

FEATURE DETECTION AND MATCHING TOWARDS AUGMENTED
REALITY APPLICATIONS ON MOBILE DEVICES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ERHAN GÜNDOĞDU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2012

Approval of the thesis:

**FEATURE DETECTION AND MATCHING TOWARDS AUGMENTED
REALITY APPLICATIONS MOBILE DEVICES**

Submitted by **ERHAN GÜNDOĞDU** in partial fulfillment of the requirements for
the degree of **Master of Science in Electrical and Electronics Engineering,**
Middle East Technical University, by,

Prof. Dr. Canan Özgen _____

Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____

Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. A. Aydın Alatan _____

Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members

Prof. Dr. Yasemin Yardımcı _____

Information Systems Dept., METU

Prof. Dr. A. Aydın Alatan _____

Electrical and Electronics Engineering Dept., METU

Assoc. Prof. Çağatay Candan _____

Electrical and Electronics Engineering Dept., METU

Dr. Zafer Arıcan _____

Turk Telecom Group R&D

Assist. Prof Afşar Saranlı _____

Electrical and Electronics Engineering Dept., METU

Date: 03.09.2012

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Lastname : Erhan Gündođdu

Signature :

ABSTRACT

FEATURE DETECTION AND MATCHING TOWARDS AUGMENTED REALITY APPLICATIONS ON MOBILE DEVICES

Gündođdu, Erhan

M.S., Department of Electrical and Electronics Engineering

Supervisor: Prof. Dr. A. Aydın Alatan

September 2012, 100 pages

Local feature detection and its applications in different problems are quite popular in vision research. In order to analyze a scene, its invariant features, which are distinguishable in many views of this scene, are used in pose estimation, object detection and augmented reality. However, required performance metrics might change according to the application type; in general, the main metrics are accepted as accuracy and computational complexity. The contributions in this thesis provide improving these metrics and can be divided into three parts, as local feature detection, local feature description and description matching in different views of the same scene. In this thesis an efficient feature detection algorithm with sufficient repeatability performance is proposed. This detection method is convenient for real-time applications. For local description, a novel local binary pattern outperforming state-of-the-art binary pattern is proposed. As a final task, a fuzzy decision tree method is presented for approximate nearest neighbor search. In all parts of the system, computational

efficiency is considered and the algorithms are designed according to limited processing time. Finally, an overall system capable of matching different views of the same scene has been proposed and executed in a mobile platform. The results are quite promising such that the presented system can be used in real-time applications, such as augmented reality, object retrieval, object tracking and pose estimation.

Keywords: Local Feature Detection, Local Feature Description, Approximate Nearest Neighbor Search, Augmented Reality.

ÖZ

TAŞINABİLİR ELEKTRONİK CİHAZLARDA ARTTIRILMIŞ GERÇEKLİK UYGULAMALARINA YÖNELİK İLĞİ NOKTASI ALGILAMA VE EŞLEME

Gündođdu, Erhan

Yüksek Lisans, Elektrik-Elektronik Mühendisliđi Bölümü

Tez Yöneticisi: Prof. Dr. A. Aydın Alatan

Eylül 2012, 100 sayfa

Yerel ilgi noktası algılama ve bu noktaların Bilgisayarla Görü uygulamalarında kullanımı günümüzde oldukça yaygınlaşmıştır. Bir sahneyi analiz edebilmek için, bu sahnenin çeşitli açılardaki imgelerinde de ayırt edilebilen ilgi noktaları poz tahmini, kamera kalibrasyonu, nesne tanıma, nesne algılama ve arttırılmış gerçeklik uygulamalarında kullanılmaktadır. Odaklanılan performans ölçütü uygulama alanına göre değişiklik göstermektedir. Genellikle bu ölçütler algoritmaların doğruluđu veya hesaplama yükü olmaktadır. Bu tezde ele alınan problemler yerel ilgi noktası algılama, yerel ilgi noktası betimleme ve betimlenen bu noktaların verimli bir biçimde eşlenmesi olarak özetlenebilir. Hesaplama karmaşıklığı açısından verimli bir ilgi noktası algılama algoritması sunulmuştur. Yerel betimleme için ise yeni bir ikili karşılaştırma deseni önerilmiştir. Hızlı eşleme için ise çoklu karar ağacına dayanan bir yöntem sunulmuştur. Belirtilen problemlerin çözümlenmesinde hesaplama karmaşıklığı yukarıda anlatılan sebeplerle ön plana çıkarılmış olup algoritmalar sınırlı kaynak varsayımına dayanarak tasarlanmıştır. Sonuç olarak, aynı sahnenin farklı açılardan çekilmiş imgelerindeki ilgi noktalarını bulup eşleyen bir sistem üretilmiştir. Bu sistem gerçek veriler üzerinde test edilmiş olup yapılan testler neticesinde bu sistemin arttırılmış

gerçeklik, nesne tanıma, nesne takibi ve poz tahmini gibi gerçek zamanlı Bilgisayarla Görü uygulamalarında ticari bir ürün olarak sunulabileceği anlaşılmıştır.

Anahtar Kelimeler: Yerel ilgi noktası algılama, yerel ilgi noktası betimleme, yaklaşık en yakın komşu eşlemesi, arttırılmış gerçeklik.

To Mom, Dad, Sister, Serhat and Ceren

ACKNOWLEDGEMENTS

I would like to express my gratitude and deep appreciation to my supervisor Prof. Dr. A. Aydın Alatan for his guidance, positive suggestions and also for the great research environment he had provided.

I would like to thank all members of Multimedia Research Group, especially to Osman Serdar Gedik, Ahmet Saraçođlu, Çađlar Aytekin, Duygu Arslan, Ozan Őener, Yađız Aksoy, Yeti Ziya Gurbuz, Mustafa Ergul, Emrah Tařlı for their assistance and friendship. We have shared, more than an office, two years with bitter sweet memories.

I would like to also acknowledge The Scientific and Technological Research Council of Turkey (TUBITAK) and Turk Telekom for their funds.

Moreover, I would like to thank to Turk Telecom R&D, especially Dr. Zafer Arıcan and İbrahim Kiremitçi for their fruitful and useful ideas.

TABLE OF CONTENTS

ABSTRACT	iv
ÖZ.....	vi
ACKNOWLEDGEMENTS	ix
LIST OF FIGURES	xii
LIST OF TABLES	xvi
CHAPTERS	1
1. INTRODUCTION	1
1.1. Motivation	1
1.2. Scope of the thesis	3
1.3. Outline of the thesis	6
2. LOCAL INTEREST POINT DETECTION METHODS	7
2.1. INTRODUCTION	7
2.2. RELATED WORK.....	8
2.2.1. Corner-like keypoint detection methods.....	8
2.2.2. Blob-like keypoints detection methods.....	14
2.3. PROPOSED KEY POINT EXTRACTION METHOD	16
2.3.1. Key point extraction using nested circles	16
2.3.2. Improving computational complexity.....	18
2.3.3. Multi-scale extension.....	19
2.4. RESULTS	21
2.4.1. Repeatability.....	21
2.4.2. Localization Error.....	26
2.4.3. Matching Score	31
2.4.4. Algorithm Specifications:.....	35
2.5. CONCLUSION.....	36
3. LOCAL INTEREST POINT DESCRIPTION METHODS.....	38
3.1. INTRODUCTION	38

3.2.	RELATED WORK	39
3.2.1.	Distribution based and differential based descriptors	39
3.2.2.	Intensity comparison based descriptors	43
3.3.	PROPOSED LOCAL BINARY DESCRIPTOR.....	48
3.4.	EXPERIMENTAL RESULTS.....	50
4.	APPROXIMATE NEAREST NEIGHBOUR FOR DESCRIPTOR MATCHING...60	
4.1.	INTRODUCTION	60
4.2.	RELATED WORK AND PROPOSED METHOD.....	61
4.3.	EXPERIMENTAL RESULTS.....	66
4.3.1.	Precision-Recall Curves	67
4.3.2.	Precision-Computation time curves for different number of trees	72
4.3.3.	Precision-Computation time curves for different branching factors	76
4.4.	CONCLUSION.....	80
5.	CONCLUSIONS, DISCUSSION AND FUTURE WORK.....	82
	REFERENCES	86
	APPENDICES	93

LIST OF FIGURES

FIGURES

Figure 1 A monument identification application using AR.....	2
Figure 2 A navigation system augmented to the streets	2
Figure 3 A mobile news report application	3
Figure 4 Illustration of a feature matching system	5
Figure 5 Illustration of the window and its change along x and y directions.	9
Figure 6 Illustration of SUSAN circles and nucleus.....	10
Figure 7 Illustration of FAST algorithm.....	11
Figure 8 Left to right: (discretized and cropped) Gaussian second order partial derivatives in y -direction and xy -direction, and their approximations thereof using box filters. The grey regions are equal to zero.	15
Figure 9 Left: The illustration of the center pixel and nested circles. Right: The pseudo-code for the algorithm.	17
Figure 10 Left: The case when C is eliminated in Purple 5-5' test. Right: The case when C is eliminated in Yellow 4-4' test.....	18
Figure 11 Scale-space interest point detection used in BRISK [16].....	20
Figure 12 Repeatability for Bike dataset (deformation type: blur)	22
Figure 13 Repeatability for Trees dataset (deformation type: blur).....	23
Figure 14 Repeatability for Bark dataset (deformation type: rotation + scale).....	23
Figure 15 Repeatability for Graffiti dataset (deformation type: affine).....	24
Figure 16 Repeatability for Leuven dataset (deformation type: illumination change).....	24
Figure 17 Repeatability for Boat dataset (deformation type: rotation + scale)	25
Figure 18 Repeatability for Wall dataset (deformation type: affine)	25
Figure 19 Average repeatability results.....	26
Figure 20 Localization error for Bark dataset (deformation type: rotation + scale).....	27
Figure 21 Localization error for Bike dataset (deformation type: blur).....	27
Figure 22 Localization error for Graffiti dataset (deformation type: affine).....	28
Figure 23 Localization error for Boat dataset (deformation type: rotation + scale).....	28

Figure 24 Localization error for Leuven dataset (deformation type: illumination change)	29
Figure 25 Localization error for Trees dataset (deformation type: blur)	29
Figure 26 Localization error for Wall dataset (deformation type: affine)	30
Figure 27 Avg. localization error	30
Figure 28 Matching score for Bark dataset (deformation type: rotation + scale)	31
Figure 29 Matching score for Bike dataset (deformation type: blur)	32
Figure 30 Matching score for Boat dataset (deformation type: rotation + scale)	32
Figure 31 Matching score for Graffiti dataset (deformation type: affine)	33
Figure 32 Matching score for Leuven dataset (deformation type: illumination)	33
Figure 33 Matching score for Trees dataset (deformation type: blur)	34
Figure 34 Matching score for Wall dataset (deformation type: affine)	34
Figure 35 A sample capture from the webcam. The diameter of the circles represents the scale of the detected points	36
Figure 36 Illustration of SIFT descriptor calculation using orientation gradients	40
Figure 37 Illustration of SURF descriptor calculation	41
Figure 38 Illustration of DAISY sampling pattern	42
Figure 39 ORB binary pattern. Left: High variance under orientation. Right: Reduced Correlation	46
Figure 40 Binary pattern of BRISK	47
Figure 41 Proposed binary pattern illustration. The end points of the lines constitute a comparison pair and the middle points of these lines are sampling points in the pattern. In proposed descriptor, there are four nested circles and the sampling points are chosen sparsely on the circles. (a) first nested circle (b) first and second nested circles (c) first, second and third nested circles (d) final descriptor pattern	49
Figure 42 Parameters of the proposed descriptor. C_1 is the distance of the circles and R_k 's are radius of the comparison circles	52
Figure 43 Recall-Precision Curves for Bark Dataset	55
Figure 44 Recall-Precision Curves for Bikes Dataset	56
Figure 45 Recall-Precision Curves for Boat Dataset	56
Figure 46 Recall-Precision Curves for Graffiti Dataset	57
Figure 47 Recall-Precision Curves for Leuven Dataset	57
Figure 48 Recall-Precision Curves for Trees Dataset	58
Figure 49 Recall-Precision Curves for Wall Dataset	58

Figure 50 Priority search result for error and the ideal error decrease [32].	64
Figure 51 Illustration of 2 dimensional descriptor space. C1 and C2 are centroids and W is the data point to be classified into buckets of C1 and/or C2.	66
Figure 52 Precision vs recall curve for Bark Dataset	68
Figure 53 Precision vs recall curve for Bike Dataset	68
Figure 54 Precision vs recall curve for Boat Dataset	69
Figure 55 Precision vs recall curve for Graffiti Dataset	69
Figure 56 Precision vs recall curve for Leuven Dataset	70
Figure 57 Precision vs recall curve for Trees Dataset	70
Figure 58 Precision vs recall curve for Wall Dataset	71
Figure 59 Precision-comp. time curve for Bark Dataset (complexity type: # of trees)	72
Figure 60 Precision-comp. time curve for Bikes Dataset (complexity type: # of trees)	73
Figure 61 Precision-comp. time curve for Boat Dataset (complexity type: # of trees)	73
Figure 62 Precision-comp. time curve for Graffiti Dataset (complexity type: # of trees)	74
Figure 63 Precision-comp. time curve for Leuven Dataset (complexity type: # of trees)	74
Figure 64 Precision-comp. time curve for Trees Dataset (complexity type: # of trees)	75
Figure 65 Precision-comp. time curve for Wall Dataset (complexity type: # of trees)	75
Figure 66 Precision-comp. time curve for Bark Dataset (complexity type: branching factor)	76
Figure 67 Precision-comp. time curve for Bikes Dataset (complexity type: branching factor)	77
Figure 68 Precision-comp. time curve for Boat Dataset (complexity type: branching factor)	77
Figure 69 Precision-comp. time curve for Graffiti Dataset (complexity type: branching factor)	78
Figure 70 Precision-comp. time curve for Leuven Dataset (complexity type: branching factor)	78
Figure 71 Precision-comp. time curve for Trees Dataset (complexity type: branching factor)	79
Figure 72 Precision-comp. time curve for Wall Dataset (complexity type: branching factor)	79
Figure 73 Two sample frames shows the matching results of the keypoints	80
Figure 74 Mobile application using an Android tablet PC	81
Figure 75 Distance histograms of Bark Dataset for proposed descriptor (256 bit)	93

Figure 76 Distance histograms of Bark Dataset for ORB descriptor (256 bit)	94
Figure 77 Distance histograms of Bikes Dataset for proposed descriptor (256 bit).....	94
Figure 78 Distance histograms of Bikes Dataset for ORB descriptor (256 bit).....	95
Figure 79 Distance histograms of Boat Dataset for proposed descriptor (256 bit)	95
Figure 80 Distance histograms of Boat Dataset for ORB descriptor (256 bit).....	96
Figure 81 Distance histograms of Graffiti Dataset for proposed descriptor (256 bit).....	96
Figure 82 Distance histograms of Graffiti Dataset for ORB descriptor (256 bit)	97
Figure 83 Distance histograms of Leuven Dataset for proposed descriptor (256 bit).....	97
Figure 84 Distance histograms of Leuven Dataset for ORB descriptor (256 bit)	98
Figure 85 Distance histograms of Trees Dataset for proposed descriptor (256 bit).....	98
Figure 86 Distance histograms of Trees Dataset for ORB descriptor (256 bit)	99
Figure 87 Distance histograms of Wall Dataset for proposed descriptor (256 bit)	99
Figure 88 Distance histograms of Wall Dataset for ORB descriptor (256 bit)	100

LIST OF TABLES

TABLES

Table 1 Deformation types in datasets	21
Table 2 Execution times of the algorithms for 800x600 frames. For all of the compared algorithms, approximate number of detected points is kept around 1000. (In machine learning version of FAST, the computation time per a 768x288 frame is declared as 1.3 ms in case of the number of detected points is approximately 500 meaning that our implementation and FAST implementation have almost the same computation time).	35
Table 3 Dataset properties with visual examples.....	53
Table 4 Fisher's criterion results for different R_k and C_k parameters. The parameter combinations giving maximum Fisher's score are shown in bold.	54
Table 5 Fisher's Criterion comparison of ORB [8] and the proposed pattern	54

CHAPTER 1

INTRODUCTION

1.1. Motivation

Feature detection and matching have been becoming popular due to the increasing number of tasks available for mobile devices. Augmented Reality (AR) is one of the vision tasks, which requires detection of keypoints and matching these points in different views and has a wide application range.

The applications that can be performed using Augmented Reality include the tasks in fabrication processes, surgery, navigation, industrial design, military, tourism, news report and architecture.

In all of the tasks, AR helps to improve efficiency of the processes. For instance, AR can be used as a tourist guide in a museum. A tourist can learn the information about a specific monument by just using his mobile phone. A sample application is illustrated in Figure 1. When fabrication processes are considered, the damage due to the mistakes of workers can be minimized using AR in assembly lines. Moreover, a soldier can be guided with the help of an intelligent map in a military task. One of the exciting applications of AR can be a navigation system with virtual arrows augmented to the streets of a city. A sample illustration is given in Figure 2.



Figure 1 A monument identification application using AR

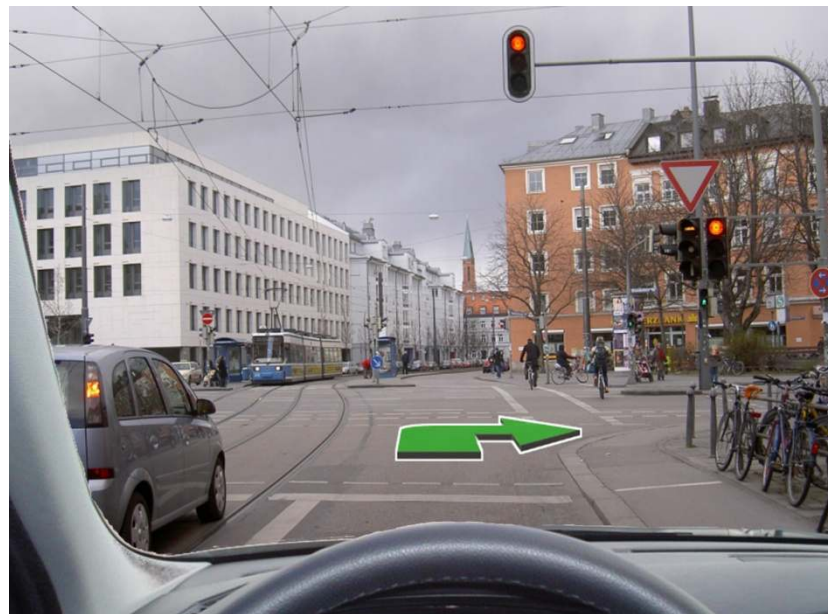


Figure 2 A navigation system augmented to the streets

In addition, AR can also serve in the area of news report. Recent technological improvements have been providing a service in newspapers. For example, a newspaper reader can watch the video of the news by just pointing the camera of a mobile device towards the image of news on the newspaper.



Figure 3 A mobile news report application

1.2. Scope of the thesis

As it can be realized from the wide application range of AR, contributions to Augmented Reality tasks may not only change the daily lives of people, but also increase the efficiency of production chain. As a matter of fact, improvements on the vision problems, which are components of AR applications, become more of an issue. Local feature extraction, local feature description and matching are examples of important parts of AR

tasks. Therefore, this thesis is performed for some of the fundamental and crucial problems in vision research, namely *local feature* (interest point) *detection*, *local feature description* and *approximate nearest neighbor search* for fast descriptor matching.

In general, decomposing an image into subparts is a feasible method, since dealing with smaller regions of an image might improve computational complexity of processing for many vision tasks. Furthermore, if these subparts also carry meaningful information, a sparse analysis on images could be possible. Therefore, the term “*local*” is a valuable notion in many vision problems.

In this work, the mainly focused problems are fast detection of repeatable points and matching these points in different views of the same scene. Although there are many well-known and leading methods that attempt to solve this problem in the literature, it is still hard to avoid the tradeoff between computational complexity and accuracy. Nevertheless, some satisfactory paths to avoid that tradeoff exist when the problem is well defined for a specific application.

If local feature detection problem is considered, there are some efficient algorithms with acceptable accuracy. Most of them are based on simple pixel intensity comparison in image space. Since the idea behind these algorithms has a potential to be improved for real-time AR tasks, the focused methods for local feature detection are intensity comparison based methods. By analyzing the nature of the relationships between local feature points, a new feature detection method, with satisfactory performance, has been proposed in this thesis.

The second step in most vision tasks is describing keypoints. State-of-the-art path for this problem is use of local descriptors. Although elegant and accurate methods [9], [12], [20], exist in the literature, most of them are not feasible to be used in real-time implementation. For mobile devices, efficiency is important. Therefore, efficient local description methods [16], [24] have been becoming frequently preferred methods. The main idea behind these descriptors is intensity comparisons in a local patch around keypoints. Easy calculation of intensity comparisons makes these algorithms computationally efficient with satisfactory performance in accuracy. Hence, local binary descriptors based on intensity comparisons are utilized for local description task. Using

the state-of-the-art methods, a new comparison pattern for local binary descriptors is proposed. The proposed pattern outperforms its counterparts in terms of accuracy performance.

One of the crucial phases of real-time AR applications is matching local descriptors of different scenes. In order to match different views of the same scene, several hundred local descriptors are required to be extracted. Using these descriptors, local descriptors of a different view of the same scene can be matched. However, the most common problem is finding the nearest descriptor vectors for all of the descriptors in the test images. As the number of descriptors increases, the computation time of matching phase increases linearly. If the AR task is assumed to contain many scenes, the number of descriptors in the database increases.

If an efficient method is required to match many descriptors, importance of indexing methods which can learn a database increases. In the literature, there are approximate nearest neighbor methods where the nearest descriptors can be calculated efficiently with convincing performance of accuracy. In this thesis, an efficient method for approximate nearest neighbor search has also been proposed. In the proposed method, a fuzzy decision is exploited to learn the descriptors in a database.

In all parts of the process including feature detection, feature description and matching, novel ideas are exploited and implemented for the sake of efficiency.

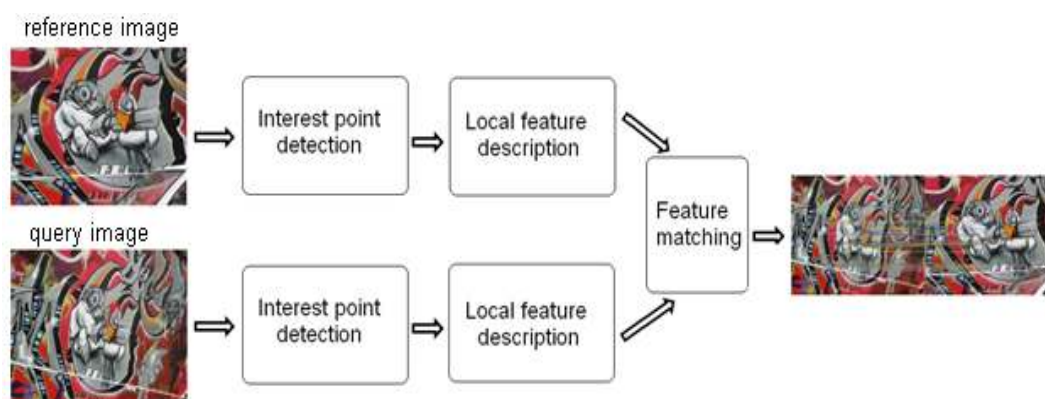


Figure 4 Illustration of a feature matching system

Figure 4 shows a typical example of matching different planar views of the same scene. Specifically, in Augmented Reality (AR) applications, a set of keypoints of a reference image is detected and described before a query image is asked. The aim is to match the keypoints of the query image and the keypoints of the reference image to calculate the geometric relation between these two images. Afterwards, this geometric relationship between these images should be used in AR applications. If the overall system is capable of running fast enough, then the resulting system could also be used in real-time.

1.3. Outline of the thesis

In Chapter 2 of the thesis, local interest point detection methods are discussed. Since this work aims at improving computational efficiency, computationally efficient methods are discussed. Moreover, a novel and efficient algorithm is presented.

In Chapter 3, local feature description methods are analyzed. Moreover, local binary descriptors are exploited for the sake of computational efficiency. The contribution of this work to local descriptors is a new and regular binary pattern outperforming its state-of-the-art counterparts.

Chapter 4 includes approximate nearest neighbor search for matching. A decision tree based approach is adopted due to its influential performance in the literature.

Finally, a conclusion and discussion chapter summarizes the overall system explaining crucial points of the methods analyzed.

CHAPTER 2

LOCAL INTEREST POINT DETECTION METHODS

2.1. INTRODUCTION

Interest point detection is an important and fundamental stage of many vision tasks, such as wide baseline matching, tracking, recognition of objects, AR, pose estimation and camera calibration. In order to accomplish these tasks, there are plenty proposed algorithms for this problem. In a well-known survey paper [1], the definition of local feature is defined as “*a local feature is an image pattern which differs from its immediate neighborhood.*” For distinguishing the difference of a point from its neighbors, there should be a cost function or a metric, comparing the points for dissimilarity. The authors [1] suggest that those differences could be intensity, color or texture. In order to provide a meaning to those differences, many cost functions exist, such as gradient [2] or a response to a specific kernel [13].

The requirements of applications determine the most feasible interest point detection algorithm. For instance, the algorithm in use should be computationally efficient, since real time applications are aimed. If the problem is pose estimation or calibration of a camera, then there should be an almost perfect localization between correspondences. Furthermore, the distinctiveness and repeatability should be prevailing performance requirements, if object recognition is desired with a high accuracy.

Some of the important performance metrics are listed below [1]:

Repeatability: In multiple images of the same scene with different deformations, the same points should *exist* in all of these images.

Accuracy: In multiple images, detected points should *correspond* to the exactly same points of the scene.

Computational Efficiency: This term can be defined as number of computations per pixel, or computation time per image.

Invariance: The detected features should be invariant to many changes, such as affine, rotation, translation, scale, etc.

Distinctiveness/informativeness: The intensity patterns underlying the detected features should show a lot of variation, such that features can be distinguished and matched.

It is almost impossible to satisfy all of the conditions above for a specific feature extraction algorithm. Instead, necessary performance metrics should be studied to be improved for the requirements of a specific task.

2.2. RELATED WORK

Local features can be mainly divided into three main classes, as *points*, *regions* and *special segments*, as stated in [1]. Interest *point* detectors will be discussed in this thesis, since the detected features is expected to be used in AR applications, which may require substantial performance on localization.

2.2.1. Corner-like keypoint detection methods

Interest points can be further classified as corner-like points and blob-like points. In [2], the intensity variation is utilized to find corner-like points, namely Harris corner detection. The basic idea behind that corner detection lies on the fact that, the average intensity variation should be high both horizontally and vertically along the region of

corners. A cost function maximizing the variation in both directions is achieved and a second moment matrix, M , is approximated by Sobel operator in order to calculate the derivatives, while a Gaussian window is utilized to weigh the neighbors according to the distance from the center point.

$$M = \begin{bmatrix} g * I_x^2 & g * (I_x I_y) \\ g * (I_x I_y) & g * I_y^2 \end{bmatrix} \quad (2.1)$$

The matrix M is the approximated second moment matrix and g represents the Gaussian kernel. Eigenvalues of M matrix is exploited to distinguish corner-like points. An illustration of the idea for this algorithm is given in Figure 5.

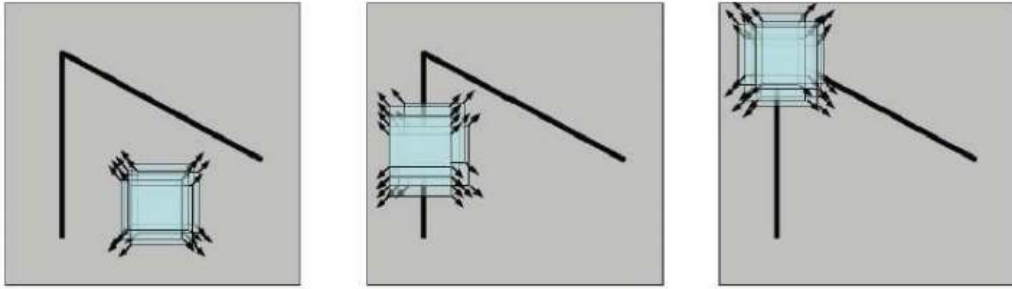


Figure 5 Illustration of the window and its change along x and y directions.

There are many studies related to Harris corner detection [2], e.g. a scale space representation is proposed; however, the main idea is the same as [2]. In another approach 0, the authors propose a new definition of feature quality. According to their definition, a good feature should be tracked with a high quality. In this method 0, the same matrix in [2] is used, but the minimum of eigenvalues of the M matrix is stated as the feature quality.

On the other hand, a feature detector, which is invariant to affine changes, is also proposed [4]. This technique provides high repeatability even in large affine deformations

of the scene. As an alternative to this feature detector, there are region detectors which are also affine invariant. One of such methods is Maximally Stable Extremal Regions MSER [43]; in MSER, extremal regions are detected according to their stability by using a thresholding procedure to gray-level images.

The methods explained above are based on the differential information of the interest points. However; calculations of gradients or differential information take long time to compute if the keypoint detection implementation is planned to run in real time. Instead of using differential or gradient information, intensity comparison based methods are improved to approximate the gradient calculation.

The method proposed by Smith *et. al.* [5] is the pioneer method using intensity comparisons. This detection method uses a circular neighborhood of the interest points, as shown in Figure 6.

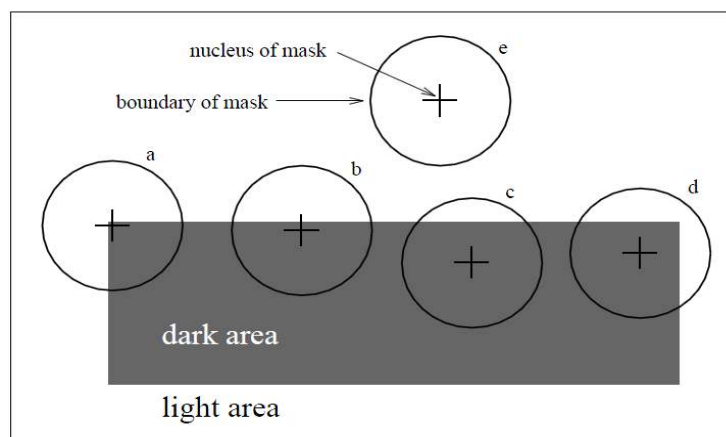


Figure 6 Illustration of SUSAN circles and nucleus.

The basic idea behind Smallest Univalued Segment Assimilating Nucleus (SUSAN) [5] is that the pixels within the circular neighborhood are classified into two, as points similar to the nucleus or otherwise. Similarity between pixels is measured using pixel intensity

comparisons. If the percentage of the dissimilar area is minimum in a local neighborhood, then that point is stated as a corner point according to SUSAN. The most prevailing advantage of SUSAN is that it does not need to calculate any derivatives or second moment matrices. Hence, it is relatively efficient compared to the detection algorithms that calculate image gradients.

Rosten *et. al.* [6] propose recently a new algorithm, Features from Accelerated Segment Test (FAST), which is quite similar to the idea of SUSAN, where there is a circle surrounding a center pixel.

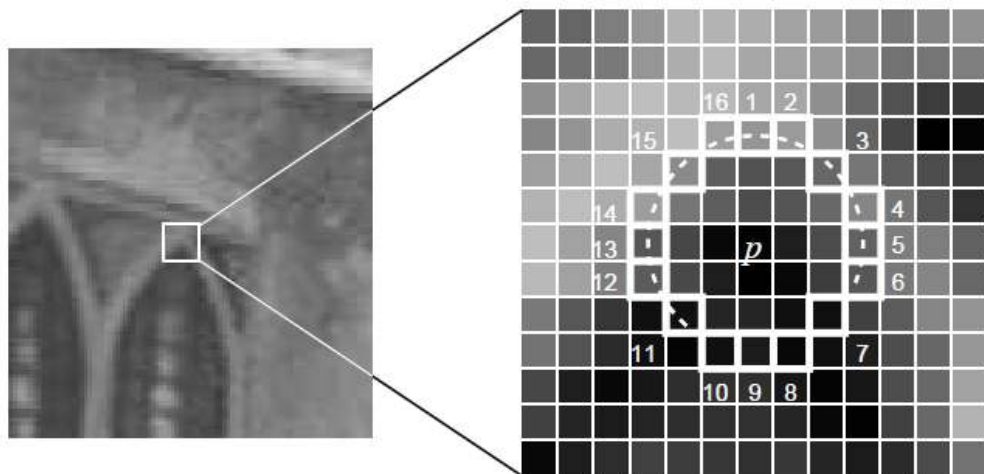


Figure 7 Illustration of FAST algorithm.

As it can be observed in Figure 7, the center pixel p is tested by comparing its intensity with the pixel intensities on the circle, whose boundary consists of 16 pixels. In FAST [6], points whose indices are equal to 1, 5, 9, 13 in Figure 7 on the left are first tested, whether at least three of them are brighter or darker from the intensity of the center pixel p . If this test is passed successfully, then an exhaustive test can be performed as follows: There should be at least n contiguous pixels, which are all brighter or darker than center pixel p . If this condition is satisfied then the algorithm declares p as a corner point.

Although the algorithm seems to be efficient and intuitive, there are a number of disadvantages, such as the first test (testing boundary points 1, 5, 9, and 13) does not generalize well for the case $n < 12$ and multiple features could be detected adjacent to one another.

In order to eliminate such disadvantages, a machine learning approach is used to decide whether a pixel is a corner [6]. Neighboring pixels are classified as similar, brighter or darker than the center pixel p . A decision tree with minimum entropy is constructed by using a training data and the minimization is achieved by using Induced Decision Trees (ID3) method [7].

After minimizations procedure, a non-maxima suppression is used to eliminate points adjacent to the corners. The non-maxima suppression score is given as:

$$V = \max\left(\sum_{x \in S_{bright}} |I_{p \rightarrow x} - I_p|, \sum_{x \in S_{dark}} |I_{p \rightarrow x} - I_p|\right) \quad (2.2)$$

In (2.2), V is the score for non-maxima suppression, I_p is the center pixel, $I_{p \rightarrow x}$ is the neighbor pixels of I_p , S_{bright} and S_{dark} are brighter and darker neighbor pixels of the center pixel, respectively.

The intuition behind the relation in (2.2) is based on selection of points as corner in their local neighborhood, if number of pixels, which are different than the center pixel, is a local maximum in terms of brightness intensity. The remaining points are declared as corners, as the output of the algorithm.

After the supervised stage of the algorithm, the constructed decision tree serves for asking minimum number of questions (or tests), which corresponds to the minimization of the entropy of the tree. FAST [6] as a result of this machine learning approach approximately asks 2.26 questions on the average per pixel, which is sufficiently efficient. However, ID3 method used for tree construction is suboptimal and might miss some important corner patterns. In spite of these disadvantages, FAST algorithm is appropriate for many vision tasks due to its computational efficiency and high repeatability performance.

In ORB [8], the corner detection task is implemented by using FAST-9 algorithm, in which 9 contiguous pixels should be all brighter or darker than the center pixel. Moreover, Harris measures of detected key points are calculated and they are ordered according to the Harris measure. First N key points are selected as corner-like points. This technique gives better performance in terms of repeatability compared to FAST, since relatively strong points are selected out of FAST corner points in terms of their Harris score.

In [15], the authors propose a novel method for fast key point extraction and use a circle around the point of interest, as in FAST [6] and SUSAN [5] methods. The idea can be summarized follows: The circle surrounding a center pixel p is tested such that the pixels diametrically opposed to each other are tested to be different than center pixel at the same time. If this constraint is satisfied, then those points are eliminated. The points, which are not eliminated, are candidates for interest points and their Laplacian of Gaussian (LOG) is calculated to suppress non-maxima neighbors. The condition for being a non-corner point and LoG approximation is given in (2.3) and (2.4).

$$\text{If } \left| I_{(p)} - I_{(p+dR_\varphi)} \right| \leq +\tau \text{ And If } \left| I_{(p)} - I_{(p-dR_\varphi)} \right| \leq +\tau \quad (2.3)$$

$$\text{LoG}(m) \approx \sum_{\varphi \in [0, \pi]} I_{(p-dR_\varphi)} - I_{(p)} + I_{(p+dR_\varphi)} \quad (2.4)$$

In (2.3) and (2.4), $I_{(p)}$ is the intensity value of the pixel p (center pixel) and $I_{(p-dR_\varphi)}$ and $I_{(p+dR_\varphi)}$ are diametrically opposed neighbour pixels of p in φ direction. In Equation (2.3), if both of the conditions are satisfied, then the point of interest is claimed to be non-corner. In the proposed keypoint extraction method, the rule for being non-corner [15] is used. Hence, this rule will be explained in detail in the proposed keypoint extraction section.

2.2.2. Blob-like keypoints detection methods

Lindeberg [46] defined “blob” like regions or points as local extremes at least in one scale of the image space. Scale-space selection of blobs is further improved by SIFT [9]. Scale-space selection of blob-like keypoints could be achieved by using a cascade filtering approach, as in the popular method Scale Invariant Feature Transform (SIFT) [9]. Lowe [9] utilizes a Gaussian function as scale-space kernel, following the works in [10] and [11] by the help of the following relation:

$$\sigma \nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \quad (2.5)$$

Equation (2.5) presents an important approximation of the partial derivative of the G function according to the scale parameter σ . In fact, this relation is an approximation for Laplacian of Gaussian (LoG), namely *Difference of Gaussian* (DoG), which is relatively easier to calculate. In [9], Lowe also shows that the approximation of LoG as DoG has no impact on the stability of extreme detection or localization, even for significant differences in scale. After calculation of DoG in many scales and octaves, each pixel coordinate is tested, whether it is a local maximum in image and scale-space. After rejecting points, which are not local extrema, then non-maxima suppression is used to eliminate edge responses. After this process, the remaining points are declared as stable key points.

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix} \quad (2.6)$$

Bay *et.al.*[12] use some previous studies on extraction of blob-like structures and conclude that Hessian-based detectors are more stable and repeatable than their Harris-based counterparts [12].

In equation (2.6), σ is the scale at which Hessian matrix is calculated. $L_{xx}(x, \sigma)$, $L_{xy}(x, \sigma)$, $L_{yy}(x, \sigma)$ are the second order derivatives of point x convolved with Gaussian kernel by scale σ . Gaussian second order derivatives, $L_{xx}(x, \sigma)$, $L_{xy}(x, \sigma)$ and $L_{yy}(x, \sigma)$

are all approximated by using simple box filters, as shown in Figure 8 and denoted as D_{xx} , D_{xy} and D_{yy} .

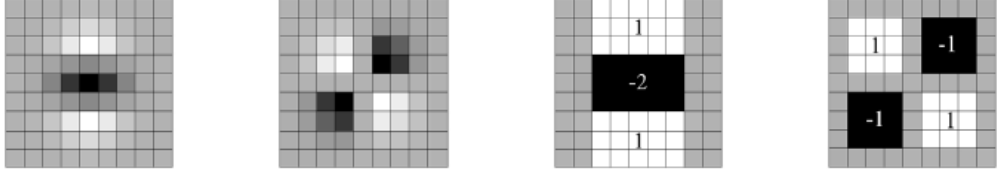


Figure 8 Left to right: (discretized and cropped) Gaussian second order partial derivatives in y -direction and xy -direction, and their approximations thereof using box filters. The grey regions are equal to zero.

For detecting blob-like structures, determinant of an approximated Hessian matrix, which uses approximated derivatives D_{xx} , D_{xy} and D_{yy} , is used as follows:

$$\det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \quad (2.7)$$

where H_{approx} is the approximated Hessian matrix by approximated partial derivatives D_{xx} , D_{xy} and D_{yy} . After calculation of (2.7), local maximum values of pixels for the determinant function of (2.7) in image and scale space are selected to be the points of interest. The keypoint extraction based on approximation of Hessian matrix [12] is more efficient than SIFT [9], while sacrificing slightly from the performance.

However; second order derivative calculations as in [9] and [12] are still not feasible for real-time applications. Therefore, further approximations of the state-of-the-art feature extraction algorithms are proposed. For instance; some researchers use box kernels as approximations of Laplacian of Gaussians to speed up computation [13]. Convolution of image and these box kernels can be computed fast enough for real-time applications. On the other hand, Dupac *et. al.* propose a new blob detector which exploits the phase of Fourier transform of the image points [14]. The main idea can be explained as follows: If

a point has a sinus-like structure within its neighborhood, then its phase becomes almost zero when the phase is approximated in the neighborhood of this point. The algorithm also exploits the phase information to find the shifts through video frames and uses this information to track previously determined blob-like points. In this work, the Zero Shift Points (ZSP) [14] is also tested in terms of repeatability and computational efficiency.

Once all of the keypoint extraction algorithms are considered, one can observe that state-of-the-art keypoint extraction methods, like Harris corner detection [2], SIFT [9] or Fast-Hessian [12] are not appropriate for real-time applications due to their computational inefficiency. Instead of these methods, pixel intensity comparison based methods [5], [6], [8], [15] and [16] could be preferred to execute in real-time even in mobile devices. As a matter of fact, a pixel intensity comparison based method is proposed in this thesis and will be explained in detail in the next section.

2.3. PROPOSED KEY POINT EXTRACTION METHOD

In this work, a key point extraction algorithm, which borrows some ideas from the method in [15], is proposed and this algorithm is based on rapid elimination of non-corner points for speed up. Furthermore, there is no need to perform non-maxima suppression to locate key points.

2.3.1. Key point extraction using nested circles

The proposed key point detection procedure is as follows: In order to test whether the point C is a corner, the pixels on the perimeter of the purple colored circle in Figure 9 is examined first. Any diametrically opposed neighbors, *e. g.* 1 and 1' in Figure 9, are compared in terms of brightness similarity with respect to the center pixel C .

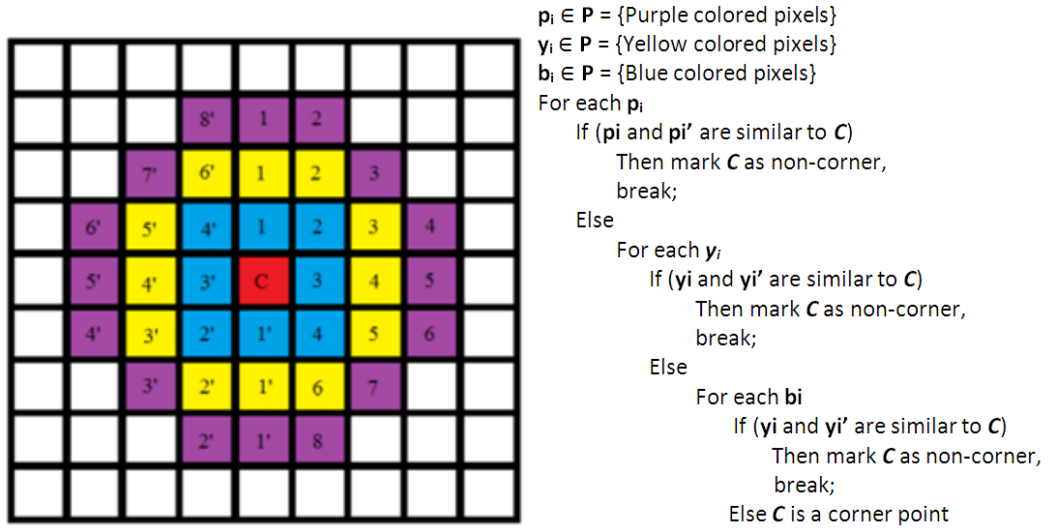


Figure 9 Left: The illustration of the center pixel and nested circles. Right: The pseudo-code for the algorithm.

Rule for the similarity condition of the diametrically opposed pixels comparing the center pixel is given by the following condition:

$$If \left| I_{(p)} - I_{(p+dR_{\phi})} \right| \leq +\tau \text{ And } If \left| I_{(p)} - I_{(p+dR_{\phi})} \right| \leq +\tau \quad (2.8)$$

where τ is a design threshold, as in FAST. If τ is selected to be a small value, then only a few points are detected; otherwise, the number of detected points increases. During in any test for the points on the perimeter, if the similarity condition is satisfied, then the point of interest is declared as non-interesting.

If none of the two opposing perimeter points are similar to the center pixel, C , then this center point is further tested by the inner circles of the purple colored circle. In the following step, the next inner loop (yellow circle in Figure 9) is tested.

If the yellow circle in Figure 9 also fails to determine a similarity of pixels at the end of this test, then the preceding inner blue circle is tested for the similarity of the diametrically opposed pixels against the center pixel, C . If this final test does not yield any two opposing perimeter points similar to C , then the point of interest is declared as an

interest point. It should be emphasized that during any stages of the tests, if the opposing pixel pairs are similar to C at the same time, then the point is marked to be non-corner (see Figure 9).

2.3.2. Improving computational complexity

In practice, there can be some redundancy during these nested tests, when a point is marked as non-corner in the early stages of these tests. For example, assume that the foreground is darker than background and the intensity difference between foreground and background is guaranteed to exceed the threshold T of the nested circle test (see Figure 9). The illustration of such an example is given in Figure 10 on the left. Blue part of the image space is assumed to be foreground of the image and the only corner point in this example is at point D , which is 3 pixels right from the center pixel C . Therefore, for such a condition as in Figure 10 on the left, if the center pixel C is tested and detected as non-corner in the outermost circle test, which is D - D' test, then it is guaranteed that the pixels A and B are also not corners. Hence, there is no need to perform any circle test to A and B points, since they are also non-corners. The elimination of testing pixels A and B (Figure 10 on the left) decreases the computational efficiency significantly.

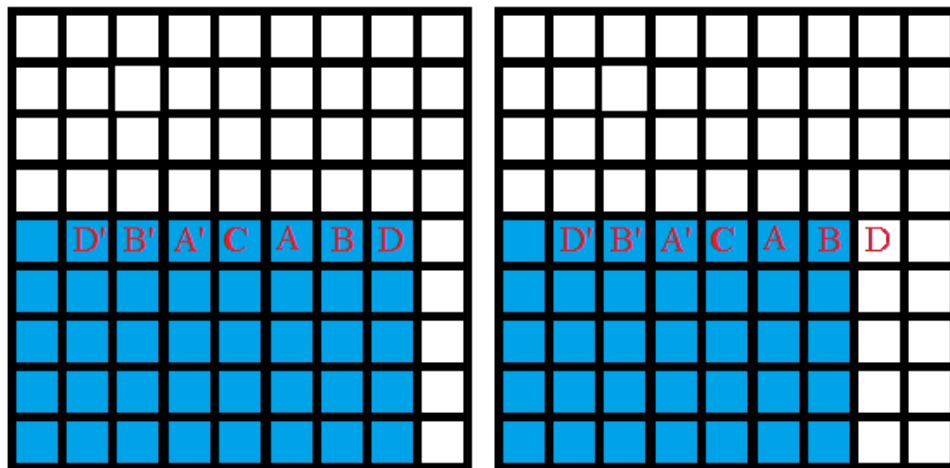


Figure 10 Left: The case when C is eliminated in Purple 5-5' test. Right: The case when C is eliminated in Yellow 4-4' test.

The computation time can be further decreased, when the center pixel is detected as non-corner during the yellow circle test of nested circle (see Figure 9). A sample illustration of such a case is shown in Figure 10 on the right. Using the same idea, if the test is failed for the pixel C in B-B' test, then it is guaranteed that point-A is not a corner. This result is due to the fact that if it is tested, it should be eliminated during *Blue 3-3'* (see Figure 9) test of nested circle.

In order to increase the computational efficiency, the horizontal tests, which are *Purple 5-5'*, *Yellow 4-4'* and *Blue 3-3'* in Figure 9, should be applied initially to eliminate candidate non-corners at the beginning of the test for that center pixel, since the redundancy is exploited along the horizontal direction.

2.3.3. Multi-scale extension

When there is a significant change on the scale of the scene, for better repeatability, there should be a multi-scale representation, which can also detect the same point in different scenes with various scales. Moreover, for a scale invariant description, a scale estimate is also required in most of the feature description algorithms that are scale invariant. Hence, a scale score should be computed in the proposed technique for description part.

In this thesis, scale invariance is provided by using the idea of [16]. In their work, FAST corner detection algorithm [6] is utilized as the baseline feature detection. Scale invariance is accomplished by image pyramids and a detection score. The detection score being used is the metric utilized by FAST to suppress non-maxima points which is shown in Equation (2.2).

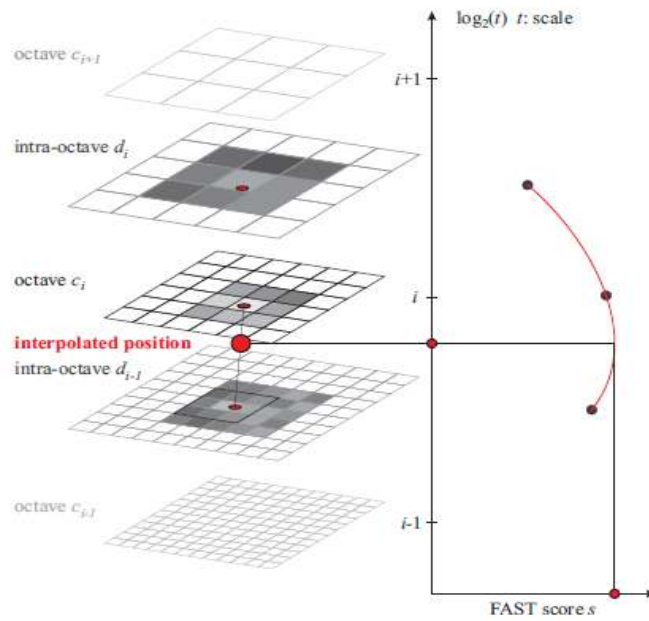


Figure 11 Scale-space interest point detection used in BRISK [16]

First of all, an image pyramid that consists of octaves and intra-octaves is generated. Intra octaves are 1.5 times larger than or smaller than their neighboring octaves. After this procedure, FAST 9-16 detector is applied for each octave and intra-octave with the same threshold τ . In our work, for the sake of simplicity and efficiency, no intra octaves are used. In [16], a curve fitting to the detected local maxima in scale and image space is applied. In the proposed algorithm, the localization procedure of the scale is not implemented. Finally, the detected points have also their scale information to be used in the description part of the work.

In the next section; repeatability, localization error and matching score results are presented for the proposed approach and compared by some state-of-the-art algorithms.

2.4. RESULTS

There are mainly three performance metrics used for comparing the algorithms namely; repeatability, matching score and localization error.

2.4.1. Repeatability

Repeatability is measured according to a similar method in [17] and it is measured between two images by extracting features in both of them. After extraction, the points in one of the images are projected on top of the other image. A nearest neighbor, which is around a predefined distance threshold (3 pixels for our tests), is accepted as “repeated” in both images. The repeatability is calculated in the reverse direction as well and then their average is calculated to obtain the final repeatability percentage. In order to project a point in one of the image to the other one, one must know a priori or estimate the homography between two images, if the scene is assumed to be planar in both images. Datasets in [18], whose homography matrices are provided, is utilized during tests. In Table 1, properties of the datasets are given. There are 5 pairs in each of these types tabulated in Table 1.

Table 1 Deformation types in datasets

Dataset Name	Deformation Type
Bark	Zoom + Rotation Change
Bikes	Blur Change
Boat	Zoom + Rotation Change
Graffiti	View Point Change
Leuven	Light Change
Wall	View Point Change
Tree	Blur Change

In all of the illustrations, the x-axis shows the amount of deformation and as the number in this axis increases the amount of deformation increases. For instance; if the deformation type is view point change, then as the numbers in the x-axis increases, the view point change also increases.

In the viewpoint change test the camera varies from a fronto-parallel view to one with significant foreshortening at approximately 60 degrees to the camera. The scale change and blur sequences are acquired by varying the camera zoom and focus respectively. The scale changes by about a factor of four. The light changes are introduced by varying the camera aperture. The x-axis in the experiments corresponds to the deformation strength in different deformation types.

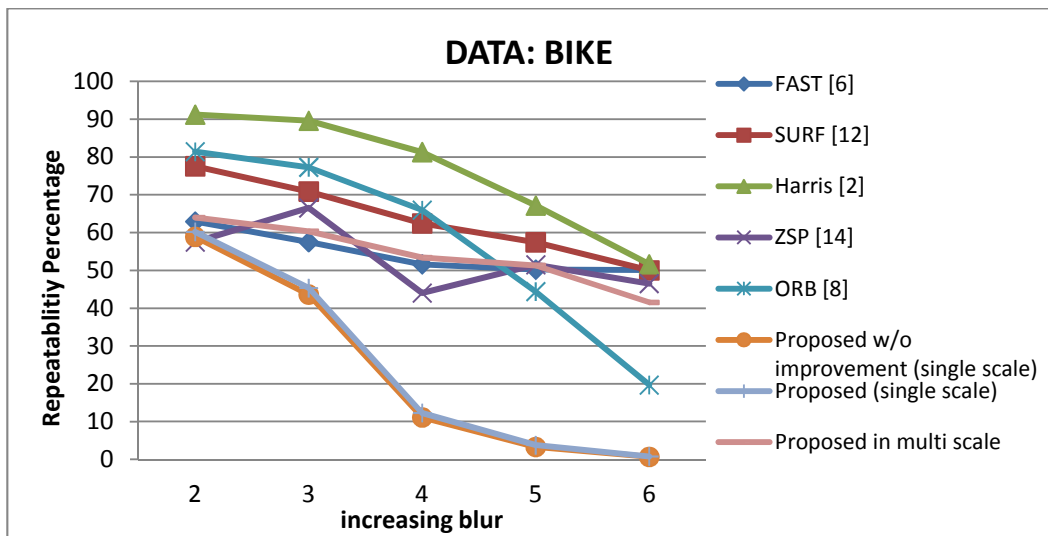


Figure 12 Repeatably for Bike dataset (deformation type: blur)

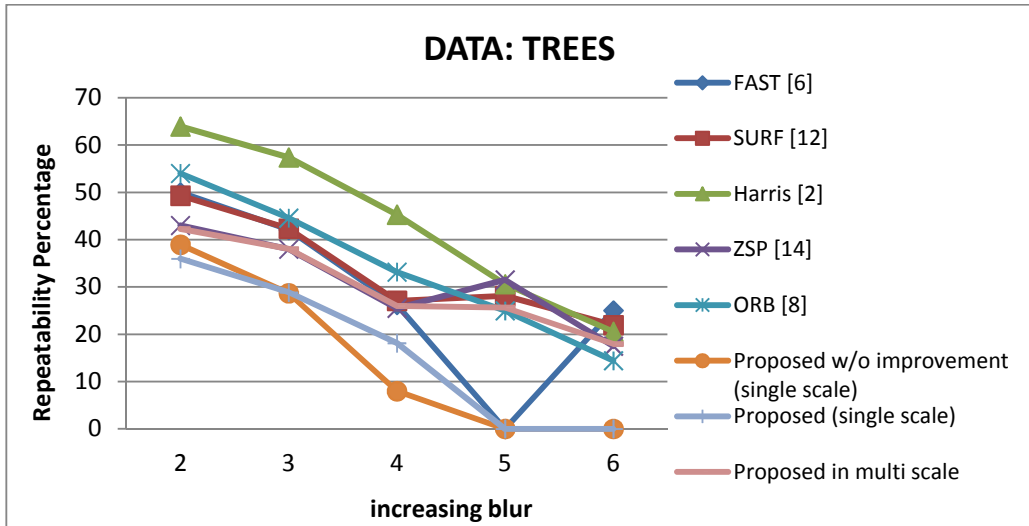


Figure 13 Repeatability for Trees dataset (deformation type: blur)

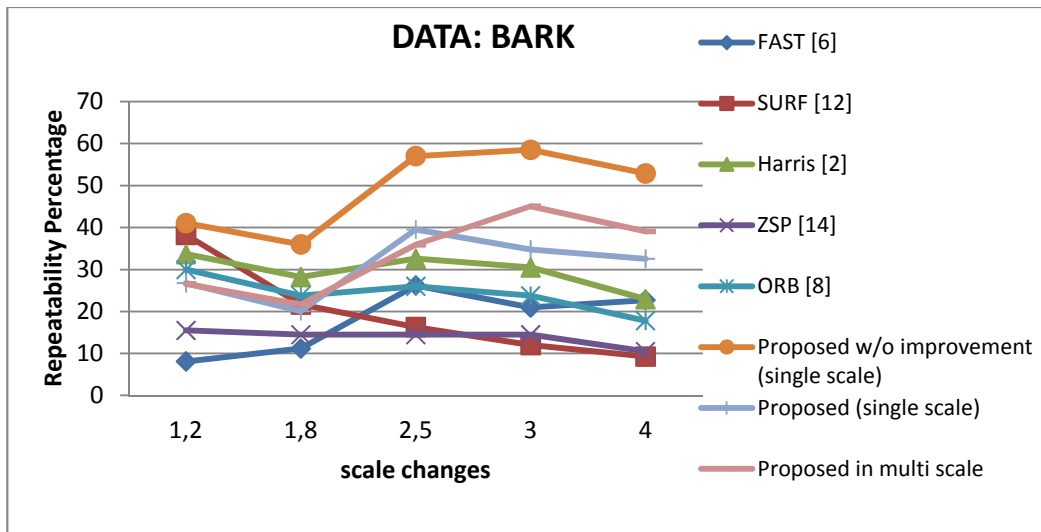


Figure 14 Repeatability for Bark dataset (deformation type: rotation + scale)

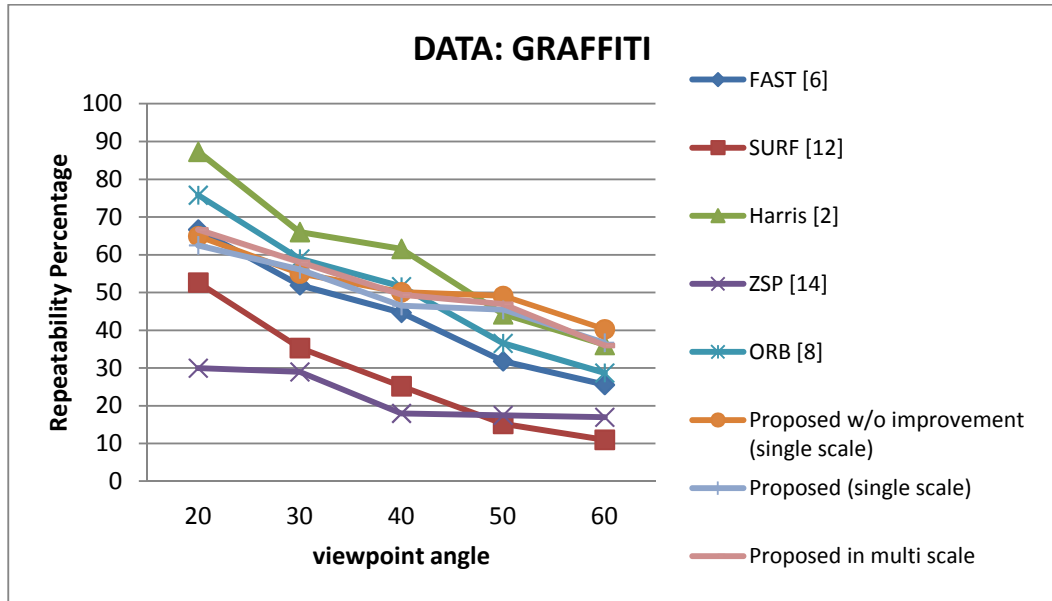


Figure 15 Repeatability for Graffiti dataset (deformation type: affine)

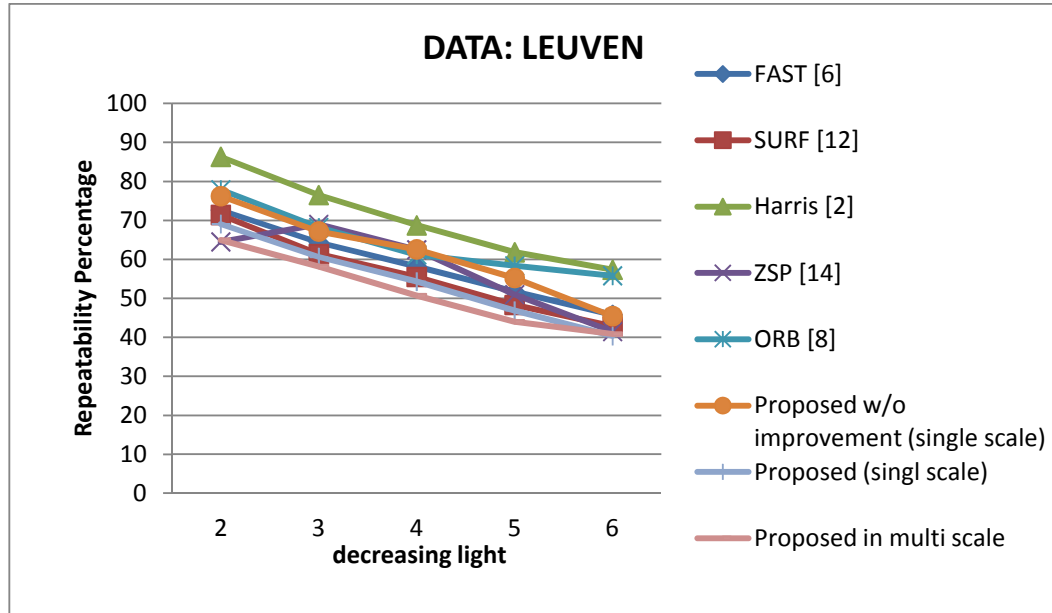


Figure 16 Repeatability for Leuven dataset (deformation type: illumination change)

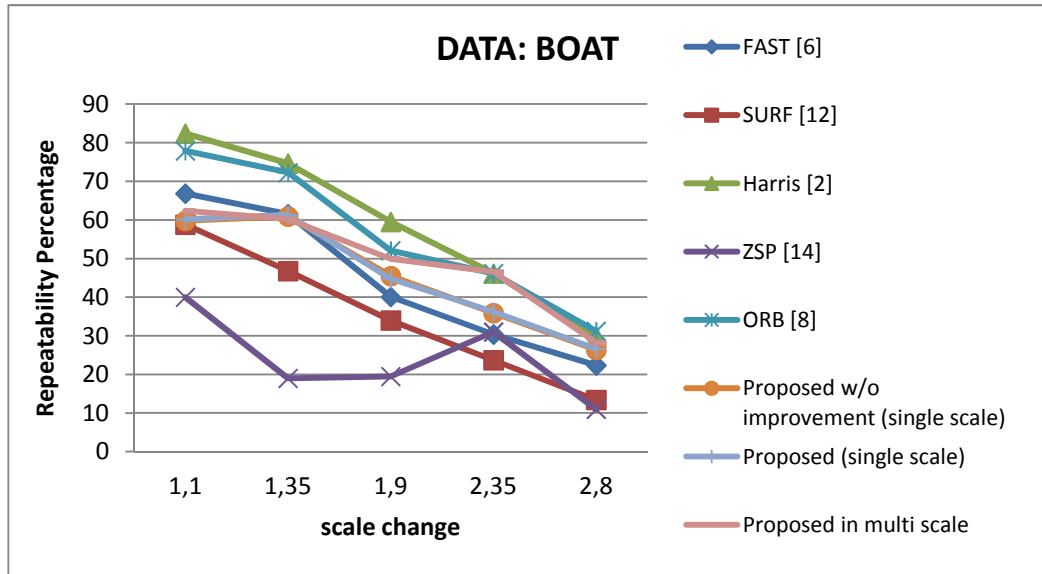


Figure 17 Repeatability for Boat dataset (deformation type: rotation + scale)

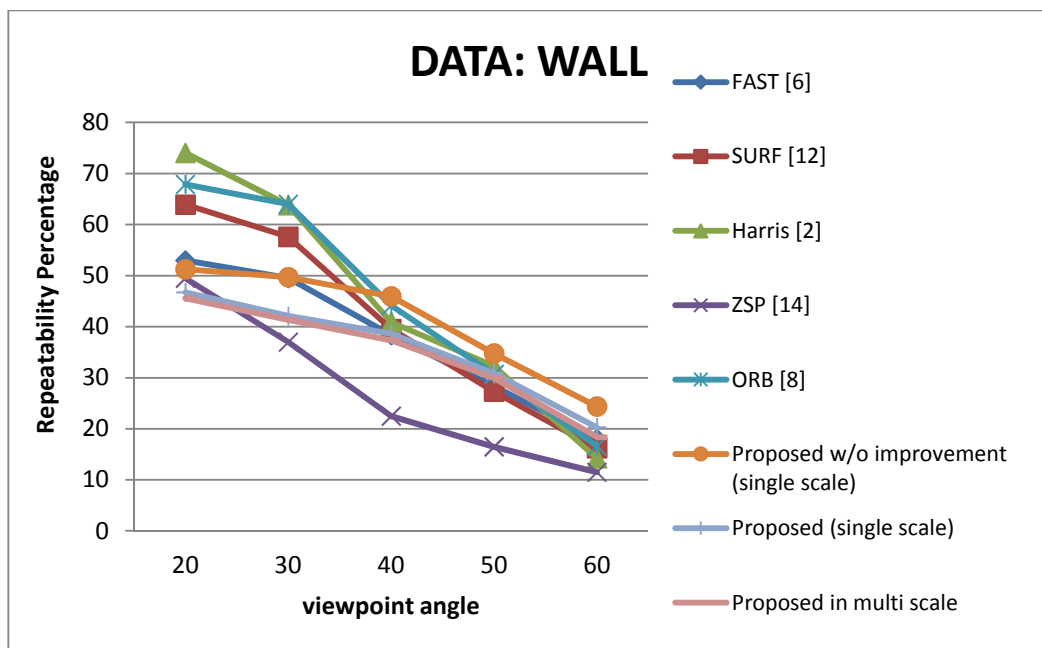


Figure 18 Repeatability for Wall dataset (deformation type: affine)

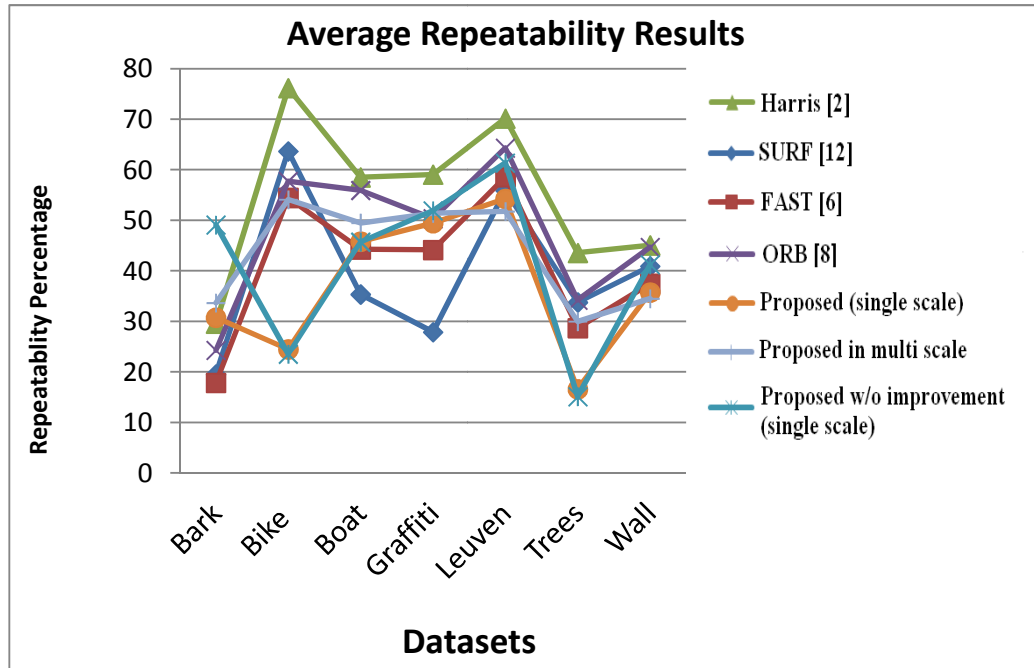


Figure 19 Average repeatability results

When the average repeatability results are compared (see Figure 19), there is a slight difference in repeatability performance between proposed detector with improvement and proposed detector without improvement. Hence, improvement on computational complexity can be preferred while sacrificing a slight performance in repeatability. Moreover, FAST [6] and proposed detector have similar performance in average. Furthermore, multi scale implementation of proposed detector is also comparable with multi scale detection methods such as SURF [12] and ORB [8].

2.4.2. Localization Error

If a point is repeatable, it is obvious that the point is close to the point in the reference image around the predefined distance threshold. The average distance between correspondences is defined as localization error for the image pairs.

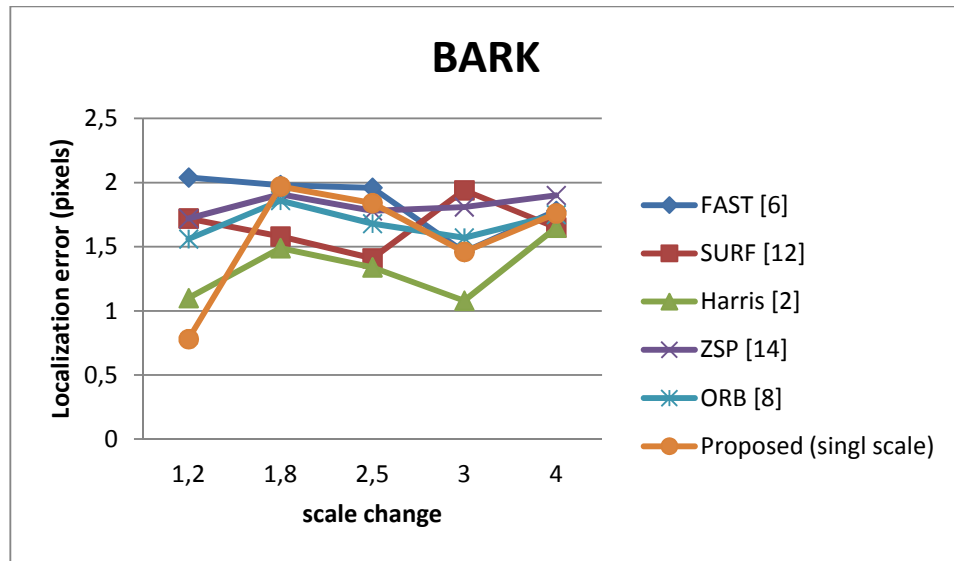


Figure 20 Localization error for Bark dataset (deformation type: rotation + scale)

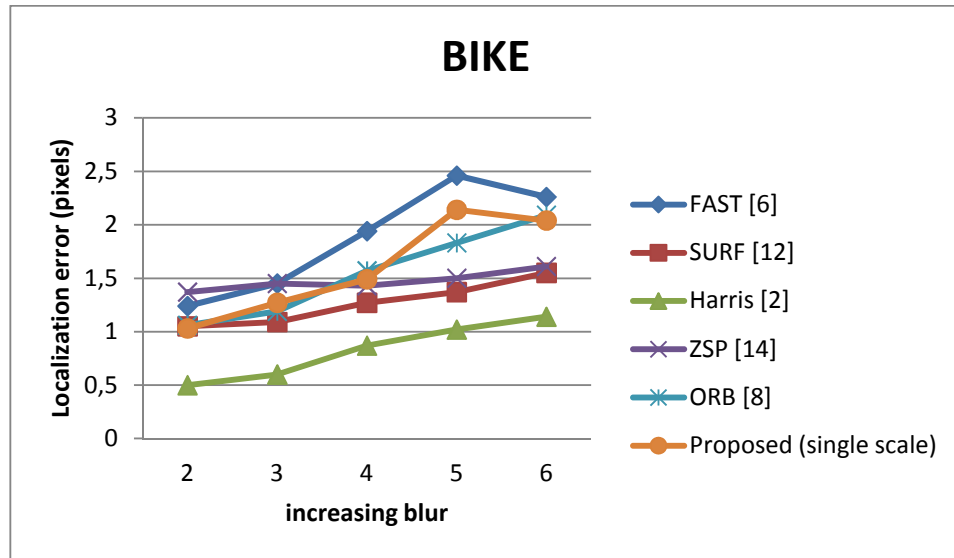


Figure 21 Localization error for Bike dataset (deformation type: blur)

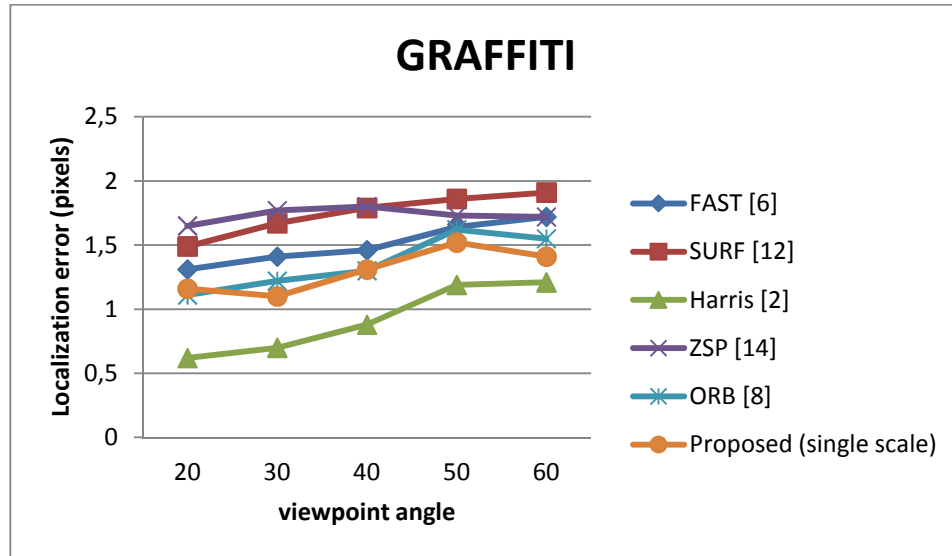


Figure 22 Localization error for Graffiti dataset (deformation type: affine)

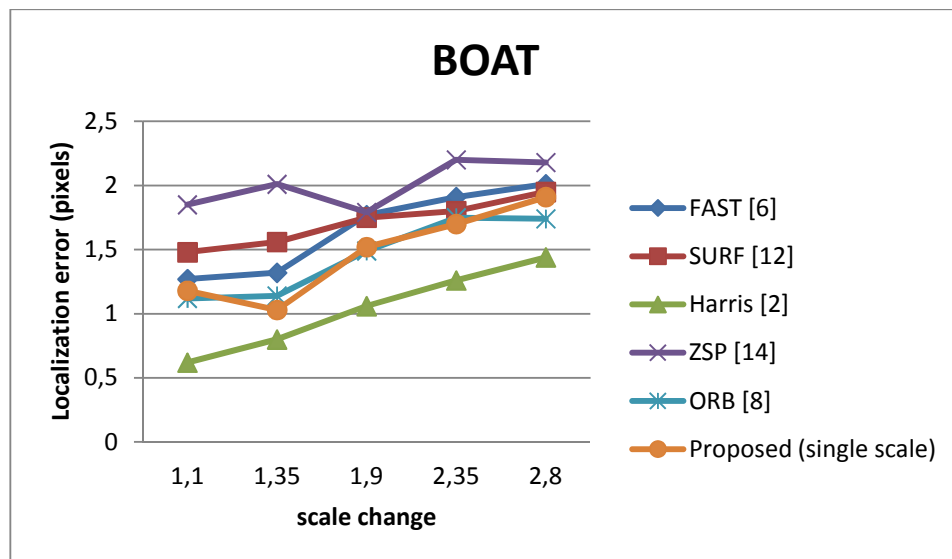


Figure 23 Localization error for Boat dataset (deformation type: rotation + scale)

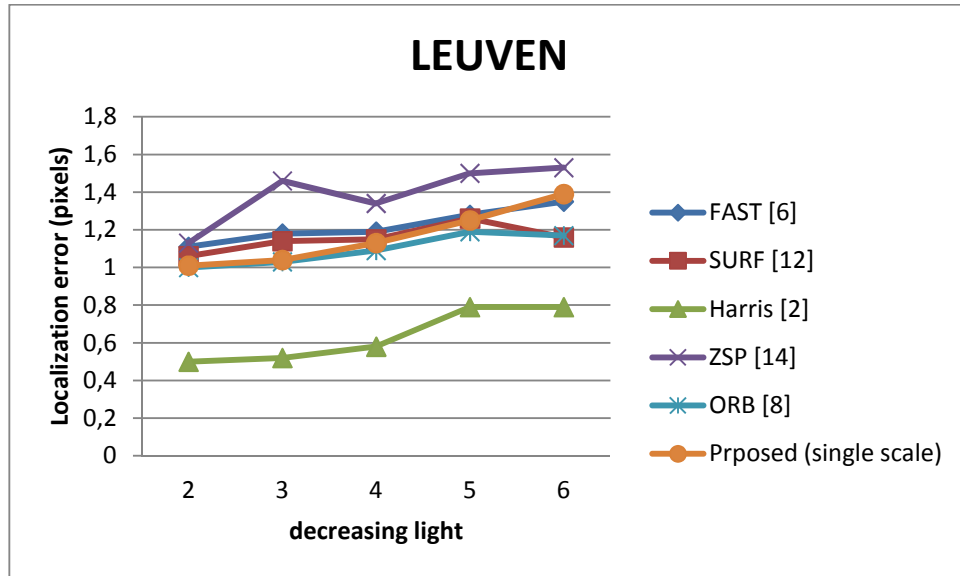


Figure 24 Localization error for Leuven dataset (deformation type: illumination change)

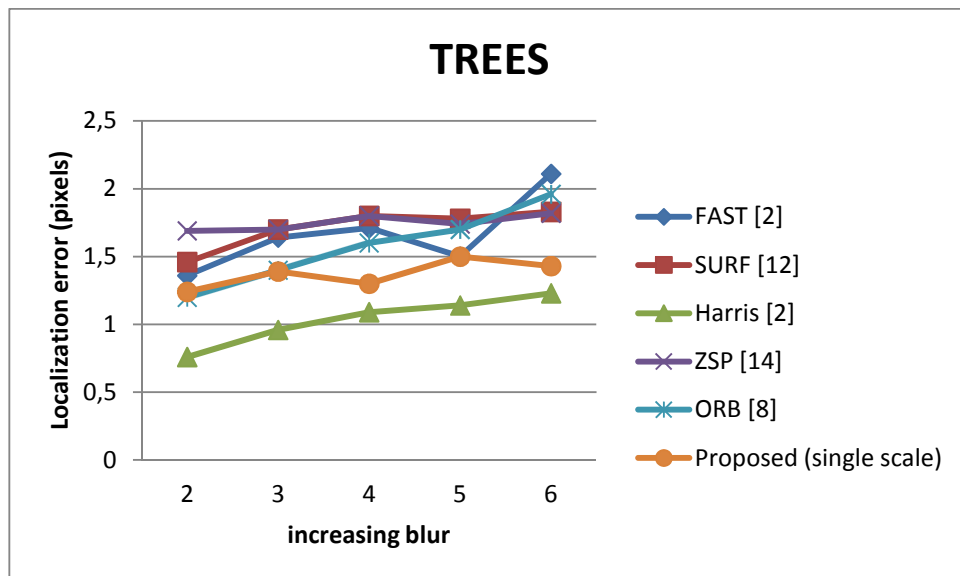


Figure 25 Localization error for Trees dataset (deformation type: blur)

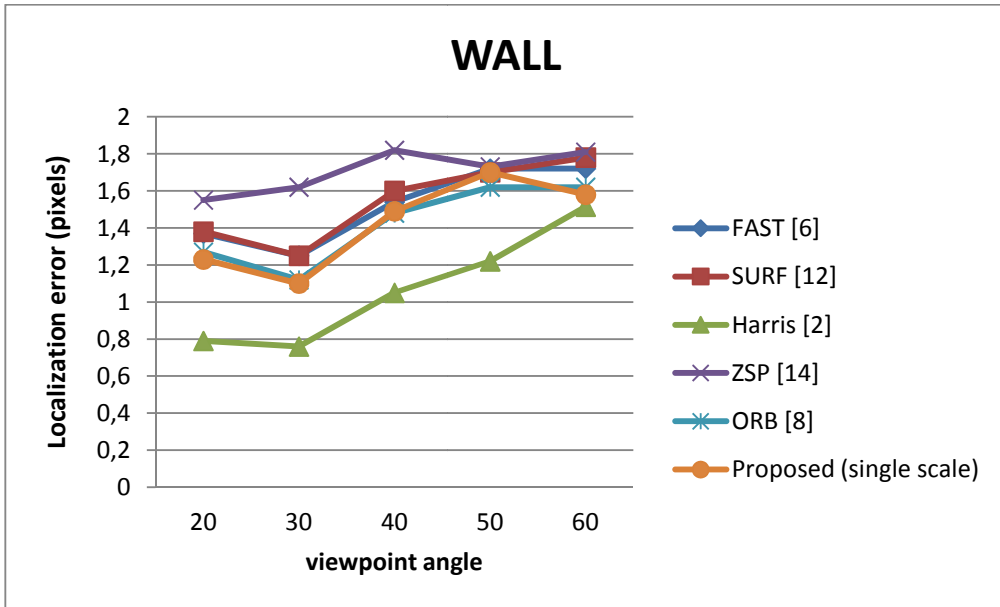


Figure 26 Localization error for Wall dataset (deformation type: affine)

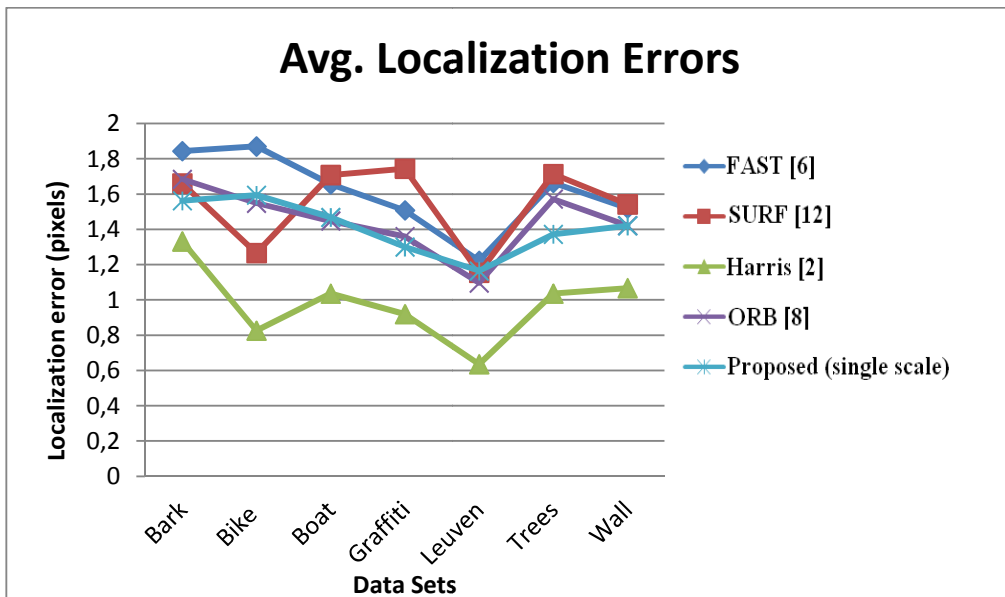


Figure 27 Avg. localization error

Figure 27 shows the average localization error results in different datasets. As can be observed from this figure, it is clear that proposed detector has better localization performance than FAST [6] and SURF [12]. This is because of the basic idea behind the proposed detection method where the pixel intensity comparisons are utilized up to the nucleus pixel.

2.4.3. Matching Score

If a point is repeatable by taking one of the images as reference, then projection of itself and its neighbors are compared with the reference image by using NCC. Calculation is achieved in the reverse direction and the average gives the matching score for that particular image pairs.

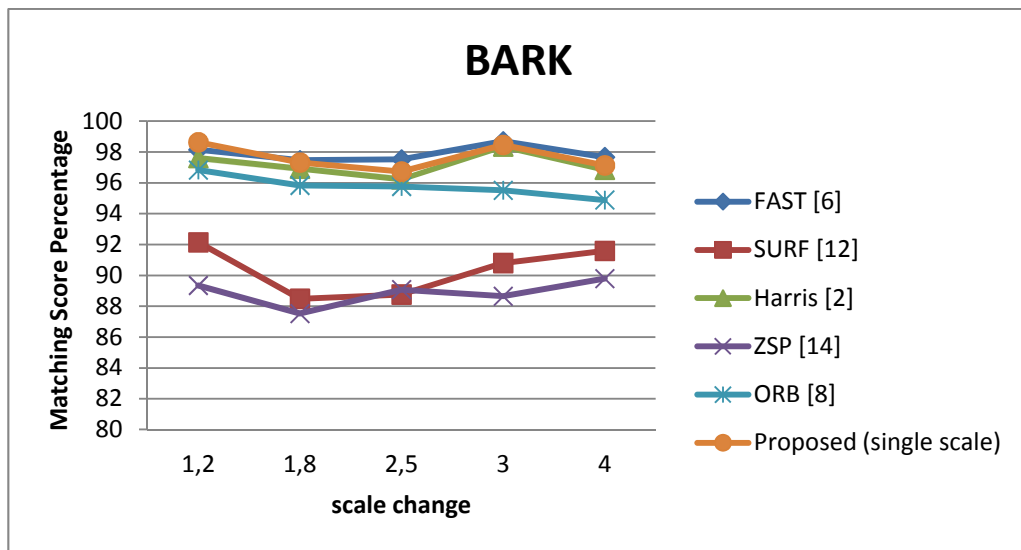


Figure 28 Matching score for Bark dataset (deformation type: rotation + scale)

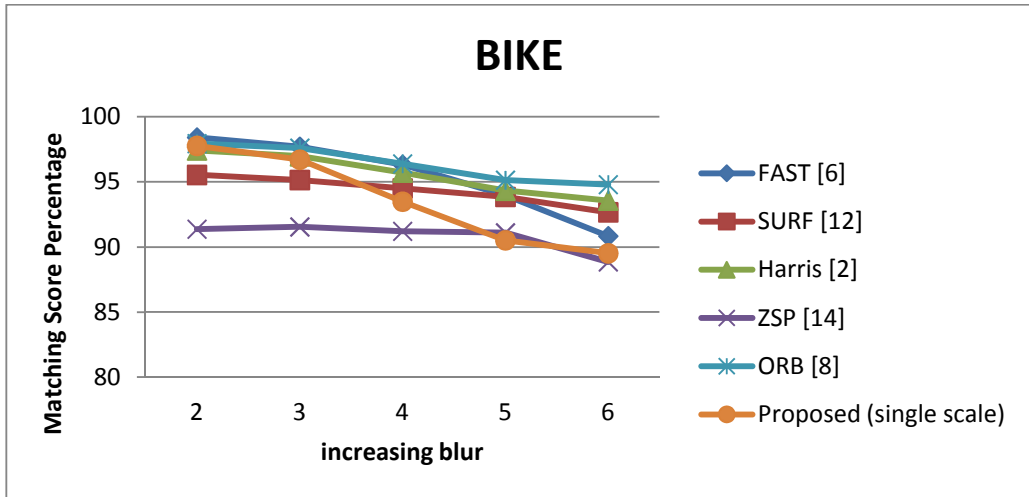


Figure 29 Matching score for Bike dataset (deformation type: blur)

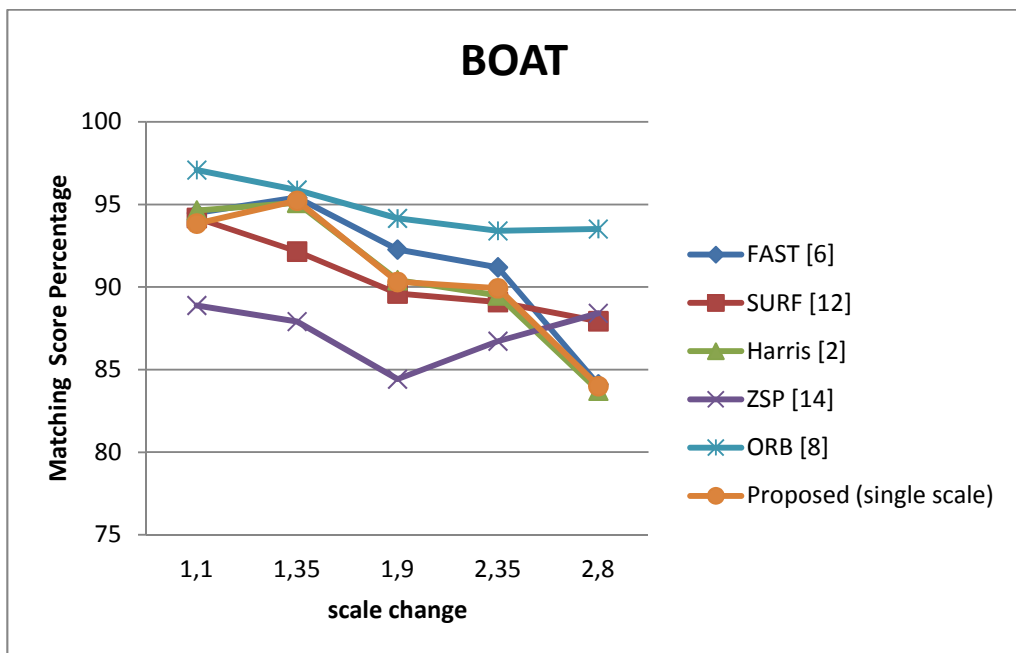


Figure 30 Matching score for Boat dataset (deformation type: rotation + scale)

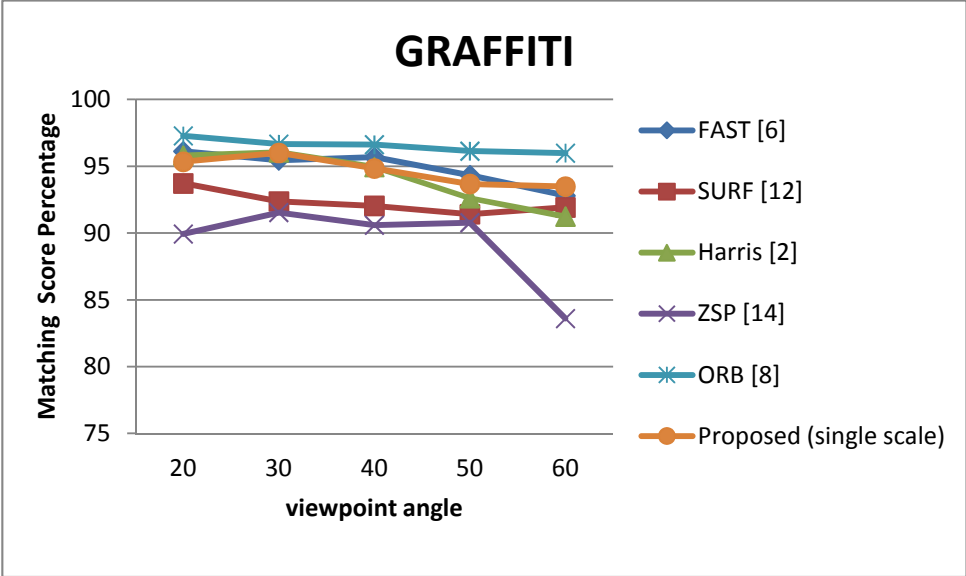


Figure 31 Matching score for Graffiti dataset (deformation type: affine)

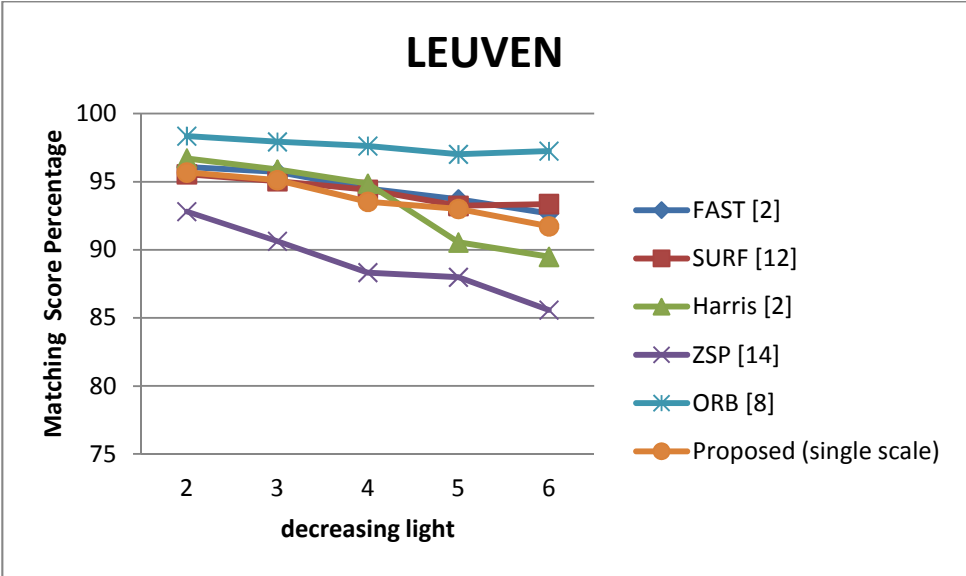


Figure 32 Matching score for Leuven dataset (deformation type: illumination)

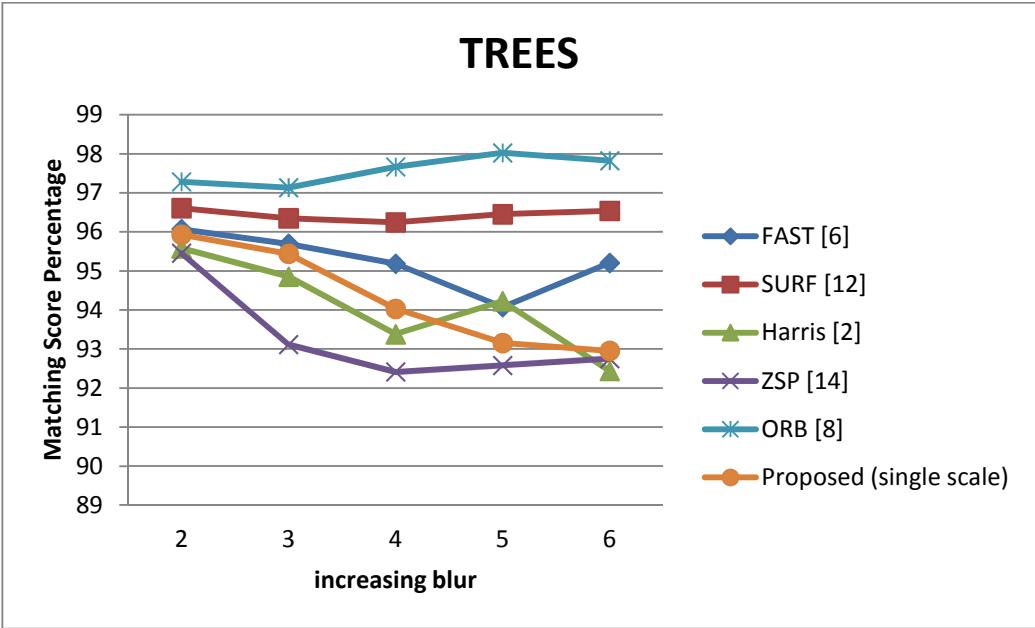


Figure 33 Matching score for Trees dataset (deformation type: blur)

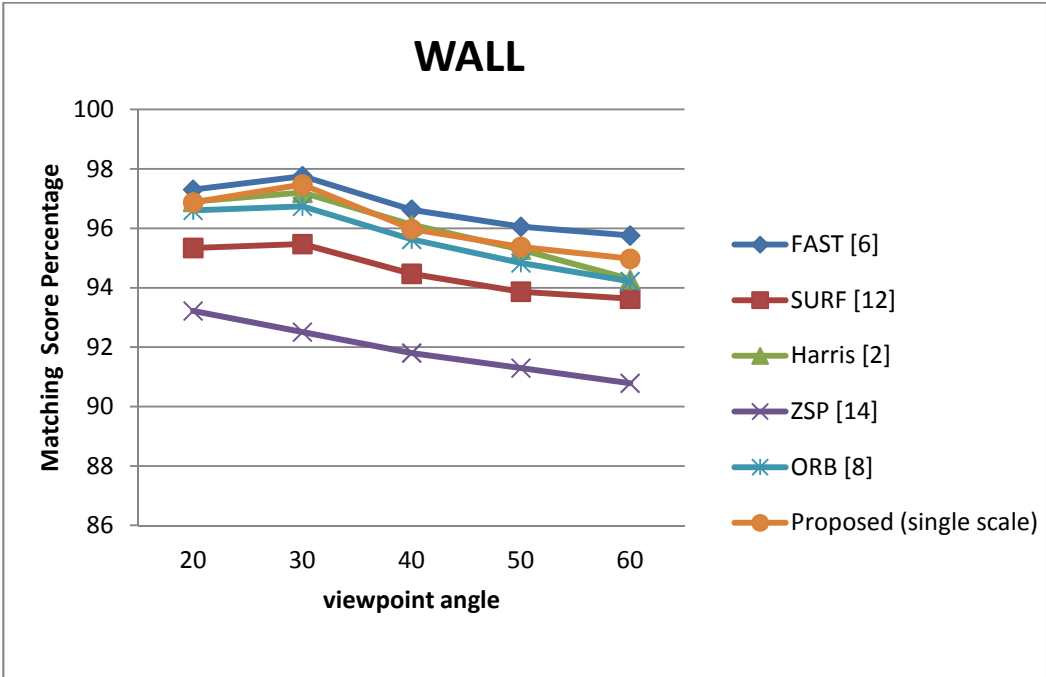


Figure 34 Matching score for Wall dataset (deformation type: affine)

2.4.4. Algorithm Specifications:

In order to compare the algorithms, some open source implementations have been used. The source code of the detection algorithm used for SURF, which is Fast-Hessian, is obtained from OpenCV 2.3 with 4 octaves and threshold minimum Hessian score is adjusted to make the number of detected points close to the number of points detected in other algorithms. Harris Corner Detection is also implemented in OpenCV in multi-scales as well as ORB. Zero Shift Points blob detector is implemented in C++. The source code of FAST algorithm is also from OpenCV 2.3, which is not the machine learning approach, but FAST-9-16 version. The proposed algorithm is implemented using OpenCV image loading and matrix entry accessing functions. Hence, the code is not fully optimized, as FAST, and the source code can be improved to make the algorithm faster.

It should be noted that, the proposed algorithm is executed on a webcam and a standard PC (2.8 GHz). A frame from the webcam with detected features is shown in Figure 35.

Table 2 shows the execution times of the algorithms for C++ implementations. The computation times are calculated using an average computation time score out of all of the datasets and these computation times are calculated a few hundred times and averaged.

Table 2 Execution times of the algorithms for 800x600 frames. For all of the compared algorithms, approximate number of detected points is kept around 1000. (In machine learning version of FAST, the computation time per a 768x288 frame is declared as 1.3 ms in case of the number of detected points is approximately 500 meaning that our implementation and FAST implementation have almost the same computation time).

	Multi Scale Harris[2]	Proposed (multi - scale)	Proposed (single scale)	SURF[12]	FAST[6]	ZSP[14]
Exec. Time per frame (ms)	72.2	9.2	3.5	140.1	10.5	920.4



Figure 35 A sample capture from the webcam. The diameter of the circles represents the scale of the detected points.

2.5. CONCLUSION

In this work, state-of-the-art interest point detection algorithms are compared as Speed Up Robust Features [12], ORB [8], Harris Corner Detector [2], Zero Shift Points [14], Features from Accelerated Segment Test [6] in terms of repeatability, localization error and matching score. Furthermore, a novel algorithm is also proposed.

In order to discuss performance of the algorithms, the algorithms which are tested in multi scale have better performance than others as FAST, proposed detector and its speed up version in the datasets in which the deformation type is blur. When the mobile applications are considered, multi scale implementations should have more computational

complexity than the ones in single scale. However, the camera might have some blur effect. Therefore, a sense of scale should be included in an efficient way for the utilized algorithms.

In the datasets Wall and Graffiti, which have the affine transformation, the proposed algorithm outperforms the best state-of-the-art methods in terms of repeatability, especially for the case with high changes in view point angle. The affine transformation rather than scale change is a crucial deformation for video sequences, if the pose of a camera is required to be calculated. Hence, the proposed method becomes advantageous for such a scenario.

In case of rotation and scale changes, the proposed algorithm prevails in terms of the trade-off between computation complexity and repeatability performance. This result is also important in augmented reality applications, such that an object should be recognized to augment a reality on top of the given scene. When the light condition is considered, almost all of the algorithms perform in the same level in terms of repeatability.

Once the localization error is analyzed, the proposed algorithm always performs better than FAST, while localization is also a crucial performance metric for pose estimation problems.

Finally, the matching score is another performance evaluation criterion which represents how well the given point resembles its correspondence in the other view of the same scene. All of the corner detection algorithms performs almost same, while blob-detector methods have a slight decrease in performance.

To conclude this chapter, the proposed algorithm can be used in mobile devices when the performances are considered, since there should be sufficient time to compute descriptors and match them in a video sequence in real-time.

CHAPTER 3

LOCAL INTEREST POINT DESCRIPTION METHODS

3.1. INTRODUCTION

Local interest point description is crucial; it is a fundamental part of many computer vision applications, such as wide baseline matching, object tracking, object recognition, augmented reality, and camera calibration. In order to describe an interest point, there are many existing methods that focus on the interest point location and its predefined neighborhood. By using this region of interest, such points can be described and matched according to the descriptor extraction method.

As technology in mobile devices has been improving sharply, the number of tasks that can be implemented in mobile devices is rising. Although these tasks are applicable in mobile world, the developers are aware of some restrictions related to resources, such as memory consumption or computation time. When these restrictions are taken into consideration, a local interest point descriptor should be computationally efficient and have less memory consumption.

In the literature, there are many feature description algorithms that have high performance in terms of precision, such as SIFT [9], SURF [12], GLOH [19] and DAISY [20]. In these algorithms, the common idea is computing the gradient information in a local patch in which floating point precision is required for representation of the local patches in terms of orientation gradients. Computations in floating point precision require relatively higher time during the calculation of the descriptor as well as its matching phase. In order

to avoid the floating point precision, floating point values are quantized so that there is zero or negligible loss in the performance [21], [22], [23]. Although such an approach is a promising idea to decrease the memory consumption and computation time in matching phase, calculation of those floating point numbers typically takes a long time with no possibility for real time implementations.

Both computation time reduction and high performance in terms of precision can be achieved up to a scale by using binary descriptors. One of the earliest local binary descriptors is proposed in BRIEF [24]. In this work, the main idea is approximating gradient information, which is usually preferred for interest point description. Instead of approximating floating points after description of features, it is proposed in [24] that one might approximate or replace gradient calculation by brightness intensity comparisons. Concatenation of such binary comparisons results with a binary string. Using this approximation, description computation, memory consumption and the calculation of the distance between descriptors are all reduced significantly.

3.2. RELATED WORK

3.2.1. Distribution based and differential based descriptors

Most of the popular local descriptors are based on calculation of histograms of a local patch to exploit different characteristics of the appearance or shape. In [25], Lazebnik *et. al.* propose a histogram based method in which two dimensions of these histograms are based on distance from the center point and pixel intensities. Zabih *et. al.* use rank and census transform to distinguish local image pairs [26] in which those transforms are based on the comparison of the center pixel intensity to its neighbor pixels. One of the expressive representations for 3D object recognition was introduced by Johnson *et. al.*[38] to exploit the range data information. This representation makes use of the point locations in a 3D interest point. This idea is later extended for images by [39]. In this description method, the distances from the interest point and pixel intensities are used for two dimensions of histograms. Nevertheless, scale-invariance and rotation invariance are not provided by the aforementioned algorithms.

In Scale Invariant Feature Transform (SIFT) [9], a keypoint is described computing gradient magnitudes and orientation in the region of interest, as shown in Figure 36. The whole procedure is performed in the estimated scale of the detected SIFT point. These gradient magnitudes are then weighted by a Gaussian window; the blue circle in Figure 36 indicates this window. Each of these gradient values is accumulated in orientation histograms (shown at right in Figure 36). The descriptor is generated by a vector containing the values of the orientation histogram entries, corresponding to occurrences of the arrows on the right side of Figure 36.

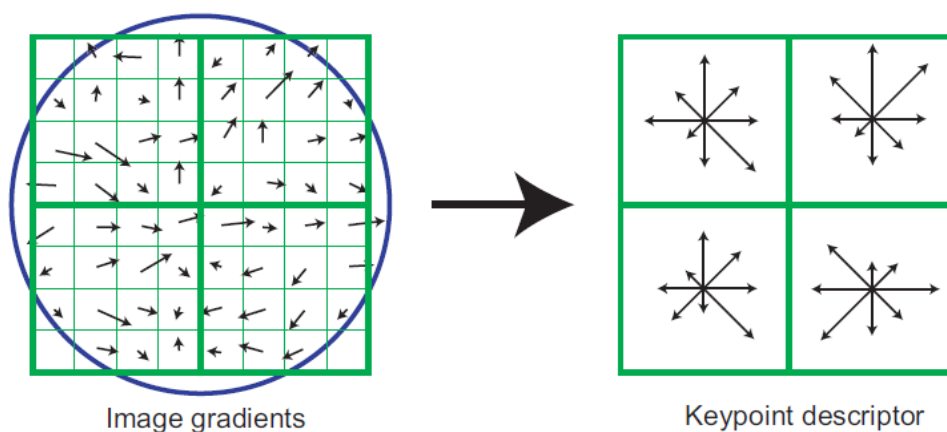


Figure 36 Illustration of SIFT descriptor calculation using orientation gradients.

Although SIFT [9], being rotation and scale invariant, is a very distinctive feature description technique, it is computationally quite inefficient to calculate all of the gradient information inside a local patch.

Hence, some coarse techniques to approximate gradient calculation are also proposed to reduce computation time, as in SURF [12]. In this work, the first step is finding a dominant orientation and then aligning the local patch according to that orientation. The oriented patch is split up into subregions (squares). Subregions are then used to calculate the Haar wavelet response in horizontal and vertical direction by the help of integral

image method to decrease computational burden. Similar to SIFT, all of the calculations including dominant orientation and histogram calculations are performed in the selected scale of SURF interest point. SURF technique selects its interest points in scale-space. An illustration is shown in Figure 37. As it can be observed from Figure 37, homogenous regions and regions with high frequency can be distinguished by the help of magnitudes of Haar wavelets. As in [9], after representing subregions by Haar wavelet responses, the descriptor vector is formed using the response entries of each sub region.

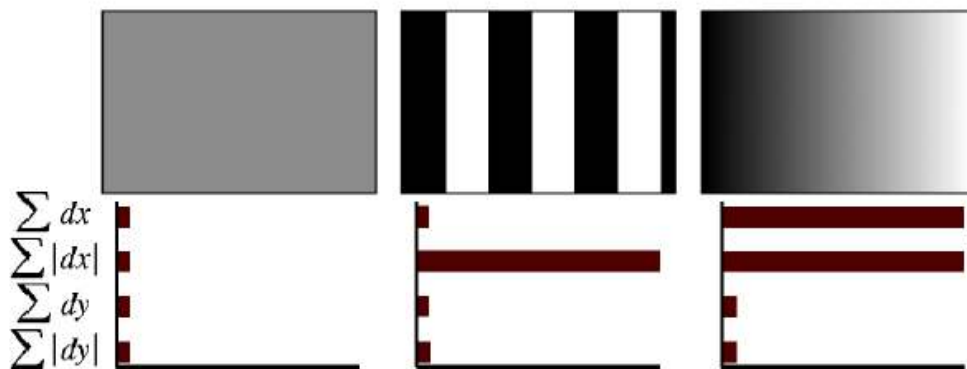


Figure 37 Illustration of SURF descriptor calculation.

Although SURF requires less computation time compared to SIFT, Bay *et. al.* [12] also argue that SURF outperforms SIFT in most cases. Therefore; if computational complexity is not a significant issue for a specific application, SURF and SIFT should be preferred for local description.

In GLOH [19], an extension of the SIFT descriptor is proposed to increase the robustness and distinctiveness of the descriptor. In this work, SIFT descriptor is computed in log polar grids instead of regular grids. In addition, the computed descriptor is also reduced by PCA. By modifying SIFT in these aspects; a slight improvement in the performances occurs. Nevertheless, the computational complexity is still a problem.

Following the exploitation of log-polar grid idea, Tola *et. al.* propose a local image descriptor [20] which is computationally efficient when it is compared to SIFT and SURF. In this work, the main idea is to sample some meaningful locations in a local patch and then utilize those points to calculate gradient and register this information into the descriptor. Such a sampling improves the performance quite significantly that state-of-the-art SIFT is outperformed by DAISY for dense depth reconstruction. The illustration of the sampling pattern is given in Figure 38.

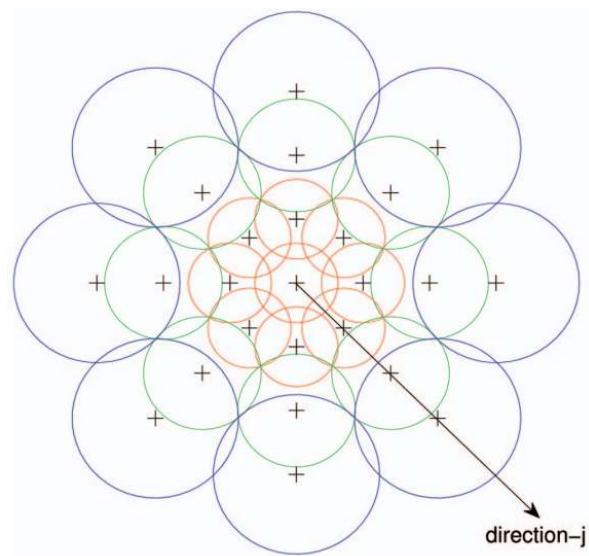


Figure 38 Illustration of DAISY sampling pattern.

In this figure, each circle represents a region whose radius is proportional to the standard deviation of the Gaussian kernels. The “+” signs are the sampling points and overlapping circular regions provides rotational robustness and smoothness. The gradients are calculated for the whole image space and their Gaussian filtered versions are saved for different variances. All of the calculated gradients of sampling point in different orientations are concatenated to extract the descriptor.

Takacs *et. al.*[44] propose a rotation invariant local feature description method based on the histogram of gradients. In this description method, gradient is calculated on a circular neighborhood of the interest point and these gradients are projected to a local radial coordinate system. This local radial coordinate system provides rotation invariance. After calculation of image gradients in different locations around the interest point, histograms of these gradients are constructed to represent the local interest point. Nevertheless, this method is not scale invariant and relatively less distinctive compared to SIFT, GLOH and DAISY.

One of the methods for representation of images is use of spatial-frequency of images. For instance, Fourier transform disintegrates the image into basis functions. Nevertheless, in their survey for local descriptors, Mikolajczyk *et. al.* [42] state that it is difficult to make use of these representations due to the fact that spatial relations between points are not explicit and basis functions are infinite. However, Gabor transform [40] is able to solve these problems, although a large number of filters are required. Gabor and wavelet transforms [41] are good candidates for texture classification with high computational complexity.

3.2.2. Intensity comparison based descriptors

When the aforementioned distribution and differential based local descriptors are considered, none of them are applicable for real-time applications due to their computational complexity. Since the aim of this work is to execute in real-time applications even by mobile devices, more efficient methods, such as local binary descriptors, are more appropriate for the attacked problem in this thesis. In this part, local binary descriptors will be discussed.

The earliest intensity based binary descriptor is proposed by Calonder *et. al.* [24]. In this work, approximation of gradient is used, instead of approximation of descriptors with floating point accuracy, since computation of floating numbers is still time-consuming. The formulation is as follows [24]:

$$\tau(\mathbf{p}, x, y) = \begin{cases} 1 & \text{if } \mathbf{p}(x) < \mathbf{p}(y) \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

where τ represents a test on patch \mathbf{p} of size $S \times S$ and $\mathbf{p}(\mathbf{x})$ is the pixel intensity in a smoothed version of \mathbf{p} at location $\mathbf{x} = (u, v)^T$. This test is performed in n_d different location pairs resulting in n_d -dimensional binary string which can also be written [24] :

$$f(\mathbf{p}) = \sum_{1 \leq i < j \leq n_d} 2^{i-1} \tau(\mathbf{p}, x_i, y_i) \quad (3.2)$$

In this binary descriptor (BRIEF), the main contributions are based on the observation that intensity comparison based binary descriptor can be used as a local descriptor and the comparison location pattern should be random. It is also claimed that the random patterns are useful than regular patterns. This hypothesis is supported by trying different patterns in the local patch. However; a regular or a deterministic pattern compared to the random ones is not sufficient to conclude that random patterns are more useful than the regular ones. This being the case, a more efficient (less number of comparisons) and legitimate binary pattern, which is deterministic, is proposed in this thesis.

Before going further detail in regular patterns, it is also worth to discuss random binary descriptors which is inferred using variety of comparative results [8]. In ORB [8], the same idea of intensity comparison is used as in [24]. The advantage of ORB descriptor is that it is rotation invariant, since a dominant orientation is calculated and the tests are performed according to that orientation.

For orientation calculation, Intensity Centroid method [27] is used in ORB. In this method, the first moments of an image is used, while these moments are defined as follows [27],

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \quad (3.3)$$

By using these moments, the centroid is computed as,

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \quad (3.4)$$

Using these moments, orientation of the patch can also be calculated as,

$$\theta = \text{atan2}(m_{01}, m_{10}) \quad (3.5)$$

In (3.5), $\text{atan2}(\cdot)$ is quadrant –aware version of tangent inverse. In [8], it is also shown that Intensity Centroid method is more robust against noise than gradient based orientation calculations. For each interest point, the orientation is calculated by the method above. The orientation angle is used to project the sampling points for comparison tests to the dominant rotation. Therefore, there is no need to rotate the whole patch, since rotating sampling pattern is more efficient. The formulation of this technique [8] is given in (3.6) and (3.7).

$$S = \begin{pmatrix} x_1, \dots, x_n \\ y_1, \dots, y_n \end{pmatrix} \quad (3.6)$$

In (3.6), (x_i, y_i) represents test location pairs. By exploiting the orientation angle that is calculated previously, R_θ is generated. After this procedure S_θ is generated as follows,

$$S_\theta = R_\theta S \quad (3.7)$$

S_θ in (3.7) is the rotated version of the binary pattern. For the sake of simplicity and reducing the computation time, angles are discretized by $2\pi/30$ and a look up table is generated for fast computation.

Moreover, they have learnt the intensity comparison which gives the closeness of the mean to 0.5. In other words; an ideal comparison result, which can be a ‘1’ or a ‘0’, should not have a mean close to ‘0’ or ‘1’. This means that this comparison almost always has the same value. This being the case, this comparison carries no information. Therefore; in ORB [8], the major claims about the sampling pattern are as follows: Each test should be independent from each other, a very high variance of binary comparisons should result and the mean of these comparisons should be approximately equal to 0.5. In order to select high quality intensity comparisons, learning is implemented to select the important comparisons out of 200K combinations. The selection procedure is as follows: They first choose 300K interest points detected by using FAST detector in multi-scale. All possible binary tests are enumerated drawn from 31x31 patches. The algorithm for constructing the ideal binary pattern is as follows:

- 1) Run each test against all training patches
- 2) Put the test into an order according to their mean value (closest to 0.5), forming the vector T
- 3) Greedy Search is applied as follows:
 - a. Put the first test into the result vector R and remove it from T
 - b. Take the next test from T , and compare it against all tests in R . If there is a correlation with any of the tests in R , eliminate it. Else add it to R .
 - c. Repeat the tests until there are 256 tests in R .

The resulting binary pattern that is obtained from training data is given in Figure 39.

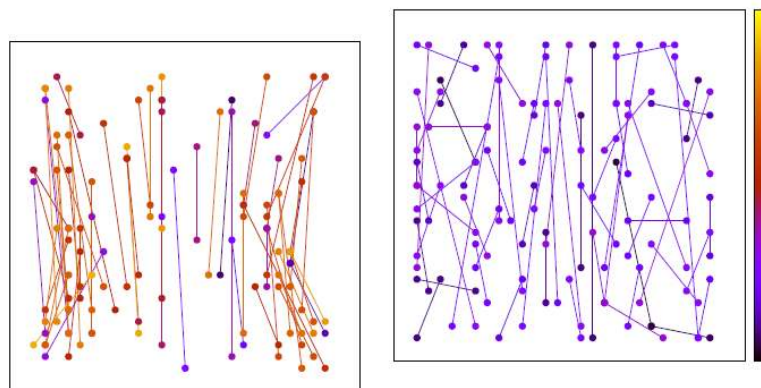


Figure 39 ORB binary pattern. Left: High variance under orientation. Right: Reduced Correlation.

After the third procedure of the binary pattern construction, selected comparisons give different values from each other in the training dataset due to the decorrelation process. Figure 39 shows the uncorrelated tests on the right. As it can be observed from this figure, the pattern seems to be non-random and there is a majority of vertical tests on the patches. This observation is due to utilizing FAST corner detector and aligning it in the dominant orientation where corner point will be possibly showing up or bottom. Hence, it is intuitive that the maximum correlation is obtained as a result of vertical comparisons. However, the inference of the ORB pattern lacks the ground truth information. In the

learning procedure of ORB [8], there is no such information that the comparisons that have high variance or etc. are giving the same results in the true correspondences and different results in the false correspondences. Since this is not accomplished, the ORB is quite arguable as the best learnt binary pattern for local patch description.

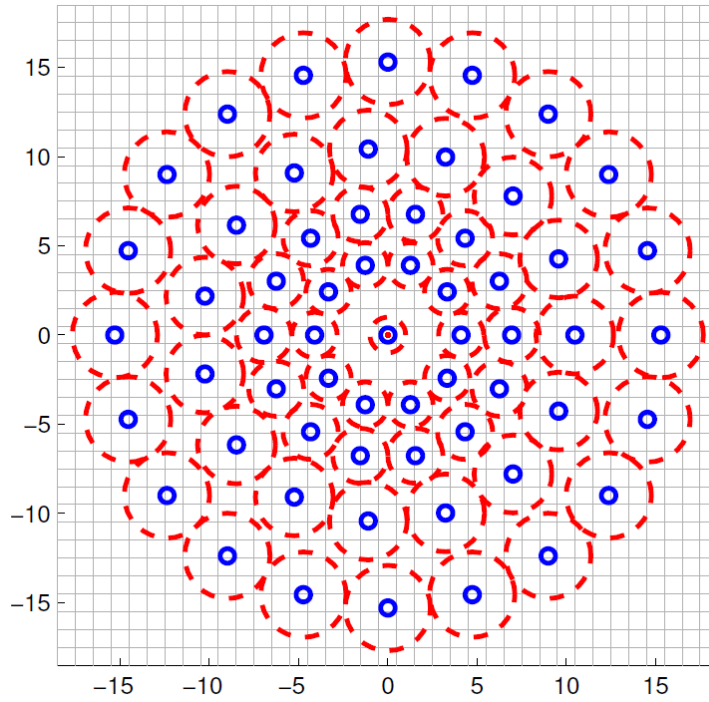


Figure 40 Binary pattern of BRISK

Once the powerful pattern of DAISY and fast computation opportunity of BRIEF descriptor are considered, both of the methods are fused in Leutenegger's work [16]. The proposed binary pattern, as well as the fundamental idea in BRISK, are illustrated in Figure 40. In this figure, there is a sampling pattern with $N=60$ points. The small blue circles represent the sampling locations. The larger red dashed circles denote standard deviation value of the Gaussian kernels. These kernels aggregate brightness intensity information in the sampling points. In other words, those sampling points are smoothed to be robust to noise. After this procedure, the binary combinations of sampling points constitute a set. The distant pairs are used to estimate the orientation of the patch. The short pairings are used to build up the binary descriptor using intensity comparisons as in

[24] and [8]. Although this seems to be an elegant idea to exploit both DAISY and ORB, the computation complexity is still above ORB descriptor due to aggregation of intensity values.

3.3. PROPOSED LOCAL BINARY DESCRIPTOR

When all of the state-of-the art local binary descriptors are considered, as well as quite distinctive SURF, DAISY algorithms, which make use of gradient information in the patch, it is worth experimenting a DAISY-type pattern and the first binary descriptor idea in [24].

To summarize, DAISY is exploiting a circular pattern for gradient histograms, which is more distinctive than sampling from rectangular grids. BRIEF is approximating the gradient calculation using intensity comparisons. In this work, it is proposed to approximate the DAISY-type descriptor using the idea of BRIEF.

The binary sampling pattern is illustrated as in Figure 41.

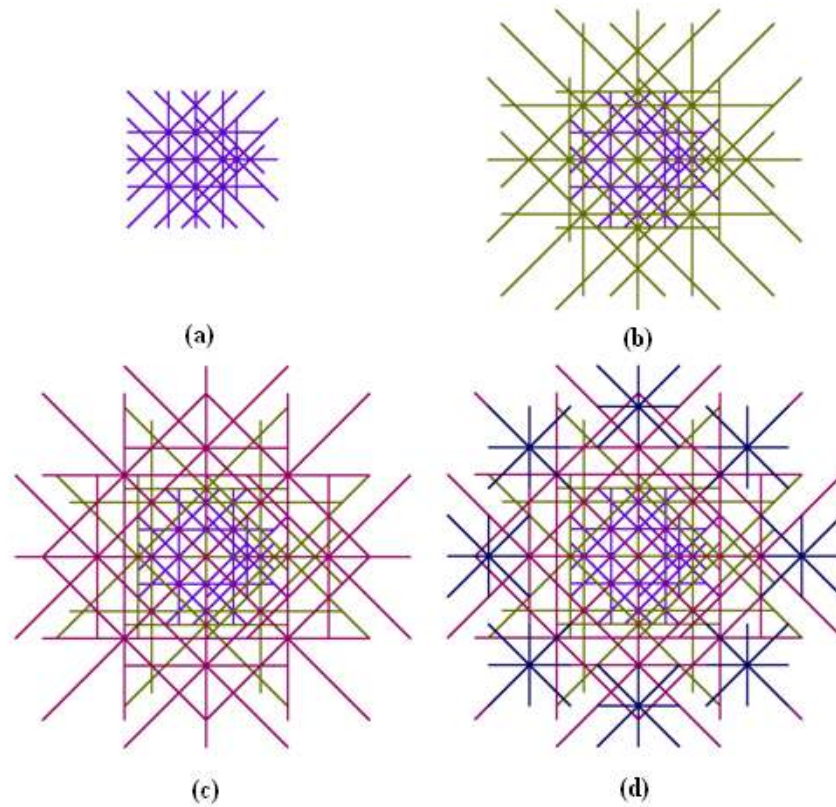


Figure 41 Proposed binary pattern illustration. The end points of the lines constitute a comparison pair and the middle points of these lines are sampling points in the pattern. In proposed descriptor, there are four nested circles and the sampling points are chosen sparsely on the circles. (a) first nested circle (b) first and second nested circles (c) first, second and third nested circles (d) final descriptor pattern

In DAISY [20], the orientation gradient histograms are calculated in the sampling pattern. In the proposed method, intensity comparisons are evaluated at sampling points in 8 directions corresponding to 4 comparisons per sampling location. In Figure 41, the end points of the lines show the pixel locations to operate intensity comparisons. One of the main advantages of using such a configuration is that the orientation gradient histogram information is compressed in the diametrically opposed end point comparisons around the sampling points. Moreover, using such a circular overlapping grid provides rotational robustness.

Through experimentation, it is argued that the proposed 128 bit descriptor has approximately the same performance as 256-bit ORB descriptor. Hence, the best

parameters are selected for the proposed 128 bit descriptor. The parameter tests and their results are explained in the experimental results part of the work.

Parameter selection is performed by a distinctiveness criterion which measures the ratio of distinctive false and true descriptor pairs. Hence, the ground truth information, which is not used in ORB, is taken into account.

Scale invariance of proposed local binary descriptor is satisfied by a similar idea of ORB [8]. For a given image, interest points are detected in multi scale and the number of detected points in a particular scale is determined proportional to the number of pixels in that image scale without eliminating keypoints in scale-space. Utilization of such a multi-scale method has the advantage of computational complexity, since there is no need to calculate a score for scale-space elimination. Furthermore, the same points are detected in different scales. This case makes more likely to find the true correspondences by the same interest point in different scales. When scale-space elimination is used in the detection phase, the noisy nature of local binary descriptors might cause matching of the unique interest point in the selected scale to fail.

3.4. EXPERIMENTAL RESULTS

During the experiments, ORB and the proposed descriptor are compared in terms of distance histograms and precision recall curves. For the sake of fairness, the idea in ORB is used as follows: The detected points in multi-scale are ordered according to their Harris score. Moreover, ORB interest point detection is used in distance histograms for fair comparison of descriptors. In precision-recall curves, SURF, ORB and proposed descriptor versions are also compared in terms of their own detection methods. Definitions of inlier-outlier, distance histograms and precision-recall used in experiments are given below.

Inlier/outlier: Let $X_1 \in View1$ and $X_2 \in View2$. Homography, H , between the images is provided as a ground truth. Projection of X_1 on to $View2$ is $X_2' = H X_1$. If there is a point X_2 such that $|X_2' - X_2| < R$, then point X_1 and X_2 are said to be repeatable points, which

constitute inlier correspondence. If the condition is not satisfied, then that correspondence is said to be an outlier.

Distance histogram: Distance histograms are used as a distinctiveness measure as in [24] and [8]. Interest points of two views for the same scene are described and matched (using brute-force-matching and Hamming distance). If a match is a true correspondence (inlier), then the distance between matched descriptors is added to the inlier distance histogram. Otherwise, it is accumulated in the outlier histogram. If the inlier and outlier histograms are well-separated and both have less variance, it indicates that the descriptor is a distinctive one. Therefore, distance histograms could be used as a distinctiveness measure. In this work, 256 bit proposed local binary descriptor and 256 bit ORB are compared in terms of distance histograms.

Precision and Recall: Assume there are two images and the corner points previously detected on these images are described and matched. At the end of the matching procedure, let N denote *number of correspondences*. To use those correspondences for homography estimation, most of the correspondences should be correct. Therefore, there should be a distance threshold T , which ignores the correspondences by larger distance value than itself. Let N_{thr} denote the number of correspondences after thresholding. The true correspondences out of N_{thr} correspondences are denoted by N_{true} . Precision and recall formulas are given in (3.8) below.

$$Precision = \frac{N_{true}}{N_{thr}} \quad \text{and} \quad Recall = \frac{N_{true}}{N} \quad (3.8)$$

Precision and recall curves are plotted using different threshold values for ORB, SURF, proposed local binary descriptor and its different versions with different length. Proposed local binary descriptor has different bit length versions as 64 bit, 128 bit and 256 bit.

Parameter selection of the proposed descriptor: As it can be seen from Figure 42, blue stars are the sampling points located on the circles. The radius differences between those circles are C_k 's and the pixel intensity comparisons are achieved in the diametrically opposed points of the comparison circles as illustrated with R_k radius. Hence, the best parameter search is performed by using our 128-bit proposed descriptor, since it performs slightly worse than its 256-bit version of the proposed descriptor. Moreover, the

parameters here are C_1 and R_1 , and all C_k 's and R_k 's are fixed at R_1 and C_1 , respectively. The reason for not varying R_k 's and C_k 's is that in the preliminary experiments it is observed that using different R_k 's or C_k 's does not improve the performance or slightly decrease the precision-recall rates.

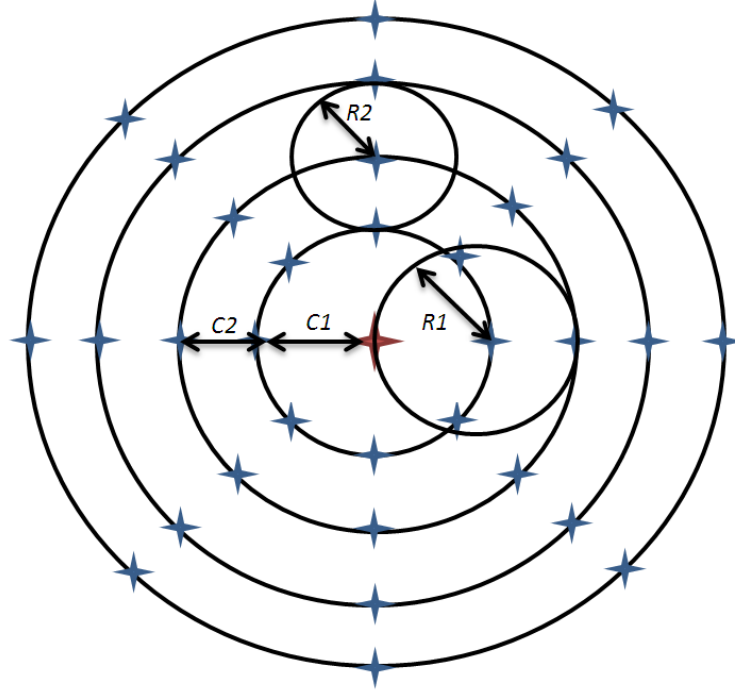


Figure 42 Parameters of the proposed descriptor. C_1 is the distance of the circles and R_k 's are radius of the comparison circles.

In order to compare the binary patterns with different parameters, distance histograms are utilized. The mean values (μ) and standard deviations (σ) of the inliers and outliers are calculated. Fisher's criterion in Equation (3.9) is used to analyze which of the patterns are most distinctive:

$$J = \frac{|\mu_1 - \mu_2|}{\sqrt{\sigma_1^2 + \sigma_2^2}} \quad (3.9)$$

In (3.9), σ_1 and σ_2 are variances of inliers and outliers, respectively and μ_1 and μ_2 are mean values of the inliers and outliers. If J becomes maximum for a specific binary pattern, it

indicates that this binary pattern is the most distinctive one for the dataset under consideration.

Dataset properties: The tests are performed on the datasets for which the homography between different viewpoints of the same scene are provided as ground truth, since the repeatability tests for feature detectors are fulfilled. As can be seen from Table 3, deformation types are given for different datasets with visual examples.

Table 3 Dataset properties with visual examples.

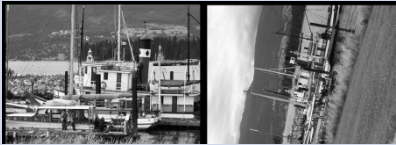



Dataset Name	Deformation Type	Visual Example
Bark, Boat	Zoom + Rotation Change	
Bikes, Tree	Blur Change	
Graffiti, Wall	View Point Change	
Leuven	Light Change	

Table 4 Fisher's criterion results for different R_k and C_k parameters. The parameter combinations giving maximum Fisher's score are shown in bold.

BARK							BIKES						
$C_1 \backslash R_1$	$R_1=2$	$R_1=3$	$R_1=4$	$R_1=5$	$R_1=6$	$R_1=7$	$C_1 \backslash R_1$	$R_1=2$	$R_1=3$	$R_1=4$	$R_1=5$	$R_1=6$	$R_1=7$
$C_1=2$	0.472	0.469	0.469	0.467	0.460	0.449	$C_1=2$	1.077	1.052	1.020	0.986	0.955	0.923
$C_1=3$	0.443	0.440	0.446	0.442	0.443	0.443	$C_1=3$	0.936	0.949	0.958	0.963	0.963	0.959
$C_1=4$	0.437	0.434	0.432	0.431	0.431	0.431	$C_1=4$	0.960	0.964	0.967	0.970	0.971	0.971
$C_1=5$	0.424	0.421	0.418	0.416	0.415	0.414	$C_1=5$	0.967	0.966	0.967	0.969	0.970	0.971
$C_1=6$	0.409	0.405	0.402	0.400	0.399	0.398	$C_1=6$	0.965	0.962	0.962	0.962	0.963	0.963
$C_1=7$	0.393	0.389	0.387	0.385	0.383	0.382	$C_1=7$	0.958	0.954	0.953	0.953	0.953	0.953
BOAT							GRAFFITI						
$C_1 \backslash R_1$	$R_1=2$	$R_1=3$	$R_1=4$	$R_1=5$	$R_1=6$	$R_1=7$	$C_1 \backslash R_1$	$R_1=2$	$R_1=3$	$R_1=4$	$R_1=5$	$R_1=6$	$R_1=7$
$C_1=2$	0.593	0.582	0.566	0.551	0.534	0.517	$C_1=2$	0.718	0.701	0.680	0.654	0.628	0.602
$C_1=3$	0.521	0.527	0.530	0.532	0.532	0.529	$C_1=3$	0.608	0.614	0.616	0.615	0.612	0.605
$C_1=4$	0.528	0.529	0.530	0.531	0.531	0.530	$C_1=4$	0.606	0.606	0.607	0.606	0.604	0.601
$C_1=5$	0.527	0.526	0.530	0.526	0.526	0.526	$C_1=5$	0.598	0.597	0.596	0.595	0.594	0.592
$C_1=6$	0.522	0.520	0.530	0.519	0.519	0.519	$C_1=6$	0.588	0.586	0.584	0.583	0.582	0.581
$C_1=7$	0.515	0.513	0.530	0.511	0.511	0.511	$C_1=7$	0.577	0.575	0.574	0.573	0.572	0.571
LEUVEN							TREES						
$C_1 \backslash R_1$	$R_1=2$	$R_1=3$	$R_1=4$	$R_1=5$	$R_1=6$	$R_1=7$	$C_1 \backslash R_1$	$R_1=2$	$R_1=3$	$R_1=4$	$R_1=5$	$R_1=6$	$R_1=7$
$C_1=2$	1.090	1.061	1.024	0.984	0.946	0.908	$C_1=2$	0.865	0.859	0.843	0.820	0.794	0.767
$C_1=3$	0.922	0.936	0.947	0.952	0.951	0.945	$C_1=3$	0.772	0.779	0.785	0.786	0.783	0.776
$C_1=4$	0.945	0.950	0.954	0.958	0.960	0.960	$C_1=4$	0.773	0.774	0.775	0.776	0.775	0.772
$C_1=5$	0.955	0.955	0.957	0.960	0.962	0.963	$C_1=5$	0.766	0.763	0.762	0.761	0.759	0.758
$C_1=6$	0.957	0.955	0.956	0.957	0.960	0.961	$C_1=6$	0.751	0.747	0.745	0.744	0.743	0.741
$C_1=7$	0.955	0.952	0.951	0.952	0.953	0.955	$C_1=7$	0.735	0.732	0.730	0.728	0.727	0.726
WALL													
$C_1 \backslash R_1$	$R_1=2$	$R_1=3$	$R_1=4$	$R_1=5$	$R_1=6$	$R_1=7$	$C_1 \backslash R_1$	$R_1=2$	$R_1=3$	$R_1=4$	$R_1=5$	$R_1=6$	$R_1=7$
$C_1=2$	0.763	0.751	0.738	0.719	0.699	0.677	$C_1=2$	0.677	0.679	0.681	0.681	0.679	0.675
$C_1=3$	0.677	0.679	0.681	0.681	0.679	0.675	$C_1=3$	0.672	0.670	0.670	0.669	0.667	0.664
$C_1=4$	0.672	0.670	0.670	0.669	0.667	0.664	$C_1=4$	0.657	0.653	0.651	0.650	0.648	0.646
$C_1=5$	0.657	0.653	0.651	0.650	0.648	0.646	$C_1=5$	0.639	0.635	0.633	0.631	0.630	0.629
$C_1=6$	0.639	0.635	0.633	0.631	0.630	0.629	$C_1=6$	0.622	0.619	0.616	0.614	0.613	0.611
$C_1=7$	0.622	0.619	0.616	0.614	0.613	0.611	$C_1=7$						

Fisher's criterion comparison results shows that dense sampling and shorter distance comparisons make the performance better. The optimum values are $C_1=2$ and $R_1=2$ for all of the datasets. In the remaining part of the thesis, $C_k=3$ and $R_k=3$ are used, since nearby comparisons might lead to a sensitivity to noise in real-time applications.

Table 5 Fisher's Criterion comparison of ORB [8] and the proposed pattern

	BARK	TREES	LEUVEN	GRAFFITI	BOAT	BIKES	BARK
Proposed (256 bit)	0.74	0.87	1.12	0.44	0.60	1.08	0.46
ORB [8]	0.68	0.80	1.09	0.60	0.57	1.02	0.34

As soon as the distance histogram comparisons between 256 bit proposed descriptor and 256 bit ORB are analyzed (see Table 5), the proposed binary pattern has more separate histograms of inliers and outliers than those of ORB, since Fisher's criterion scores are higher in the proposed pattern than ORB. The detailed visualization of these histograms is shown in Appendix A.

The next experimental results present the recall-precision curves of the algorithms. In all of the curves, precision and recall percentages are normalized to 1.

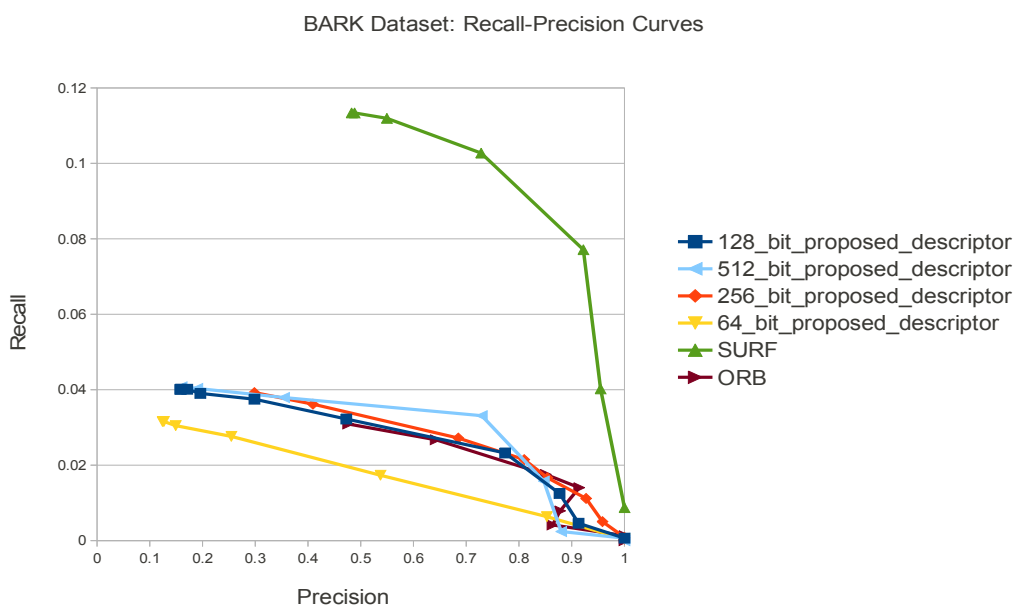


Figure 43 Recall-Precision Curves for Bark Dataset

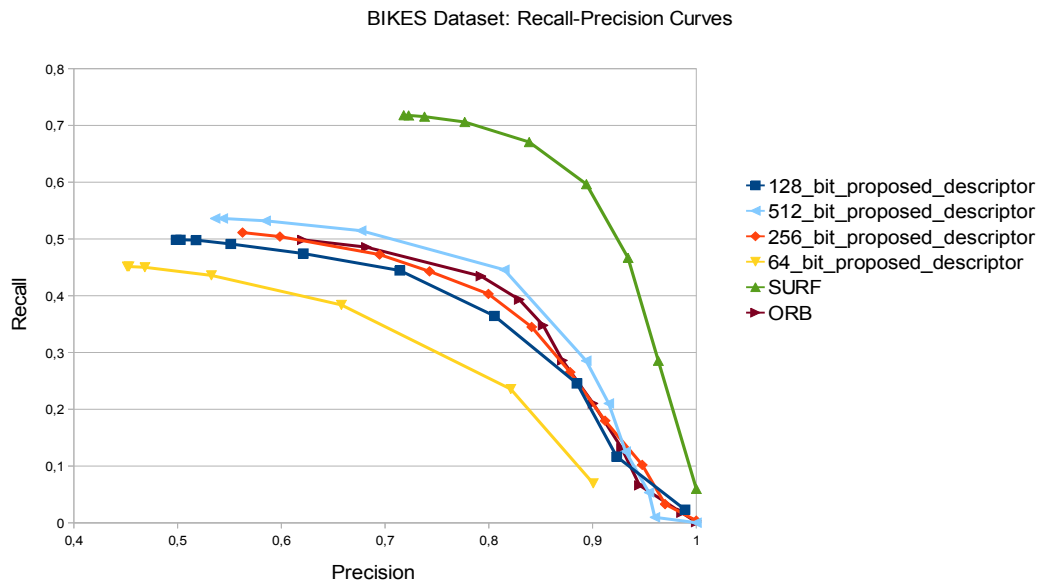


Figure 44 Recall-Precision Curves for Bikes Dataset

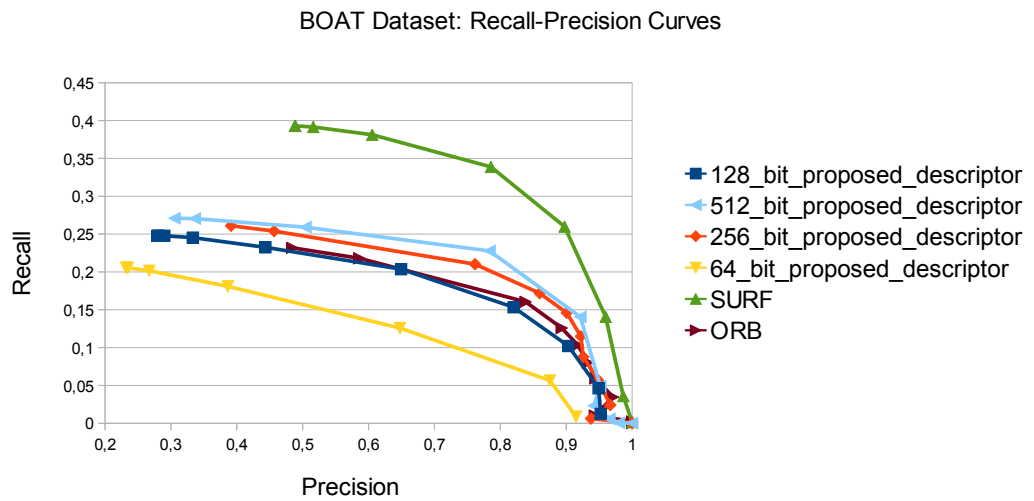


Figure 45 Recall-Precision Curves for Boat Dataset

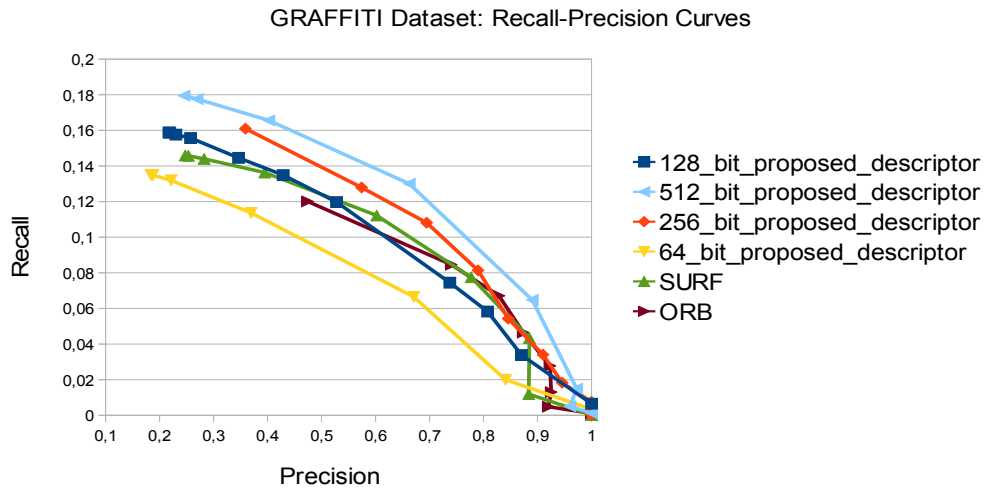


Figure 46 Recall-Precision Curves for Graffiti Dataset

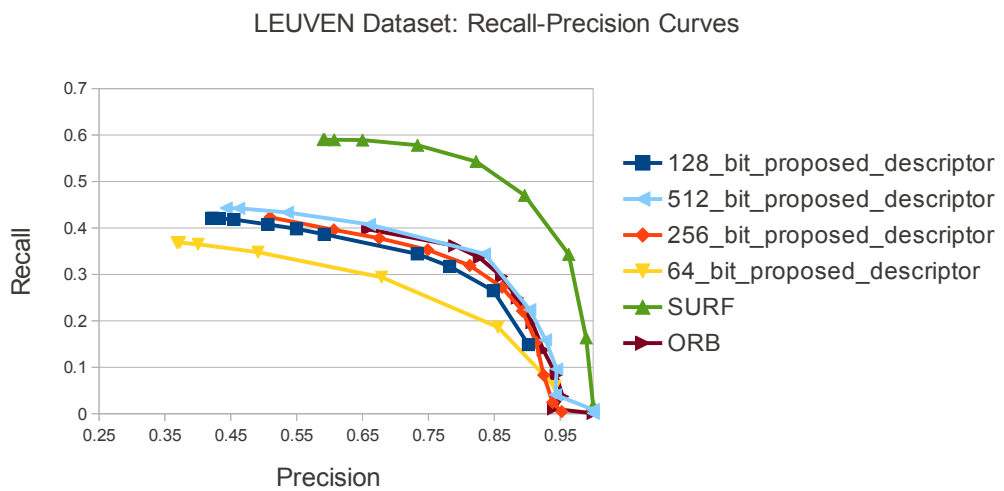


Figure 47 Recall-Precision Curves for Leuven Dataset

TREES Dataset: Recall-Precision Curves

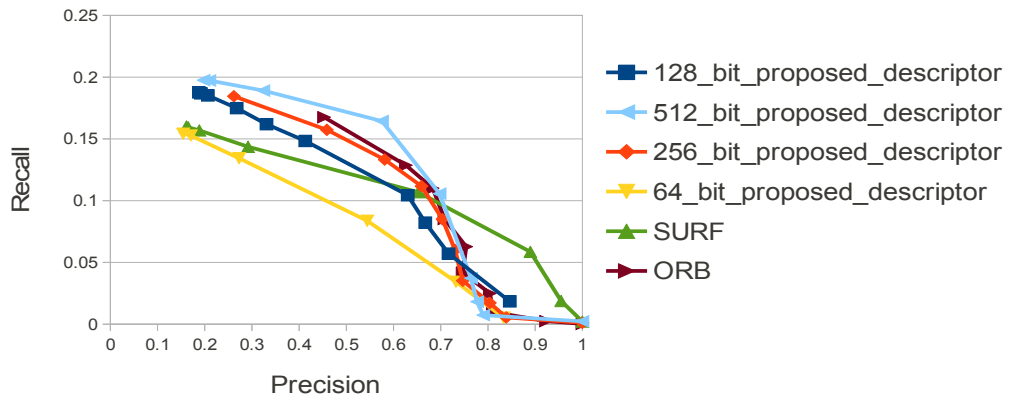


Figure 48 Recall-Precision Curves for Trees Dataset

WALL Dataset: Recall-Precision Curves

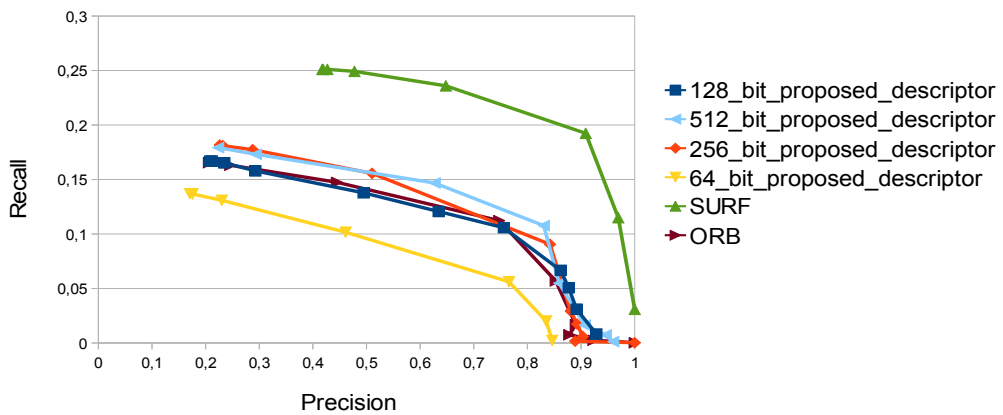


Figure 49 Recall-Precision Curves for Wall Dataset

As it can be observed from Figure 43 to Figure 49, the proposed binary pattern for different bit lengths, 256 bit ORB and SURF are compared. In almost all of the datasets, the 128 bit version of the proposed binary pattern has almost the same performance as

256 bit ORB does in terms of precision-recall curves. Hence, the proposed 128 bit descriptor is used in our real-time implementations, since it is easier to calculate when compared to 256 bit and 512 bit versions and has a slight performance decrease against 256 bit and 512 bit versions. This being the case, it can be claimed that the proposed descriptor pattern saturates at the bit length of 128 approximately.

When 1K keypoints are detected and their descriptors are extracted, description extraction time for 1K keypoints is computed as approximately 150 ms by the SURF algorithm. The parameters of SURF descriptor are selected as default in OpenCV implementation (4 octaves and 2 layers). Description extraction time for 256 bit local binary descriptor is computed as approximately 20 ms for 1K keypoints. Once those computation times are compared, powerful computational efficiency of local binary descriptors arises and it is convenient to use local binary descriptors for real time applications.

CHAPTER 4

APPROXIMATE NEAREST NEIGHBOUR FOR DESCRIPTOR MATCHING

4.1. INTRODUCTION

In application areas, in which local feature descriptors are used, one of the main goals is matching descriptors of different views of the same scene. In order to match descriptors, the fundamental and ideal method is linear search since linear search finds the closest descriptor in the descriptor database to the query descriptor.

Assume there is N number of d -dimensional descriptors in the system database. A query vector q asks the nearest descriptor out of N descriptors in the d -dimensional descriptor space. Linear search is the exact answer to find the nearest vector to query descriptor out of the descriptors in the system database and it can be defined as computing the distance from the query point to every point in the database. This has a computational complexity $O(Nd)$ where N and d are number of descriptor vectors in the database and the dimension, respectively.

Linear search does not seem to be feasible for large database applications and the cases in which higher dimensional descriptor vectors are used. Dimensionality reduction is one of the methods to decrease computation time of linear search. Nevertheless, this is not appropriate for the problem in this thesis, since binary descriptors are required for real-time application of the problem and dimensions of binary vectors cannot be reduced. Moreover, use of descriptors which are not binary vectors is computationally inefficient.

At that point, approximate nearest neighbor search can be accepted as a solution to decrease the computation time of search. Definition of nearest neighbor search and approximate nearest neighbor search are given as follows:

$$X^* = \mathit{arg} \min_{X \in D_N} \mathbf{p}(X, q) \quad (4.1)$$

where $D_N = \{X_1, X_2 \dots X_N, \}$ is a set of descriptors in the dataset, X^* is the *nearest neighbor* to q and \mathbf{p} is the distance metric for the descriptor space.

$$\mathbf{p}(X_{ANN}, q) < (1 + \Delta) \mathbf{p}(X^*, q) \quad (4.2)$$

where Δ stands for any positive real number and X_{ANN} is the *approximate nearest neighbor* to q .

Hamming distance can be used in binary space for approximate nearest neighbor search. For binary descriptors, Hamming distance between vectors \mathbf{x} and \mathbf{y} is equal to number of ones in $\mathbf{x} XOR \mathbf{y}$.

State-of-the-art approximate nearest neighbor search is discussed in the related work part.

4.2. RELATED WORK AND PROPOSED METHOD

There are many indexing methods for approximate nearest neighbor problem. Nevertheless, many of them are not appropriate for binary vectors. In the literature, there are many state-of-the-art algorithms which are feasible for higher dimensional spaces.

Locality sensitive hashing [28] is one of the methods for approximate nearest neighbor. This algorithm is based on projection of vectors onto a space with a smaller dimension. This projection is achieved by using random coordinates of the descriptor space and the projected vectors are used to form buckets of projections with different coordinate values. As those buckets are constructed, approximate nearest neighbor can be obtained by using nearest neighbor search in these buckets.

There is also another approach, namely *K-D trees*, to approximate nearest neighbor search. In [29], K-D trees method is proposed as a generalization of a binary tree to high dimensional spaces. Afterwards, its modified versions [30] and [31] are also proposed to optimize the order of the coordinates. The idea behind *K-D trees* is that the database is clustered hierarchically according to the coordinates of the descriptors. The descriptors in the database are classified into two clusters in each stage of the hierarchical classification according to the value in one of the coordinates of the vector. In other words, the data is split into two halves by a hyperplane orthogonal to a selected dimension. At the end of this procedure, the descriptors in the database are kept in the leaves of this tree. Once the vector with minimum distance to the query vector is required to be searched out of the database, the query vector is matched by the constructed tree. There is also a *K-D trees* algorithm using multiple *K-D trees* [32], in which the order of the coordinates is selected randomly. Nevertheless, the computational complexity of this kind of algorithms increases, as the dimension of the descriptor space is enlarged.

Another tree-based approach to approximate nearest neighbor problem is hierarchical k-means [33], [34]. In this approach, an initial k-means clustering is performed on the training data where the number of clusters is k . After this initial approach is performed, the data is split into k clusters, where each cluster consists of the descriptor vectors closest to a particular cluster center. The same process is then recursively applied to each of the clusters. The algorithm is stopped, when there is l number of levels in the hierarchy. Finally, a tree with l levels is constructed. In the search part, a query vector is propagated down the tree by comparing the query vector to the k cluster centers and choosing the nearest one. There are also other tree based hierarchical methods, such as vantage-point trees [35] or spill trees [36].

In [37], Trzcinski *et. al.* analyzed a number of methods in the literature related to approximate nearest neighbor search. Moreover, they also proposed a method, called *Parc-Trees*, in which multiple random trees are used. The *Parc-Trees* algorithm is given below:

\mathcal{S} : set of descriptors

- I. Choose k random samples out of \mathcal{S}
- II. Partition \mathcal{S} into k subsets according to the nearest samples

III. Repeat the same procedure until the number of elements in the subset is less than k .

This algorithm gives one random tree result. In *Parc-Trees*, multiple random trees are generated. In the search phase, a query vector is propagated downward along all the trees and the best result is selected as approximate nearest neighbor.

In [37], k-d trees, vantage point trees, locality sensitive hashing, hierarchical k-means algorithms are compared against *Parc-Trees* algorithm. Comparative results all of the algorithms, except *Parc-Trees*, yields undesired performance in binary spaces. For k-d trees method, the performance is quite sensitive to noise, since the data is split according to one dimension and the error in that dimension of the query vector affects the performance significantly. The reason for hierarchical k-means and vantage point trees fail in binary space is that there are many equidistant vectors to two random points. Such a *thick boundary problem* in binary space causes most of the algorithms to fail. In order to overcome this disadvantage of binary space, *Parc-Trees* method prefers using multiple random trees. Using multiple trees seems to compensate for the false decisions of another tree or trees.

Another important consideration is that use of a single unique tree is disadvantageous, even if many nodes are visited during the search algorithm. Suppose each search result in a tree is independent from each other and it has a failure probability of p_e . Searching independently n times should decrease the probability to p_e^n . This should be straight line in the logarithmic scale. In [32], a study on the probability error versus maximum number of searched nodes is performed. The resulting graph is shown in Figure 50.

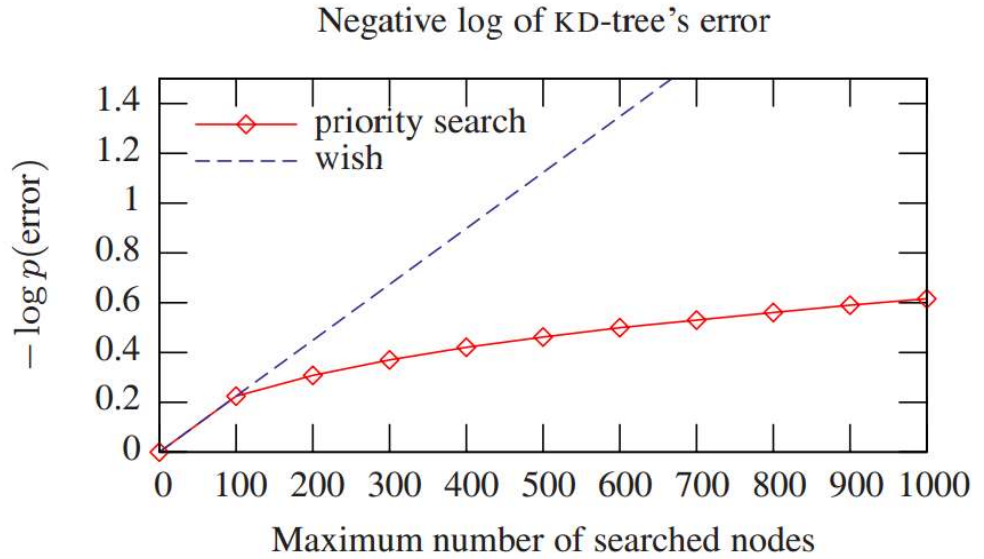


Figure 50 Priority search result for error and the ideal error decrease [32].

As it can be observed from Figure 50, priority search is not close for representing the independent searches. This being the case, one can claim that usage of one tree with multiple searches does not improve the performance of the approximate nearest neighbor search results, since the searches are not independent from each other. Therefore, use of multiple random trees prevails and is another reason for *Parc-Trees* giving promising results.

In order to handle the uncertainty in the *thick* boundaries of the binary space, a loose criterion is used in some methods, such as [33], [35] and [37]. In [36], a loose criterion is used while constructing a hierarchical binary tree. Assume $C1$ and $C2$ are two centroids of the data space. Those centroids are selected among the data set as the most distant centroids. As it can be observed from Figure 51, the middle point of these centroids, M is obtained. By using an overlapping distance threshold t , the data space is split into two subparts. The region between L and R lines in Figure 51 is the region of points all of which are sent to both of the subparts of the data space. The crucial point in this partitioning is overlapping size t . If t is not selected small enough, the construction of tree may not terminate due to the result that the subparts may be equal to the parent part, if the overlapping size is high enough. Since the partitioning of the data is performed hierarchically, the overlapping size t should be shrunk as the space, which the subparts

fill, gets smaller. Since the space of the subparts cannot be calculated during tree construction and optimum overlapping size may change according to dataset type, the method in [36] avoids this problem by a hybrid tree. If overlapping sub-region idea does not change partitioning significantly, then the non-overlapping classical nearest neighbor partitioning is used. By controlling the overlapping partitioning and the overlapping size in a reasonable range, the termination of the tree construction is prohibited. To sum up, Spill Trees [36] is a modified version of hierarchical k-means algorithm using a branching factor 2, since binary trees are used.

In this work, some advantageous parts of the algorithms are merged. The proposed approximate nearest neighbor is a modified version of *Parc-Trees* algorithm following the idea of spill trees [36].

In the proposed method, the data is split into k according to k centroids $C1, C2 \dots Ck$, that are selected randomly. For the sake of simplicity assume k is two and centroids are $C1$ and $C2$. In classical hierarchical K-means, if the distance of the vector W shown in Figure 51 to $C1$ is smaller than the distance of that vector to $C2$, the vector W will go to the group of centroid $C1$. Let $d1 = p(C1, W)$ and $d2 = p(C2, W)$, while p is the distance metric and W is the vector to be grouped in the database. If $\max(d1, d2) < \alpha * \min(d1, d2)$, then the vector W is sent to both of the groups of $C1$ and $C2$. Here parameter α is a predefined threshold for learning stage. This loose criterion makes the approximate nearest neighbor more accurate, since the vectors in the approximately equidistant boundaries of the centroids are clustered into both of the groups. This means that within the loose criterion, there cannot be a search propagating in the wrong direction due to those thick boundaries.

In both the proposed algorithm and *Parc-Trees*, multiple random trees are generated to perform approximate nearest neighbor. In the construction stage, the centroids to cluster the database are selected randomly. After that procedure, the database is clustered according to the nearest centroids. Unlike *Parc-Trees*, the clustering of the database is achieved according to the loose criterion similar to [36]. Spill trees [36] uses a unique binary tree whereas the proposed algorithm is using multiple random trees and does not have to be binary. The other difference is that proposed algorithm selects centroids randomly, while spill trees choose the farthest pairs as centroid vectors. This is not feasible in our case, since the proposed method is a modified version of *Parc Trees*

algorithm, where the branching factor does not have to be two. Another advantage of this proposed algorithm is that its learning stage is fast enough, since there is no need to consider about random centroid selection. Furthermore, by exploiting the loose criterion in spill trees, the thick boundary problem is handled better than *Parc-Trees* does since the descriptors which are in the boundaries of the clusters are accepted as members of both of the clusters. Therefore, a query vector near a cluster boundary will go to the both of the clusters due to the fuzzy decision tree structure.

The computational complexity of the proposed system is approximately $O(\mathbf{d} \mathbf{k} \log_k \mathbf{N})$ where \mathbf{d} is dimension, \mathbf{k} is branching factor and \mathbf{N} is the number of vectors in the database. This is valid, of course, if the tree is dividing the database in a balanced way.

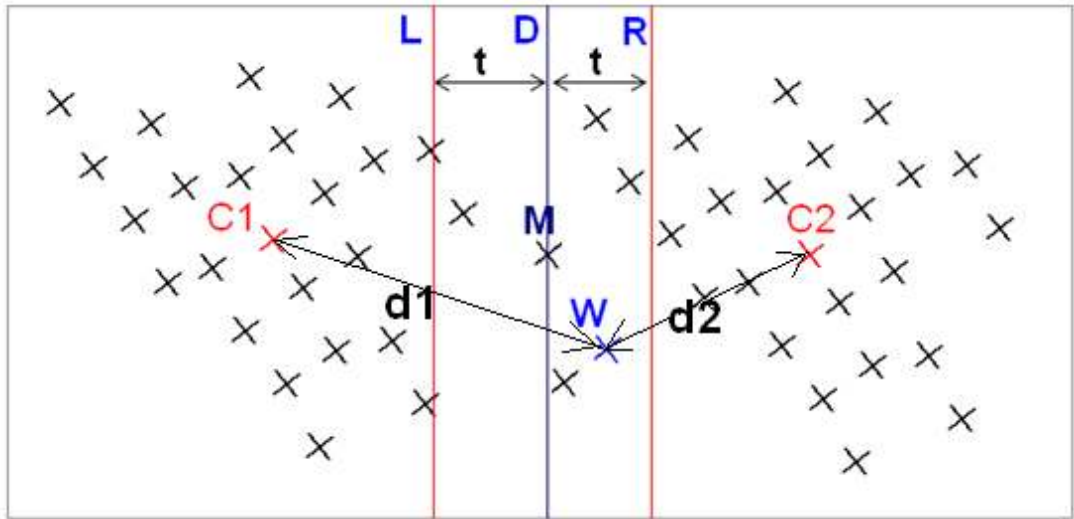


Figure 51 Illustration of 2 dimensional descriptor space. C1 and C2 are centroids and W is the data point to be classified into buckets of C1 and/or C2.

4.3. EXPERIMENTAL RESULTS

Performance of the proposed approximate nearest neighbor is compared with *Parc-Trees* algorithm [37] since it is proven in that work that *Parc-Trees* has better performance than K-D trees, vantage point tree and hierarchical k-means approach in terms of precision versus computation time curves. In our experiments, three types of experiments are

conducted, as Precision-Recall Curves, Precision-Computation time curves for different number of trees and Precision-Computation time curves for different number of branching factors. In all of these experiments, the proposed 128 bit local binary descriptor and the proposed corner detector is used in multi-scales. Moreover, the idea of Harris Score ordering in ORB is also used due to the powerful repeatability results of ORB corner detector. The parameter for loose criterion α is chosen as 1.1, since in the datasets used in experiments, the value 1.1 is small enough to terminate the construction of tree.

4.3.1. Precision-Recall Curves

To plot precision –recall curves, a distance threshold Thr is used. Approximately 1K points are matched to 1K points in different datasets where different views of the same scene are given [18]. Brute-force-matching result (linear search) is used as ground truth which indicates the correspondences with minimum distance performing an exhaustive search. Assume there are N number of correspondences. Correspondences by a Hamming distance under Thr are selected and number of thresholded correspondences is denoted by N_{thr} . The true correspondences out of thresholded correspondences are extracted using ground truth correspondences of brute-force-matching. Number of true correspondences is denoted by N_{true} . Precision and recall definitions for our experiments are given below.

$$precision = N_{true}/N_{thr} \quad (4.3)$$

$$recall = N_{true} / N \quad (4.4)$$

In precision-recall graphs, precision and recall percentages are normalized to 1.

The curves of precision versus recall are plotted using different thresholds.

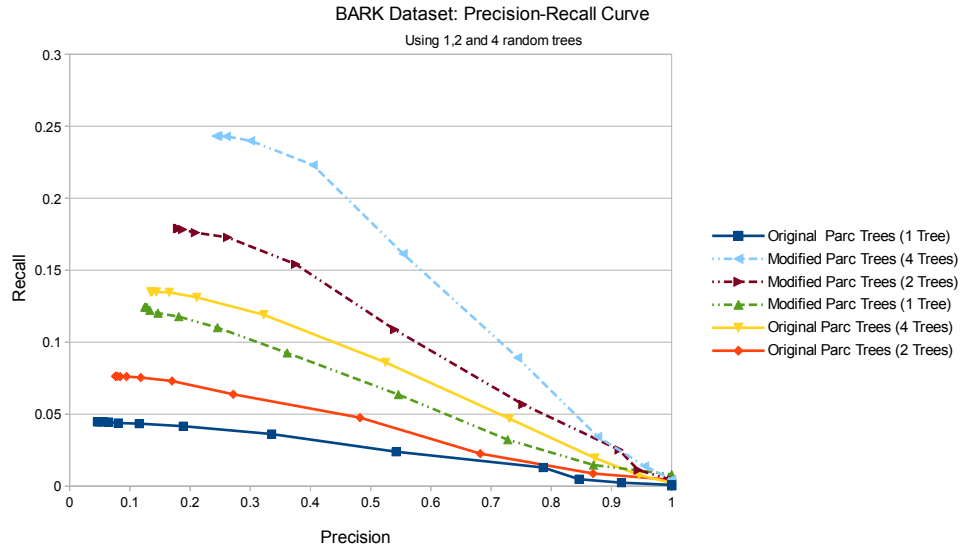


Figure 52 Precision vs recall curve for Bark Dataset

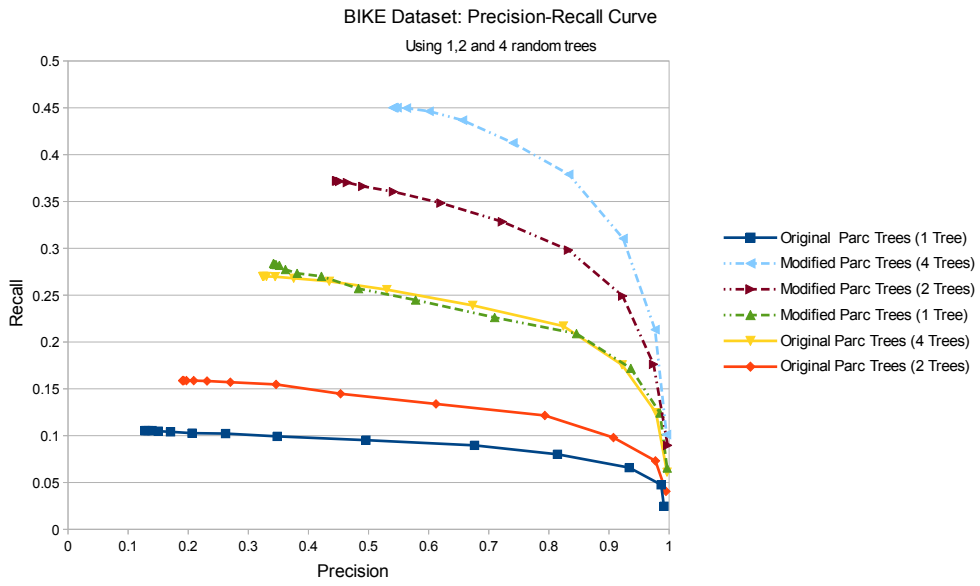


Figure 53 Precision vs recall curve for Bike Dataset

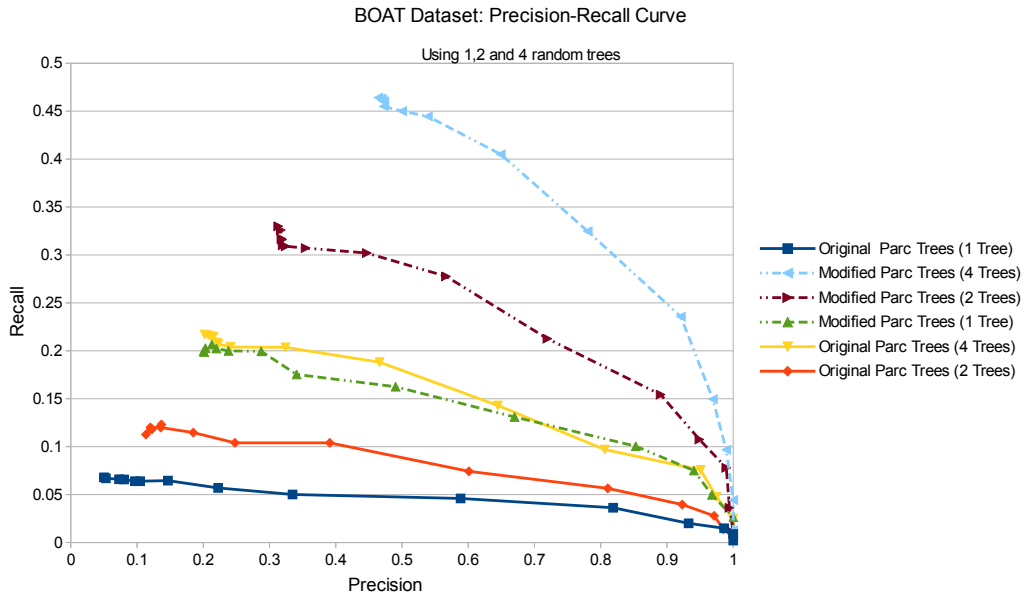


Figure 54 Precision vs recall curve for Boat Dataset

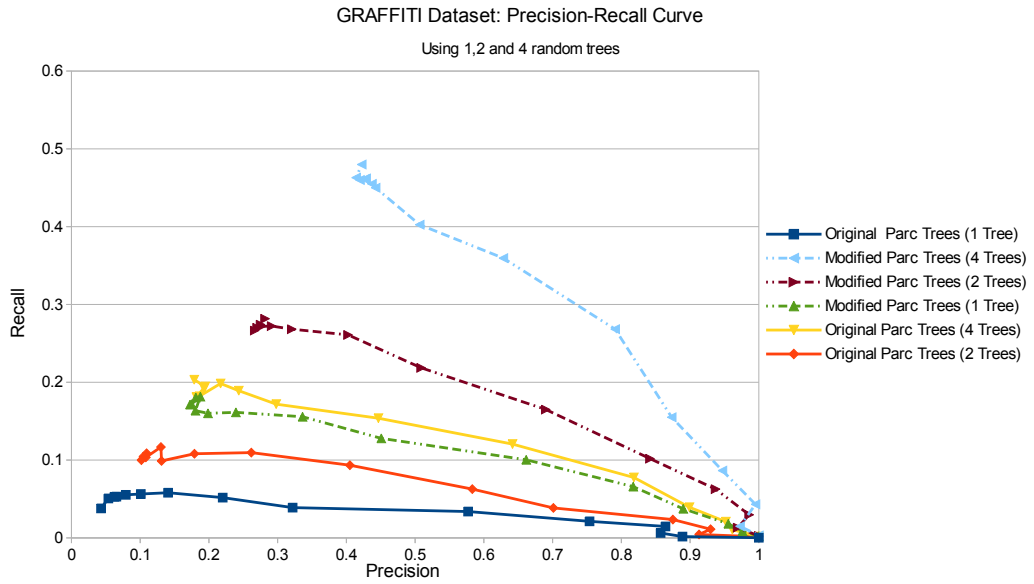


Figure 55 Precision vs recall curve for Graffiti Dataset

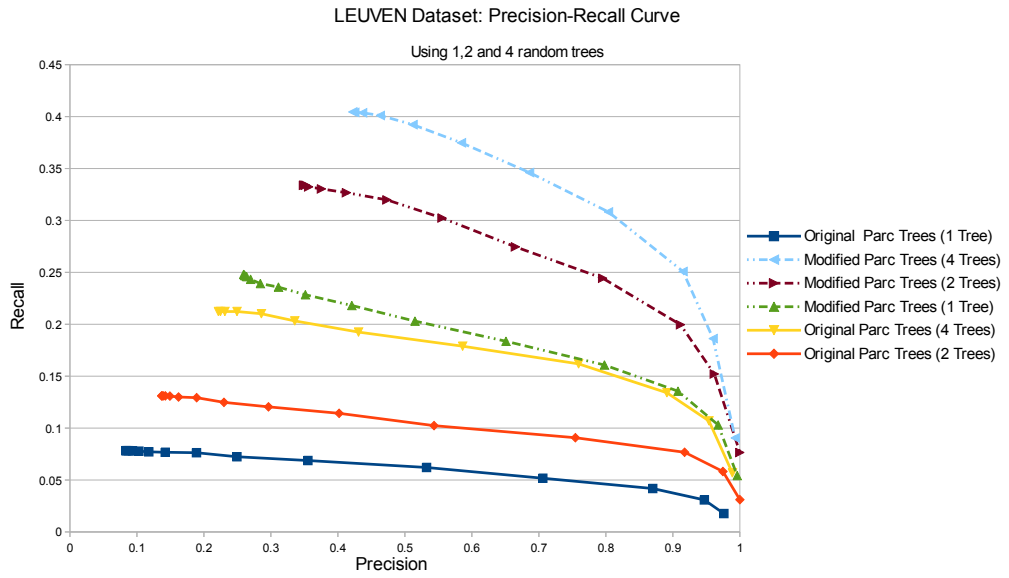


Figure 56 Precision vs recall curve for Leuven Dataset

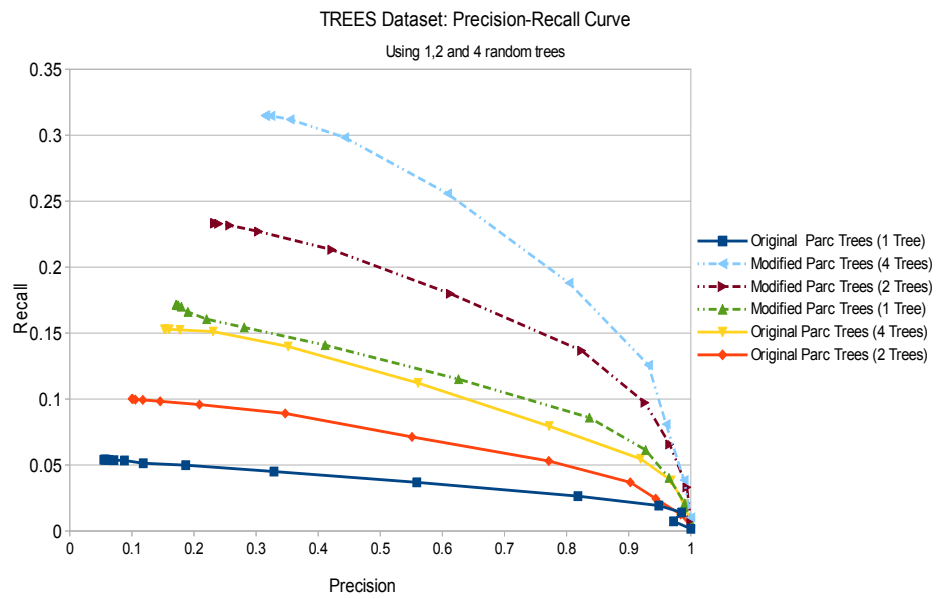


Figure 57 Precision vs recall curve for Trees Dataset

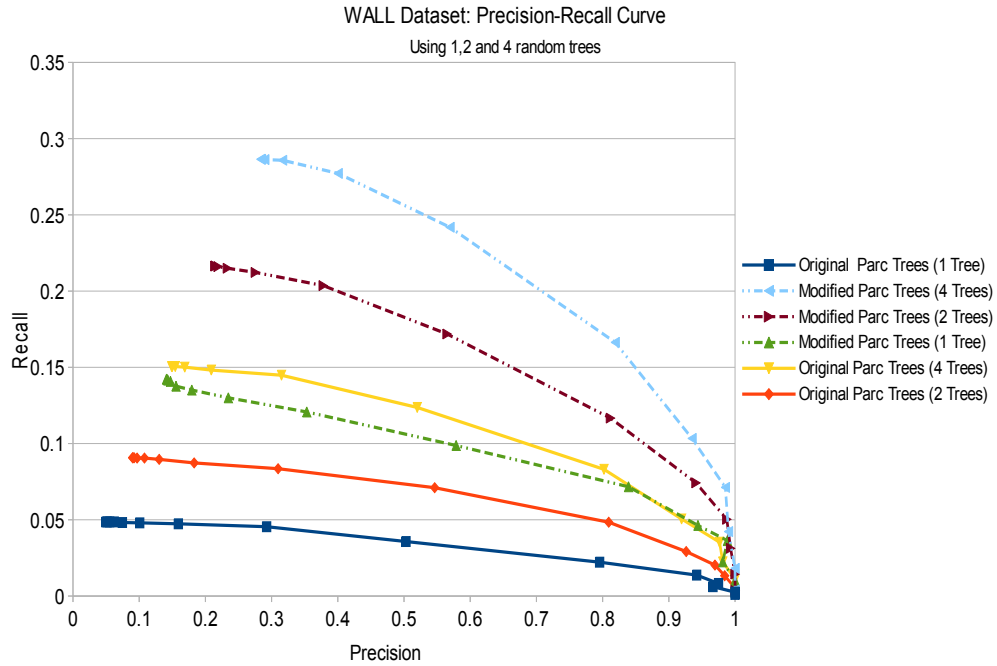


Figure 58 Precision vs recall curve for Wall Dataset

Figure 52 to Figure 58 are precision-recall results of the proposed (denoted as Modified Parc Trees in the legend) algorithm and original *Parc-Trees* algorithm for different number of trees. Proposed algorithm with a single tree has almost equal performance with original *Parc-Trees* algorithm. Proposed algorithm with 1 tree has a computational burden which is half of the original *Parc-Trees* algorithm approximately. Although the proposed algorithm with 1 tree is expected to have quarter computation time of the original *Parc-Trees* algorithm with 4 trees, it is half, since the number of levels in the trees in the proposed algorithm has number of levels as twice as original *Parc-Trees* algorithm. These results can further be observed from precision-computation time curves.

4.3.2. Precision-Computation time curves for different number of trees

For precision – computation time curves, the same definition of the precision is used in section 4.3.1. The time is calculated using the clocks of the computer. The computation time is calculated using the clocks of the computer and all of the computations are calculated using 1K training and 1K test vectors. The curves are plotted for different threshold values. This means that the selected threshold value eliminates correspondences with Hamming distances higher than that value. After this elimination, the precision values are calculated for correct matches out of the correspondences thresholded. In these curves, Precision percentage values are normalized to 1.

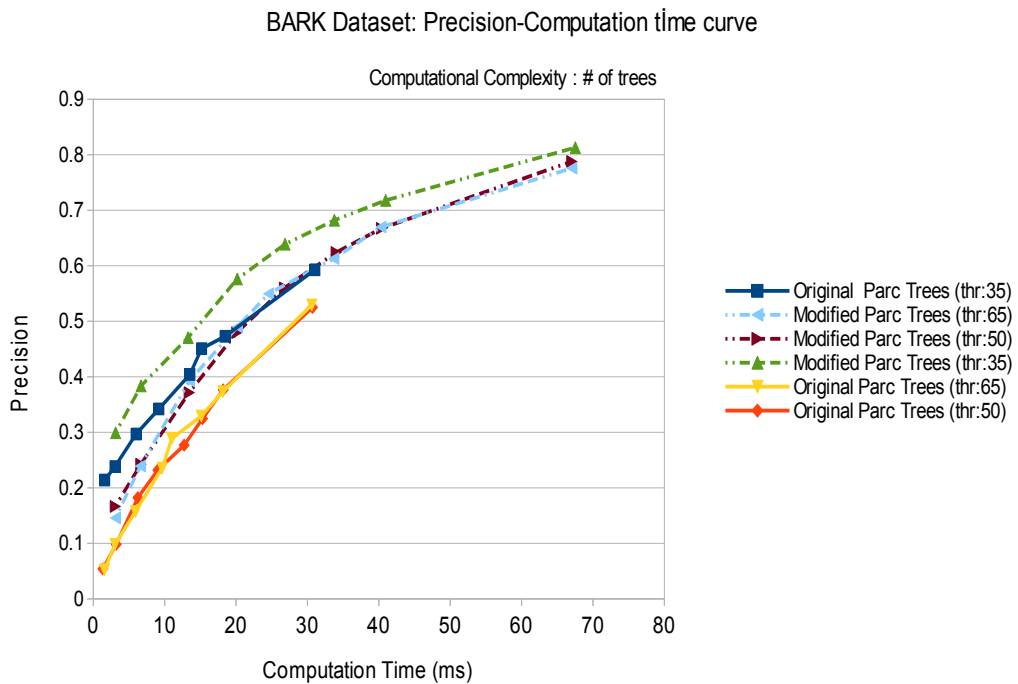


Figure 59 Precision-comp. time curve for Bark Dataset (complexity type: # of trees)

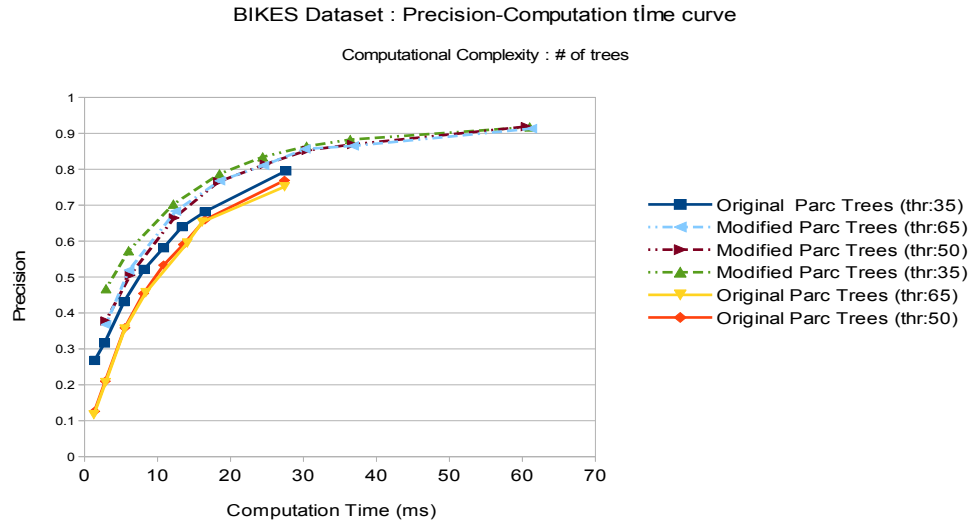


Figure 60 Precision-comp. time curve for Bikes Dataset (complexity type: # of trees)

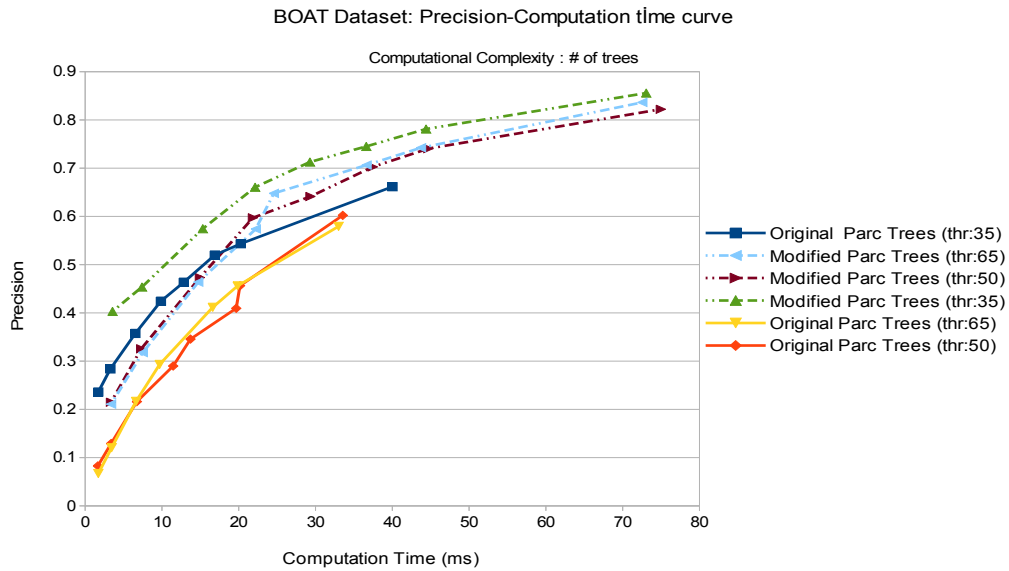


Figure 61 Precision-comp. time curve for Boat Dataset (complexity type: # of trees)

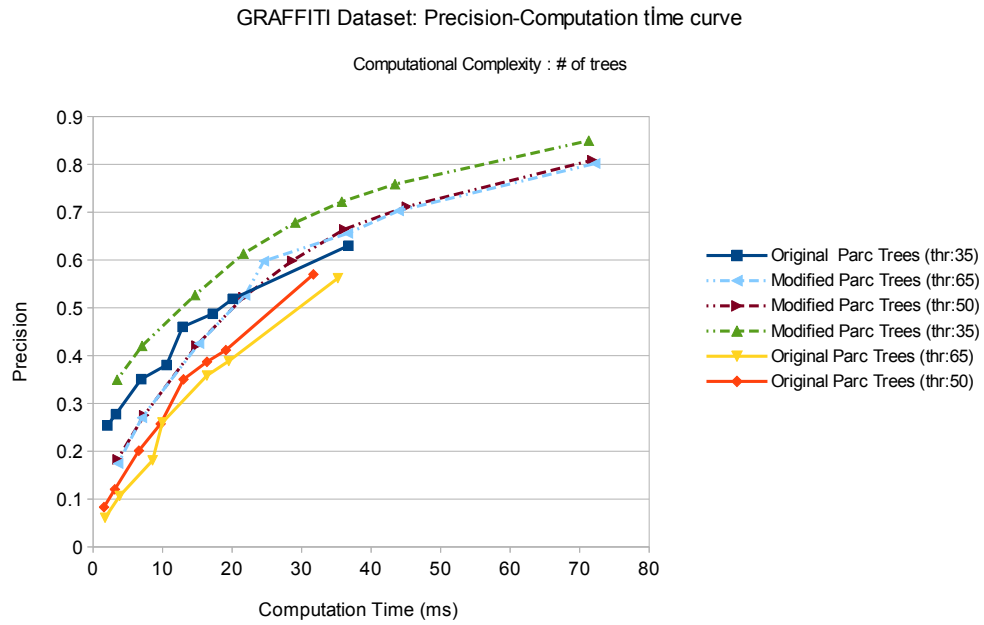


Figure 62 Precision-comp. time curve for Graffiti Dataset (complexity type: # of trees)

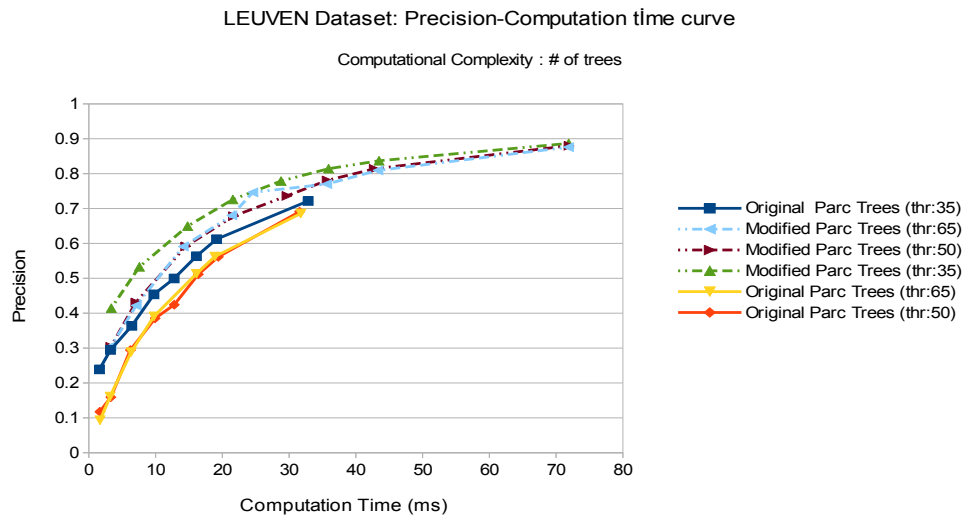


Figure 63 Precision-comp. time curve for Leuven Dataset (complexity type: # of trees)

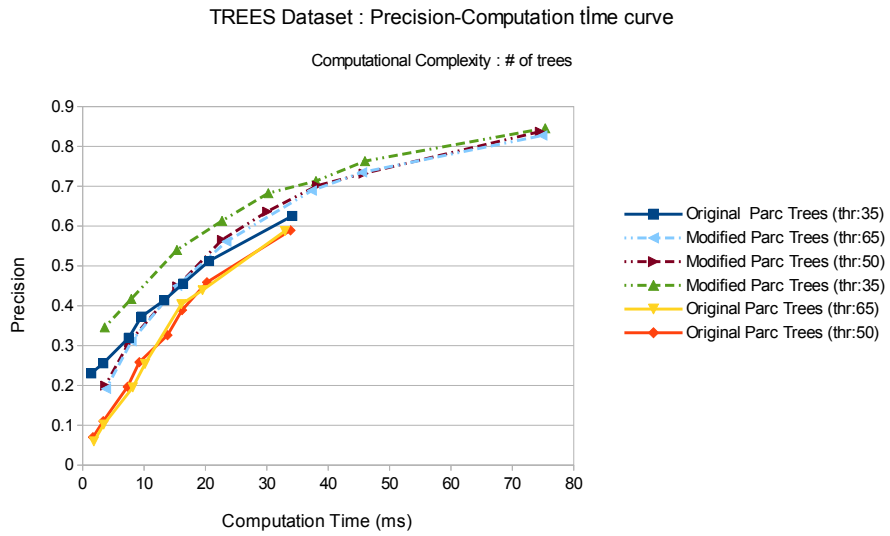


Figure 64 Precision-comp. time curve for Trees Dataset (complexity type: # of trees)

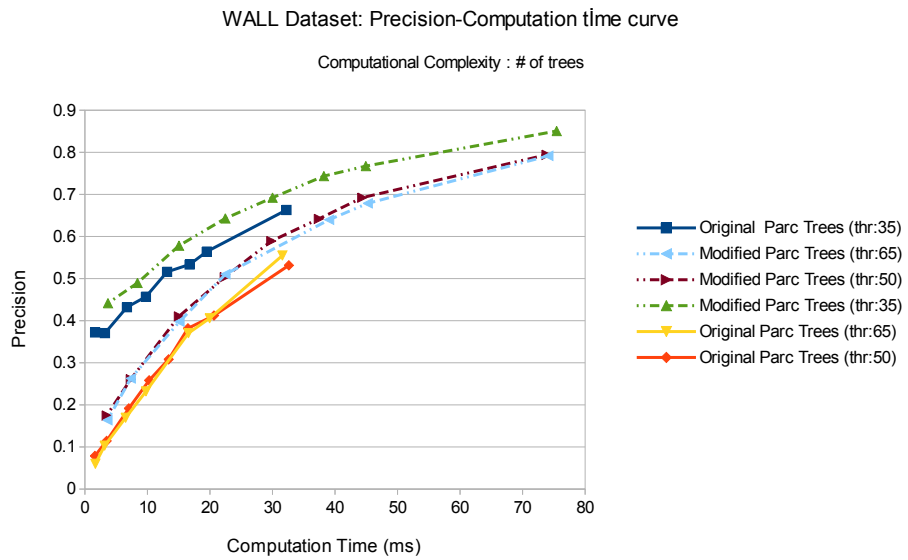


Figure 65 Precision-comp. time curve for Wall Dataset (complexity type: # of trees)

As it can be seen from Figure 59 to Figure 65, the proposed matching method performs better than *Parc-Trees* algorithm and using less number of trees than *Parc-Trees* algorithm, the proposed algorithm matches descriptor vectors in high precision.

4.3.3. Precision-Computation time curves for different branching factors

For precision – computation time curves, the same definition of the precision is used in section 4.3.1. The time is calculated using the clocks of the computer. The curves are plotted for different threshold values. The trade-off for computational complexity is here branching factor k since computation time increases as the branching factor increases. As it is stated in the previous section, precision percentage values are normalized to 1.

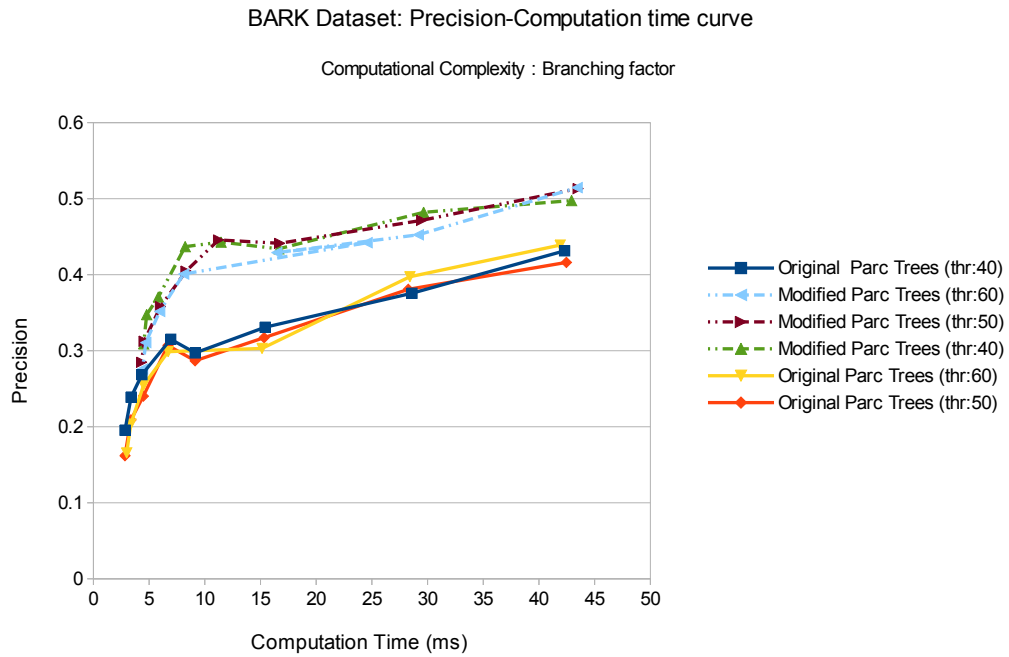


Figure 66 Precision-comp. time curve for Bark Dataset (complexity type: branching factor)

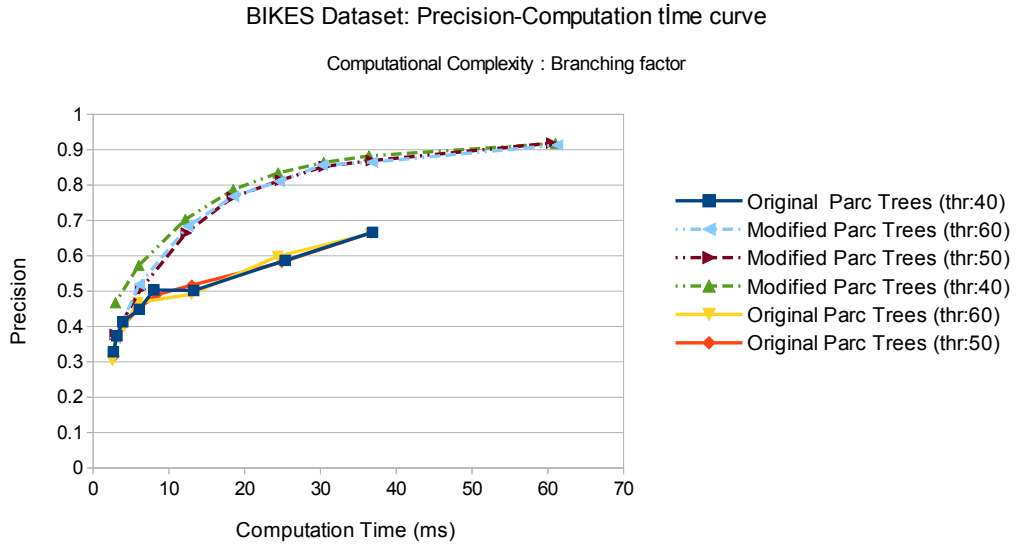


Figure 67 Precision-comp. time curve for Bikes Dataset (complexity type: branching factor)

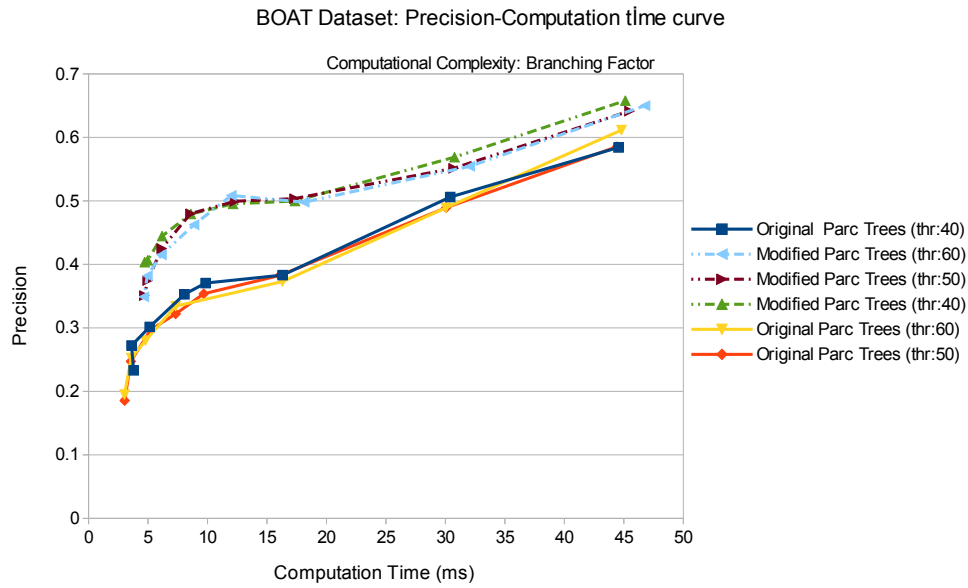


Figure 68 Precision-comp. time curve for Boat Dataset (complexity type: branching factor)

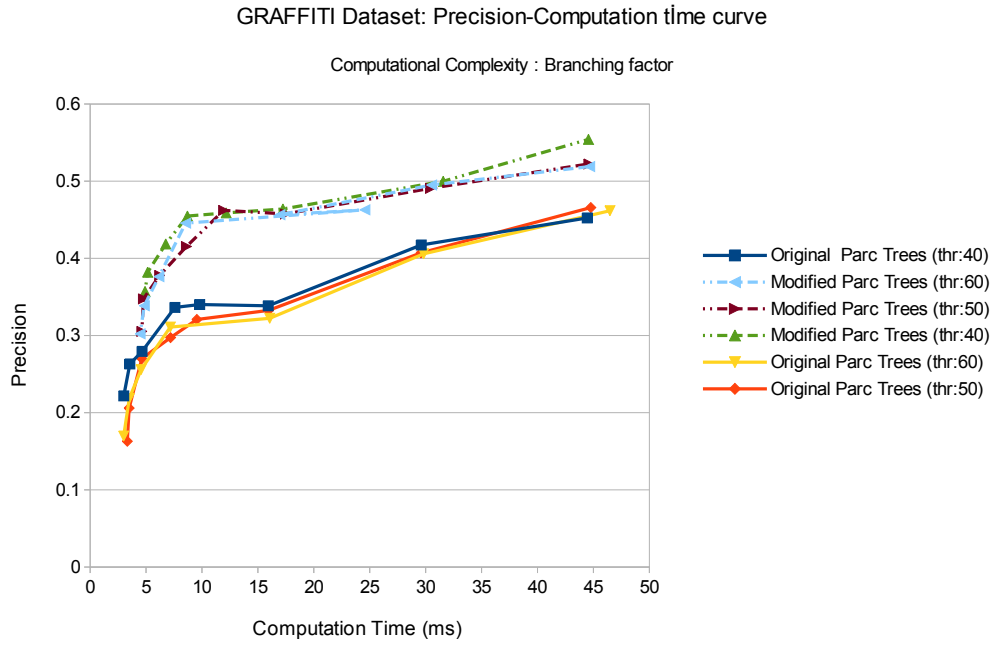


Figure 69 Precision-comp. time curve for Graffiti Dataset (complexity type: branching factor)

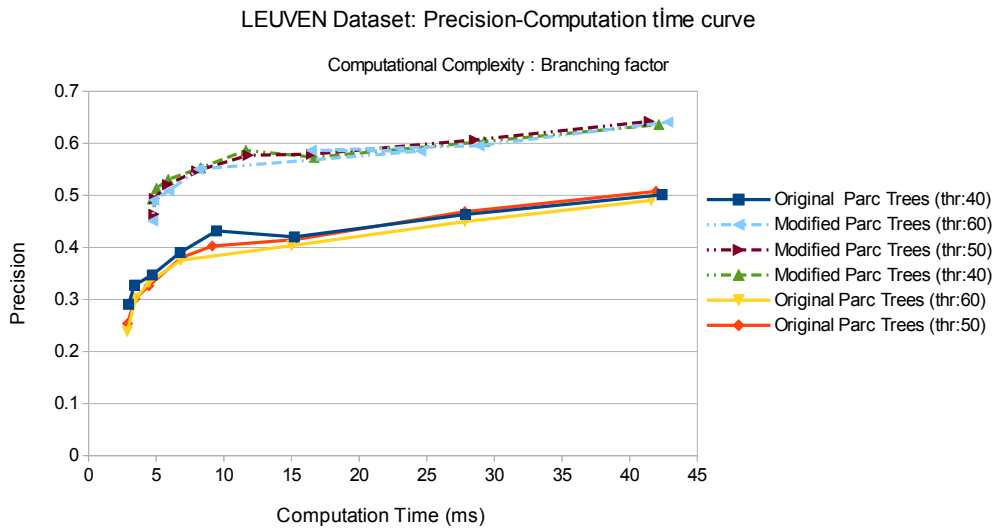


Figure 70 Precision-comp. time curve for Leuven Dataset (complexity type: branching factor)

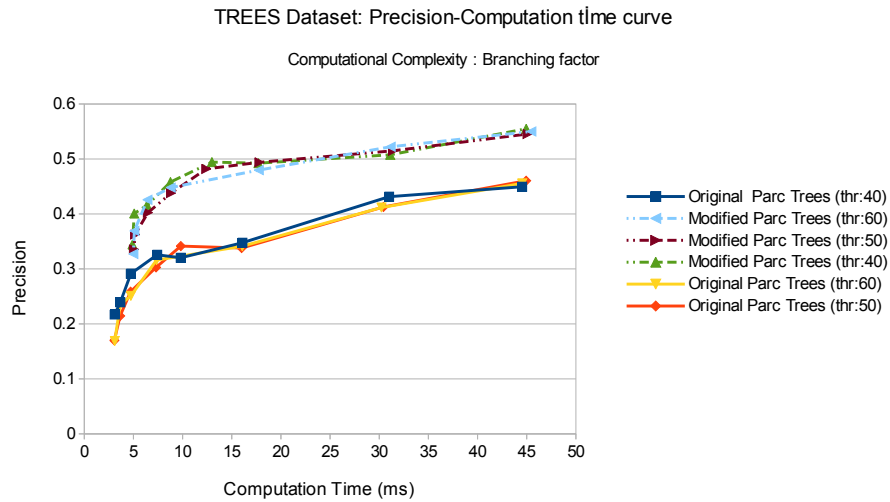


Figure 71 Precision-comp. time curve for Trees Dataset (complexity type: branching factor)

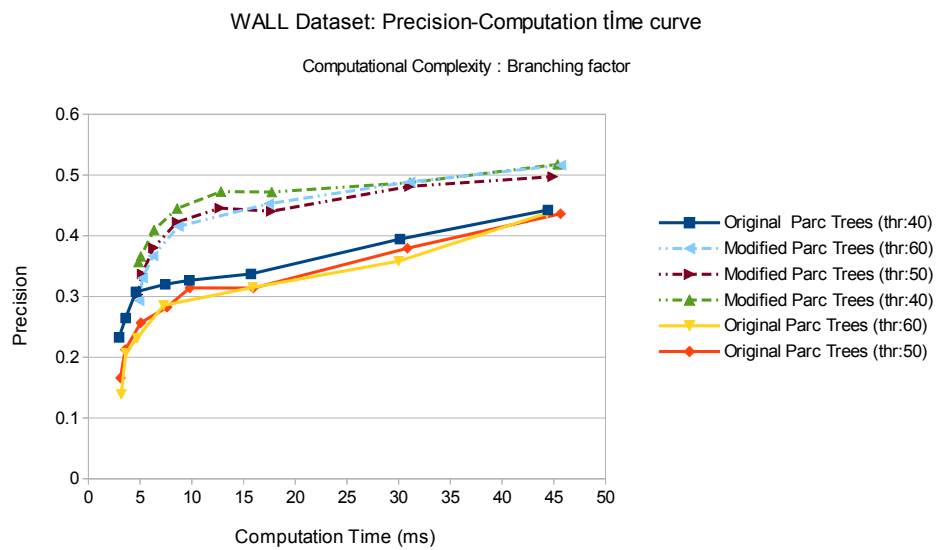


Figure 72 Precision-comp. time curve for Wall Dataset (complexity type: branching factor)

From Figure 66 to Figure 72, precision-computation time curves for different branching factors in different datasets are presented. It is obvious that proposed algorithm yields a

better precision by using less number of branching factor, compared to *Parc-Trees* algorithm.

4.4. CONCLUSION

When all of the experiments for different conditions are considered, the proposed method is able to beat *Parc-Trees* algorithm performance. Although the proposed method is random as *Parc-Trees* algorithm, proposed method outperforms *Parc-Trees* even using a single tree, since it is able to handle thick boundary problem in binary space. For real-time applications, the proposed matching algorithm can run approximately in 2 ms satisfying the conditions for estimating homography between two planar views, when there are 1K points in reference image and 300 points in the query image.

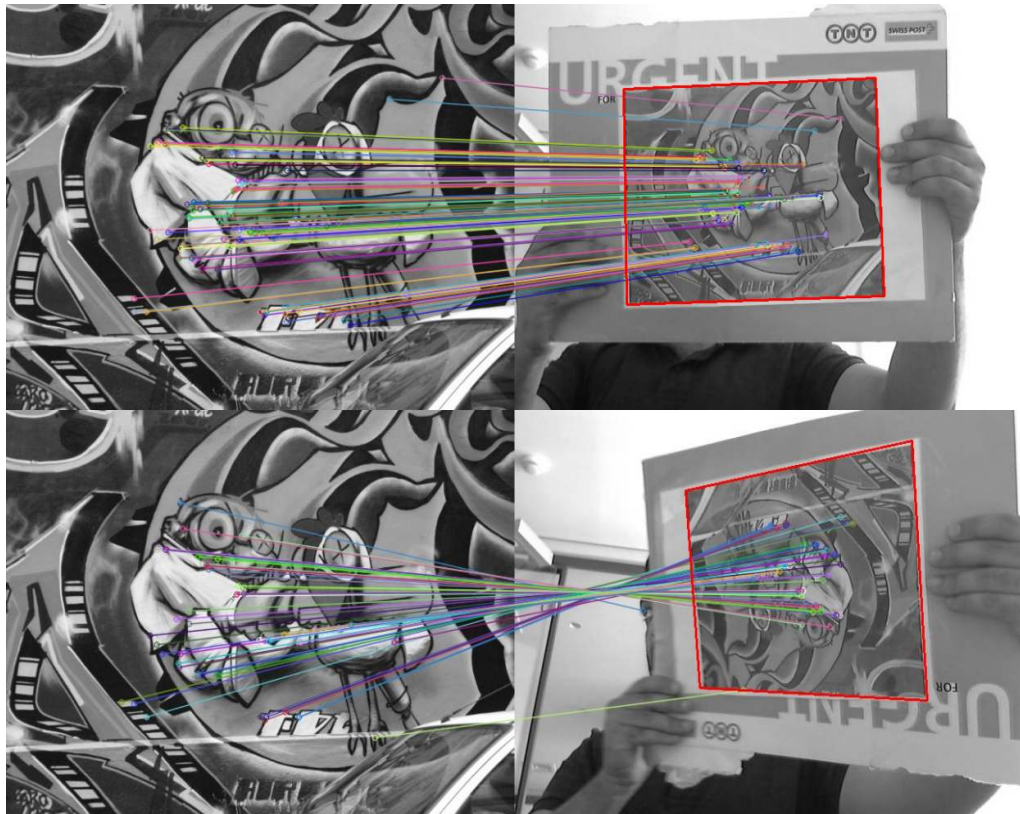


Figure 73 Two sample frames shows the matching results of the keypoints



Figure 74 Mobile application using an Android tablet PC

CHAPTER 5

CONCLUSIONS, DISCUSSIONS AND FUTURE WORK

In this thesis, important problems of Computer Vision; namely, local feature detection, local feature description, and methods for matching descriptors for real-time implementations have been analyzed and new methods have been proposed for those problems. Although the problems analyzed in this thesis are well-known and various solutions exist, the main focus in this thesis is to observe and analyze the trade-off between computational efficiency and accuracy of the algorithms for real-time implementations.

Once the literature is considered in terms of efficient corner detection methods, the proposed corner detection algorithm in this thesis is a promising candidate for real-time applications, since it has a competitive repeatability performance and an efficient computation complexity.

In FAST [6], the average number of comparisons per pixel is stated as 2.26. Although, number of comparisons depends on the texture of the scene, the proposed key point detection method performs a minimum of 0.5 and a maximum of 2.2 comparisons per pixel in average in the datasets used in the experiments. This means that most of the time the proposed key point extraction algorithm is more efficient than FAST. Therefore, if the necessary optimizations are fulfilled in the source code of the proposed algorithm, it is clear that the proposed key point extraction method will be less time consuming than FAST.

View point change is one of the frequent conditions for AR applications for planar scenes. When the deformation conditions for repeatability experiments are considered, the proposed key point extraction method outperforms FAST in the datasets where deformation type is viewpoint change. Hence, the proposed method is proven to be advantageous for AR applications of planar scenes.

One of the important performance metrics for pose estimation, which is required for AR applications, is localization error. The proposed method has less localization error than FAST in all of the experimental dataset due to the nature of the proposed method. Therefore, one should use the proposed extraction method if the localization error is required to be as small as possible.

Local feature description is another important problem for vision research, since it is proven to be successful in various vision applications. As this work pays attention to computational efficiency of local descriptor, local binary comparison idea is embraced due to its computational efficiency.

The proposed local binary pattern clearly outperforms ORB [8] binary pattern in most cases in terms of distinctiveness. This being the case, precision-recall curves of the proposed binary pattern reveals better results than ORB. The proposed 256 bit descriptor results in higher precision rates within the same recall rates or vice versa. This result also shows that regular patterns may be more distinctive than random binary patterns. This is because of the fact that ORB pattern is learnt without using the ground truth information in which true correspondences gives the same result and false correspondences gives different result for a specific comparison test. Therefore, a further study on local binary description should be determining the optimum binary pattern as a future work. The literature lacks this research, since there is no algorithm which can learn the best binary descriptor pattern by using the ground truth information of local correspondences.

Moreover, multi-thread implementations of the descriptor algorithms can be preferred if there are more than one processor. If multi-thread option is available, then different color channels can be utilized for local description such that local description can be performed in each channel separately and fused to have the appropriate descriptor matches.

Approximate nearest neighbor search is an old research area, where there are many improvements and methods to solve approximate nearest neighbor problem. Nevertheless, construction of an approximate nearest neighbor search algorithm requires high computational efficiency especially in large database applications and when the number of dimensions is relatively high. In order to avoid these drawbacks, methods using multiple random trees are feasible for efficiency if the algorithms are expected to run in mobile devices. Therefore, an approximate nearest neighbor search method based on multiple random trees is proposed in this thesis. Moreover, it is proven that fuzzy decisions in the construction of the decision tree improve performance significantly.

In the implementation phase, high precision rates should be guaranteed to have sufficient correspondences between two scenes. Hence, low recall rates with high precision are acceptable for augmented reality applications for planar scenes. Once the precision-recall and precision-computation time curves are analyzed, the proposed algorithm outperforms *Parc-Trees* [37] significantly that the precision values in the proposed method becomes 1.5 times greater than in *Parc-Trees* within the same computation time or the same recall rates. Therefore, the proposed idea prevails for augmented reality implementations in real-time.

Taking computational efficiency and real-world conditions into account, a feature detection and matching system has been built to be implemented in mobile devices. Moreover, in the AR implementation of the system, Harris score ordering idea in ORB is adapted to the system due to its prevailing performance in terms of repeatability. This overall system has been tested in real-world data, which is a webcam with a resolution of 480x640. Using 1K keypoints for target/reference image and 300 keypoints for test frame, the keypoint detection, keypoint description using 256 bit binary descriptors and the proposed matching method with 3 trees take 15 ms in a 2.8 GHz desktop PC. A sample illustration is shown in Figure 73. This figure shows the matching result of reference image and test frame. In addition, homography between two planar surfaces are calculated and the bounding box of the reference image is drawn in the test image using this homography matrix.

Moreover, an Android application [45] has also been implemented for this system in a 1.2 GHz (Tegra 3 Quad core) tablet PC. The computation time of three phases (detection,

description and matching) in the tablet PC is approximately 150 ms. The captured frame for this application is shown in Figure 74.

According to the real world experiments in desktop and tablet PC's, one can say that the proposed methods perform promisingly in terms of efficiency and accuracy. This being the case, overall system can be used in real-time implementations especially in Augmented Reality, pose estimation, object recognition, object tracking.

REFERENCES

- [1] 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 <http://dx.doi.org/10.1561/06000000017>.
- [2] C. Harris and M. Stephens. “A combined corner and edge detector,”. In *Proceedings of 4th Alvey Vision Conference*, pages 147–151, 1988.
- [3] J. Shi and C. Tomasi. “Good Features to Track, ” *IEEE conference on Computer Vision and Pattern Recognition*, pages 593-600, 1994.
- [4] K. Mikolajczyk, & C. Schmid . “An affine invariant interest point detector, ”. In *Proceedings of the 7th European conference on computer vision (ECCV)* (pp. 128–142), 2002.
- [5] S. M. Smith and J. M. Brady, “SUSAN—A New Approach to Low Level Image Processing, ” *International Journal of Computer Vision*, 23(1), 45–78, 1997.
- [6] E. Rosten, and T. Drummond, “Machine Learning for High Speed Corner Detection,” *In Proceedings of the European Conference on Computer Vision (ECCV)*, 2006.

- [7] J. R. Quinlan. "Induction of decision trees. Machine Learning, " Vol. 1, pages 81-106,(1986) *Signal Processing*, 2011
- [8] E. Rublee, V. Rabaud, K.Konolige and G. Bradski , "ORB: an efficient alternative to SIFT or SURF, " *Intenational Conference on Computer Vision*, 2011.
- [9] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints, " *International Journal of Computer Vision* Vol. 60pages 91-110,2004.
- [10] J. J. Koenderink, "The structure of images. Biological Cybernetics, ", 50:363-396, 1984.
- [11] T. Lindeberg "Scale-space theory: A basic tool for analysing structures at different scales, ", *Journal of Applied Statistics*, 21(2):224-270, 1994
- [12] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. "SURF: Speeded up robust features,". *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008.
- [13] M. Agrawal, K. Konolige, and M. R. Blas, "CenSurE: Center surround extremas for realtime feature detection and matching, " *InProceedings of the European conference on computervision (ECCV)* ,Vol. 5305, pp. 102–115, 2008
- [14] J. Dupac and J. Matas, "Ultra Fast Tracking Based on Zero Shift Points, " *International Conference on Acoustics, Speech and Signal Processing*, 2011.

- [15] V. Lepetit and P. Fua, "Keypoint Recognition Using Randomized Trees, " *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 9, Sept, 2006.
- [16] S. Leutenegger, M. Chli and R. Y. Siegwart "BRISK: Binary Robust Invariant Scalable Keypoints," *International Conference on Computer Vision (ICCV)*, 2011.
- [17] K. Mikolajczyk and C. Schmid, "Scale & Affine Invariant Interest Point Detector, " *International Journal of Computer Vision (IJCV)* 60(1), 63–86, 2004.
- [18] "Test data for affine invariant regions" Internet: <http://www.robots.ox.ac.uk/~vgg/research/affine/index.html>, 15th July 2007 [9th September 2011]
- [19] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, 27:1615–1630, 2005.
- [20] E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. In *CVPR*, 2008.
- [21] T. Tuytelaars and C. Schmid, "Vector Quantizing Feature Space with a Regular Lattice," *Proc. Int'l Conf. Computer Vision*, 2007.

- [22] S. Winder, G. Hua, and M. Brown, "Picking the Best Daisy," Proc.IEEE Conf. Computer Vision and Pattern Recognition, June 2009.
- [23] M. Calonder, V. Lepetit, K. Konolige, J. Bowman, P. Mihelich, and P. Fua, "Compact Signatures for High-Speed Interest Point Description and Matching," Proc. IEEE Int'l Conf. Computer Vision, Sept. 2009.
- [24] Calonder M., Lepetit V., Strecha C., Fua P.: BRIEF: Binary Robust Independent Elementary Features. 11th European Conference on Computer Vision (ECCV), Heraklion, Crete. LNCS Springer, September 2010.
- [25] S. Lazebnik, C. Schmid, and J. Ponce. Sparse texture representation using affine-invariant neighborhoods. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin, USA, pages 319-324, 2003
- [26] R. Zabih and J. Woodill. Non-parametric local transforms for computing visual correspondance. In Proceedings of the 3rd European Conference on Computer Vision, Stockholm, Sweden, pages 151-158, 1994.
- [27] P. L. Rosin. Measuring corner properties. *Computer Vision and Image Understanding*, 73(2):291 – 307, 1999.
- [28] A. Gionis, P. Indik, and R. Motwani, "Similarity Search in High Dimensions Via Hashing," in International Conference on Very Large Databases, 2004.

- [29] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.
- [30] J. Beis and D. Lowe, “Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces,” in *Conference on Computer Vision and Pattern Recognition*, 1997, pp. 1000–1006.
- [31] S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Wu, “An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions,” *Journal of the ACM*, vol. 45, pp. 891–923, 1998.
- [32] C. Silpa-Anan and R. Hartley, “Optimised kd-Trees for Fast Image Descriptor Matching,” in *Conference on Computer Vision and Pattern Recognition*, 2008.
- [33] K. Fukunaga and P. Narendra, “A Branch and Bound Algorithm for Computing K-Nearest Neighbors,” *IEEE Transactions on Computing*, vol. 24, pp. 750–753, 1975.
- [34] D. Nister and H. Stewenius, “Scalable Recognition With a Vocabulary Tree,” in *Conference on Computer Vision and Pattern Recognition*, 2006.
- [35] P. Yianilos, “Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces,” in *Fourth ACM-SIAM Symposium on Discrete Algorithms*, 1993.

- [36] T. Liu, A. Moore, A. Gray, and K. Yang, "An Investigation of Practical Approximate Nearest Neighbor Algorithm," in *Advances in Neural Information Processing Systems*, 2004.
- [37] T. Trzcinski, V. Lepetit, P. Fua. Thick Boundaries in Binary Space and their Influence on Nearest-Neighbor Search. Ecole Polytechnique Fédérale de Lausanne, Technical Report, Nr. 167061, 2011.
- [38] A. Johnson and M. Hebert, "Object Recognition by Matching Oriented Points," *Proc. Conf. Computer Vision and Pattern Recognition*, pp. 684-689, 1997.
- [39] S. Lazebnik, C. Schmid, and J. Ponce, "Sparse Texture Representation Using Affine-Invariant Neighborhoods," *Proc. Conf. Computer Vision and Pattern Recognition*, pp. 319-324, 2003.
- [40] D. Gabor, "Theory of Communication," *J. IEE*, vol. 3, no. 93, pp. 429-457, 1946.
- [41] J.K.M. Vetterli, *Wavelets and Subband Coding*. Prentice Hall, 1995.
- [42] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors", *IEEE Transactions of Pattern Analysis and Machine Intelligence*, Vol. 27, No. 10, October, 2005.
- [43] Matas, J. Chum, O., Urban, M., and Pajdla, T. 2002. Robust wide-baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference*, Cardiff, UK, pp. 384–393.

- [44] G. Takacs, V. Chandrasekhar, S. Tsai, D. Chen, R. Grzeszczuk, and B. Girod, "Unified real-time tracking and recognition with rotation-invariant fast features", *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2010.
- [45] Internet: <http://developer.android.com/index.html>, [20th July 2012]
- [46] Lindeberg, T. "Detecting salient blob-like image structures and their scales with a scale-space primal sketch: a method for focus-of-attention". *International Journal of Computer Vision*, 11(3): 283-318, 1993.

APPENDICES

7.1. APPENDIX A

In this Appendix, distance histogram illustration of inliers and outliers for proposed 256 bit binary pattern and ORB are shown.

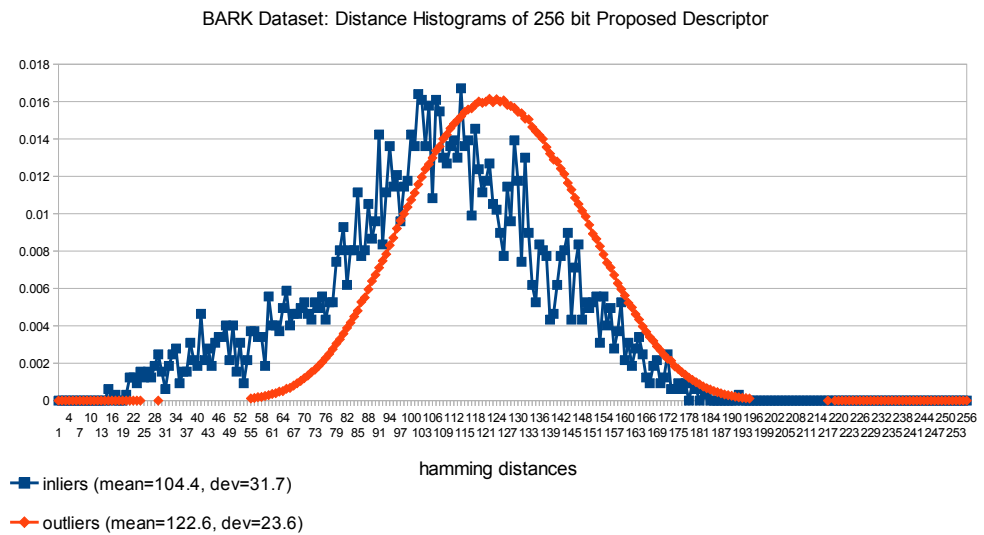


Figure 75 Distance histograms of Bark Dataset for proposed descriptor (256 bit)

BARK Dataset: Distance Histograms of 256 bit ORB Descriptor

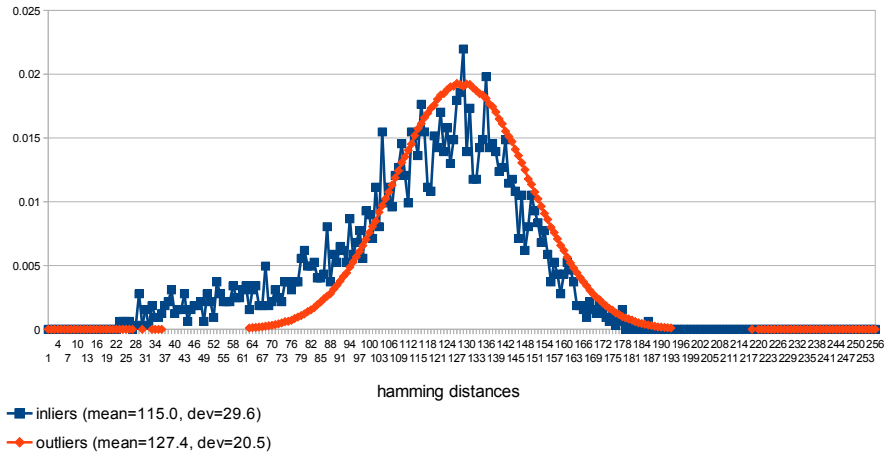


Figure 76 Distance histograms of Bark Dataset for ORB descriptor (256 bit)

BIKES Dataset: Distance Histograms of 256 bit Proposed Descriptor

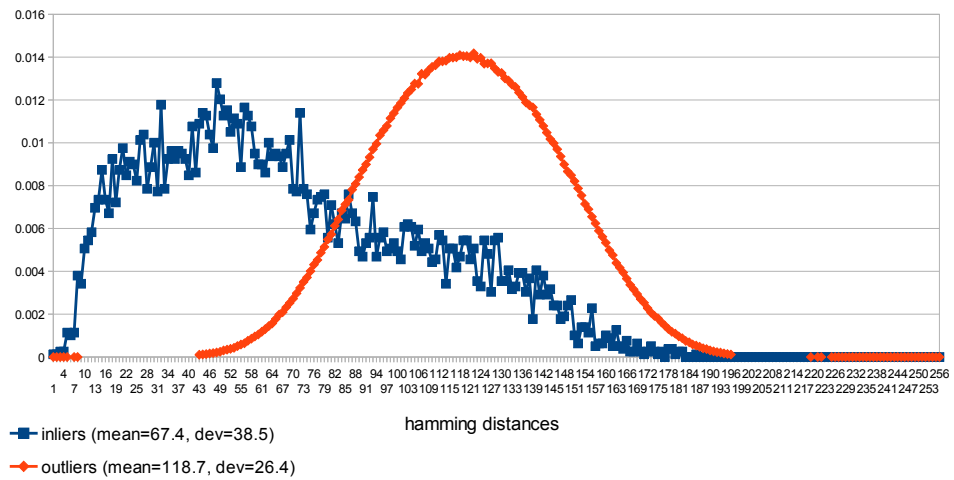


Figure 77 Distance histograms of Bikes Dataset for proposed descriptor (256 bit)

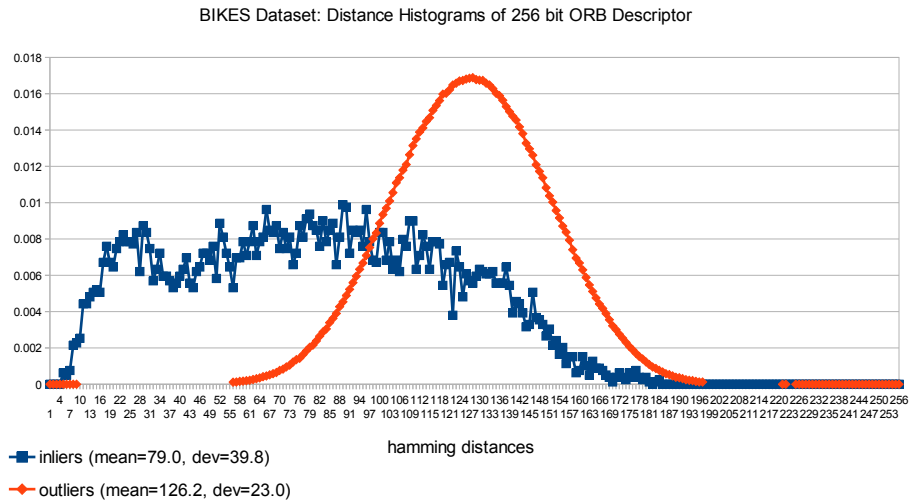


Figure 78 Distance histograms of Bikes Dataset for ORB descriptor (256 bit)

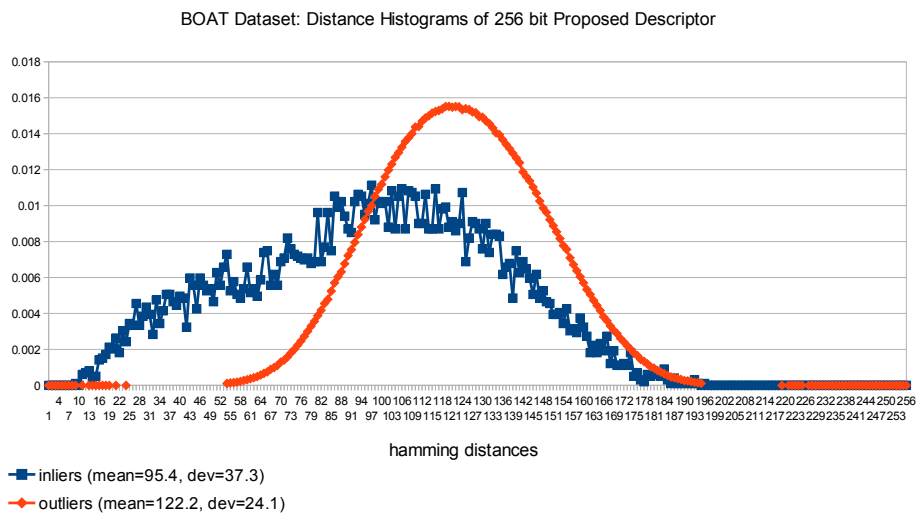


Figure 79 Distance histograms of Boat Dataset for proposed descriptor (256 bit)

BOAT Dataset: Distance Histograms of 256 bit ORB Descriptor

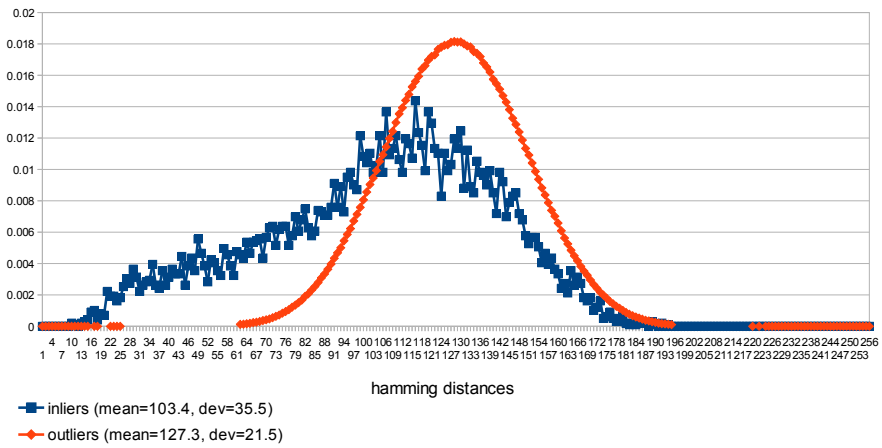


Figure 80 Distance histograms of Boat Dataset for ORB descriptor (256 bit)

GRAFFITI Dataset: Distance Histograms of 256 bit Proposed Descriptor

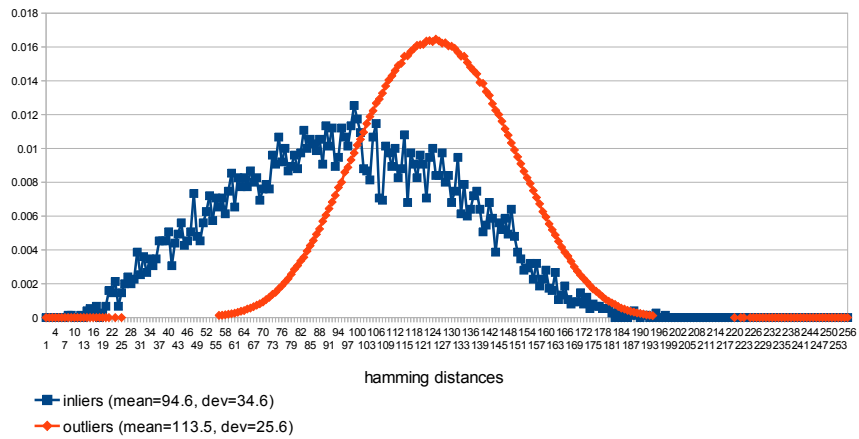


Figure 81 Distance histograms of Graffiti Dataset for proposed descriptor (256 bit)

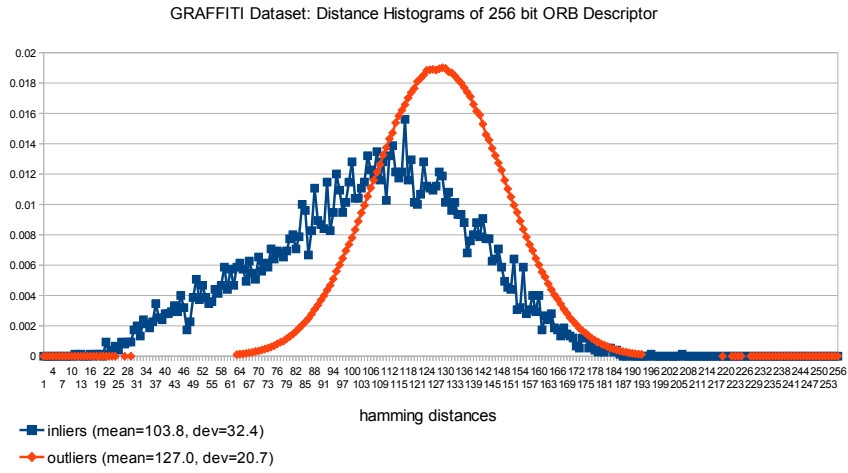


Figure 82 Distance histograms of Graffiti Dataset for ORB descriptor (256 bit)

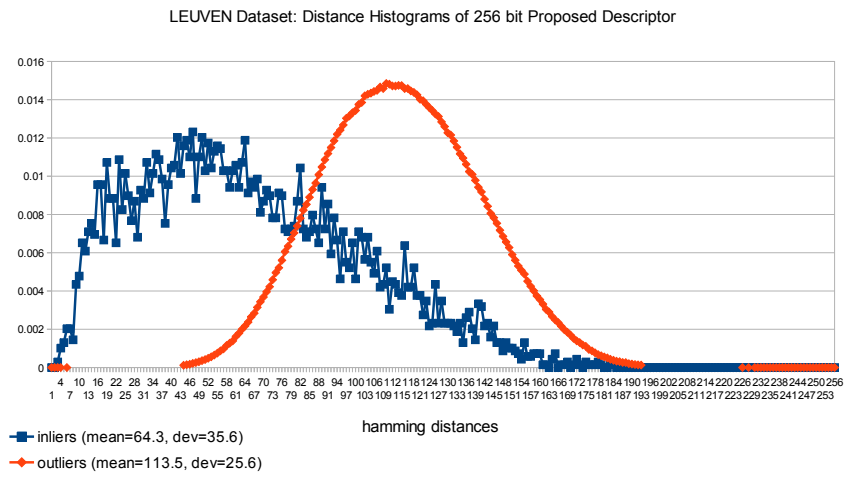


Figure 83 Distance histograms of Leuven Dataset for proposed descriptor (256 bit)

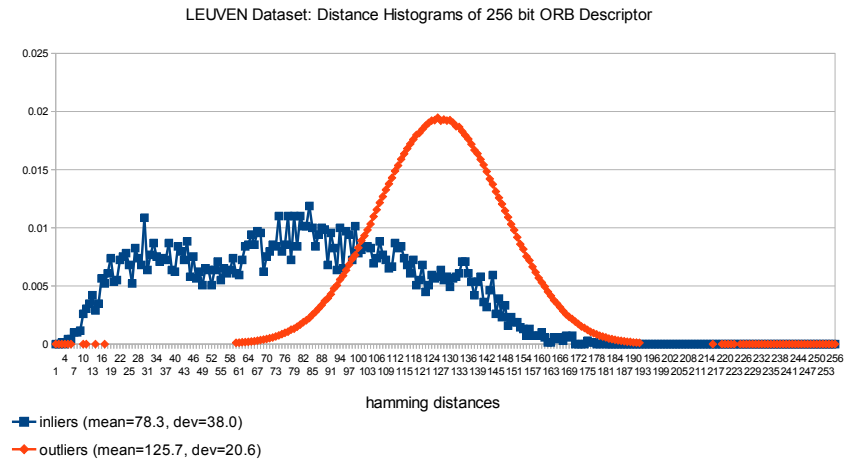


Figure 84 Distance histograms of Leuven Dataset for ORB descriptor (256 bit)

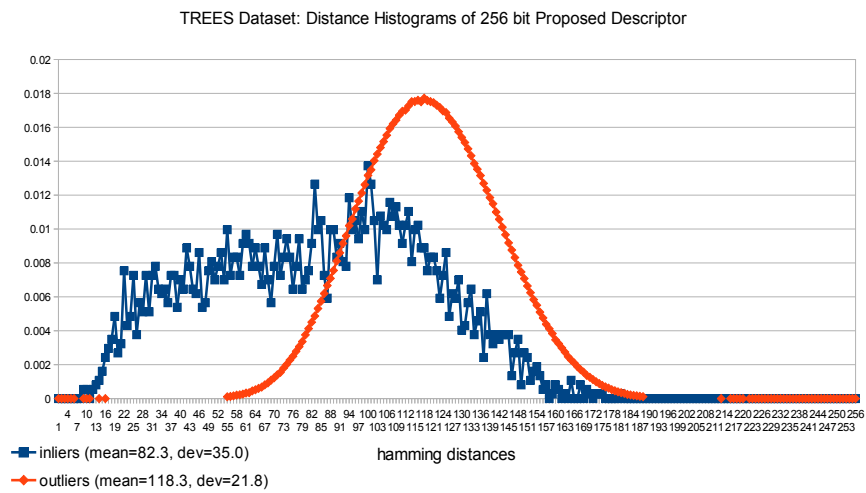


Figure 85 Distance histograms of Trees Dataset for proposed descriptor (256 bit)

TREES Dataset: Distance Histograms of 256 bit ORB Descriptor

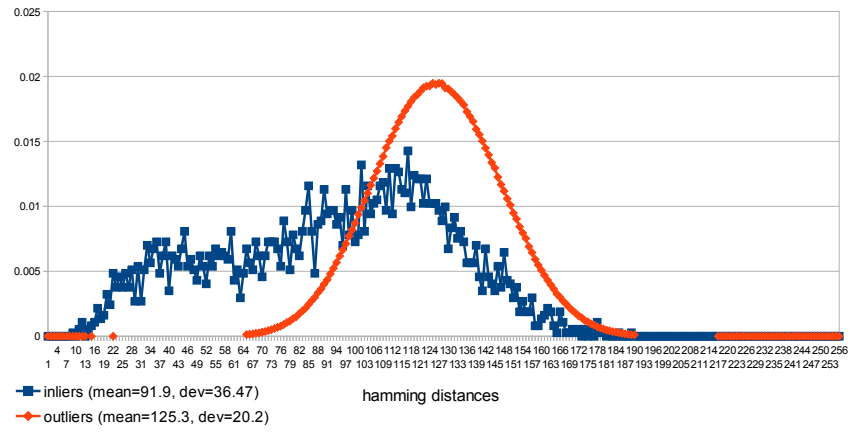


Figure 86 Distance histograms of Trees Dataset for ORB descriptor (256 bit)

WALL Dataset: Distance Histograms of 256 bit Proposed Descriptor

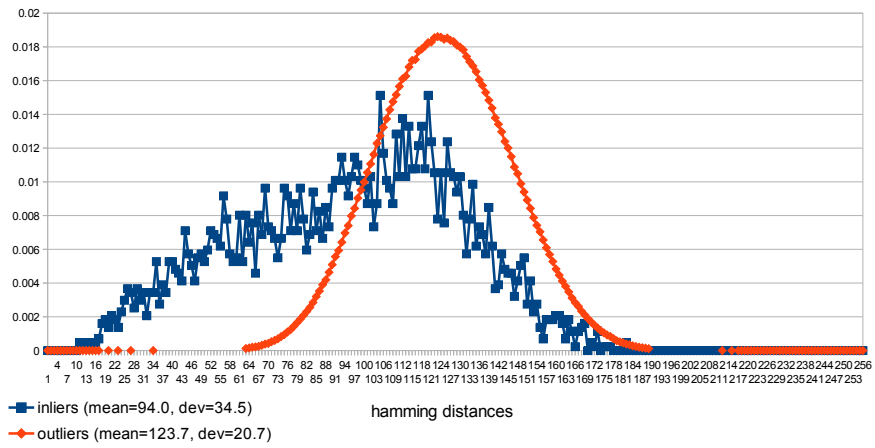


Figure 87 Distance histograms of Wall Dataset for proposed descriptor (256 bit)

WALL Dataset: Distance Histograms of 256 bit ORB Descriptor

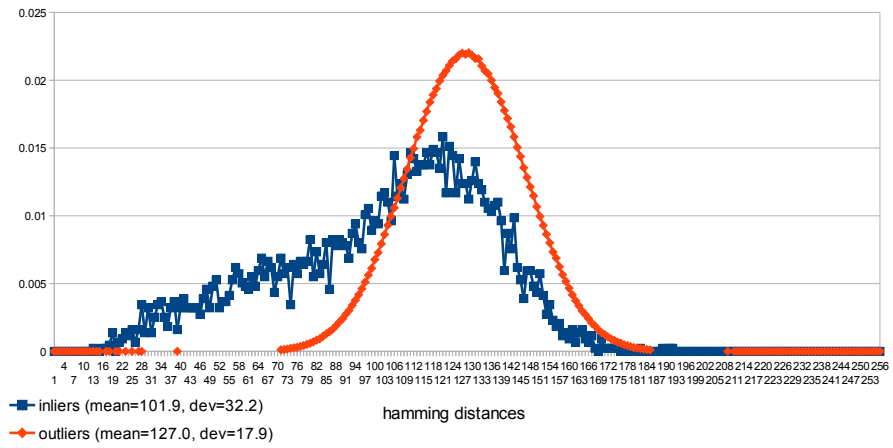


Figure 88 Distance histograms of Wall Dataset for ORB descriptor (256 bit)