$f_y = 5.2556 \times 10^2$	$t_y = 2.6748 \times 10^2$
$f_{xd} = 5.9421 \times 10^2$	$t_{xd} = 3.3931 \times 10^2$
$f_{yd} = 5.9104 \times 10^2$	$t_{yd} = 2.4274 \times 10^2$
$k_1 = 2.6452 \times 10^{-1}$	$p_1 = -1.9922 \times 10^{-3}$
$k_2 = -8.3991 \times 10^{-1}$	$p_2 = 1.4372 \times 10^{-3}$
$k_3 = 9.1192 \times 10^{-1}$	

these values, camera distortion coefficients and transformation matrices can be calculated using the process described in [208] and [209]. The various parameters involved are summarised in the following formulae, while their calibrated parameters are shown in Table 7.1.

$$\text{Sensor}_{RGB} = \begin{bmatrix} f_x & 0 & t_x \\ 0 & f_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{7.1}$$

$$\text{Sensor}_{Depth} = \begin{bmatrix} f_{xd} & 0 & t_{xd} \\ 0 & f_{yd} & t_{yd} \\ 0 & 0 & 1 \end{bmatrix} \tag{7.2}$$

$$\text{Distortion Coefficients} = \begin{bmatrix} k1 & k2 & p1 & p2 & k3 \end{bmatrix} \tag{7.3}$$

The rotation matrix **R** and translation matrix **T** that result from these are:

$$\mathbf{R} = \begin{pmatrix} 9.99 \times 10^{-1} & 1.26 \times 10^{-3} & -1.75 \times 10^{-2} \\ -1.48 \times 10^{-3} & 9.99 \times 10^{-3} & -1.23 \times 10^{-2} \\ 1.75 \times 10^{-2} & 1.23 \times 10^{-2} & 9.99 \times 10^{-1} \end{pmatrix} \tag{7.4}$$

$$\mathbf{T} = \begin{pmatrix} 1.9985 \times 10^{-2} \\ -7.4424 \times 10^{-4} \\ -1.0917 \times 10^{-2} \end{pmatrix} \tag{7.5}$$

The Kinect distance sensor returns a depth 'image' in which each pixel corresponds to the distance of the object in a colour image pixel, with some translation and rotation due to the physical separation of the two sensors on the device. Therefore, to find the depth of colour image pixels, one projects each depth image point in real world by calculating their 3D coordinates using

$$\mathbf{P}_{3D} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = \begin{pmatrix} xt_{xd}\frac{d_m}{f_{xd}} \\ yt_{yd}\frac{d_m}{f_{yd}} \\ d_m \end{pmatrix} \tag{7.6}$$

where d_m is the distance in metres calculated using

$$d_m = 0.1236 * \tan\left(\frac{D}{2842.5}\right) + 1.1863 \tag{7.7}$$

where D is the raw depth value obtained from the Kinect. This conversion is important because the rotation and translation matrices **R** and **T** are calculated in metres in the real world. From these 3D coordinates, 2D projections onto the colour image can be computed using

$$P_{2d} = \begin{pmatrix} p_{x_{2d}} \\ p_{y_{2d}} \\ p_z \end{pmatrix} = \mathbf{R}\mathbf{P}_{3D} + \mathbf{T} \tag{7.8}$$

$$P_{x_{2d}} = \frac{P_{x_{2d}}}{P_z} \tag{7.9}$$

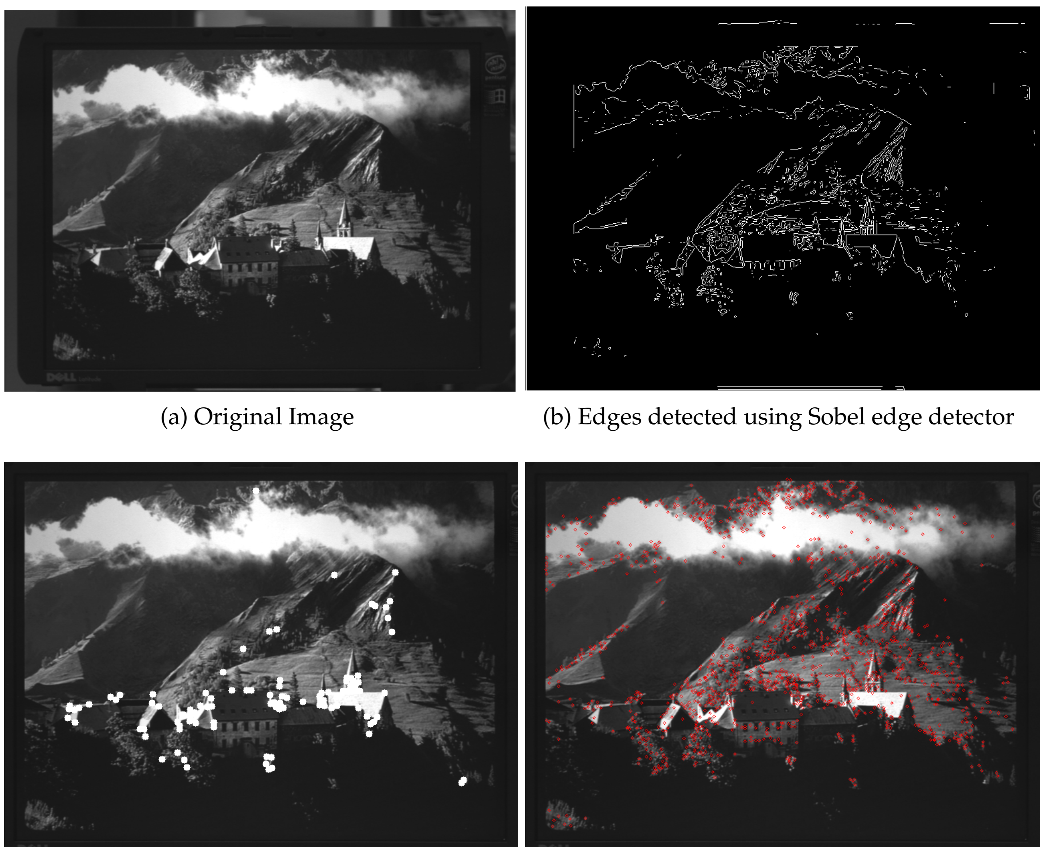
$$P_{y_{2d}} = \frac{P_{y_{2d}}}{P_z} \tag{7.10}$$

so that the colour image pixel corresponding to a depth image pixel can be determined. To speed up the process, a 2D look-up table can be generated, indexed on 2D projections of depth image pixels.

7.3 Low level image features for navigation systems

Image understanding and matching have been long-standing, important research areas in devising safe navigation systems for both robots and humans [210]. Vision-based navigation systems commonly use feature-based matching in video frames. These image features can be blobs, edges, corners or regions. A blob is an image area with a significant intensity difference from its neighbourhood, *e.g.* a dark spot in a bright region or a bright spot in a dark region [94]. An edge is the boundary between image regions, usually identified in terms of colour or intensity difference, and is important when segmenting image regions [211]. Similarly, image regions are also used as features to be matched for segmentation tasks [85]. Finally, corner points are the image areas that correspond to sharp changes in the direction of boundaries [76].

Although any of these local image features can be used for this application,



(a) Original Image

(b) Edges detected using Sobel edge detector

(c) Corners detected using H&S detector

(d) Blobs detected using SIFT detector

Figure 7.2: Results of local image feature detectors that can be used to develop a vision based navigation system

the sensitivity of infra-red sensor to reflective materials means that the use of blobs for depth calculation can degrade the system's performance, particularly outdoors. Furthermore, extracting regions takes more time than corners but gives less information about image content, for the reasons discussed in earlier chapters. Figure 7.2 shows that if the detected features are edges or blobs, the result is every textured area in the image. This might be useful for applications where identification of the whole image content is required, such as segmentation, panorama stitching and homography estimation; however, in this navigation application, meaningful image areas such as obstacles are important, and the result in Figure 7.2c shows that the corner points identify important image content.

Corners are particularly attractive because they are fast to compute and lie on the boundaries of obstacles. Consequently, corner points have been chosen to find image locations that can correspond to obstacles.

After detecting interest points in images, matching them in subsequent video frames is required for a navigation or tracking application. The best-known methods in the literature for tracking features are homography-based matching [71], visual odometry, optical flow and particle filtering. Visual SLAM [72] and related systems have also been developed using matching of local image features [115] or corners [118]. Despite all these efforts, there are limitations of vision algorithms to detect and match features under geometric and photometric transformations [15], resulting in low repeatability scores and unstable responses at different scales, which makes it difficult to obtain consistent, continuous information for safe navigation. This research explores whether one possible solution is to combine different sensors, such as the Kinect camera and its infra-red sensor.

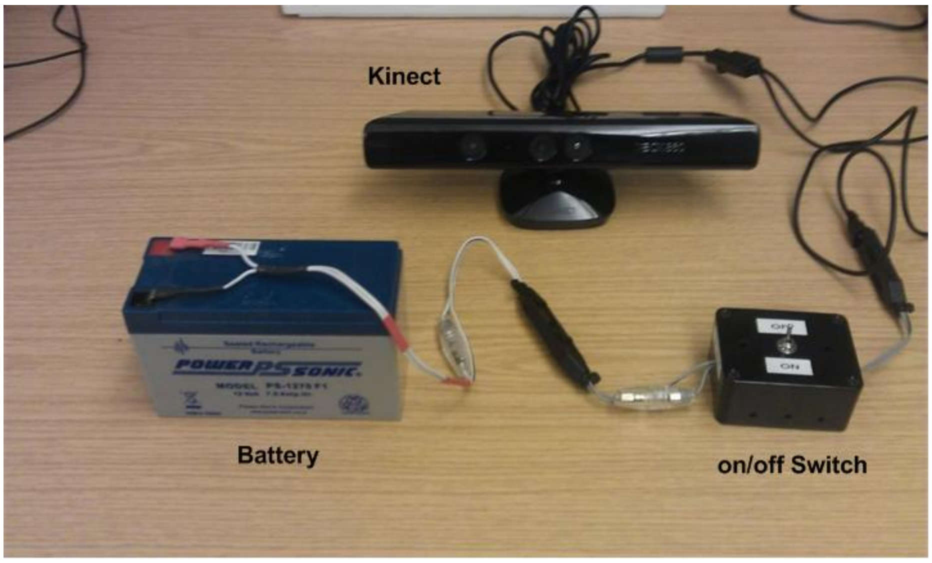


Figure 7.3: Components of the navigation system, excluding the laptop or equivalent on which all processing is performed

7.4 A Kinect-based navigation system

The navigation system described here requires a standard Kinect sensor, a battery and a laptop, all of which are carried in a backpack or shoulder bag by the user. The Kinect is powered by a sealed lead acid 12 V, 7 A battery shown in Figure 7.3, the output of which is fed through a DC to DC converter to ensure a stable 12 V supply. The capacity of the battery is enough to power both Kinect and laptop for 3 hours. The Kinect sensor is carried in front of the user using a strap around their neck; although manageable, this is bulky and the authors would expect a production system to re-package the sensors into a unit that is physically smaller and easier to manage, perhaps shoulder-mounted.

7.4.1 Software issues

The proposed system uses the H&S corner detector to find corners in RGB images acquired by the Kinect's camera, with their depth being obtained from the depth

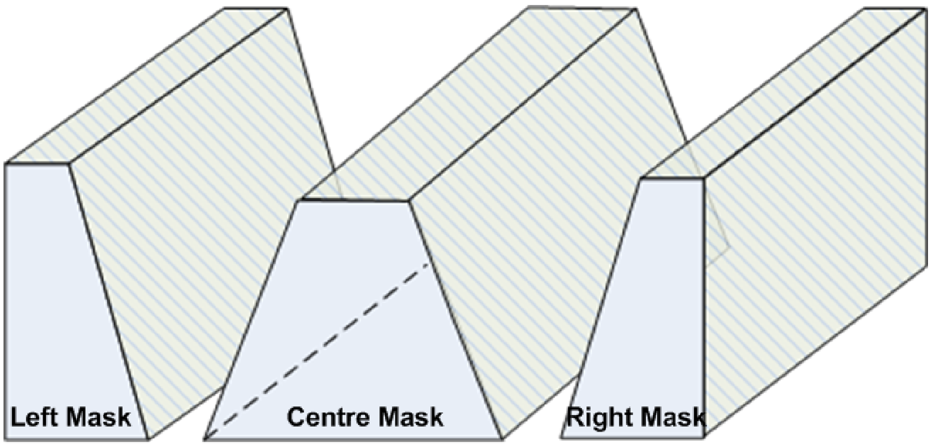


Figure 7.4: Image masks showing which parts of the colour and depth images are processed (in white) for a user travelling straight ahead (center mask). These white regions are termed 'safe navigation regions.' The equivalent masks used to assess whether to turn to the left or right are shown to the sides of the center mask.

lookup table constructed from 2D projections of depth image pixels using the measurements described in section 7.2.1.

It is important to bear in mind that not all values from the Kinect distance sensor are correct due to the problems alluded to in section 7.1. Hence, the system searches for a sensible depth value in a 10×10-pixel neighbourhood around colour image corners.

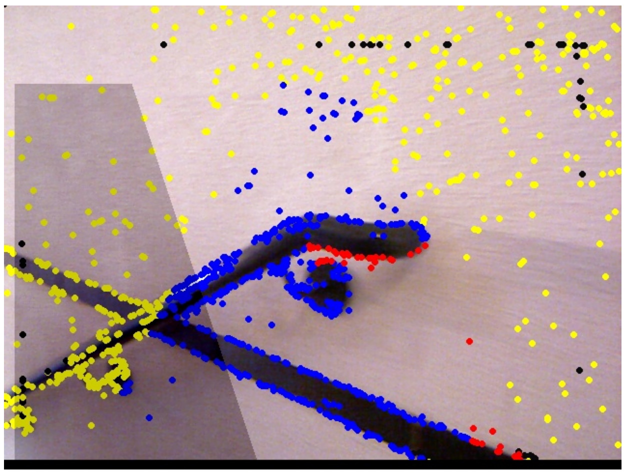
To enable a partially sighted person to navigate freely, the entire image is divided into regions, shown in Figure 7.4. The white pixels in each region need to be obstacle-free for the person to move in that direction, and there are separate mask images for (from left to right in the figure) turning to the left, walking straight ahead and turning to the right. These white regions are termed 'safe navigation regions'. The width of the safe navigation region in the middle mask image is set to match the area in which a person might walk between video frames at a reasonable walking pace, so is wider near the person and narrower at greater distances. The minimum depth of each window is kept the same, half the length of an average human arm (0.3 m (later updated to 1.85m)) so that the person can reach out to find any obstacle that has been detected.

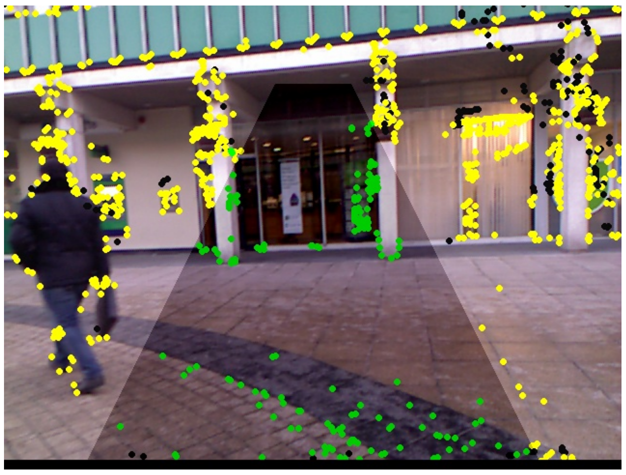
7.4.2 Processing

The system checks for two types of obstacles, those that can completely block progress and smaller ones such as chairs that can be avoided by changing direction. For the former, the image feature detector may fail if the surface is featureless so the system checks all depth values inside the safe navigation region (Figure 7.4) and if the mean depth becomes less than a threshold the user is notified to stop using voice synthesis. Otherwise, the feature detector finds corner points and, depending on the depth of corners, warnings may be issued to change directions to the right or left. The system employs the processed depth (I_d) and colour (I_c) images from the Kinect in a three-stage processing algorithm described in the following paragraphs.

1. **Pre-processing.** Calibration parameters are applied to both input images (I_d and I_c) so that distortions can be removed from them. Then, corner points are found in the colour image while the depth image's data are converted into meters, from which the 2D projection of all points is found and the 2D look-up table described in section 7.2 calculated. For fast processing, this stage works in separate threads because calculations on I_c and I_d are independent of each other.

2. **Navigation processing.** The central safe navigation region is scanned for a wall-like obstacles by averaging the depth values. If the mean depth is less than a threshold (0.3 m in this work), the system activates the alarm-generation module and warns the user to stop. Otherwise, the system looks for each corner point's depth within the central safe navigation region. If


(a) Left region


(b) Middle region

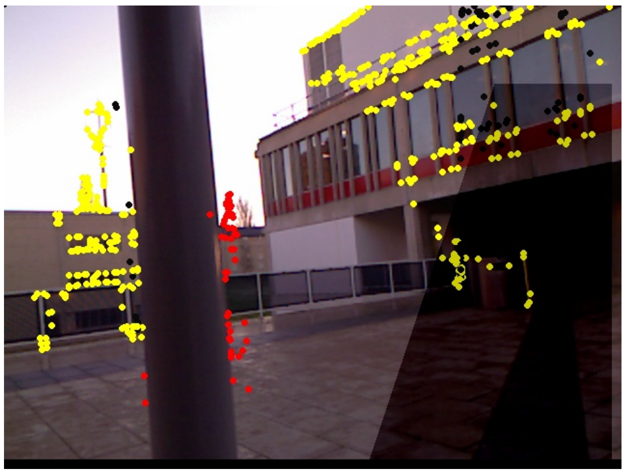

(c) Right region

Figure 7.5: Navigation from analysing data in the image masks of Figure 7.4. In (a), the user is asked to move to the left; in (b), the user can move forward; and in (c) the user is given an alarm and encouraged to move to the right.

one or more corner point appears to be close to the person (its distance is less than the threshold), the system starts counting frames for which these points remain a hazard. The walking speed of a blind person is typically 2 to 3 miles per hour [212] so if hazardous corner pixels remain in the active region for five frames, it checks the right and left safe navigation regions for a clear route, selecting the one with no wall-like obstacle and where the nearest corners are further from the user. Once the left or right window is selected the system initiates the alarm-generation module to inform the user. If no obstacle is found, the system remains quiet.

3. **Alarm.** If an alarm is triggered during processing, voice synthesis is used to alert the user, generating *e.g.* *"stop please"* if there is an obstacle in the way. For a left or right turn, the system generates *"slightly left is better"* or *"slightly right is better"*.

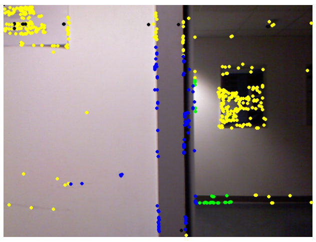
Figure 7.5 illustrates three different scenarios. Corner points are presented in five different colours:

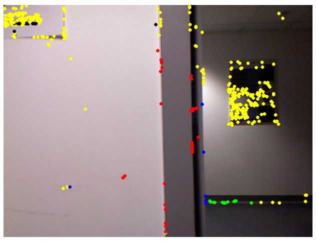
- **yellow** points are non-hazard corner outside the safe navigation regions;
- **green** points indicate potential hazards ($d_m > 1$ m and inside the safe navigation regions);
- **blue** points are potential hazards ($1.0\,\text{m} > d_m > 0.3\,\text{m}$ and inside the safe navigation regions);
- **red** points are hazards ($d_m < 0.3$ m inside the safe navigation regions);
- **black** pixels indicate a blind spot in depth image.

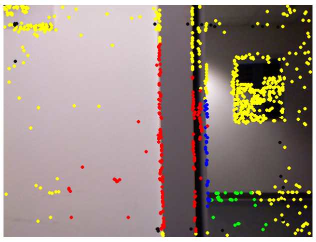
In Figure 7.5b system does not find any wall or obstacle so all the corner points are green and there will be no warning to the user. In Figure 7.5a there are some points which lie inside central safe navigation region and cross the safe distance threshold (red-coloured corners) so the system checks right and left safe navigation regions and finds the left one to be the safer, and so prompts user to move left. Conversely, Figure 7.5c shows where, after finding an obstacle in the central safe navigation region, the system identifies the right as being safer and prompts the user to move that way.

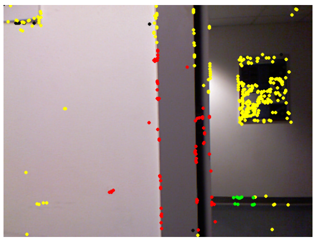
7.5 Results from real-world videos

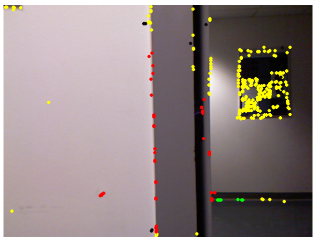
This section presents results from illustrative real-world navigation sequences in which the equipment was body-mounted on a blindfolded user, in both indoor and outdoor sequences conditions. The current system can process Kinect data at 5 frames per second (on an Intel Core 2 quad 2.83 GHz processor), which is fast enough to warn the trial subject of any collisions. In each of Figures 7.6 to 7.9, the first row of images was taken near the start of the video while the rest of the rows show gradually decreasing distances from obstacles. The colour of the corner points indicate whether they are potential hazards or not, using the annotation scheme described in the previous section. Each row contains a sequence of three frames of video showing a gradual decrease in distance from obstacle. The audio output of the system is displayed at the bottom of each video frame, when there is no warning the system remains silent, indicated by “—–" as system’s output.

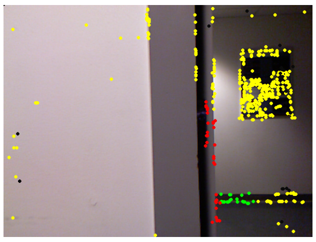

(a) —-

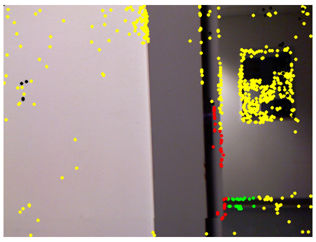

(b) count start

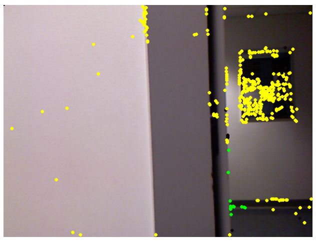

(c) counting

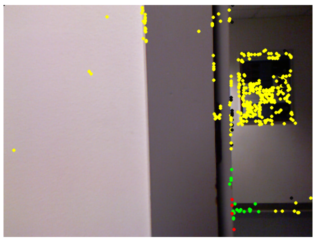

(d) counting

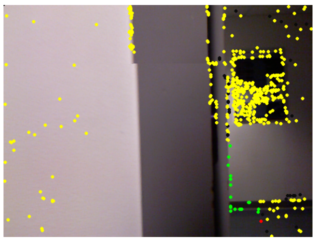

(e) counting

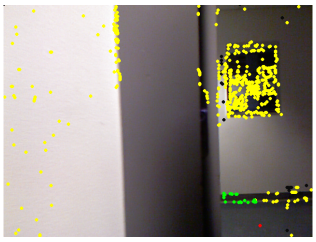

(f) *Slight right is better*

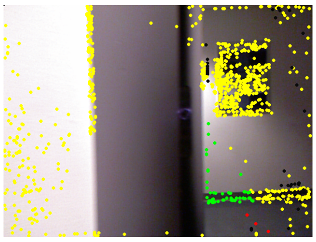

(g) count start

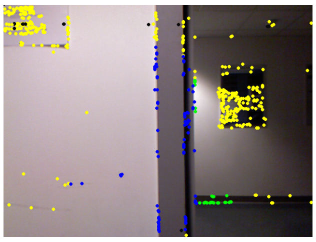

(h) —-

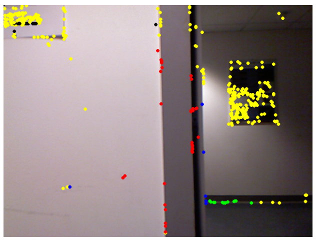

(i) —-


(j) —-


(k) —-


(l) —-


(m) —-


(n) count start

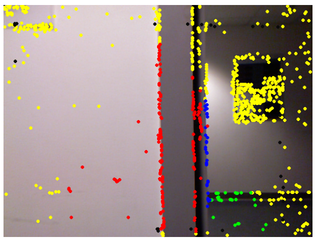
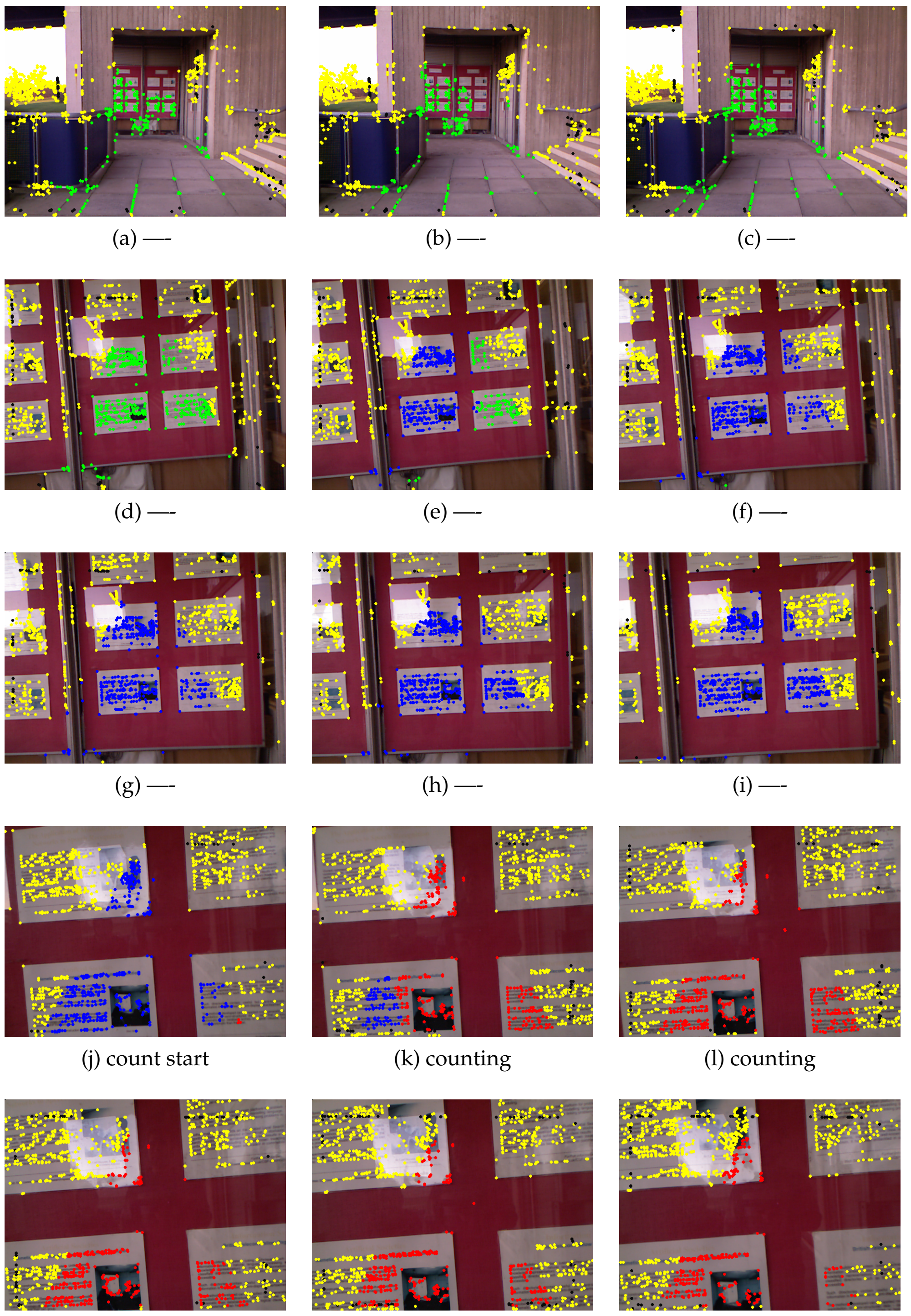

(o) count start

Figure 7.6: The navigation system's visual output on an indoor video sequence of walking through a corridor coming in front of a wall


(a) —- (b) —- (c) —-

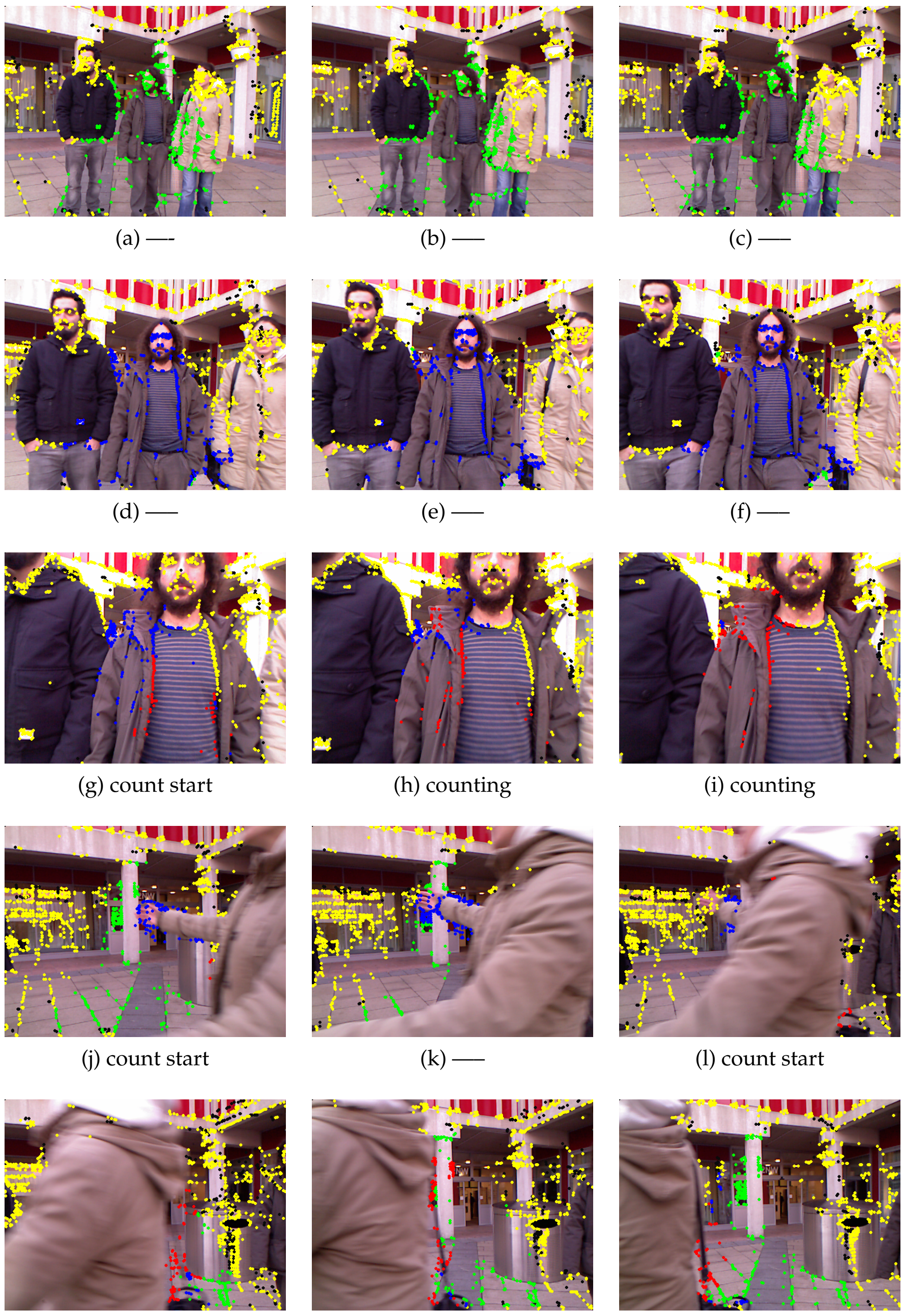
(d) —- (e) —- (f) —-

(g) —- (h) —- (i) —-

(j) count start (k) counting (l) counting

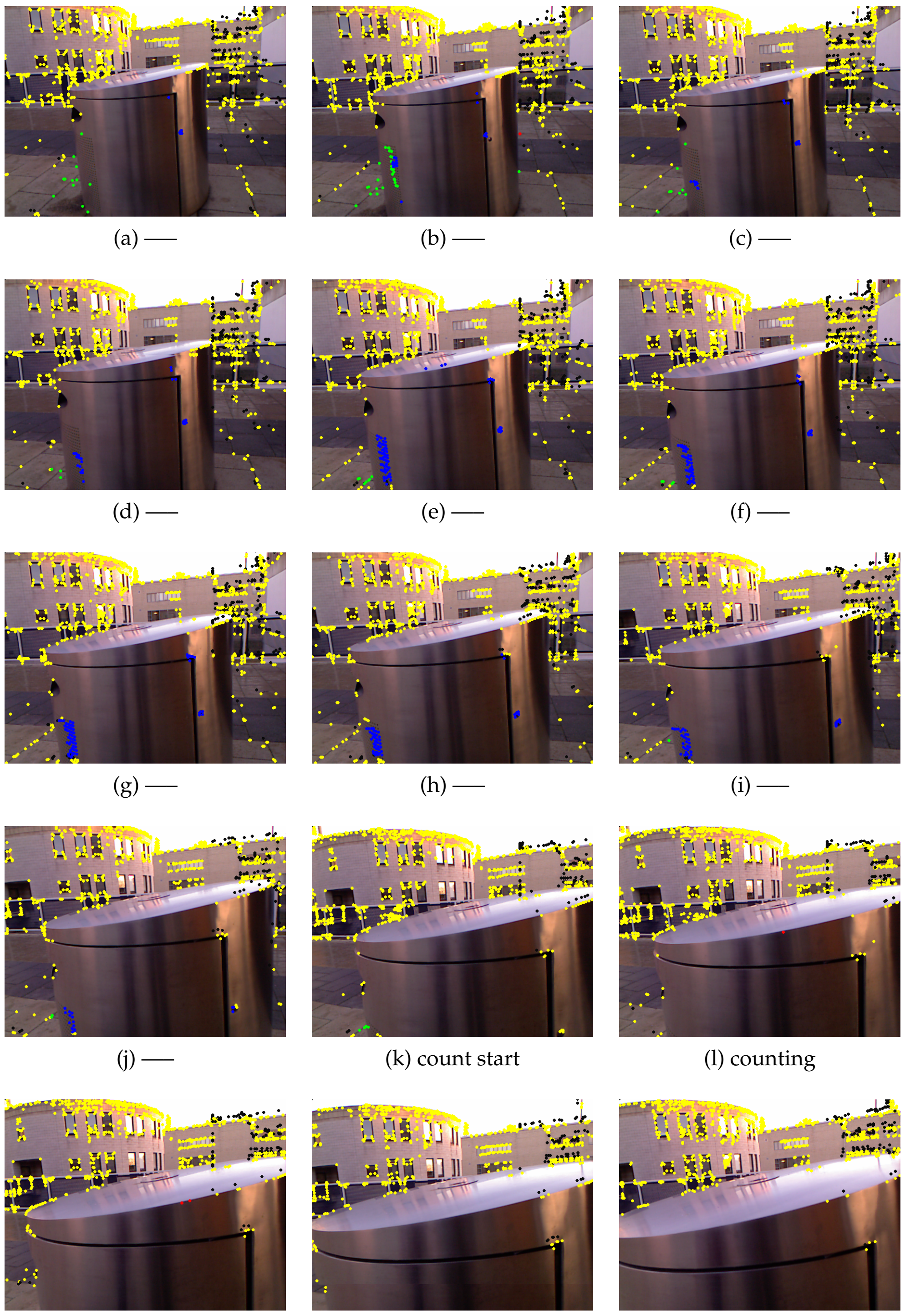
(m) counting (n) *Stop please* (o) count start

Figure 7.7: The navigation system's visual output on an outdoor video sequence, walking towards a wall with noticeboard



(a) —-
(b) —–
(c) —–
(d) —–
(e) —–
(f) —–
(g) count start
(h) counting
(i) counting
(j) count start
(k) —–
(l) count start
(m) counting
(n) counting
(o) counting

Figure 7.8: The navigation system's visual output on an outdoor video sequence with people walking around the subject



(a) —— (b) —— (c) ——

(d) —— (e) —— (f) ——

(g) —— (h) —— (i) ——

(j) —— (k) count start (l) counting

(m) counting (n) counting (o) *Stop please*

Figure 7.9: The navigation system's visual output on an outdoor video sequence in which the subject encountered an obstacle

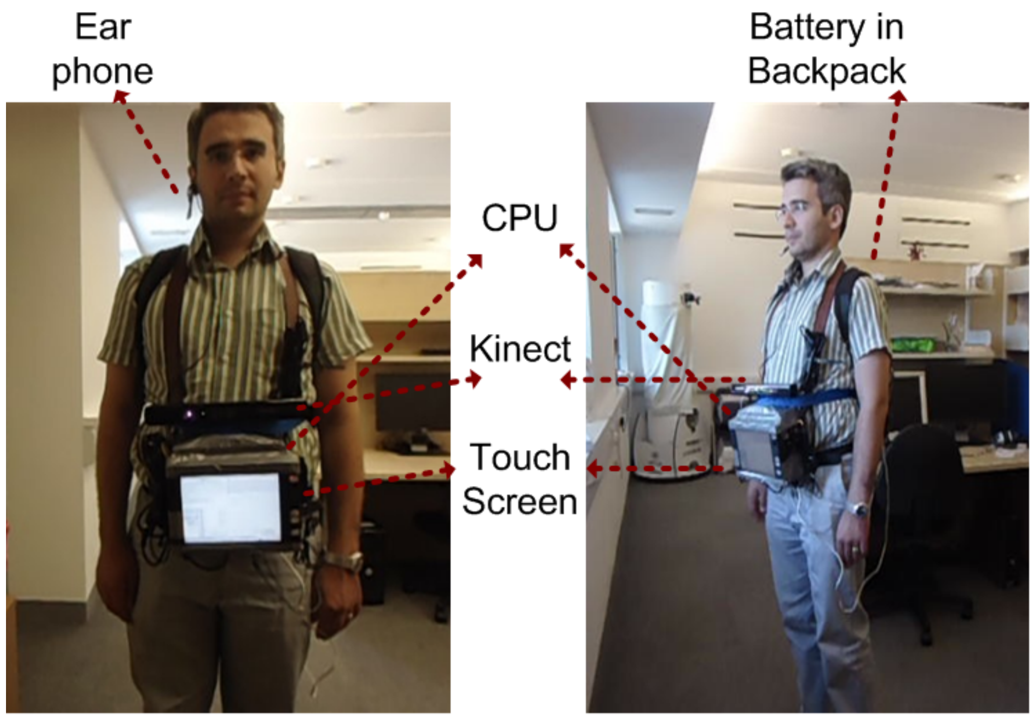


Figure 7.10: Hardware setup for testing the system

7.6 System testing

The proposed system aims to provide navigation assistance to visually-impaired people, so testing by its probable users was crucial. To circumvent any potential problems, a two-stage experiment was designed. In the first stage, a blindfolded person was asked to use the system to navigate in an unknown environment containing obstacles, while in the second stage, a visually impaired person was requested to try the system and assess its accuracy in real time.

Figure 7.10 shows the hardware setup used to perform these experiments. The durations of the experiments were kept short because the equipment was bulky and heavy, though a belt through waist was also used to provide some extra support and to distribute the equipment's weight evenly. The Kinect and a touchscreen monitor were mounted on a portable Central Processing Unit (CPU)

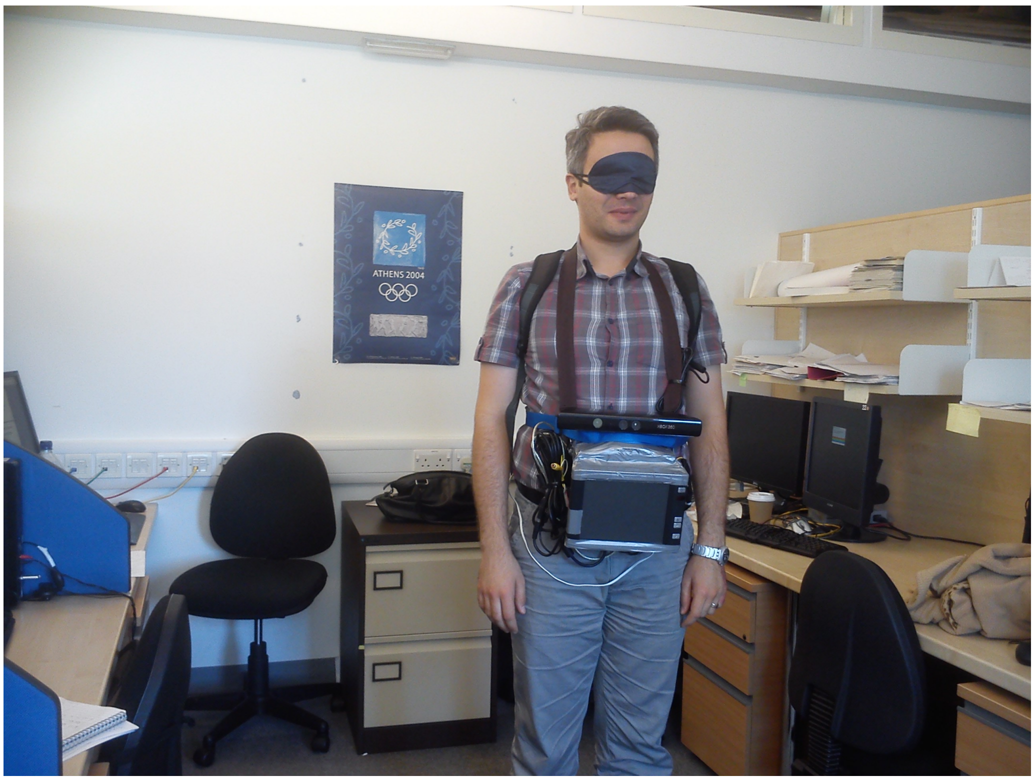


Figure 7.11: System testing by a blindfolded person

and all of the three devices (CPU, Kinect and monitor) were powered by the battery shown in Figure 7.3, which was placed in a backpack. The results of the experiments, along with the users' feedback, is given below.

7.6.1 A blindfolded user

Image 7.11 shows how a person with proper visual sense tries to sense the environment using the Kinect-based system instead of his own eyes. The system was required to be calibrated according to the user's height, setting the distance threshold and the number of frames per response to match their walking speed. (Initially, the system had been tuned to respond after 5 frames but, based on preliminary user feedback, it was adapted to make this variable down to 2 frames per response.)

The user was able to move in different indoor and outdoor locations with good confidence of the system's guidance. However, as expected, the user found the system response time to be slow. This appears to be because a person with good visual sense does not naturally walk as slowly as a blind person walks, and has difficulty walking at that pace. This became quite obvious when the same user was asked to use a white cane as mobility assistant (shown in Figure 7.12). Their stress level was high during this experiment, with the user keeping his other arm close to his body to save himself from collisions. Similarly, the walking speed was significantly slower — showing that using an unfamiliar mobility aid affects the user's confidence.

The following paragraphs are feedback from the blindfolded user who trialled the navigation aid:

> *"Having tested with both the Kinect aided navigation system and the white cane, as a novice user for both tools, I can say that the former is easier to use since it requires less effort to navigate while I was blind-folded. It helped me to navigate in a relatively crowded environment which I was not familiar with and kept me from tripping on most obstacles in front of me. I hit some of the obstacles while using the white cane.*
>
> *One shortcoming of the navigation system may be that it cannot currently provide a "feel" of the surface such as how rough or smooth the it is when compared to a white cane since the system is sensing the environment from a distance. I think this would be a good area to investigate as future research.*
>
> *Overall, I believe that the system is a very intuitive and natural way of interacting with a visually-impaired person due to its environment sensing and speech features."*

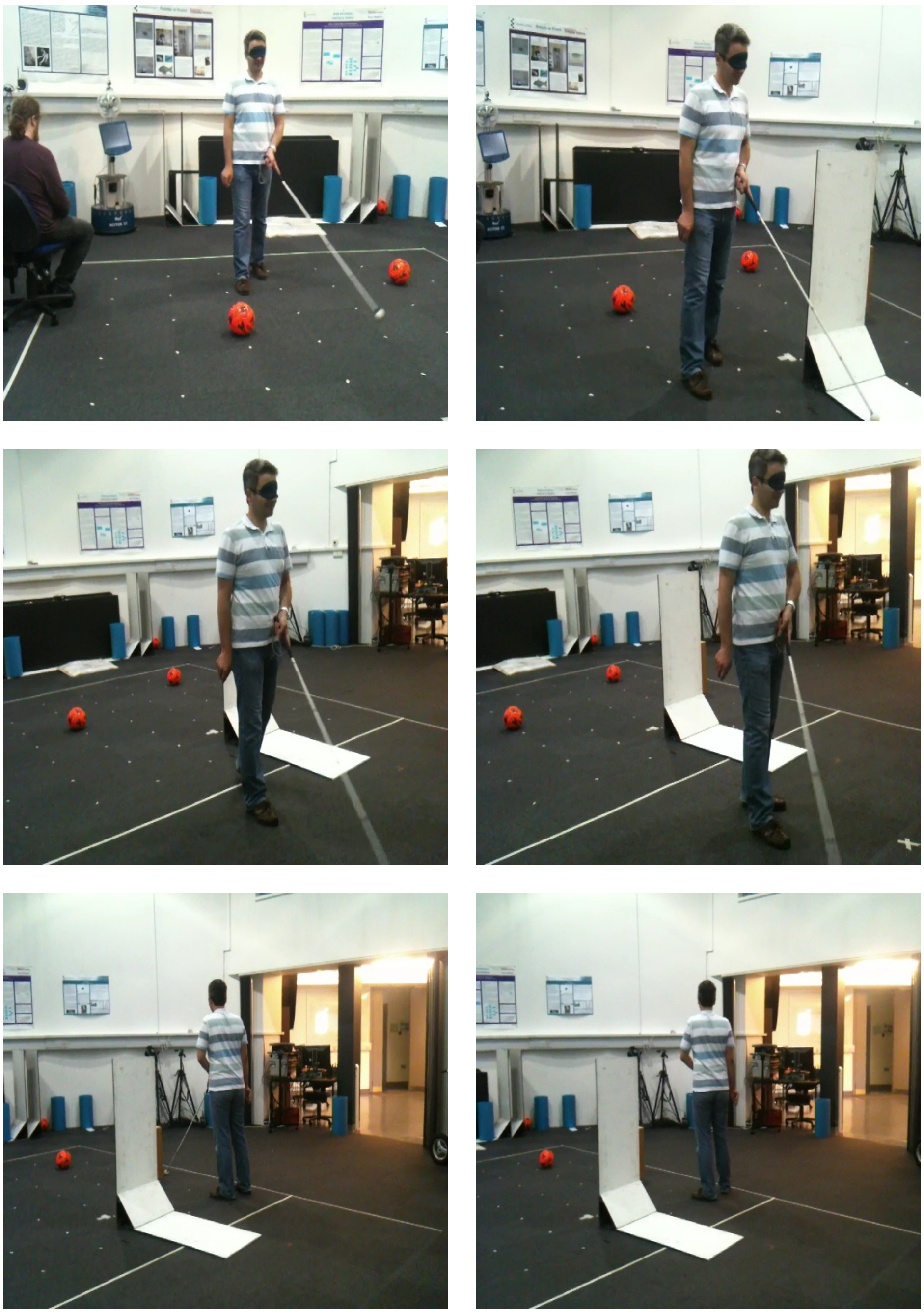


Figure 7.12: A blindfolded person using white cane as mobility aid

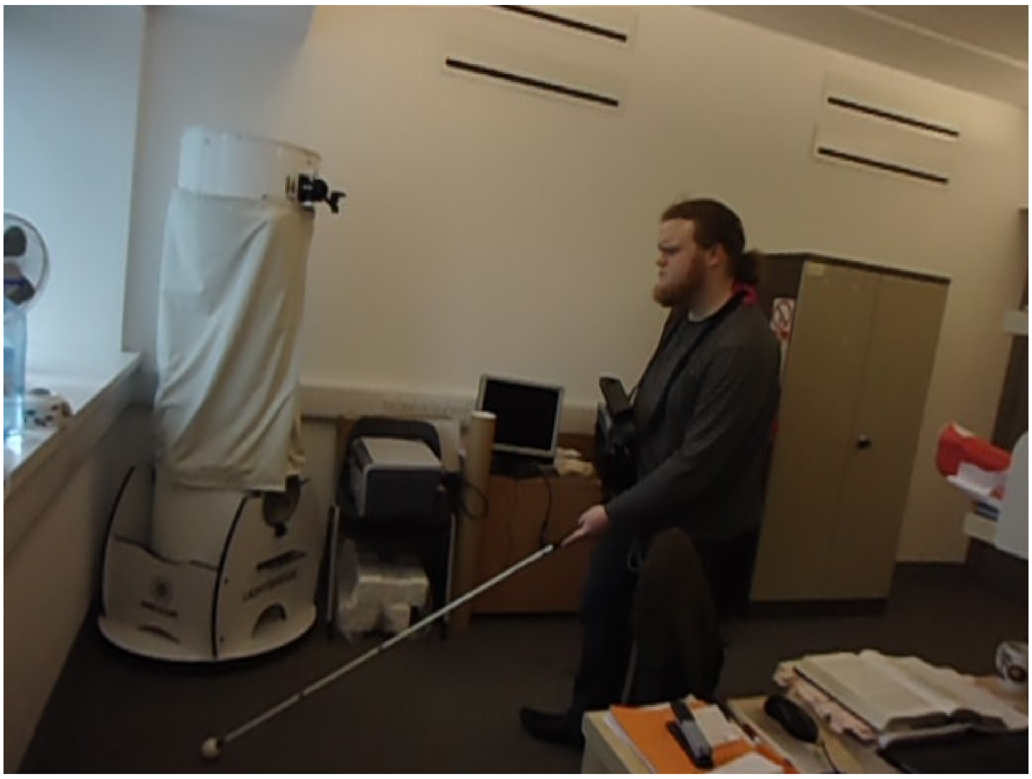


Figure 7.13: Calibrating Kinect-aided navigation system by letting a visually impaired person to walk using it along with white cane

7.6.2 A visually impaired user

Dr. Keith Currie, a born blind man, participated in the experiment to test the Kinect-aided navigation system. Keith was enthusiastic and optimistic about the whole experiment because of his interest in such kind of mobility aids. The author admires his courage and bravery for testing a development system. Figure 7.13 shows him using the system along with a white cane, allowing him to familiarize himself with the system and to calibrate it according to his needs.

In order to increase his comfort level with the system, the system was tuned to communicate at every alternate frame, by saying *"Move Ahead"*, even if no obstacle was detected. In this way, the user was able to know that the system was working and he can take further steps without bumping into an obstacle. Fur-

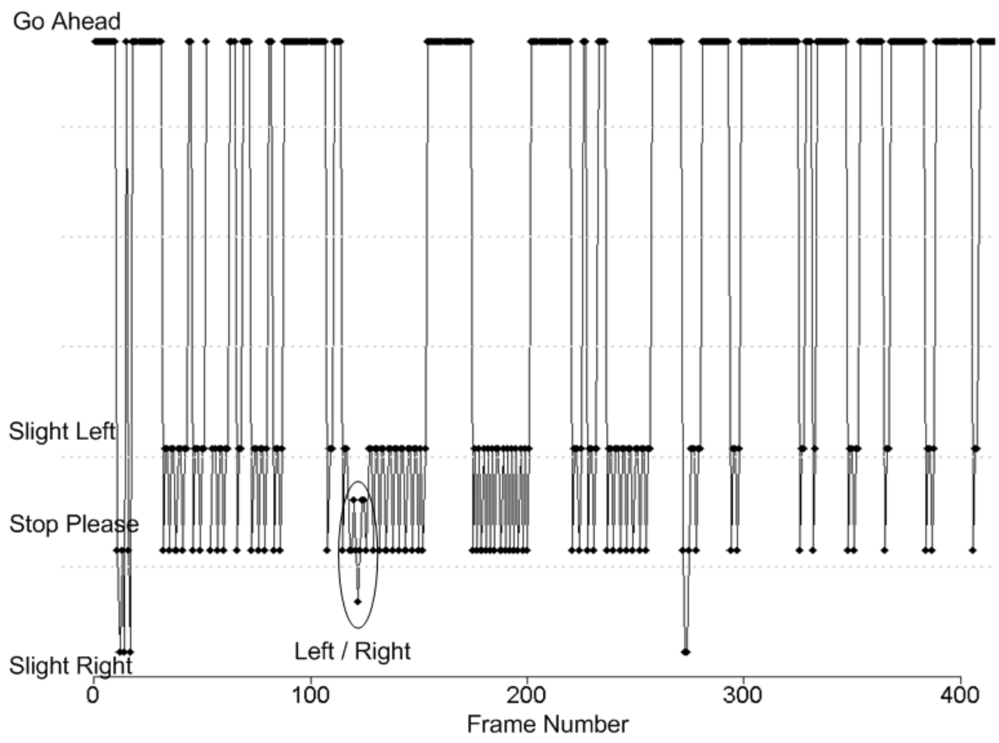
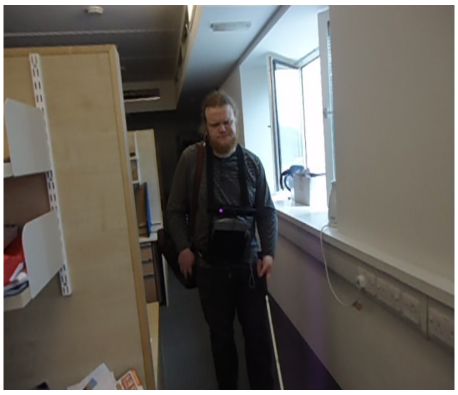


Figure 7.14: Kinect-aided navigation system's response when the user was walking through a corridor. Each response is after two frames therefore, actual number of frames processed is 800.

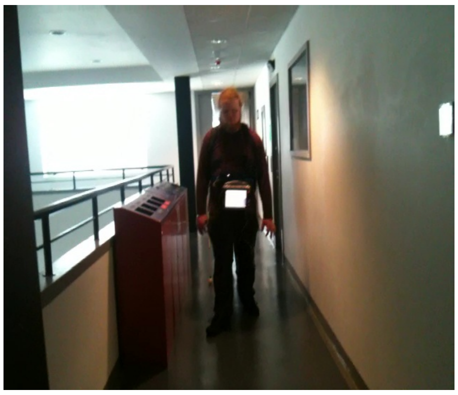
thermore, letting him use white cane along with the automated system helped identify the best distance threshold for a blind person. Initially, the distance threshold was set to 0.3 m, less than his white cane which was approximately 1.42 m long. As a consequence, Keith was able to sense obstacles before the system and therefore, did not find it helpful. However, after changing the distance threshold to 1.8 m, the system was able to notify the user before the white cane, improving its usefulness.

(a) Lab

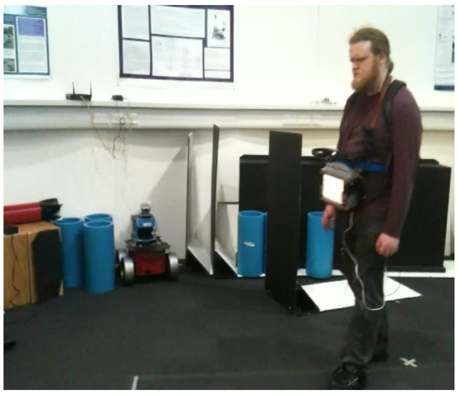

(b) Lab

(c) Open square

(d) Stairs


(e) Corridor

(f) Corridor


(g) Robotics Lab

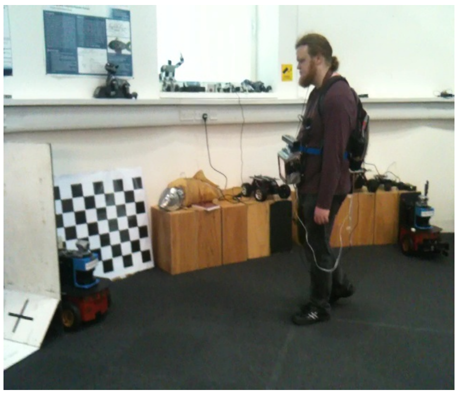

(h) Robotics Lab

Figure 7.15: *Keith Curie* walking at different locations using Kinect-aided navigation system

The overall system's value can only be judged from its users' feedback, which is given below, but to show the system's verbal responses Figure 7.14 presents its output for a sequence in which Keith walked from a corridor to an open space (shown in Figure 7.15d). The system's response at indoor locations was found to be more appropriate and better-timed than in outdoor locations. This was not unexpected because strong sunlight and shadows create blind spots in the depth images captured by Kinect and these affect the system's accuracy.

7.6.3 Feedback from the blind user, *Dr. Keith Currie*

"I first became aware of the Kinect-Aided Navigation system when I responded to a request for assistance from users of white canes and guide dogs. I am a lifelong user of navigation aids, and agreed to participate in a number of tests of the system in order to help determine its use-value for visually impaired people. Despite possessing a deep interest in adaptive technologies, I rarely make use of mobility aids that are more sophisticated than a standard issue white cane on account of the prohibitive cost associated with such products. I was extremely excited by the prospect of investigating a device that made use of relatively inexpensive technologies. The tests were performed on two separate occasions: the first was conducted in a large office in the computer sciences department, and the second was conducted partially in square 1, concluding indoors in one of the work labs.

On the first occasion we tested the use of the system in conjunction with the white cane, and on the second we tested it without making use of any other mobility aids. My initial impression of the system after the first set of tests was mixed, and I found that I tended to rely on my cane in preference to the system throughout. I found it difficult to achieve the extremely slow pace the system required in order to be effective. I was impressed by the range and *sensitivity of the sensors, though the reach of the cane exceeded the reach of the sensors and was thus able to detect obstacles before the sensors could. I was additionally concerned that the device presently lacked portability; it currently comprises a backpack and a small but heavy terminal that is suspended in front of the user's chest via a neck strap, and while the weight of the device is hardly noticeable when it is first worn it becomes progressively less comfortable. The system was also prone to crashing, and seemed to become easily confused when it encountered multiple obstacles; this meant that it was only effective for very short periods of time.*

After the first test the system had been slightly recalibrated to accommodate the length of my white cane, and the initial experiments we conducted on the second day of testing seemed to indicate that the two mobility aids could potentially be used in conjunction to significant effect. These experiments were conducted outside, but unfortunately had to be abandoned as the sensors proved unable to cope with direct sunlight. We subsequently tested the system indoors, and without the white cane. I found that there was a marked difference in my response to the system during these final experiments. I seldom walk independently when outside of familiar environments, but whenever I do so I am able to move about with a fair degree of confidence, relying on my hearing and sense of balance when encountering and negotiating obstacles; in this regard I found the system quite useful, as it was able to provide me with fair warning of approaching obstacles before I encountered them directly. Despite being somewhat heavy to carry, the system does not require the active use of the wearer's hands, and this further improved my confidence when using it. In addition, I found that I was more attentive to the spoken instructions that the system provided than I had been on the previous occasion, and was thus better able to gain a sense of the system's *perceptivity. I found that it was able to detect large obstacles such as walls and pillars, but lacked the depth perception required to detect smaller objects, leaving the user open to collisions with chairs, tables, and anything else at waist height or lower.*

After conducting these tests and reflecting on their outcome my overall impression of the Kinect-Aided Navigation System is a positive one, but I believe it will require significant further development before it can be put to meaningful use by the visually impaired community. The verbal instructions that it provides are easy to comprehend, but are limited to directional advice and do not presently identify approaching obstacles. Perhaps other audible indicators like bleeps or clicks might prove useful in this regard. Another significant problem with the system is that it requires the user to walk at an extremely slow pace in order for it to prove effective, making it an inappropriate navigation aid for guide dog users, who generally walk at a rapid pace. The tests prove that the device can be calibrated to work in conjunction with a white cane, and indeed an additional aid like a white cane is necessary at this time as the system is unable to identify those lower obstacles that the cane is specifically designed to detect. With a little further development it is conceivable that the system could be put to good use on its own, but it is my conclusion that the designers will want to consider the use of the system in conjunction with a white cane as the default for most future users. In conclusion I am confident that, with some further refinement to its design and sensitivity, the Kinect-Aided Navigation System will make a valuable contribution to the field of assistive technology, and to the confidence and mobility of any future users."

7.7 Current limitations and future directions

The system is capable of identifying obstacles at reasonable distances and speeds; however, based on the users' feedback, it needs improvement in the following aspects:

- equipment weight
- false reflections detected by the infra-red sensor in strong sunlight
- distance limitation of the infra-red sensor
- surface texture identification

As suggested by Keith, if an automated navigation system can be combined with a white cane, one can have a safe and reliable mobility aid. This is mainly because surface textures and low-level obstacles can be identified easily using a white cane, whereas an automated navigation system can help locating head-level obstacles and identify obstacles blocking one's path. (Of course, one must also bear in mind that Keith is familiar with using his cane but not the computer-based system; one would therefore expect him to find the former to be easier to use.)

The audio output of the system was not appreciated by the blind user as rightly it halts the environmental sounds which can be helpful in navigation, however, during the developmental stage this type of output is considered more helpful. In future this can be replaced by any other suitable form of output such as beep sounds or vibrations etc.

All the problems experienced with the current system appear to be due to the infra-red sensor. In the future, replacing it with some other depth sensor, such as a laser striper, might yield more accurate responses. However, the concept of combining vision with sensed imagery proves to work well. Android mobile phones are equipped with GPS, inertial sensor and an RGB camera, a combination that may well be worth exploring for this kind of systems.

7.8 Remarks

A navigation system for visually-impaired people has been designed, implemented and assessed in both indoor and outdoor environments. Input from the Kinect's camera and distance sensor compensates for limitations of each individual sensor.

The system was tested on both a blindfolded user and a visually-impaired person. Both users found system to be promising and highlighted its potential in becoming a good navigational aid in future. Although some problems were experienced with the Kinect in outdoor locations, it was found to be reasonably reliable indoors. The proposed solution also provides a strong justification for using hybrid technologies, because of the inability of all sensors to work under all environmental conditions (sunlight, rain, etc.).

System testing by a single person is clearly not enough, because of some genuine differences in peoples behaviour in different situations [23]. A guide dog user can walk very fast as compared to a white cane user, similarly, a blind person from birth can walk at a reasonable speed in know area (Dr. Keith was familiar to university's architectural layout). Therefore, it can be expected that a different user may find the system more useful and reliable.

CHAPTER 8

CONCLUSION

8.1 Thesis contributions

The aim of the research was to produce a robust, working navigation system for the blind. At the start of project, a vision based approach using feature matching seemed the most sensible, though the subsequent appearance of the Microsoft Kinect made a hybrid video-and-depth approach more attractive. The major problem with using features was that it was unclear which detector and descriptor to use, largely because existing evaluations did not yield useful indicators of performance on real world problems. Moreover, evaluations were not necessarily statistically meaningful. Hence, a substantial part of the work reported in chapters 3–4 is concerned with approaches to evaluation that are statistically meaningful. Well-established tests such as McNemar's test and ANOVA are able to perform statistically meaningful comparisons and this work has established that they can be used to compare the performances of feature detectors and descriptors. Furthermore, the results in chapter 4 indicate that the databases most widely used for evaluations are barely large enough. The results in chapter 4 also suggest that performance depends on the angle through which features are rotated, so chapter 5 presented a comprehensive assessment of the performances of corner detectors with regard to angle, both of the corner itself and of its orientation. Corner detectors were considered because more general feature detectors do not run in real time on commodity hardware. This is believed to be the first such assessment.

Chapter 6 presented a novel pair of descriptors for corners that allow the boundaries of objects in successive frames to be matched. One of these descriptors in particular encodes the angle of the corner, amongst other information. It was shown that the performances of these descriptors are indistinguishable from that of BRIEF, and that they can be calculated and matched in real time.

Having established this, chapter 7 presented a complete navigation system using corners — but, because of the advent of the Kinect, it was no longer necessary to match corners. The navigation system includes a number of features that accommodate real-world problems such as spurious depth values from Kinect and the provision of useful feedback to the user. Evaluations with both blindfolded and blind people showed that the system is effective. This hybrid system is more robust than a purely vision-based or purely Kinect-based system would be because the deficiencies in one approach can be ameliorated by the other.

Although the research has demonstrated proof of concept, the equipment is both bulky and heavy (and expensive, as a laptop or equivalent is involved). Clearly, the miniaturization of electronic hardware continues apace, and it is likely that both sensing and the processing will be available in a small enough form factor within 5 years.

Is this type of navigation system likely to replace the ubiquitous white cane? The author thinks it unlikely in the short term, both because of the bulk and weight issues mentioned above and because blind people are familiar with their white cane. However, for a person who is newly blinded (due to age or accident), this is less clear-cut. In particular, this kind of system is likely to be preferable to a white cane in crowds. In the longer term, the author thinks it is highly likely that vision-based systems will become more prevalent for blind navigation.

8.2 Future directions

Considering the shortcomings of performance evaluation metrics previously used to assess the performance of vision algorithms (see chapters 3 and 4), the application of statistical tests other than McNemar's test and ANOVA can be explored. However, the main limitation of most statistical tests is their data requirements, such as a Normal distribution or the homogeneity of variance. Consequently, it may be best to explore non-parametric tests, as they should be able to characterize an algorithm's performance in a statistically valid way without transforming actual data. Examples of such tests are the Wilcoxon signed-ranks test, Wilcoxon matched pairs signed-ranks test, Spearman's rank-order correlation coefficient, Kendall's rank correlation coefficient, the Kruskal-Wallis test, Friedman's two-way analysis of variance by ranks and the Kolmogorov-Smirnov test for a single sample.

Another important factor affecting algorithms' performances is the amount and type of data used. In future, the performance of feature operators, including corner detectors, can be and should be performed on significantly larger amounts of data with more variety of imagery. The investigations in chapter 4 are limited to standard datasets, and capturing larger amount of image data — and with an automated method of generating ground truth — would clearly improve the situation.

In chapter 5, a method of finding corner points in real images to establish ground truth was proposed. This can be further refined by removing the most erroneous part (human guess) of finding corner locations, replacing it by asking the user to draw geometric shapes over objects in images; the corner points of these geometric shapes can then be correlated to the object's corner points. It will only replace the human guess part of the technique and therefore will help in reducing error. Non-geometric shapes can be handled using the end of lines and dots as corner points.

The two descriptors proposed in this work, AMIE and CMIE, are not scale invariant, so a logical extension of this work is to make them scale invariant — but without compromising on execution speed, which is the main requirement of any vision application in which they can be employed.

Based on the limitations of the Kinect-aided navigation system presented in chapter 7, some modifications can be applied, such as replacing the infra-red sensor with a laser striper. These are currently substantially more expensive but can vastly increase the reliably of the system by giving much more accurate depth reading in any environment. Similarly, the miniaturization of the processing hardware can be achieved by using current mobile technology, such as Android phones, which come with with camera, inertial sensor and GPS. Combining these three sensors can add multiple functionalities to the navigation system, for example using the GPS of the user to plan routes based on road maps.

8.3 Closing remarks

A medical cure for blindness may be very far in the future but developing a good, efficient and easy-to-handle automated navigation assistant for people with visual impairment seems relatively close. With rapid advancements in technology, we can easily imagine a blind person walking comfortably through crowds with a mobile-size guiding device in his/her shirt pocket. The way of getting to this point is described by Elbert Hubbard as

> *"A little more persistence, a little more effort, and what seemed hopeless failure may turn to glorious success."*

— The End —

Appendices

APPENDIX A

CORRECTIONS TO REDUCE TYPE-I ERROR

Following are the tables of P for different Z-values and the associated corrected value using three different methods (Bonferroni,Benjamini & Hochberg, Benjamini & Yekutieli). Table A.1 presents one-tailed test results, while Table A.2 contains results for two-tailed test.The concerns over the application of these corrections are also obvious from these tables, where each method can be made to produce different result by adjusting P. If Bonferroni correction rejects null hypothesis ($P < \alpha = 0.05$), other two corrections accept it ($P > \alpha$).

Table A.1: Z-vlaues, corresponding P-value and corrected P-values for one tailed prediction test

Z-value	**P values (one tailed test)**	**Bonferroni correction**	**Benjamini & Yekutieli correction**	**Benjamini & Hochberg correction**
0	0.500	0.500	1.000	0.500
1	0.250	0.250	1.000	0.254
2	0.148	0.148	0.719	0.153
3	0.102	0.102	0.504	0.107
4	0.078	0.078	0.392	0.083
5	0.063	0.063	0.322	0.069
6	0.053	0.053	0.276	0.059
7	0.045	0.045	0.239	0.051
8	0.040	0.040	0.216	0.046
9	0.035	0.035	0.193	0.041
10	0.032	0.032	0.180	0.038
11	0.029	0.029	0.166	0.035
12	0.026	0.026	0.152	0.032
13	0.024	0.024	0.143	0.031
14	0.023	0.023	0.140	0.030
15	0.021	0.021	0.131	0.028
16	0.020	0.020	0.127	0.027
17	0.019	0.019	0.124	0.026
18	0.018	0.018	0.120	0.026
19	0.017	0.017	0.116	0.025
20	0.016	0.016	0.112	0.024
21	0.015	0.015	0.107	0.023
22	0.014	0.014	0.103	0.022
23	0.014	0.014	0.103	0.022
24	0.013	0.013	0.101	0.021
25	0.013	0.013	0.101	0.021
26	0.012	0.012	0.098	0.021
27	0.012	0.012	0.098	0.021
28	0.011	0.011	0.095	0.020
29	0.011	0.011	0.095	0.020
30	0.011	0.011	0.095	0.020
31	0.010	0.010	0.095	0.020
32	0.010	0.010	0.095	0.020
33	0.010	0.010	0.095	0.020
34	0.009	0.009	0.095	0.020
35	0.009	0.009	0.095	0.020
36	0.009	0.009	0.095	0.020
37	0.009	0.009	0.095	0.020
38	0.008	0.008	0.095	0.020
39	0.008	0.008	0.095	0.020
40	0.008	0.008	0.095	0.020
41	0.008	0.008	0.095	0.020
42	0.008	0.008	0.095	0.020
43	0.007	0.007	0.095	0.020
44	0.007	0.007	0.095	0.020
45	0.007	0.007	0.095	0.020
46	0.007	0.007	0.095	0.020
47	0.007	0.007	0.095	0.020
48	0.007	0.007	0.095	0.020
49	0.006	0.006	0.095	0.020
50	0.006	0.006	0.095	0.020
51	0.006	0.006	0.095	0.020
52	0.006	0.006	0.095	0.020
53	0.006	0.006	0.095	0.020
54	0.006	0.006	0.095	0.020
55	0.006	0.006	0.095	0.020
56	0.006	0.006	0.095	0.020
57	0.006	0.006	0.095	0.020
58	0.005	0.005	0.095	0.020
59	0.005	0.005	0.095	0.020
60	0.005	0.005	0.095	0.020

Table A.2: Z-values, corresponding P-values and suggested correction

Z-value	**P values (two tailed test)**	**Bonferroni correction**	**Benjamini & Yekutieli correction**	**Benjamini & Hochberg correction**
0	1.000	1	1.000	1.000
1	0.500	1	1.000	0.508
2	0.295	1	1.000	0.305
3	0.205	1	1.000	0.216
4	0.156	1	0.784	0.167
5	0.126	1	0.645	0.137
6	0.105	1	0.547	0.116
7	0.090	1	0.477	0.102
8	0.079	1	0.427	0.091
9	0.070	1	0.386	0.082
10	0.063	1	0.354	0.075
11	0.058	1	0.332	0.071
12	0.053	1	0.310	0.066
13	0.049	1	0.292	0.062
14	0.045	1	0.274	0.058
15	0.042	1	0.262	0.056
16	0.040	1	0.255	0.054
17	0.037	1	0.241	0.051
18	0.035	1	0.233	0.050
19	0.033	1	0.225	0.048
20	0.032	1	0.224	0.048
21	0.030	1	0.215	0.046
22	0.029	1	0.213	0.045
23	0.028	1	0.211	0.045
24	0.027	1	0.209	0.045
25	0.025	1	0.199	0.042
26	0.024	1	0.196	0.042
27	0.024	1	0.196	0.042
28	0.023	1	0.196	0.042
29	0.022	1	0.196	0.042
30	0.021	1	0.194	0.041
31	0.021	1	0.194	0.041
32	0.020	1	0.194	0.041
33	0.019	1	0.194	0.041
34	0.019	1	0.194	0.041
35	0.018	1	0.194	0.041
36	0.018	1	0.194	0.041
37	0.017	1	0.194	0.041
38	0.017	1	0.194	0.041
39	0.016	0.976	0.194	0.041
40	0.016	0.976	0.194	0.041
41	0.016	0.976	0.194	0.041
42	0.015	0.915	0.194	0.041
43	0.015	0.915	0.194	0.041
44	0.014	0.854	0.194	0.041
45	0.014	0.854	0.194	0.041
46	0.014	0.854	0.194	0.041
47	0.014	0.854	0.194	0.041
48	0.013	0.793	0.194	0.041
49	0.013	0.793	0.194	0.041
50	0.013	0.793	0.194	0.041
51	0.012	0.732	0.194	0.041
52	0.012	0.732	0.194	0.041
53	0.012	0.732	0.194	0.041
54	0.012	0.732	0.194	0.041
55	0.012	0.732	0.194	0.041
56	0.011	0.671	0.194	0.041
57	0.011	0.671	0.194	0.041
58	0.011	0.671	0.194	0.041
59	0.011	0.671	0.194	0.041
60	0.011	0.671	0.194	0.041

BIBLIOGRAPHY

[1] M. August Colenbrander, “International standards: Visual standards – aspects and ranges of vision loss with emphasis on population surveys.” *International Council of Opthalmology, 29th International Congress of Opthalmology*, 2002.

[2] G. Lacey and K. Dawson-Howe, “Personal Adaptive Mobility Aid (PAM-AID) for the Infirm and Elderly Blind,” 2007.

[3] A. M. Flynn, “Combining sonar and infrared sensors for mobile robot navigation,” *The International Journal of Robotics Research*, vol. 7, no. 6, pp. 5–14, 1988.

[4] Y. Zhang, “A survey on evaluation methods for image segmentation,” *Pattern recognition*, vol. 29, no. 8, pp. 1335–1346, 1996.

[5] F. Mohanna and F. Mokhtarian, “Performance evaluation of corner detection algorithms under similarity and affine transforms,” *BMVC 2001*.

[6] P. Rockett, “Performance assessment of feature detection algorithms: A methodology and case study on corner detectors,” *Image Processing, IEEE*

Transactions on, vol. 12, no. 12, pp. 1668–1676, 2004.

[7] P. Tissainayagam and D. Suter, "Assessing the performance of corner detectors for point feature tracking applications," *Image and Vision Computing*, vol. 22, no. 8, pp. 663–679, 2004.

[8] W. Wang and R. Dony, "Evaluation of image corner detectors for hardware implementation," in *Electrical and Computer Engineering, 2004. Canadian Conference on*, vol. 3. IEEE, 2004, pp. 1285–1288.

[9] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Gool, "A comparison of affine region detectors," *International journal of computer vision*, vol. 65, no. 1, pp. 43–72, 2005.

[10] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1615–1630, 2005.

[11] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Foundations and Trends in Computer Graphics and Vision*, vol. 3, no. 3, pp. 177–280, 2008.

[12] Kinect hacks. 2013 Jan 16. [Online]. Available: http://www.kinecthacks.com

[13] E. Bostanci, N. Kanwal, and A. F. Clark, "Extracting planar features from kinect sensor," in *Computer Science and Electronic Engineering Conference (CEEC), 2012 4th*. IEEE, 2012, pp. 111–116.

[14] E. Bostanci, N. Kanwal, and A. Clark, "Kinect-driven augumentation of the real world for cultural heritage," in *UKSIM'13, IEEE, Cambridge*, 2013.

[15] N. Kanwal, S. Ehsan, and A. Clark, "Are performance differences of interest operators statistically significant?" in *Computer Analysis of Images and*

Patterns. Springer, 2011, pp. 429–436.

[16] N. Kanwal, S. Ehsan, E. Bostanci, and A. F. Clark, "A statistical approach for comparing the performances of corner detectors," in *Communications, Computers and Signal Processing (PacRim), 2011 IEEE Pacific Rim Conference on*. IEEE, 2011, pp. 321–326.

[17] ——, "Evaluating the angular sensitivity of corner detectors," in *Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–4.

[18] N. Kanwal, E. Bostanci, and A. Clark, "Describing corners using angle, mean intensity and entropy of informative arcs," *Electronics letters*, vol. 48, no. 4, pp. 209–210, 2012.

[19] N. Kanwal, B. Erkan, and A. F. Clark, "Kinect aided navigation system for visually impaired people," in *Recognition and Action for Scene Understanding (REACTS), York, UK, August 30, 2013, Proceedings*. Springer, 2013.

[20] B. Punani and N. Rawal, *Visual Impairment Handbook*. Blind People's Association, 2000.

[21] D. Clark-Carter, A. Heyes, and C. Howarth, "The efficiency and walking speed of visually impaired people," *Ergonomics*, vol. 29, no. 6, pp. 779–789, 1986.

[22] S. Shoval, I. Ulrich, J. Borenstein *et al.*, "Computerized obstacle avoidance systems for the blind and visually impaired," *Intelligent systems and technologies in rehabilitation engineering*, pp. 414–448, 2000.

[23] N. A. Bradley and M. D. Dunlop, "Investigating context-aware clues to assist navigation for visually impaired people," in *Proceedings of Workshop on Building Bridges: Interdisciplinary Context-Sensitive Computing, University of Glasgow*, 2002.

[24] Hordle ce primary school. 2013 July 22. [Online]. Available: http://hordle.schoolblogger.co.uk/2012/04/guide-dogs-fundraising/

[25] I. Ulrich and J. Borenstein, "The guidecane-applying mobile robot technologies to assist the visually impaired," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 31, no. 2, pp. 131–136, 2001.

[26] J. Strong, "Infrared range finder," Oct.24 1961, uS Patent 3,005,913.

[27] D. Huang, "Laser range finder and method to measure a distance," Mar.20 2007, uS Patent 7,193,692.

[28] T. Suginouchi, K. Saito, and M. Hashimoto, "Ultrasonic range finder," Nov.11 2005, wO Patent 2,005,106,530.

[29] S. Willis and S. Helal, "A passive rfid information grid for location and proximity sensing for the blind user," *University of Florida Technical Report*, pp. 1–20, 2004.

[30] ——, "Rfid information grid for blind navigation and wayfinding," in *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on*. IEEE, 2005, pp. 34–37.

[31] S. Chumkamon, P. Tuvaphanthaphiphat, and P. Keeratiwintakorn, "A blind navigation system using rfid for indoor environments," in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008. 5th International Conference on*, vol. 2. IEEE, 2008, pp. 765–768.

[32] M. Hebert, "Active and passive range sensing for robotics," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 102–110.

[33] M.-C. Amann, T. Bosch, M. Lescure, R. Myllyla, and M. Rioux, "Laser ranging: a critical review of usual techniques for distance measurement," *Optical Engineering*, vol. 40, no. 1, pp. 10–19, 2001.

[34] R. G. Dorsch, G. Häusler, and J. M. Herrmann, "Laser triangulation: fundamental uncertainty in distance measurement," *Applied Optics*, vol. 33, no. 7, pp. 1306–1314, 1994.

[35] D. R. Paschotta, "Distance measurements with lasers," *Encyclopedia of laser physics and technology ()*, 2009.

[36] L. C., "Sensory aid for the blind," *Electronics*, p. 116, 1946.

[37] J. Benjamin, N. Ali, and A. Schepis, "A laser cane for the blind," in *Proceedings of the San Diego Biomedical Symposium*, vol. 12, no. 53-57, 1973.

[38] L. Kay, "An ultrasonic sensing probe as a mobility aid for the blind," *Ultrasonics*, vol. 2, no. 2, pp. 53–59, 1964.

[39] L. Russell, "Travel path sounder," in *Proceedings of Rotterdam Mobility Res. Conference*, 1965.

[40] D. Bissitt and A. D. Heyes, "An application of biofeedback in the rehabilitation of the blind," *Applied Ergonomics*, vol. 11, no. 1, pp. 31–33, 1980.

[41] S. Tachi, K. Tanie, K. Komoriya, and M. Abe, "Electrocutaneous Communication in Seeing-eye Robot (MELDOG)," in *Proceedings 4th Annual Conference IEEE Engineering in Medicine and Biology Society*, 1982, pp. 356–361.

[42] L. Ran, S. Helal, and S. Moore, "Drishti: An integrated indoor/outdoor blind navigation system and service," in *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*. IEEE, 2004, pp. 23–30.

[43] S. Cardin, D. Thalmann, and F. Vexo, "Wearable obstacle detection system

for visually impaired people," in *VR workshop on haptic and tactile perception of deformable objects*, 2005, pp. 50–55.

[44] B. Cyganek and J. Borgosz, "Computer platform for transformation of visual information into sound sensations for vision impaired persons," in *Computer Vision Systems*. Springer, 2003, pp. 182–191.

[45] S. Se and M. Brady, "Stereo vision-based obstacle detection for partially sighted people," *Computer VisionÃćâĈňâĂİACCV'98*, pp. 152–159, 1998.

[46] s. Se and M. Brady, "Vision-based detection of staircases," in *Fourth Asian Conference on Computer Vision ACCV*, vol. 1, 2000, pp. 535–540.

[47] S. Se, "Zebra-crossing detection for the partially sighted," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2. IEEE, 2000, pp. 211–217.

[48] P. B. Meijer, "An experimental system for auditory image representations," *Biomedical Engineering, IEEE Transactions on*, vol. 39, no. 2, pp. 112–121, 1992.

[49] G. Sainarayanan, "On intelligent image processing methodologies applied to navigation assistance for visually impaired," *Ph.D. Thesis*, 2002.

[50] M. Capp and P. Picton, "The optophone: an electronic blind aid," *Engineering Science and Education Journal*, vol. 9, no. 3, pp. 137–143, 2000.

[51] J. Guerrero, R. Martinez-Cantin, and C. Sagüés, "Visual map-less navigation based on homographies," *Journal of Robotic Systems*, vol. 22, no. 10, pp. 569–581, 2005.

[52] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge Univ Press, 2000, vol. 2.

[53] J. D. Anderson, D.-J. Lee, and J. K. Archibald, "Embedded stereo vision

system providing visual guidance to the visually impaired," in *Life Science Systems and Applications Workshop, 2007. LISA 2007. IEEE/NIH*. IEEE, 2007, pp. 229–232.

[54] H. J. Andersen, K. Kirk, T. Dideriksen, C. Madsen, M. Holte, and T. Bak, "Obstacle detection by stereo vision, introducing the pq method," in *IEEE Conference Proceedings of the 2nd International Conference on Informatics in Control, Automation and Robotics*, vol. 2, 2005, pp. 250–257.

[55] N. Bourbabkis, "Sensing surround 3-d space for navigation of the blind," *Engineering in Medicine and Biology MAgazine*, vol. 27, no. 1, 2008.

[56] N. Molton, *Computer vision as an aid for the visually impaired, PhD Thesis*. University of Oxford, 1998.

[57] C. Rasmussen, "A hybrid vision+ ladar rural road follower," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 156–161.

[58] ——, "Roadcompass: following rural roads with vision+ ladar using vanishing point tracking," *Autonomous Robots*, vol. 25, no. 3, pp. 205–229, 2008.

[59] ——, "Texture-based vanishing point voting for road shape estimation." in *BMVC*, 2004, pp. 1–10.

[60] M. Zöllner, S. Huber, H.-C. Jetter, and H. Reiterer, *NAVI–A Proof-of-Concept of a Mobile Navigational Aid for Visually Impaired Based on the Microsoft Kinect*. Springer, 2011.

[61] W. K. Meers, S., "A substitute vision system for providing 3d perception and gps navigation via electro-tactile stimulation," *School of IT and Computer Science University of Wollongong*, 2005.

[62] D. Dakopoulos and N. G. Bourbakis, "Wearable obstacle avoidance elec-

tronic travel aids for blind: a survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, no. 1, pp. 25–35, 2010.

[63] S. Mann, J. Huang, R. Janzen, R. Lo, V. Rampersad, A. Chen, and T. Doha, "Blind navigation with a wearable range camera and vibrotactile helmet," in *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 2011, pp. 1325–1328.

[64] J. Cunha, E. Pedrosa, C. Cruz, A. J. Neves, and N. Lau, "Using a depth camera for indoor robot localization and navigation," *DETI/IEETA-University of Aveiro, Portugal*, 2011.

[65] L. Xia, C.-C. Chen, and J. Aggarwal, "Human detection using depth information by kinect," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*. IEEE, 2011, pp. 15–22.

[66] A. Khan, F. Moideen, J. Lopez, W. L. Khoo, and Z. Zhu, "Kindectect: kinect detecting objects," in *Computers Helping People with Special Needs*. Springer, 2012, pp. 588–595.

[67] F. Tarsha-Kurdi, T. Landes, P. Grussenmeyer, and M. Koehl, "Model-driven and data-driven approaches using lidar data: analysis and comparison," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 35, no. 3/W49A, pp. 87–92, 2007.

[68] Ø. Due Trier, A. K. Jain, and T. Taxt, "Feature extraction methods for character recognition-a survey," *Pattern recognition*, vol. 29, no. 4, pp. 641–662, 1996.

[69] R. J. Campbell and P. J. Flynn, "A survey of free-form object representation and recognition techniques," *Computer Vision and Image Understanding*, vol. 81, no. 2, pp. 166–210, 2001.

[70] E. Malis and S. Benhimane, "A unified approach to visual tracking and servoing," *Robotics and Autonomous Systems*, vol. 52, no. 1, pp. 39–52, 2005.

[71] S. Benhimane and E. Malis, "Homography-based 2d visual tracking and servoing," *The International Journal of Robotics Research*, vol. 26, no. 7, pp. 661–676, 2007.

[72] E. Eade and T. Drummond, "Edge landmarks in monocular SLAM," *Image and Vision Computing*, vol. 27, no. 5, pp. 588–596, 2009.

[73] A. Bimbo, F. Dini, G. Lisanti, and F. Pernici, "Exploiting distinctive visual landmark maps in pan-tilt-zoom camera networks," *Computer Vision and Image Understanding*, 2010.

[74] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International Journal of Computer Vision*, vol. 74, no. 1, pp. 59–73, 2007.

[75] Feature detection (computer vision). 2013 July 20. [Online]. Available: http://en.wikipedia.org/wiki/Feature_detection_(computer_vision)

[76] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15. Manchester, UK, 1988, p. 50.

[77] E. Rosten, R. Porter, and T. Drummond, "FASTER and better: A machine learning approach to corner detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, pp. 105–119, 2010. [Online]. Available: http://lanl.arXiv.org/pdf/0810.2434

[78] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, 2004.

[79] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features,"

in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 404–417.

[80] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[81] Y. Ke and R. Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors," in *CVPR (2)*, 2004, pp. 506–513.

[82] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," *Computer Vision–ECCV 2010*, pp. 778–792, 2010.

[83] D. Park, Y. Jeon, and C. Won, "Efficient use of local edge histogram descriptor," in *Proceedings of the 2000 ACM workshops on Multimedia*. ACM, 2000, p. 54.

[84] M. Bober, "MPEG-7 visual shape descriptors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 716–719, 2001.

[85] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Gool, "A comparison of affine region detectors," *International Journal of Computer Vision*, vol. 65, no. 1, pp. 43–72, 2005.

[86] N. Medathati and J. Sivaswamy, "Local descriptor based on texture of projections," in *Proceedings of the Seventh Indian Conference on Computer Vision, Graphics and Image Processing*. ACM, 2010, pp. 398–404.

[87] J.-n. Liu and G.-h. Zeng, "Improved global context descriptor for describing interest regions," *Journal of Shanghai Jiaotong University (Science)*, vol. 17, pp. 147–152, 2012.

[88] J. Liu and G. Zeng, "Description of interest regions with oriented local self-similarity," *Optics Communications*, vol. 285, no. 10, pp. 2549–2557, 2012.

[89] F. von Hundelshausen and R. Sukthankar, "D-nets: Beyond patch-based

image descriptors," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2941–2948.

[90] X. Guo and X. Cao, "Mift: A framework for feature descriptors to be mirror reflection invariant," *Image and Vision Computing*, 2012.

[91] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–511.

[92] M. Brown and D. G. Lowe, "Invariant features from interest point groups." in *BMVC*, no. s 1, 2002.

[93] S. Ehsan, A. F. Clark, W. M. Cheung, A. M. Bais, B. I. Menzat, N. Kanwal, and K. D. McDonald-Maier, "Memory-efficient design strategy for a parallel embedded integral image computation engine," in *Proceedings of the 2011 Irish Machine Vision and Image Processing Conference*. IEEE Computer Society, 2011, pp. 107–108.

[94] T. Lindeberg and J. Gårding, "Shape-adapted smoothing in estimation of 3-d shape cues from affine deformations of local 2-d brightness structure," *Image and vision computing*, vol. 15, no. 6, pp. 415–434, 1997.

[95] E. Seemann, B. Leibe, K. Mikolajczyk, and B. Schiele, "An evaluation of local shape-based features for pedestrian detection," in *Proc. BMVC*, vol. 4. Citeseer, 2005.

[96] S. Eshan, N. Kanwal, E. Bonstanci, A. F. Clark, and K. D. McDonald-Maier, "Analysis of interest point distribution in surf octaves," in *3rd International Conference on Machine Vision*, 2010, pp. 411–415.

[97] S. Ehsan, N. Kanwal, A. F. Clark, and K. D. McDonald-Maier, "An algorithm for the contextual adaption of surf octave selection with good

matching performance: best octaves," *Image Processing, IEEE Transactions on*, vol. 21, no. 1, pp. 297–304, 2012.

[98] A. Gruen, "Adaptive least squares correlation: a powerful image matching technique," *South African Journal of Photogrammetry, Remote Sensing and Cartography*, vol. 14, no. 3, pp. 175–187, 1985.

[99] C. Silpa-Anan and R. Hartley, "Optimised kd-trees for fast image descriptor matching," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[100] M. Everingham, L. Gool, C. Williams, and A. Zisserman, "Pascal visual object classes challenge results," *Available from www. pascal-network. org*, 2005.

[101] C. Vestri, R. Wybo, and S. Bougnoux, "Transition: a relevant image feature for fast obstacle detection," in *16th World Congress on Intelligent Transport Systems, Stockholm, Sweden*, 2009.

[102] P. Azad, T. Asfour, and R. Dillmann, "Combining Harris interest points and the SIFT descriptor for fast scale-invariant object recognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), St. Louis, USA*, 2009.

[103] M. Marji and P. Siy, "Polygonal representation of digital planar curves through dominant point detection–a nonparametric algorithm," *Pattern Recognition*, vol. 37, no. 11, pp. 2113–2130, 2004.

[104] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof, "PROST: Parallel Robust Online Simple Tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[105] T. Sikora, "The MPEG-7 Visual standard for content description-an overview," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 696–702, 2001.

[106] A. Çapar, B. Kurt, and M. Gö kmen, "Affine Invariant Gradient Based Shape Descriptor," *Multimedia Content Representation, Classification and Security*, pp. 514–521.

[107] Z. Chen and S. Sun, "A Zernike Moment Phase Based Descriptor for Local Image Representation and Matching." *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society*, 2009.

[108] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys (CSUR)*, vol. 38, no. 4, p. 13, 2006.

[109] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3d visual slam with a hand-held rgb-d camera," in *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden*, vol. 2011, 2011.

[110] B. Williams and I. Reid, "On combining visual slam and visual odometry," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 3494–3500.

[111] E. Bostanci, A. F. Clark, and N. Kanwal, "Vision-based user tracking for outdoor augmented reality," in *Computers and Communications (ISCC), 2012 IEEE Symposium on*. IEEE, 2012, pp. 566–568.

[112] E. Bostanci, N. Kanwal, and A. Clark, "User tracking methods for augmented reality," *International Journal of Computer Theory and Engineering*, vol. 5, no. 1, pp. 93–98, 2013.

[113] J. Campbell, R. Sukthankar, and I. Nourbakhsh, "Techniques for evaluating optical flow for visual odometry in extreme terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[114] D. Scaramuzza and R. Siegwart, "Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles," *IEEE Transactions*

on Robotics, vol. 24, no. 5, pp. 1015–1026, 2008.

[115] M. Agrawal, K. Konolige, and M. Blas, "Censure: Center surround extremas for realtime feature detection and matching," *Computer Vision–ECCV 2008*, pp. 102–115.

[116] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge Univ Pr, 2003.

[117] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, "A Constant-Time Efficient Stereo SLAM System," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2009.

[118] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Computer Vision–ECCV 2006*, pp. 430–443, 2006.

[119] N. Funk, "A study of the Kalman filter applied to visual tracking," *Project report CMPUT*, vol. 652.

[120] K. Nummiaro, E. Koller-Meier, and L. Van Gool, "Object tracking with an adaptive color-based particle filter," *Pattern Recognition*, pp. 353–360, 2002.

[121] Business dictionary. 2013 July 28. [Online]. Available: http://www.businessdictionary.com/definition/performance.html

[122] L. B. Lusted, "Signal detectability and medical decision-making," *Science*, vol. 171, no. 3977, pp. 1217–1219, 1971.

[123] T. fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[124] L. Hamel, "Model assessment with roc curves," *The Encyclopedia of Data Warehousing and Mining, Idea Group Publishers,*, 2008.

[125] B. Hanczar, J. Hua, C. Sima, J. Weinstein, M. Bittner, and E. R. Dougherty, "Small-sample precision of roc-related estimates," *Bioinformatics*, vol. 26,

no. 6, pp. 822–830, 2010.

[126] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[127] D. Hand, "Measuring classifier performance: a coherent alternative to the area under the roc curve," *Machine learning*, vol. 77, no. 1, pp. 103–123, 2009.

[128] C. D. Manning and H. Schütze, *Foundations of statistical natural language processing*. MIT Press, 1999, vol. 999.

[129] R. S. Galen and S. R. Gambino, *Beyond normality: the predictive value and efficiency of medical diagnoses*. Wiley New York, 1975.

[130] F. J. Provost, T. Fawcett, and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms." in *ICML*, vol. 98, 1998, pp. 445–453.

[131] C. Schmid, R. Mohr, and C. Bauckhage, "Evaluation of interest point detectors," *International Journal of computer vision*, vol. 37, no. 2, pp. 151–172, 2000.

[132] S. Ehsan, N. Kanwal, A. Clark, and K. McDonald-Maier, "Improved repeatability measures for evaluating performance of feature detectors," *Electronics letters*, vol. 46, no. 14, pp. 998–1000, 2010.

[133] T. Randen and J. Husoy, "Filtering for texture classification: A comparative study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 4, pp. 291–310, 1999.

[134] M. Varma and A. Zisserman, "Texture classification: Are filter banks necessary?" in *IEEE Computer Society Conference on Computer Vision and Pattern*

Recognition, vol. 2, 2003.

[135] G. Carneiro and A. Jepson, “Phase-based local features,” *Lecture Notes in Computer Science*, pp. 282–296, 2002.

[136] C. Valgren and A. Lilienthal, “SIFT, SURF and seasons: Long-term outdoor localization using local features,” in *Proceedings of the European Conference on Mobile Robots (ECMR)*. Citeseer, 2007, pp. 253–258.

[137] S. Smith and J. Brady, “SUSAN: A new approach to low level image processing,” *International journal of computer vision*, vol. 23, no. 1, pp. 45–78, 1997.

[138] C. Tomasi and T. Kanade, “Detection and Tracking of Point Features,” *Image Rochester NY*, no. April, 1991.

[139] J. Shi and C. Tomasi, “Good features to track,” in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, IEEE. IEEE Computer Society, 1994, pp. 593–600.

[140] I. Guyon, J. Makhoul, R. Schwartz, and V. Vapnik, “What size test set gives good error rate estimates?” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 1, pp. 52–64, 2002.

[141] T. V. Perneger, “What’s wrong with bonferroni adjustments,” *British Medical Journal*, vol. 316, pp. 1236–1238, 1998. [Online]. Available: http://www.bmj.com/cgi/content/full/316/7139/1236

[142] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[143] S. Ehsan, N. Kanwal, A. F. Clark, and K. D. McDonald-Maier, “Measuring the coverage of interest point detectors,” in *Image Analysis and Recognition*.

Springer, 2011, pp. 253–261.

[144] E. Bostanci, N. Kanwal, and A. F. Clark, "Feature coverage for better homography estimation: An application to image stitching," *International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2012.

[145] M. D. Dunlop and M. Baillie, "Paper rejected (p> 0.05): An introduction to the debate," *Human-Computer Interaction and Innovation in Handheld, Mobile and Wearable Technologies*, p. 323, 2011.

[146] M. Saag, W. Powderly, G. Cloud, P. Robinson, M. Grieco, P. Sharkey, S. Thompson, A. Sugar, C. Tuazon, J. Fisher *et al.*, "Comparison of amphotericin B with fluconazole in the treatment of acute AIDS-associated cryptococcal meningitis," *New England Journal of Medicine*, vol. 326, no. 2, pp. 83–89, 1992.

[147] N. Uemura, S. Okamoto, S. Yamamoto, N. Matsumura, S. Yamaguchi, M. Yamakido, K. Taniyama, N. Sasaki, and R. Schlemper, "Helicobacter pylori infection and the development of gastric cancer," *New England Journal of Medicine*, vol. 345, no. 11, p. 784, 2001.

[148] R. Frothingham, "Rates of torsades de pointes associated with ciprofloxacin, ofloxacin, levofloxacin, gatifloxacin, and moxifloxacin," *Pharmacotherapy*, vol. 21, no. 12, pp. 1468–1472, 2001.

[149] M. Eliasziw and A. Donner, "Application of the mcnemar test to non-independent matched pair data," *Statistics in medicine*, vol. 10, no. 12, pp. 1981–1991, 1991.

[150] L. Gillick and S. J. Cox, "Some statistical issues in the comparison of speech recognition algorithms," in *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*. IEEE, 1989, pp. 532–535.

[151] V. L. Durkalski, Y. Y. Palesch, S. R. Lipsitz, and P. F. Rust, "Analysis of clus-

tered matched-pair data," *Statistics in medicine*, vol. 22, no. 15, pp. 2417–2428, 2003.

[152] B. Wellner, A. McCallum, F. Peng, and M. Hay, "An integrated, conditional model of information extraction and coreference with application to citation matching," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. AUAI Press, 2004, pp. 593–601.

[153] M. Gönen, K. S. Panageas, and S. M. Larson, "Statistical issues in analysis of diagnostic imaging experiments with multiple observations per patient1," *Radiology*, vol. 221, no. 3, pp. 763–767, 2001.

[154] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, 1947.

[155] R. P. Crease, "Discovery with statistics," *Physics World*, vol. 23, no. 8, p. 19, Aug. 2010.

[156] H. Abdi, *Bonferroni and Šidák corrections for multiple comparisons*. Thousand Oaks, CA: Sage, 2007.

[157] Y. Hochberg and Y. Benjamini, "More powerful procedures for multiple significance testing," *Statistics in medicine*, vol. 9, no. 7, pp. 811–818, 1990.

[158] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: a practical and powerful approach to multiple testing," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 289–300, 1995.

[159] Y. Benjamini and D. Yekutieli, "The control of the false discovery rate in multiple testing under dependency," *Annals of statistics*, pp. 1165–1188, 2001.

[160] M. W. Vasey and J. F. Thayer, "The continuing problem of false positives in

repeated measures anova in psychophysiology: A multivariate solution," *Psychophysiology*, vol. 24, no. 4, pp. 479–486, 1987.

[161] H. O. Hartley, "The use of range in analysis of variance," *Biometrika*, vol. 37, no. 3/4, pp. 271–280, 1950.

[162] E. Peaeson and H. Haetlet, "Biometrika tables for statisticians," *Biometrika Trust*, 1976.

[163] W. R. Rice, "Analyzing tables of statistical tests," *Evolution*, vol. 43, no. 1, pp. 223–225, 1989.

[164] D. B. Duncan, "Multiple range and multiple f tests," *Biometrics*, vol. 11, no. 1, pp. 1–42, 1955.

[165] M. Barrow, *Statistics for economics, accounting and business studies*. Pearson Education, 2009.

[166] D. C. Hoaglin and R. E. Welsch, "The hat matrix in regression and anova," *The American Statistician*, vol. 32, no. 1, pp. 17–22, 1978.

[167] J. Cohen, "Eta-squared and partial eta-squared in fixed factor anova designs." *Educational and Psychological Measurement*, 1973.

[168] A. L. Blackwell, P. D. Thomas, S. Emery, and K. Wareham, "Health gains from screening for infection of the lower genital tract in women attending for termination of pregnancy," *The Lancet*, vol. 342, no. 8865, pp. 206–210, 1993.

[169] N. Cornelis and L. Van Gool, "Fast scale invariant feature detection and matching on programmable graphics hardware," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*. IEEE, 2008, pp. 1–8.

[170] H. Deng, E. N. Mortensen, L. Shapiro, and T. G. Dietterich, "Reinforcement

matching using region context," in *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*. IEEE, 2006, pp. 11–11.

[171] D. Kerr, S. Coleman, and B. Scotney, "Fesid: Finite element scale invariant detector," in *Image Analysis and Processing–ICIAP 2009*. Springer, 2009, pp. 72–81.

[172] R. Lakemond, C. Fookes, and S. Sridharan, "Affine adaptation of local image features using the hessian matrix," in *Advanced Video and Signal Based Surveillance, 2009. AVSS'09. Sixth IEEE International Conference on*. IEEE, 2009, pp. 496–501.

[173] A. K. Jain, J. E. Lee, and R. Jin, "Graffiti-id: Matching and retrieval of graffiti images," in *Proceedings of the First ACM workshop on Multimedia in forensics*. ACM, 2009, pp. 1–6.

[174] M. Toews and W. Wells, "Sift-rank: Ordinal description for invariant feature correspondence," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 172–177.

[175] X. Guo and X. Cao, "Triangle-constraint for finding more good features," in *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE, 2010, pp. 1393–1396.

[176] S. A. de Araújo and H. Y. Kim, "Color-ciratefi: A color-based rst-invariant template matching algorithm," in *IWSSIP-17th International Conference on Systems, Signals and Image Processing*, 2010.

[177] M. Guney and N. Arica, "Maximally stable texture regions," in *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE, 2010, pp. 4549–4552.

[178] P. Kapsalas and S. Kollias, "Affine morphological shape stable boundary regions (ssbr) for image representation," in *Image Processing (ICIP), 2011*

18th IEEE International Conference on. IEEE, 2011, pp. 3381–3384.

[179] T. H. Rassem and B. E. Khoo, "New color image histogram-based detectors," in *Visual Informatics: Sustaining Research and Innovations*. Springer, 2011, pp. 151–163.

[180] A. Issac, C. S. Velayutham *et al.*, "Saddlesurf: A saddle based interest point detector," in *Mathematical Modelling and Scientific Computation*. Springer, 2012, pp. 413–420.

[181] P. Martins, C. Gatta, and P. Carvalho, "Feature-driven maximally stable extremal regions." in *VISAPP (1)*, 2012, pp. 490–497.

[182] X. Y. Wang, P. P. Niu, H. Y. Yang, and L. L. Chen, "Affine invariant image watermarking using intensity probability density-based harris laplace detector," *Journal of Visual Communication and Image Representation*, vol. 23, no. 6, pp. 892–907, 2012.

[183] M. Zhang, Y. Zhang, and J. Wang, "Eliminating false matches using geometric context," in *Contemporary Research on E-business Technology and Strategy*. Springer, 2012, pp. 325–334.

[184] Q. Zhu, X. Liu, C. Cai, and Q. Liu, "Image local invariant feature description fusing multiple information," in *Fifth International Conference on Machine Vision (ICMV 12)*. International Society for Optics and Photonics, 2013, pp. 87 830E–87 830E.

[185] H. O. Hartley, "The maximum f-ratio as a short-cut test for heterogeneity of variance," *Biometrika*, vol. 37, no. 3/4, pp. 308–312, 1950.

[186] P. I. Rockett, "Performance assessment of feature detection algorithms: a methodology and case study on corner detectors," *Image Processing, IEEE Transactions on*, vol. 12, no. 12, pp. 1668–1676, 2003.

[187] X. C. He and N. H. Yung, "Curvature scale space corner detector with adaptive threshold and dynamic region of support," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 2. IEEE, 2004, pp. 791–794.

[188] L. Martinez-Fonte, S. Gautama, W. Philips, and W. Goeman, "Evaluating corner detectors for the extraction of man-made structures in urban areas," in *Geoscience and Remote Sensing Symposium, 2005. IGARSS'05. Proceedings. 2005 IEEE International*, vol. 1. IEEE, 2005.

[189] F. Mokhtarian and F. Mohanna, "Performance evaluation of corner detectors using consistency and accuracy measures," *Computer Vision and Image Understanding*, vol. 102, no. 1, pp. 81–94, 2006.

[190] V. Rodehorst and A. Koschan, "Comparison and evaluation of feature point detectors," in *Proc. 5th International Symposium Turkish-German Joint Geodetic Days" Geodesy and Geoinformation in the Service of our Daily Life", Berlin, Germany*, 2006.

[191] A. Heyden and K. Rohr, "Evaluation of corner extraction schemes using invariance methods," in *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, vol. 1. IEEE, 1996, pp. 895–899.

[192] F. Chabat, G. Z. Yang, and D. M. Hansell, "A corner orientation detector," *Image and Vision Computing*, vol. 17, no. 10, pp. 761–769, 1999.

[193] A. Gil, O. M. Mozos, M. Ballesta, and O. Reinoso, "A comparative evaluation of interest point detectors and local descriptors for visual slam," *Machine Vision and Applications*, vol. 21, no. 6, pp. 905–920, 2010.

[194] Z. Zheng, H. Wang, and E. Khwang Teoh, "Analysis of gray level corner detection," *Pattern Recognition Letters*, vol. 20, no. 2, pp. 149–162, 1999.

[195] X. He and N. Yung, "Corner detector based on global and local curvature

properties," *Optical Engineering*, vol. 47, p. 057008, 2008.

[196] F. Mokhtarian and R. Suomela, "Robust image corner detection through curvature scale space," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 12, pp. 1376–1381, 1998.

[197] M. Koch and R. Kashyap, "Using polygons to recognize and locate partially occluded objects," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 4, pp. 483–494, 2009.

[198] T. Pavlidis, "Algorithms for shape analysis of contours and waveforms," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 4, pp. 301–312, 2009.

[199] J. Tangelder and R. Veltkamp, "A survey of content based 3D shape retrieval methods," *Multimedia Tools and Applications*, vol. 39, no. 3, pp. 441–471, 2008.

[200] F. Sadlo, T. Weyrich, R. Peikert, and M. Gross, "A practical structured light acquisition system for point-based geometry and texture," in *Point-Based Graphics, 2005. Eurographics/IEEE VGTC Symposium Proceedings*. IEEE, 2005, pp. 89–145.

[201] H. Wang and M. Brady, "Real-time corner detection algorithm for motion estimation," *Image and Vision Computing*, vol. 13, no. 9, pp. 695–703, 1995.

[202] J. Bresenham, "A linear algorithm for incremental digital display of circular arcs," *Communications of the ACM*, vol. 20, no. 2, pp. 100–106, 1977.

[203] M. Padma and P. Vijaya, "Entropy based texture features useful for automatic script identification," *International Journal on Computer Science and Engineering*, vol. 2, no. 2, pp. 115–120, 2010.

[204] A. L. Barbieri, G. De Arruda, F. A. Rodrigues, O. M. Bruno, and L. d. F.

Costa, "An entropy-based approach to automatic image segmentation of satellite images," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 3, pp. 512–518, 2011.

[205] A. M. Steane, "Error correcting codes in quantum theory," *Physical Review Letters*, vol. 77, no. 5, p. 793, 1996.

[206] R. M. Andersen, T. Jensen, P. Lisouski, K. A. Mortensen, K. M. Hansen, T. Gregersen, and P. Ahrendt, "Kinect depth sensor evaluation for computer vision applications," *Technical report ECE-TR-6*, 2012.

[207] D. Herrera C, J. Kannala, and J. Heikkilä, "Accurate and practical calibration of a depth and color camera pair," in *Computer Analysis of Images and Patterns*. Springer, 2011, pp. 437–445.

[208] N. Burrus. Kinect calibration. 2013 Jan 29. [Online]. Available: http://nicolas.burrus.name/index.php/Research/KinectCalibration

[209] E. Bostanci, N. Kanwal, and A. Clark, "Extracting planar features from kinect sensor," in *Computer Science and Electronic Engineering Conference (CEEC), 2012 4th*, 2012, pp. 111–116.

[210] D. Sales, D. Correa, F. Osório, and D. Wolf, "3d vision based autonomous navigation system using ann and kinect sensor," in *Conference Proceedings EANN*, 2012.

[211] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 6, pp. 679–698, 1986.

[212] M. D. Colenbrander, "Exploratory simulation of pedestrian crossings at roundabouts." *Journal of Transportation Engineering ASCE*, pp. 211–218, 2003.