

CREAK descriptor evaluation for monocular visual SLAM

DN Smulders

 orcid.org/0000-0003-4523-7848

Dissertation accepted in fulfilment of the requirements for the
degree *Masters engineering in Electronic and Computer
Engineering* at the North-West University

Supervisor: Prof. K.R. Uren

Graduation: May 2020

Student Number: 24058319

ABSTRACT

This dissertation evaluates the novel Colour-based Retina Keypoint (CREAK) feature descriptor with the current state of the art, Fast Retina Keypoint (FREAK) feature descriptor in the context of a visual Simultaneous Localization and Mapping (SLAM) implementation. SLAM implementations are often the solution to autonomous navigation applications because of its profound error corrective capabilities, and therefore the FREAK and CREAK descriptors were evaluated in the context of SLAM. The concept of SLAM can be described as the method in which a robot builds a map of its surrounding environment, whilst simultaneously tracking its own movement through the map. Although the SLAM problem is considered solved on a conceptual level, there is always room for improvement, and the simplest place to look for improvement is in the initial phases of the SLAM algorithm, which provides the information that SLAM will use to estimate the map and robot's position.

One such phase is the descriptor algorithm used. In the specific case of visual SLAM, the vSLAM implementation is dependent on an estimation of the robot's current pose, the location of the observed landmarks in the map, and the location of the robot in relation to the landmarks. This dissertation explores the algorithm that defines the appearance of each landmark, such that they can be recognized and robustly identified. Such a feature defining algorithm is known as a feature descriptor. Two variations of feature descriptors are discussed, namely FREAK – which is more well known in computer vision with a reputation of being robust and efficient, and CREAK – which has been proposed more recently, and although similar to FREAK, boasts superior robustness due to its novel ability to consider colour information in its description.

The descriptors are used in a monocular Visual Odometry (VO) setting, and the trajectory determined on the KITTI Vision Benchmark Suite dataset. Results are obtained, documented and discussed. Finally, SLAM is implemented with the Extended Kalman Filter (EKF) where matched features along with their estimated map coordinates are used as the observed landmarks.

It is shown that the CREAK descriptor is not necessarily a better descriptor compared to FREAK when implementing EKF-SLAM, however, the most significant finding is that of computational times. Despite FREAK being slightly faster than CREAK for monocular VO alone, CREAK is significantly faster than FREAK for the EKF-SLAM implementation presented, due to the nature of CREAK generating fewer – but equally accurate – matches per frame.

Keywords – CREAK, FREAK, keypoint descriptor, SLAM, monocular SLAM, evaluation

TABLE OF CONTENTS

ABSTRACT	I
TABLE OF CONTENTS.....	II
LIST OF FIGURES.....	VI
LIST OF TABLES	IX
NOMENCLATURE.....	X
CHAPTER 1 - INTRODUCTION.....	1
1.1 Background and Motivation.....	1
1.2 Aim and Problem Statement.....	2
1.3 Objectives and Methodology	2
1.3.1 Objectives.....	2
1.3.1.1 Feature Descriptor	3
1.3.1.2 Visual Odometry	3
1.3.1.3 SLAM	3
1.3.1.4 Evaluation Environment.....	3
1.3.1.4.1 Verification.....	4
1.3.1.4.2 Validation.....	4
1.3.2 Methodology.....	4
1.3.2.1 Feature Descriptor	4
1.3.2.2 Visual Odometry	5

1.3.2.3	SLAM	5
1.3.2.4	Evaluation Environment.....	5
1.3.2.4.1	Verification.....	5
1.3.2.4.2	Validation.....	6
1.4	Publications	6
1.5	Document Layout	7
CHAPTER 2 - LITERATURE SURVEY		8
2.1	Introduction	8
2.2	Vision-based SLAM	8
2.3	Feature Detectors	9
2.3.1	SIFT	9
2.3.2	SURF	10
2.3.3	FAST	10
2.3.4	ORB	11
2.4	Feature Descriptors.....	11
2.4.1	FREAK	11
2.4.2	CREAK.....	13
2.5	Visual Odometry	14
2.6	Conclusion.....	15
CHAPTER 3 - FEATURE DESCRIPTORS.....		16
3.1	Introduction	16
3.2	FREAK Design	16

3.3	CREAK Design.....	18
3.4	Simulation Environment	20
3.5	Simulation Results	22
3.5.1	OpenCV FREAK Results	22
3.5.2	Re-created FREAK Results	24
3.5.3	Re-created CREAK Results.....	30
3.5.4	CREAK Experimental Results.....	36
3.6	Discussion	38
3.7	Conclusion.....	39
 CHAPTER 4 - SLAM IMPLEMENTATION		40
4.1	Introduction	40
4.2	SLAM Implementation	40
4.2.1	Monocular Visual Odometry.....	40
4.2.2	Landmark 3D Localization	42
4.2.3	EKF-SLAM	46
4.3	Simulation Environment	49
4.4	Simulation Results	50
4.4.1	Threshold Sweep Results	50
4.4.2	Visual Odometry Results	52
4.4.3	EKF-SLAM Results.....	56
4.5	Discussion	59
4.6	Conclusion.....	60

CHAPTER 5 – CONCLUSIONS AND RECOMMENDATIONS	62
5.1 Introduction	62
5.2 Conclusion.....	62
5.3 Future Work	63
5.4 Closure	64
REFERENCES.....	65
APPENDIX A - PUBLISHED WORK.....	69
APPENDIX B - THRESHOLD SWEEP RESULTS	75
APPENDIX C - VISUAL ODOMETRY RESULTS	79

LIST OF FIGURES

Figure 1.1 - SLAM Conceptual Overview [9].....	1
Figure 2.1 - The vSLAM Process.....	9
Figure 2.2 - FAST Pixel Selection [18].....	10
Figure 2.3 - (a) Density of Ganglion Cells on the Retina [2] and (b) Human Retina into Computer Vision [2].....	12
Figure 2.4 - (a) FREAK Sampling Pattern [2] vs (b) CREAK Sampling Pattern [1]	13
Figure 2.5 - Rod and Cone Distribution on the Retina [31].....	14
Figure 3.1 - FREAK Algorithmic Process.....	16
Figure 3.2 - (a) Freak Sampling Pattern [2] and (b) Orientation Pairs [2]	18
Figure 3.3 - CREAK Algorithmic Process	19
Figure 3.4 - (a) CREAK Sampling Pattern [1] and (b) Orientation Pairs [1]	20
Figure 3.5 - Simulation Environment Operational Flow	21
Figure 3.6 - OpenCV FREAK Descriptor "Graffiti" 1 1 Results	25
Figure 3.7 - OpenCV FREAK Descriptor "Graffiti" 1 2 Results	25
Figure 3.8 - OpenCV FREAK Descriptor "Graffiti" 1 3 Results	26
Figure 3.9 - OpenCV FREAK Descriptor "Graffiti" 1 4 Results	26
Figure 3.10 - OpenCV FREAK Descriptor "Bark" 1 1 Results	27
Figure 3.11 - OpenCV FREAK Descriptor "Bikes" 1 1 Results	27
Figure 3.12 - OpenCV FREAK Descriptor "Bikes" 1 2 Results	27
Figure 3.13 - OpenCV FREAK Descriptor "Bikes" 1 3 Results	28
Figure 3.14 - OpenCV FREAK Descriptor "Leuven" 1 1 Results	28
Figure 3.15 - OpenCV FREAK Descriptor "Leuven" 1 2 Results	28

Figure 3.16 - OpenCV FREAK Descriptor "Leuven" 1 3 Results	29
Figure 3.17 - OpenCV FREAK Descriptor "Leuven" 1 4 Results	29
Figure 3.18 - OpenCV FREAK Descriptor "Leuven" 1 5 Results	29
Figure 3.19 - OpenCV FREAK Descriptor "Leuven" 1 6 Results	30
Figure 3.20 - CREAK Descriptor "Graffiti" 1 1 Results	31
Figure 3.21 - CREAK Descriptor "Graffiti" 1 2 Results	31
Figure 3.22 - CREAK Descriptor "Graffiti" 1 3 Results	32
Figure 3.23 - CREAK Descriptor "Graffiti" 1 4 Results	32
Figure 3.24 - CREAK Descriptor "Graffiti" 1 5 Results	32
Figure 3.25 - CREAK Descriptor "Bark" 1 1 Results	33
Figure 3.26 - CREAK Descriptor "Bark" 1 2 Results	34
Figure 3.27 - CREAK Descriptor "Bikes" 1 1 Results	35
Figure 3.28 - CREAK Descriptor "Bikes" 1 2 Results	35
Figure 3.29 - CREAK Descriptor "Bikes" 1 3 Results	35
Figure 3.30 - FREAK 1 5 Experimental Results	37
Figure 3.31 - FREAK 1 6 Experimental Results	37
Figure 3.32 - CREAK 1 5 Experimental Results.....	37
Figure 3.33 - CREAK 1 6 Experimental Results.....	38
Figure 4.1 - Monocular Visual Odometry Operational Flow.....	41
Figure 4.2 - Landmark Localization Operational Flow	43
Figure 4.3 - Relative 3D Landmark Vector.....	44
Figure 4.4 - Localizing Landmark from Different Viewpoints	44
Figure 4.5 - Determining Closest Points Between Two Lines	46

Figure 4.6 - EKF-SLAM Operational Flow.....	47
Figure 4.7 - SLAM Simulation Environment Operational Flow	49
Figure 4.8 - Monocular Visual Odometry Feature Matching in the KITTI [5] dataset 00 where Red circles represent newly observed landmarks, and Cyan lines are drawn towards the landmark's previous position	50
Figure 4.9 - FREAK Threshold Sweep.....	51
Figure 4.10 - CREAK Threshold Sweep	51
Figure 4.11 - (a) FREAK Visual Odometry and (b) CREAK Visual Odometry Trajectory for Dataset 00	54
Figure 4.12 - (a) FREAK Visual Odometry and (b) CREAK Visual Odometry Trajectory for Dataset 01	54
Figure 4.13 - (a) FREAK Visual Odometry and (b) CREAK Visual Odometry Trajectory for Dataset 02	55
Figure 4.14 - (a) FREAK Visual Odometry and (b) CREAK Visual Odometry Trajectory for Dataset 03	55
Figure 4.15 - (a) FREAK Visual Odometry and (b) CREAK Visual Odometry Trajectory for Dataset 04	55
Figure 4.16 - (a) FREAK EKF-SLAM and (b) CREAK EKF-SLAM Trajectory for Dataset 00....	58
Figure 4.17 - (a) FREAK EKF-SLAM and (b) CREAK EKF-SLAM Trajectory for Dataset 01....	58
Figure 4.18 - (a) FREAK EKF-SLAM and (b) CREAK EKF-SLAM Trajectory for Dataset 02....	58
Figure 4.19 - (a) FREAK EKF-SLAM and (b) CREAK EKF-SLAM Trajectory for Dataset 03....	59
Figure 4.20 - (a) FREAK EKF-SLAM and (b) CREAK EKF-SLAM Trajectory for Dataset 04....	59
Figure C.1 - Average Error for FREAK and CREAK with Various Keypoints.....	80
Figure C.2 - Average FPS for FREAK and CREAK with Various Keypoints.....	80

LIST OF TABLES

Table 3.1 - OpenCV FREAK Descriptor "Graffiti" Results	22
Table 3.2 - OpenCV FREAK Descriptor "Bark" Results	23
Table 3.3 - OpenCV FREAK Descriptor "Bikes" Results	23
Table 3.4 - OpenCV FREAK Descriptor "Leuven" Results	24
Table 3.5 - Re-created FREAK Descriptor Results	24
Table 3.6 - CREAK Descriptor "Graffiti" Results	30
Table 3.7 - CREAK Descriptor "Bark" Results	33
Table 3.8 - CREAK Descriptor "Bikes" Results	34
Table 3.9 - CREAK Descriptor "Leuven" Results	36
Table 3.10 - FREAK vs CREAK "Graffiti" Experiment Results	36
Table 4.1 - Monocular Visual Odometry Translation Results	52
Table 4.2 - Monocular Visual Odometry Rotation Results	53
Table 4.3 - EKF-SLAM Translation Results	56
Table 4.4 - EKF-SLAM Rotation Results	56
Table 4.5 - Average Detected Features vs Average Matched Features per Frame	57
Table B.1 - FREAK Threshold Sweep Translation Results	75
Table B.2 - FREAK Threshold Sweep Rotation and Computational Time Results	76
Table B.3 - CREAK Threshold Sweep Translation Results	77
Table B.4 - CREAK Threshold Sweep Rotation and Computational Time Results	78
Table C.1 - Visual Odometry Results for FREAK with Various Keypoints	78
Table C.2 - Visual Odometry Results for CREAK with Various Keypoints	78

NOMENCLATURE

List of Abbreviations

BRIEF	Binary Robust Independent Elementary Features
CREAK	Colour-based Retina Key-point
DoG	Difference of Gaussians
EKF	Extended Kalman Filter
FAST	Features from Accelerated Segment Test
FLANN	Fast Library for Approximate Nearest Neighbours
FREAK	Fast Retina Key-point
KF	Kalman Filter
LoG	Laplace of Gaussians
ORB	Oriented FAST and Rotated BRIEF
RANSAC	Random Sample Consensus
RGB	Red Green Blue
RGB-D	Red Green Blue – Distance
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SURF	Speeded Up Robust Features
VO	Visual Odometry
VSLAM	Visual SLAM

Notation

x	Scalar
X	Axis
$\mathbf{x}$	Vector
$\mathbf{X}$	Matrix
$\mathbf{P}_{a b}$	EKF covariance matrix prediction step (a) and update step (b)

CHAPTER 1 - INTRODUCTION

1.1 Background and Motivation

In our current day and age, we have the luxury of advanced computing power at our disposal, and as such, the industry is filled with advances and innovation with regards to automation and autonomous navigation. Annually, at conferences across the globe, academics are presenting new solutions and improvements to the Simultaneous Localization and Mapping (SLAM) problem. SLAM, in short, attempts to build a map of the robot's surroundings whilst simultaneously navigating and localizing itself within the unknown terrain around it. The challenge in SLAM is the inter-dependency between the robot's pose estimate and the estimate for the surrounding map, where both are based on the other's information. Figure 1.1 displays an overview of the SLAM process on a conceptual level. At each time step, SLAM estimates the pose of the robot in the state x_k based on the previously estimated state x_{k-1} and control input u_k . The landmarks are observed by the robot, and the estimated position for each landmark j observed is stored in z_k . The positions of all the observed landmarks as well as the pose for the

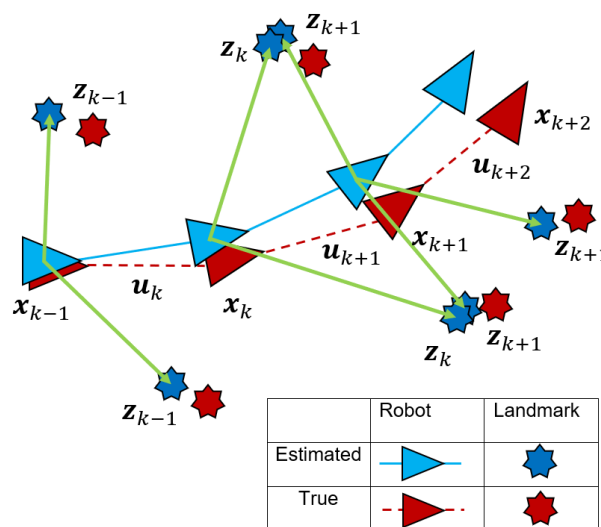


Figure 1.1 - SLAM Conceptual Overview [9]

robot are updated at each time step as the entire map is updated.

Although SLAM has already been proven a theoretical success, there is much room for improvement regarding efficiency. At the time of this dissertation, SLAM is not always practical for small autonomous vehicles that require real-time navigation as the computational requirements for SLAM are too demanding.

It is believed that the solution towards a real-time SLAM lies with visual sensors due to the affordability, light weight, and simplicity of camera sensors. However, the use of camera vision comes with a trade-off. The standard deviation of the noise in depth perception grows quadratically as the object in question moves further from the lens. Furthermore, retrieving depth information from monocular vision is impossible with the camera sensor alone, and determining depth from relative pose estimates provides the distance in terms of relative units at best. Camera vision is also highly dependent on the accuracy and reliability of feature extraction and matching. Thankfully, feature description continues to provide novel methods and approaches, and as such the reliability of a visual-based SLAM is ever increasing.

This dissertation investigates such a novel descriptor named the Colour-based Retina Keypoint (CREAK) [1] descriptor as a feature descriptor having taken inspiration from the inner workings of the human retina. CREAK had made improvements on the Fast Retina Keypoint (FREAK) [2] descriptor by taking colour space information into account, thus leading to a more robust and reliable descriptive algorithm, at the cost of increased computational power.

1.2 Aim and Problem Statement

This study aims to evaluate the performance of the CREAK algorithm as a viable descriptor against FREAK, specifically in the context of a visual-based SLAM (vSLAM) implementation. Only monocular camera input is used for visual odometry, in the pursuit of a SLAM solution performing in real-time. The solution must be applicable to an unmanned vehicle.

Therefore, the problem statement is as follows:

How does the CREAK descriptor perform with respect to the FREAK descriptor in a basic vSLAM environment in the context of an unmanned vehicle application?

1.3 Objectives and Methodology

1.3.1 Objectives

The aim of this dissertation is to evaluate the CREAK descriptor in the specific context of monocular visual SLAM. In order to achieve this end result, the main research problem is subdivided into objectives, each of which contains a sub-problem that requires solving before the next step can commence. Successful implementation of each of the following objectives include an extent of verification as well as to ensure that the solution found for each problem is valid and provides factual results. This section discusses the various aforementioned objectives.

1.3.1.1 Feature Descriptor

The core of this dissertation relies on the successful re-creation of the CREAK descriptor as proposed in the conference article by Chen *et.al* [1]. The OpenCV library does contain an open-source implementation for FREAK by Alahi *et.al* [2]. However, for this objective, it is necessary to ensure that the performance of the CREAK descriptor is not hindered by sub-optimal programming practices, resulting in a biased comparison against FREAK.

1.3.1.2 Visual Odometry

An implementation of SLAM usually obtains the robot's odometry as derived from the robot's control input. However, no such information exists. Therefore, it is required to perform monocular visual odometry (VO) on the system to provide the required state estimate.

Monocular VO does come with its fair share of hurdles, of which the most prominent include inaccuracy due to monocular vision having only a single visual input. This means that depth perception is impossible with a single frame, and highly complex when taking previous frames into account and often computationally expensive. Therefore, the translation and rotation estimate is highly prone to error. For this objective to be considered completed successfully, monocular VO needs to be achieved within acceptable error. Since the wider scope of this research topic approaches a real-time SLAM solution, monocular VO is attempted whilst keeping computational efficiency in mind.

Another hindrance of using monocular VO is the scale dependence. Any translation and rotation provided by monocular VO must be multiplied by a scale factor in order to provide the real-world odometry.

1.3.1.3 SLAM

The purpose of this dissertation is to evaluate CREAK against FREAK specifically in a SLAM implementation. Therefore, this objective requires a functional SLAM implementation, on which to evaluate the performance of the FREAK and CREAK descriptors. In order to determine whether the SLAM implementation is successful, the SLAM method needs to be tested in isolation. Ensuring that the SLAM implementation operates correctly is essential should the desired 'fair and unbiased' evaluation of the descriptors be obtained.

1.3.1.4 Evaluation Environment

In order to advocate the success of the solution presented in this dissertation, the complete solution will be broken up into its two separate evaluation definitions, namely verification and

validation. Verification will quantify the results obtained and ensure that the solutions presented function just as desired.

Validation will demonstrate that the problem statement had been answered effectively, and that all independent functions contained within this dissertation can be combined to achieve the correct end result.

1.3.1.4.1 Verification

In the context of this dissertation, verification will ensure that each different sub problem had been implemented correctly, as well as quantify measurements of accuracy, defining how well the implementation functions independently. Successful verification of the work performed in this dissertation entail that each of the following sub-problems are solved and then shown to be within acceptable error.

- FREAK re-creation
- CREAK re-creation
- Monocular VO implementation
- SLAM implementation

1.3.1.4.2 Validation

Validation will quantify the effectiveness of the complete SLAM application on an unmanned vehicle and in the context of the problem statement of this dissertation, the presented solutions will be valid if a fair and unbiased evaluation of the CREAK descriptor can be performed in the specific context of visual SLAM on an unmanned vehicle.

1.3.2 Methodology

Now that the various objectives had been discussed, this section will present the methodology followed to accomplish each required task. For each aforementioned objective, a summary will be presented to address the process followed for each sub-problem that required solving.

1.3.2.1 Feature Descriptor

In order to eliminate bias in evaluation resulting from possible sub-optimal programming from our side, both FREAK as well as CREAK will be re-created from scratch, and due to the fact that all results, such as computational times, are relative to the machine on which tests were performed, the comparison of the descriptors can be trusted to be unbiased and unhindered.

To ensure that the descriptors have been re-created successfully from a functional point of view, the re-created FREAK descriptor must be compared to the OpenCV implementation of FREAK. This will allow us to point out any discrepancies and therefore verify the results of our FREAK implementation. In order to verify the CREAK implementation, the re-creation of the results found in the aforementioned article by Chen *et.al* is attempted in order to compare their results between FREAK and CREAK, to our implementations of FREAK and CREAK. The results will be gathered from the datasets provided by the University of Oxford Visual Geometry Group [3].

1.3.2.2 Visual Odometry

In order to perform visual odometry, a method will be followed similar to that by Singh in [4], where the Essential Matrix is used to estimate the odometry between two monocular frames at two successive time steps. This will be performed on the KITTI Vision Benchmark Suite [5] odometry and SLAM dataset. The dataset consists of visual input from a pair of stereo cameras, but for the purposes of performing monocular odometry, only one of the pair of images will be used. The dataset is coupled with ground truth data, thus allowing for a quantifiable result in the form of translational and rotational error.

In this study, the aim is merely to evaluate the CREAK descriptor against FREAK, and as such we can argue that the method for determining the scale factor will not influence the evaluation. Therefore, for the purposes of simplicity, it is assumed that the translational scale factor had been pre-derived from an additional sensor such as GPS or derived from control input and will simply use the scale calculated from the ground truth.

1.3.2.3 SLAM

In order to ensure that SLAM had been implemented successfully, the implementation of EKF-SLAM will be similar to the approach by Riisgaard and Blas in [6] and Blanco in [7]. In order to verify the implementation, a simulation environment needs to be set up to provide the system with landmarks along with their determined world coordinates, and the robot's estimated state odometry. These estimated results can be compared to ground truth information in order to provide a quantifiable result in the form of translational and rotational error correction. The KITTI Vision Benchmark Suite [5] will once again be used.

1.3.2.4 Evaluation Environment

1.3.2.4.1 Verification

The following sub-problems will be verified by using the methods discussed below:

- The re-created FREAK descriptor will be compared to that of the FREAK class found in the OpenCV library. The matching evaluation will be performed on the University of Oxford Visual Geometry Group [3] datasets, where matching robustness and computational times are measures quantitatively.
- The re-created CREAK descriptor can then be compared to the re-created FREAK descriptor, with a similar method followed as described above by using the University of Oxford Visual Geometry Group datasets and measuring the matching robustness and computational times.
- The monocular VO implementation will be verified by following a method similar to that by Singh in [4], where the Essential Matrix is used to estimate the odometry between two monocular frames at two successive time periods on the KITTI Vision Benchmark Suite [5] odometry and SLAM dataset. The results can be quantitatively evaluated by measuring the translation and rotational error between the VO and the ground truth included in the dataset.
- The SLAM implementation can be verified by combining the descriptors and monocular VO information and implementing them as input to the EKF-SLAM algorithm. Estimations from the EKF-SLAM filter can be measured quantitatively by comparing the localization before and after the application of SLAM to the ground truth. The KITTI Vision Benchmark Suite odometry and SLAM datasets are used.
Thereafter, the CREAK descriptor can then be evaluated against the FREAK descriptor in a SLAM specific implementation.

1.3.2.4.2 Validation

The implemented solutions in this dissertation would collectively be considered valid should all of the following questions be answered with ‘Yes’:

- Is the CREAK descriptor evaluated in a SLAM implementation?
- Is the EKF-SLAM implementation shown to be viable on an unmanned vehicle?
- Is the comparison of the evaluations between the FREAK and CREAK descriptors considered fair in the context of SLAM?

1.4 Publications

The article “CREAK descriptor evaluation for visual odometry” is a side product from this dissertation, which explores the CREAK descriptor against the FREAK descriptor in the context of visual odometry. This article evaluates the two descriptors on the KITTI Vision Benchmark

Suite dataset and documents the descriptor's performance with varying amounts of features detected.

The article was presented at the Pattern Recognition South Africa (PRASA) 2019 conference and is attached in APPENDIX A – Published Work.

1.5 Document Layout

In this section, the strategy followed is described, corresponding to the various chapters in this dissertation.

In Chapter 2, background is given on all major concepts used to achieve the final visual SLAM solution, and the literature presenting the aforementioned concepts are discussed. In the literature study, the basic conceptual process of SLAM is defined, discussing the techniques used in our approach, such as the Extended Kalman Filter (EKF) used for state estimation and the Essential Matrix used to estimate the pose from visual odometry. Furthermore, Chapter 2 provides a discussion of the feature detectors used to extract features at each frame, and finally the FREAK and CREAK descriptors.

Chapter 3 will begin the design and implementation phases of the FREAK and CREAK descriptors. The design phase of FREAK and CREAK will provide an in-depth discussion regarding the process followed upon the recreation of the descriptors, as well as explanations regarding the functionality of both FREAK and CREAK. The performance of the descriptors will then be verified to ensure a successful and unbiased recreation.

Chapter 4 will be focussed on the application of a visual SLAM solution, where the design and implementation process of monocular visual odometry, as well as SLAM itself, will be discussed in greater detail. The method in which landmarks' 3D coordinates were extracted from monocular vision will also be discussed in detail. Next, the simulation environment will be defined, within which the implementation of the visual odometry and SLAM will be verified and validated, such as to confidently continue. The results are then discussed, and the evaluation of CREAK as a descriptor in the context of visual-based monocular SLAM will be given. Chapter 5 will conclude this dissertation with a summary of the findings in the study, and recommendations for future research and how this study can aid in the long-sought-after, real-time SLAM solution.

CHAPTER 2 - LITERATURE SURVEY

2.1 Introduction

This chapter provides the required background for the major concepts used in the vSLAM implementation. The technologies used in this study, as well as similar novel methods, are discussed to advocate the selection between various alternatives where applicable. Each relevant method is briefly discussed in concept, referring to the appropriate sources.

2.2 Vision-based SLAM

Simultaneous Localization and Mapping (SLAM) refers to the general concept of simultaneously estimating a pose for a given robot, as well as a map within which the robot moves as stated by Burgard *et al.* [8] as well as Whyte and Bailey [9]. The solution to the SLAM problem is unique to the system on which SLAM is incorporated and is usually designed to utilize some form of sensory data retrieved. Should the sensors be cameras, such as in [10] and [11], the SLAM problem becomes categorized as vision-based SLAM (vSLAM) and requires additional steps to translate sensory data into map coordinates.

Due to the unavoidable uncertainty that is present when estimating a robot's pose and location, especially when making use of monocular vision such as in [12], a probabilistic approach to SLAM is presented in [13], where Smith *et al.* applied the Extended Kalman Filter (EKF) described in [14] and [15] to the SLAM problem. EKF-SLAM is especially useful when considering a system that has higher uncertainty due to lots of input noise or a monocular vision system where localization of points cannot be triangulated as easily, as opposed to stereo vision systems. EKF-SLAM functions by estimating the robot pose based on control information in the update step, then the measurement step considers the observed information along with the estimated robot pose and adjusts all estimations in the map accordingly. After each time step, the system converges to a more accurate state.

According to Brink in [11] and Riisgaard and Blas in [6], the vSLAM process can be summarized to follow an order of operation similar to Figure 2.1 at each time step, where the first step is to extract landmarks from the visual input, then to derive the odometry estimate for the robot. The 3D coordinates for each of the landmarks are then determined. The estimated robot odometry is used to generate a pose estimate for the robot in the EKF's update step. Thereafter, the newly extracted landmarks are compared to pre-existing landmarks in order to determine if they are the same landmark in question. For each landmark associated with an existing landmark, the EKF is updated to estimate any innovations in the robot's pose or landmarks' coordinates in the

measurement step. Otherwise, each landmark determined to be a new landmark is appended to the system map. This process is described in detail by Blanco in [7] and Solà in [16].

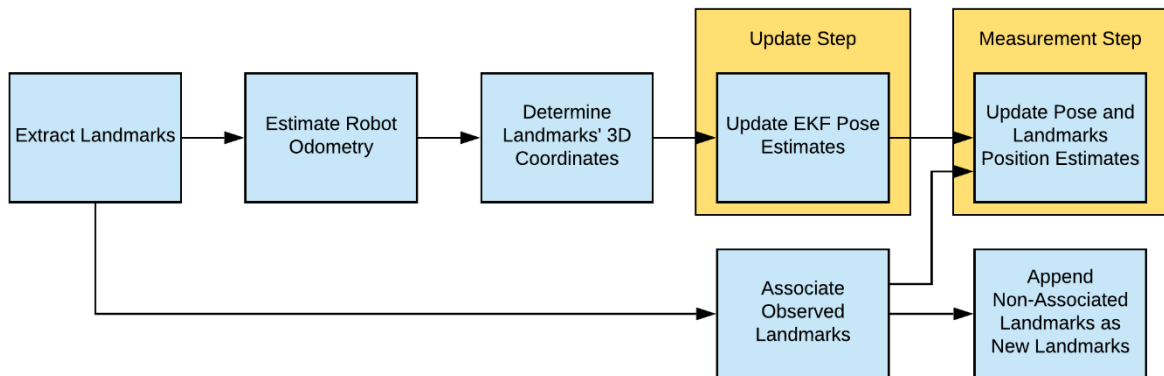


Figure 2.1 - The vSLAM Process

2.3 Feature Detectors

Detecting features is often the first step in many vision-based applications such as tracking, vision-based identification, and SLAM. Feature detectors, as the name implies, serve the purpose of detecting features or points of interest on an image or frame, usually in the form of corners or lines. According to *Visvanathan* in [17], a point of interest is a well-defined position that can be robustly detected, has a high local information content and should be repeatably detected across different viewpoints of the same scene.

2.3.1 SIFT

In 2004, Lowe came up with a new feature detection algorithm, the Scale Invariant Feature Transform (SIFT) [19]. The Harris Corner detection as in [21] is fast and effective, whilst maintaining rotation invariance. However, Harris Corner detection is not scale-invariant. The SIFT algorithm solves this problem by detecting the scale-space extreme. The Laplace of Gaussian (LoG) is found in each filtered scale-space for various σ values. The LoG acts as a blob detector that detects blobs of various sizes based on a change of the scale factor σ . Because LoG proves to be computationally expensive, the SIFT algorithm instead determines the Difference of Gaussian (DoG) between two Gaussian blurred images scaled by two different factors of σ on different octaves of the image.

The potential features are detected across images by searching for local extrema over scale and space, by comparing a pixel with its 8 neighbours and 9 corresponding pixels in the next and previous scales. The SIFT algorithm further assigns orientation to each detected feature to

make it rotation invariant and describes the feature by creating an orientation histogram, represented by a vector to form the descriptor.

2.3.2 SURF

Bay *et al.* proposed an improved feature detector to the SIFT algorithm, named the Speeded Up Robust Features (SURF) [20] feature detector. Like SIFT, SURF approximates the LoG. Unlike SIFT, SURF determines the LoG by using a box filter to assign a pixel a value based on the average of the nearest neighbouring pixels. The benefit of this is that a box filter can easily be calculated with the help of integral images and can be done in parallel for different scales.

SURF assigns orientation by determining a wavelet response in the vertical as well as horizontal directions. This is also a trivial feat when making use of integral images for any scale. SURF does have the option to increase performance by disabling the orientation component, which maintains robustness up to approximately $\pm 15^\circ$.

2.3.3 FAST

Rosten and Drummond proposed the Features from Accelerated Segment Test (FAST) feature detector [18] as a solution to the real-time shortcomings that part with previous detectors, such as the Scale Invariance Feature Transform (SIFT) [19], the Speeded Up Robust Features (SURF) [20] and Harris corner detection [21]. FAST proved to be a highly efficient detector, as shown in experimental results performed by Miksik and Mikolajczyk in [22] where the FAST detector achieved the lowest computation time as well as the largest number of features. As shown in Figure 2.2 below, FAST functions by firstly selecting a pixel of interest p . Thereafter, a circle of 16 pixels surrounding p is compared against a threshold intensity value. Should a contiguous number of N pixel intensities surrounding p be within a threshold value, p will be considered a feature.

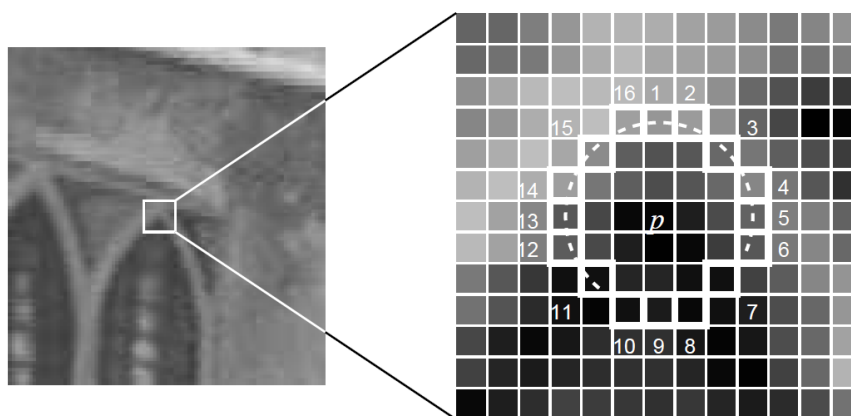


Figure 2.2 - FAST Pixel Selection [18]

2.3.4 ORB

More recently, Rublee *et al.* in [23] proposed the Oriented FAST and Rotated BRIEF (ORB) detector. Rublee noticed that FAST does not contain any measure of how well the corners are defined, and therefore employed a method to make use of a Harris corner measure as in [24] to place each FAST feature in order according to its cornerness. ORB then allows the top N features to be selected, allowing the luxury of selecting only the most robust N features. However, despite the apparent advantages of ORB, Feng *et al.* in [25] and Patel *et al.* in [26] determined that when coupled with a separate descriptor, ORB would achieve slightly slower overall performance, and fewer matches when compared to the original FAST detector. However, ORB does support its own feature descriptor, which is the BRIEF descriptor. BRIEF on its own does not support any rotation, so ORB corrects this by altering BRIEF according to the orientation of the features.

2.4 Feature Descriptors

Arguably, the most important step in any feature tracking or SLAM application is feature description. Feature descriptors serve the purpose of allocating each feature with a unique form of identification, and this can be categorized into two main groups: floating-point-based and binary descriptors. The SIFT and SURF detectors mentioned in a previous section can serve both the functionality of detectors as well as descriptors. When used as descriptors, both SIFT and SURF fall into the category of floating-point based descriptors, which means that they each use a vector of histograms to describe the feature. In order to match these features, the Euclidian distance is calculated, which can be computationally expensive.

Calonder *et al.* introduced the Binary Robust Independent Elementary Features (BRIEF) [27] descriptor as a solution to computationally expensive matching. This was the genesis of binary descriptors, which use a string of binary bits as their descriptor and as such, the Hamming distance can instead be calculated in order to determine matches. Soon other binary descriptors were introduced such as the Binary Robust Invariant Scalable Keypoints (BRISK) [28], DAISY [29], LATCH [30] and ORB, which also boasts in the dual functionality of detector as well as descriptor.

2.4.1 FREAK

Adhering to the trend of binary descriptors, Alahi *et al.* proposed the Fast Retina Keypoint Descriptor (FREAK) [2], which is modelled after the human retina. The retina consists of multiple concentric circles, that each has a concentration of ganglion cells that gets denser the closer they are to the centre of the eye (known as the fovea) as shown in Figure 2.3 (a). The

various elements present in the human retina that provide vision are translated into computer terms as shown in Figure 2.3 (b). The photoreceptors determine brightness, and thus in computer terms, they determine the intensity of pixels within an image. The ganglion cells responsible for comparing inputs from photoreceptors are then used to compare pixel intensities at different areas on the image to one another. Clusters of ganglion cells are referred to as kernels.

Since the spatial distribution of ganglion cells diminishes with exponential proportion to the distance from the fovea, they are segmented into three areas from inner to the outer retina: fovea, parafovea, and perifovea. Each area is responsible for capturing a specific resolution where the fovea has the highest resolution due to it having the highest concentration of ganglion cells, and vice versa for the perifovea. Figure 2.5 demonstrates how this had been recreated in the FREAK descriptor.

The FREAK descriptor achieves its novelty by the creation of a specific sampling pattern which consists of equally spaced points on concentric circles, similar to that in DAISY [29]. In order to match the retinal model, different kernel sizes are used, with smaller, more dense kernels used nearer to the centre, similar to that used initially in BRISK [28]. The respective average intensities of each kernel are compared to one another, and a bit is set depending on the difference in intensities. However, since there are 43 kernels, leaving a total of 903 possible pairs, not all pairs need to be compared as not all convey useful information. Therefore, an approach similar to ORB [23] is taken to learn the best 512 matching pairs for the descriptor from training data. Since 512 matching pairs are considered, the descriptor is 512 bits (or 64 bytes) in size.

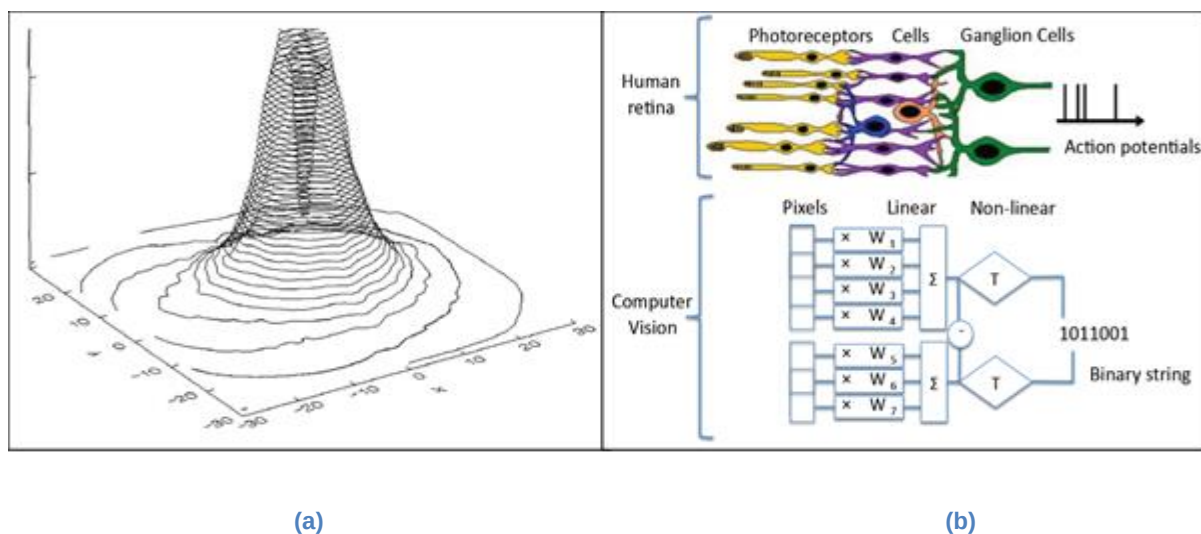


Figure 2.3 - (a) Density of Ganglion Cells on the Retina [2] and (b) Human Retina into Computer Vision [2]

2.4.2 CREAK

The Colour-based Retina Keypoint descriptor (CREAK) [1], proposed by Chen *et al.* is a novel descriptor modelled after the human retina and is built upon the foundations laid by FREAK. Due to the nature of modern binary descriptors such as ORB, BRIEF and FREAK, only the grey intensity values of images are considered. This negates a significant amount of descriptive information that can be achieved by considering the blue, red and green colour spaces as well. According to Chen *et al.* taking the colour-space information into account will significantly aid in creating a descriptor that is more robust and discriminative.

Similar to FREAK, the CREAK descriptor functions with a deliberate sampling pattern, recreated to mimic the human retina and the distribution of ganglion cells. However, in the CREAK descriptor, significant alterations had been made to the design of the sampling pattern, as shown in Figure 2.4. Due to overlapping that occurs in the innermost kernels of FREAK, spatial redundancy is high and therefore CREAK promotes better discriminative power by shrinking the size of the kernels in question, hence reducing redundancy. This is further supported by the photoreceptive cones (responsible for colour vision) being denser in the centre of the fovea [31] as shown in Figure 2.5.

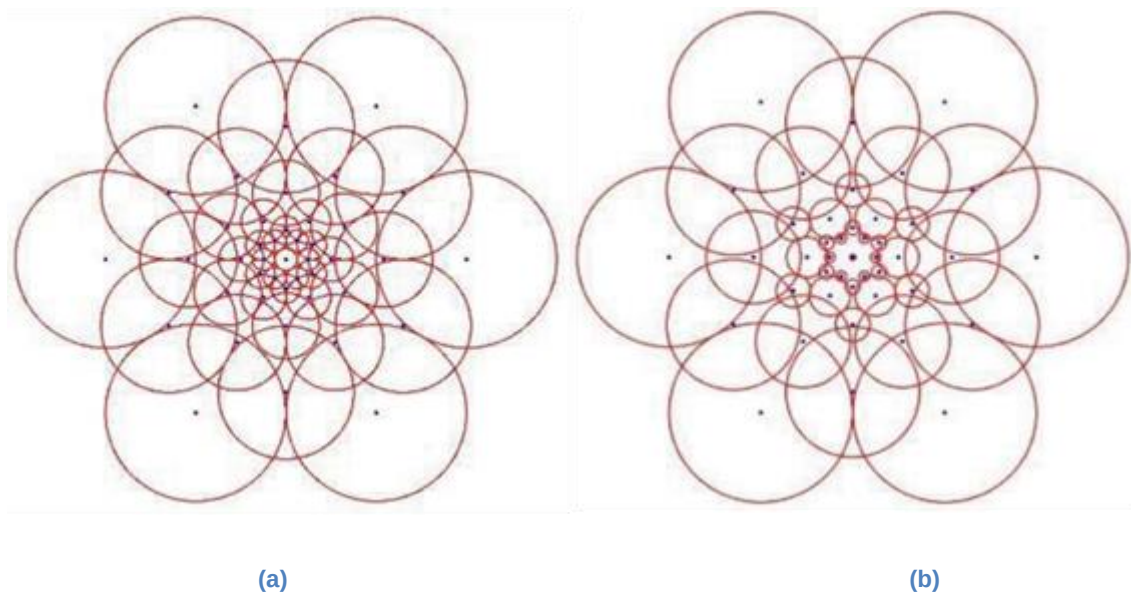


Figure 2.4 - (a) FREAK Sampling Pattern [2] vs (b) CREAK Sampling Pattern [1]

Similar to FREAK, not all pairs need to be compared as not all convey useful information. Therefore, the same approach is taken to learn the best 64 matching pairs for each colour space in the descriptor from training data. Since there are 64 pairs per colour space and 3

colour spaces, the result is a total of 192 matching pairs, meaning that the descriptor is 192 bits (or 24 bytes) in size.

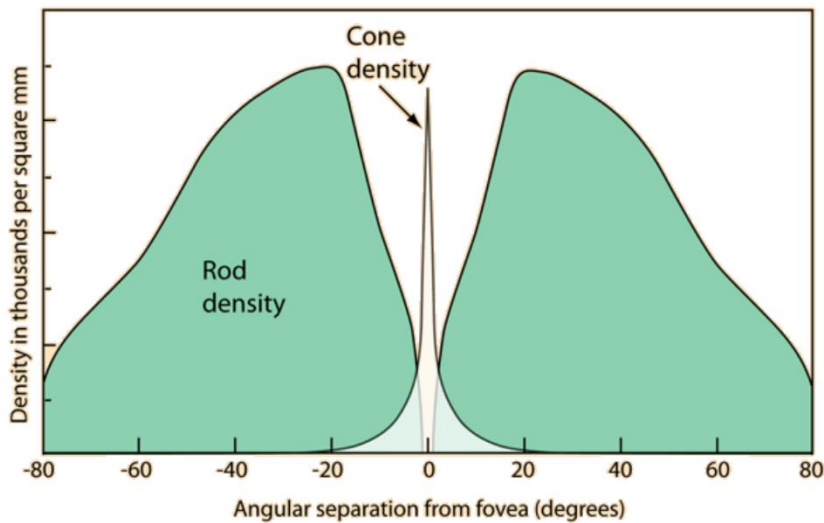


Figure 2.5 - Rod and Cone Distribution on the Retina [31]

2.5 Visual Odometry

Visual Odometry (VO), is the concept of determining a robot's trajectory by using visual input and was proposed by Nistér in [32]. This can be achieved by either using one camera, known as monocular odometry such as SVO proposed by Forster *et al.* in [33], or by using two cameras, known as stereo odometry such as in Circular FREAK-ORB (CFORB) [34]. A third approach exists that makes use of a special piece of camera equipment, known as an RGB-D camera. RGB-D cameras capture images with red, green, blue and distance parameters— where D is the distance that a pixel is from the camera, such as in [35], [36] and [37].

For the purposes of this dissertation, only solutions that apply to monocular vision had been looked at. When it comes to monocular VO, Nistér proposed a solution to the classic five-point problem in [38]. The five-point problem is to find possible solutions for relative camera motion between two different views given five corresponding points on each of the views. Nistér made use of the Essential Matrix proposed by Longuet-Higgins in [39] that draws up a 3x3 matrix $\mathbf{E}$ from which the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$ can be determined. The combination of $\mathbf{R}$ and $\mathbf{t}$ can be used to determine the relative pose of the robot.

To prevent inaccurate odometry due to errors in the feature matching, Random Sampling Consensus (RANSAC) [40] is introduced as an outlier detector. With RANSAC, each time the Essential Matrix is determined, five random points are selected, and the Essential Matrix $\mathbf{E}$

estimated. Then all other corresponding points are tested for compliance with $\mathbf{E}$. After a pre-set number of iterations, the matrix $\mathbf{E}$ with the most inliers is used.

2.6 Conclusion

SLAM is a broad concept that ranges in complexity depending on the implementation used. The EKF algorithm is a natural choice for a vSLAM implementation due to its proficiency in noisy environments, which is the case in monocular vision. The FAST detector is currently the best choice of detector due to its high speed and large number of features detected. The FREAK and CREAK descriptors, based and modelled after the human retina, are both novel descriptors where FREAK is known for its reliable performance. CREAK claims to surpass FREAK by including the functionality of considering colour space information in its description process. With the introductions to the essential algorithms out of the way, the context is set to implement the feature detectors, descriptors, and vSLAM.

CHAPTER 3 - FEATURE DESCRIPTORS

3.1 Introduction

In this chapter, the designs behind the FREAK and CREAK descriptors are discussed in detail. In order to achieve a fair evaluation between FREAK and CREAK in the context of vSLAM, FREAK and CREAK are re-created in a similar fashion, both upon the same foundation of FREAK. The complete re-creation process of the descriptors, as well as their simulation results, are documented in this chapter.

3.2 FREAK Design

The Fast Retina Keypoint Descriptor, as proposed by Alahi *et al.* in [2] is a binary descriptor that provides a unique string of 512 bits that effectively describes and identifies each feature detected within an image. Figure 3.1 shows the general FREAK process. FREAK only operates with greyscale images. Therefore, the first step is to determine whether the image is greyscale

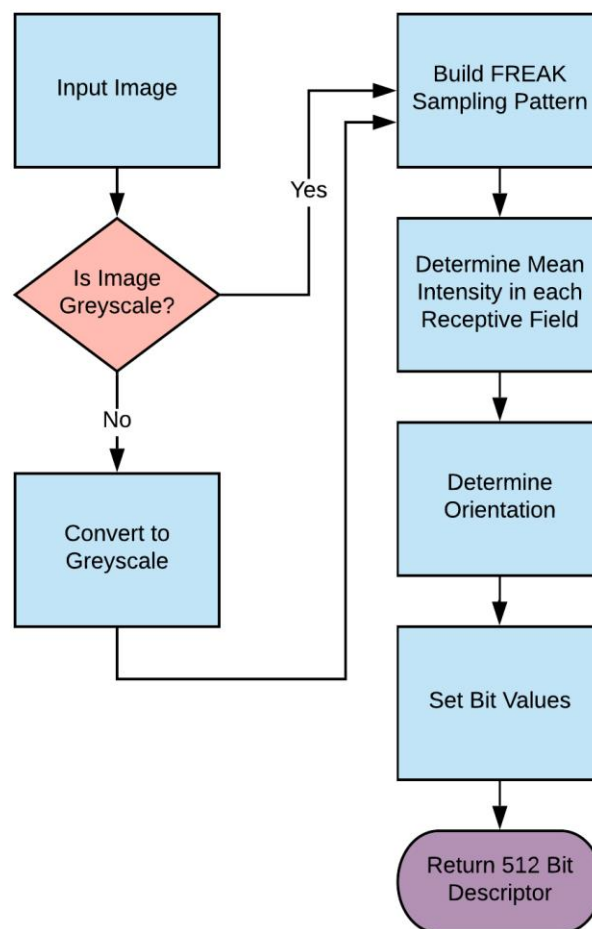


Figure 3.1 - FREAK Algorithmic Process

or not, and to convert it to greyscale if the latter is true. The algorithm then proceeds to consider the pixel location of each feature detected and build the sampling pattern around the feature in question. The sampling pattern is based on the layout of rods and cones in the human retinal system and is shown in Figure 3.2 (a), where each circle is known as a kernel and represents the photoreceptive detectors in the retina. The mean pixel intensity of each of the 43 individual kernels is determined, and the orientation of the descriptor is determined by cross-referencing specific 45 pairs of kernels, as shown in Figure 3.2 (b) and by making use of (3.1). The formula for determining the feature's orientation is given by

$$O = \frac{1}{M} \sum_{P_0 \in G} \frac{[I(P_0^1) - I(P_0^2)](P_0^1 - P_0^2)}{\|P_0^1 - P_0^2\|}, \quad (3.1)$$

where G is the set of all possible pairs used to compute the gradients, and M is the number of pairs used in G , which in the case of FREAK is 45. P_0 is the 2D vector of spatial coordinates of the centre of the kernel and P_0^1 and P_0^2 represent the two points that form the pair. $I(P_0^i)$ refers to the intensity of the point P_0^i . Once the orientation is determined, the descriptor bit-string is set by comparing the 512 pre-determined, best combination of pairs from the 43 kernels to one another and a bit value set for each pair. The best combination of pairs was determined by following a method similar to ORB in [23] in order to learn the best 512 pairs from training data. The sampling pattern corresponding to the matching pairs is rotated by the rotation O as determined in (3.1), in order to achieve a descriptor that is rotation invariant. Equations (3.2) and (3.3) describe how the bits are set for the FREAK descriptor F by comparing each kernel to its pre-determined pair:

$$F = \sum_{0 \leq a < N} 2^a T(P_a), \quad (3.2)$$

where F is a binary vector of length 512, P_a is a pair of receptive fields (kernels) and N is the desired size of the descriptor $T(P_a)$, defined by:

$$T(P_a) = \begin{cases} 1 & \text{if } I(P_a^1) - I(P_a^2) > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (3.3)$$

$I(P_a^1)$ is the smoothed intensity of the first receptive field of the pair P_a and $I(P_a^2)$ is the smoothed intensity of the second receptive field and $a \in [0, 511]$. Finally, the 512-bit long array is returned as the complete descriptor F .

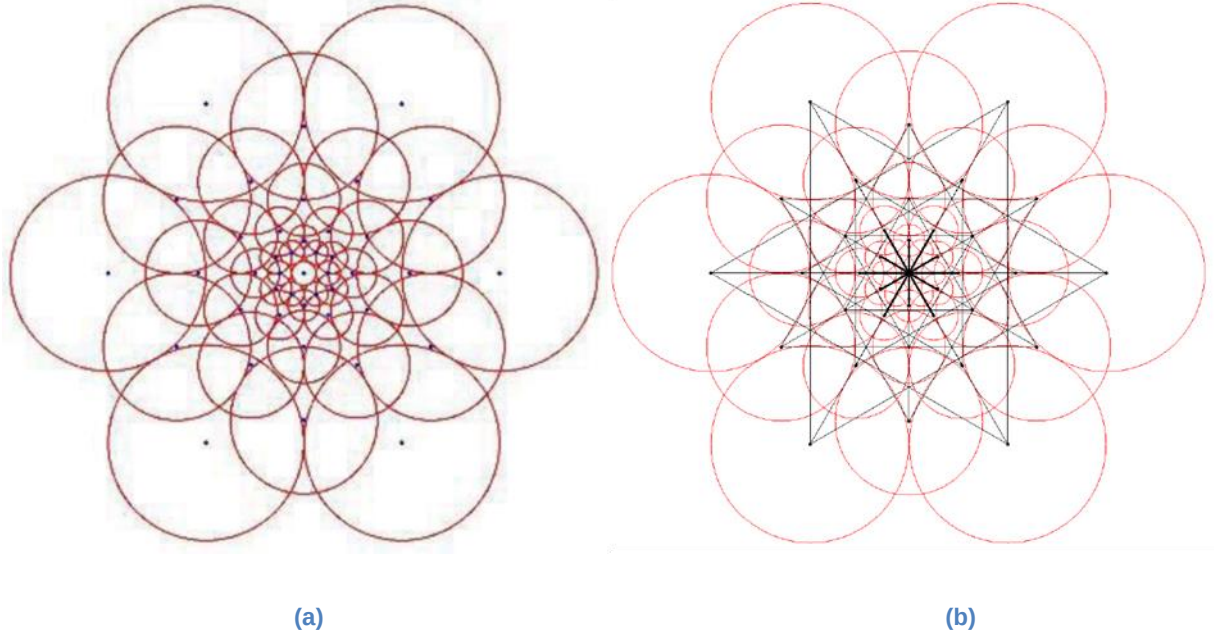


Figure 3.2 - (a) Freak Sampling Pattern [2] and (b) Orientation Pairs [2]

3.3 CREAK Design

The Colour-based Retina Keypoint Descriptor as proposed by Chen *et al.* in [1] is based on the foundation laid by the FREAK descriptor, and as such is modelled after the human retina with minor notable improvements. The CREAK descriptor provides a unique string of 192 bits in length, that describes a feature previously detected within an image. The most significant difference of CREAK compared to FREAK is that, as the name implies, CREAK takes the colour space information of the red, green and blue channels into account. This leads to a more robust feature description.

Figure 3.3 shows the general CREAK process, where the first step is to determine whether the image is greyscale or not, and to return with an error if it is indeed greyscale. The CREAK detector algorithm then splits the 3-channel RGB image into three separate 1-channel images, each representing the blue, green and red colour-spaces. The algorithm then proceeds to consider the pixel location of each feature detected and build the sampling pattern around the feature in question. The sampling pattern is based on the layout of rods and cones in the human retinal system, as is with FREAK; however, the sizes of the circles are changed slightly to accommodate the now included colour-photoreceptors. This is shown in Figure 3.4 (a), where each circle is referred to as a kernel and represents the density of photoreceptive detectors in the retina. The mean pixel intensity of each of the 43 individual kernels are determined, and the orientation of the descriptor determined by cross-referencing a specific 57 pairs of kernels as opposed to the 45 pairs in FREAK, as shown in Figure 3.4 (b) and by making use of the same Equation 3.1. This is repeated for each of the 3 colour spaces.

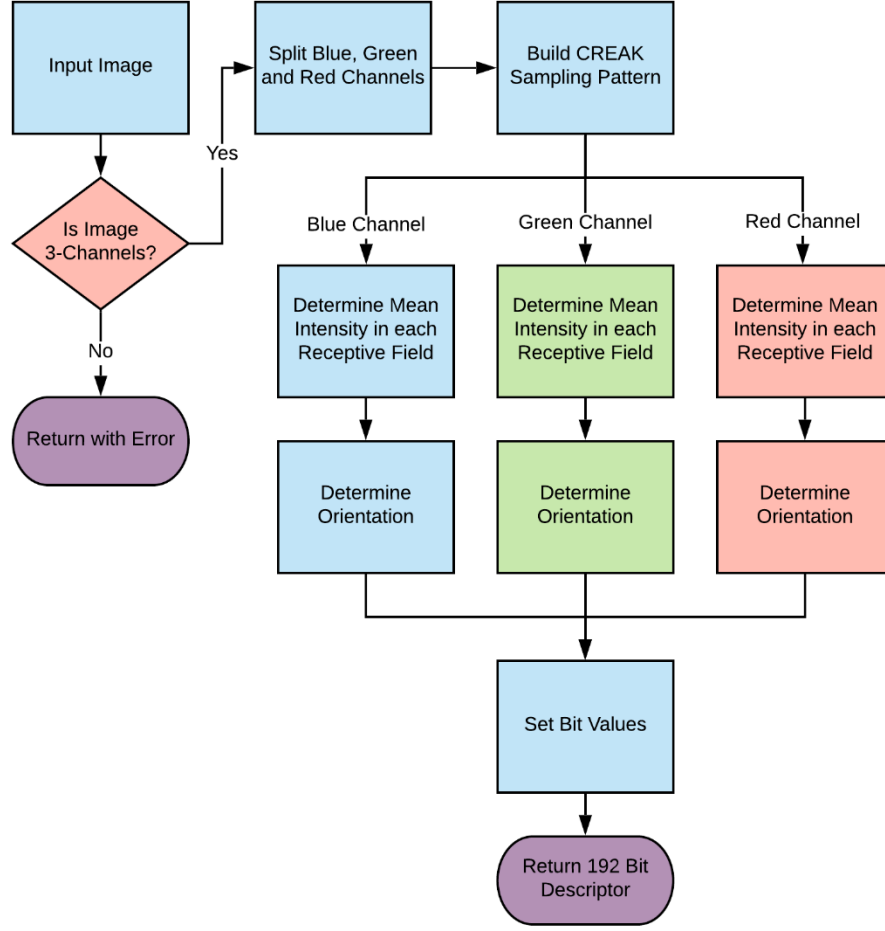


Figure 3.3 - CREAK Algorithmic Process

Once orientation is determined, the 64 best pre-determined combinations of pairs for each colour space will be determined by following a method similar to ORB in [23] in order to learn the best 64 pairs from training data. The pairs are then compared to one another using the same formula as Equation 3.2 and 3.3 as in FREAK, in order to set a 64-bit array. Once there are three 64-bit arrays, one for each colour-space, they will be concatenated as described by (3.4) to provide a single descriptor $\mathbf{D}$ that is 192 bits in length, that is:

$$\mathbf{D} = \sum_{i=0}^{63} [2^i T(B_i) + 2^{i+64} T(G_i) + 2^{i+128} T(R_i)], \quad (3.4)$$

where B , G and R represent the colour-channels blue, green and red, while B_i , G_i and R_i are the colour test pairs of receptive fields for their corresponding channels respectively, and $T(B_i)$, $T(G_i)$, $T(R_i)$ are determined by (3.3) as in FREAK.

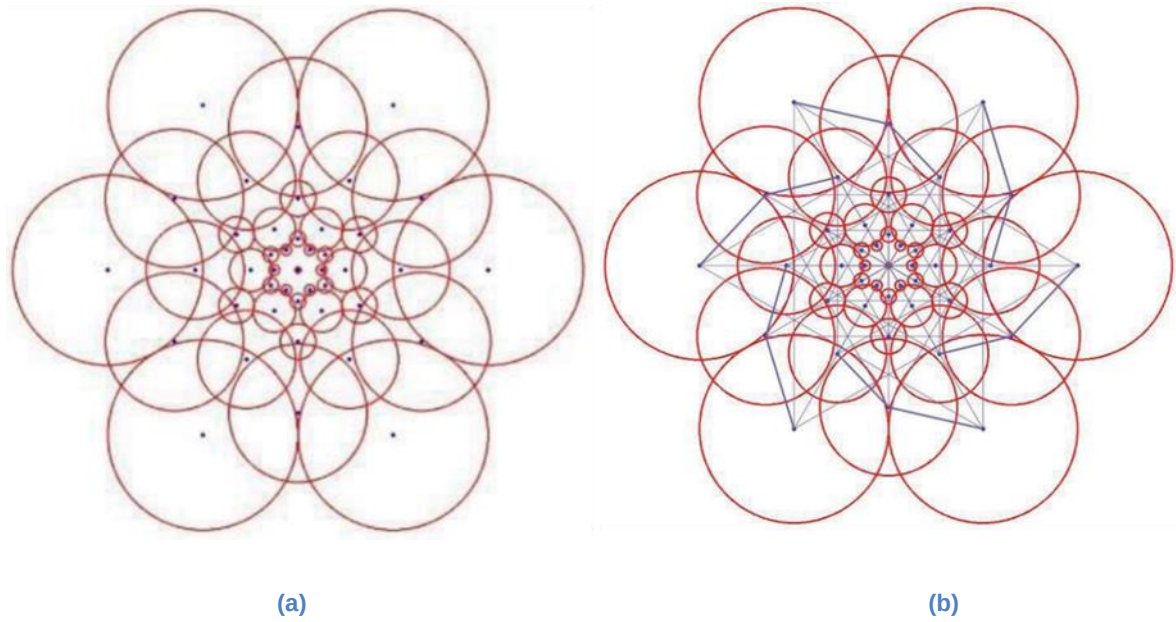


Figure 3.4 - (a) CREAK Sampling Pattern [1] and (b) Orientation Pairs [1]

3.4 Simulation Environment

Although the FREAK descriptor does have a pre-existing, usable function in the OpenCV library [42], it is necessary to re-create FREAK because CREAK will be built upon the functionality of FREAK. Thus, to ensure that results obtained are not inaccurate due to sub-optimal programming practices affecting the efficiency of the algorithm, the FREAK descriptor will also be re-created with the same programming practices. The re-created FREAK can have its results compared to the existing OpenCV implementation of FREAK, and the re-created CREAK results compared to those found by Chen *et al.* Thus, the successful re-creation of the FREAK descriptor can be ensured, and with reasonable certainty deduce that CREAK had been re-created successfully as well.

Both descriptors will be re-created in the Microsoft Visual Studio 2017 environment in C++, with the inclusion of the OpenCV library for necessary functions. The purpose of the simulation software will be to verify the re-creation of the FREAK and CREAK descriptors, and therefore ensure that any later evaluation of the descriptors is fair and unbiased. To evaluate the robustness of the descriptors, the University of Oxford: Visual Geometry Group dataset [3] will be used. The datasets contain various image scenes with varying levels of transformations applied. Figure 3.5 shows the operational flow of the simulation software and can be described as follows: two colour test images will be input, of which one will be a reference image, and the other the same scene with the aforementioned transformation applied. Since neither FREAK nor CREAK has a built-in feature detector, features are to be detected independently for each image by using the Feature Accelerated Segment Test (FAST) feature detector. FAST had been

selected over ORB due to ORB's inferior performance in terms of computational times as well as number of matches, as found by Patel *et al.* in [26]. The detected features will then all be described by using the FREAK and CREAK descriptors, and then features between the two images are matched to one another by thresholding the Hamming distance between the descriptors. The matched features are then displayed visually, where the results can be manually evaluated, and any false positives detected. The computational times are recorded for further evaluation. This process will be repeated for each of the transformation changes, namely: "Graffiti" which represents a change in the viewpoint of the scene, "Bark" which represents a rotation and zoom transformation, "Bikes" which represent a blur transformation and "Leuven" which represents a luminance change. These transformations should sufficiently simulate the real-world effects that an unmanned vehicle might encounter.

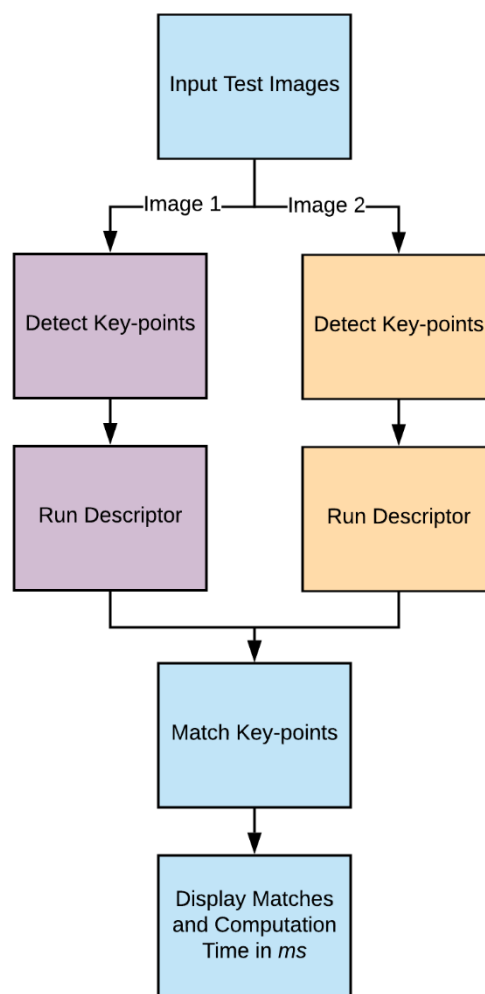


Figure 3.5 - Simulation Environment Operational Flow

3.5 Simulation Results

3.5.1 OpenCV FREAK Results

Using the aforementioned simulation environment, the pre-existing OpenCV FREAK function is used to set a reliable baseline, against which the re-created FREAK descriptor is evaluated. For evaluation, the “Graffiti”, “Bark”, “Bikes” and “Leuven” datasets from the University of Oxford: Visual Geometry Group dataset [3] is used, which represents a transformation in viewpoint, rotation and zoom, blur and luminance respectively. Each dataset consists of 6 images, with the first image acting as the reference image, and the last (6th) image representing the transformation with the most extreme difficulty. Table 3.1 shows the results obtained on the “Graffiti” dataset by using the OpenCV FREAK descriptor. The notation $1|n$ represents the first image matched with the n^{th} image in the dataset. The “Total Features” column represents the number of features detected and described in the n^{th} image. The ‘Time’ column represents the total time taken from the input of the images to the final match between the described features. The number of correct matches is determined by hand, where the output image is analysed, and the outliers counted manually.

Table 3.1 - OpenCV FREAK Descriptor "Graffiti" Results

Matching Pairs ($1 n^{\text{th}}$)	OpenCV FREAK – Graffiti (Viewpoint Change)			
	Total Features on n^{th} image	# Matches between 1 and n	# Correct Matches between 1 and n	Time (s)
1 1	423	423	423	0,058
1 2	577	30	30	0,059
1 3	662	1	1	0,053
1 4	611	1	1	0,058
1 5	646	0	0	0,049
1 6	589	0	0	0,051

It is apparent from Table 3.1, that the increase in extremity of the transformation has the expected results, that is, matching features from viewpoints that have not undergone drastic change remains a trivial feat, and that matching features from significantly different viewpoints becomes much more difficult as can be seen from the 3rd change onwards, where 1|3 and 1|4 only manage one correct match and 1|5 and 1|6 did not manage any matches. It should be noted that the threshold used in determining matches by comparing the Hamming distance between the descriptors had been set to a strict value, eliminating nearly all incorrect matches, but at the trade-off of fewer total matches in general. The same threshold had been used in the entirety of this dissertation and had been set to a strict value with foresight of visual odometry in later chapters, which is very vulnerable to incorrect matches. Table 3.2 shows the results

obtained on the “Bark” dataset by using the OpenCV FREAK descriptor. As stated previously, the notation $1|n$ represents the first image matched with the n^{th} image in the dataset.

Table 3.2 - OpenCV FREAK Descriptor “Bark” Results

Matching Pairs ($1 n^{\text{th}}$)	OpenCV FREAK – Bark (Rotation/Zoom Change)			
	Total Features on n^{th} image	# Matches between 1 and n	# Correct Matches between 1 and n	Time (s)
1 1	16	16	16	0,084
1 2	4	0	0	0,083
1 3	12	0	0	0,085
1 4	79	0	0	0,081
1 5	99	0	0	0,114
1 6	143	0	0	0,116

Table 3.2 paints a picture similar to that in Table 3.1, however, there is a significant difficulty in matching features that had undergone a rotation and zoom transformation, since only the trivial case where no rotation or zoom had been implemented resulted in any matches. Table 3.3 shows the results obtained on the “Bikes” dataset by using the OpenCV FREAK descriptor.

Table 3.3 - OpenCV FREAK Descriptor “Bikes” Results

Matching Pairs ($1 n^{\text{th}}$)	OpenCV FREAK – Bikes (Blur Transform)			
	Total Features on n^{th} image	# Matches between 1 and n	# Correct Matches between 1 and n	Time (s)
1 1	371	371	371	0,177
1 2	23	12	11	0,1
1 3	5	2	2	0,099
1 4	0	0	0	0
1 5	0	0	0	0
1 6	0	0	0	0

With the “Bikes” dataset in Table 3.3, the detector failed to provide any features when the level of blur became too significant. This is due to the nature of FAST and other corner-detection-based feature detectors, which identify a feature based on the sharp intensity changes of the pixels surrounding a feature. When the image becomes blurred, pixel sharpness is naturally reduced, and thus the detection of corners becomes more difficult. It is also worth noting that this dataset resulted in an incorrect match at the 2nd level transformation, despite the strict matching threshold in place.

Table 3.4 shows the results obtained on the “Leuven” dataset by using the OpenCV FREAK descriptor.

Table 3.4 - OpenCV FREAK Descriptor "Leuven" Results

Matching Pairs (1 n^{th})	OpenCV FREAK – Leuven (Luminance Change)			
	Total Features on n^{th} image	# Matches between 1 and n	# Correct Matches between 1 and n	Time (s)
1 1	629	629	629	0,101
1 2	495	146	146	0,119
1 3	419	70	70	0,097
1 4	349	63	63	0,099
1 5	253	31	31	0,152
1 6	170	14	13	0,162

Upon simulating results on the “Leuven” dataset, the findings are concurrent with the previously simulated datasets. However, it is noteworthy that this dataset obtained the second incorrect match for the FREAK descriptor.

3.5.2 Re-created FREAK Results

With the results of FREAK from OpenCV as a baseline, the same simulations were performed on the re-created FREAK descriptor, and the results are shown in Table 3.5. It was found that in terms of matching, the results obtained in our re-created FREAK descriptor are identical to that of the OpenCV equivalent FREAK descriptor. In each experiment, the number of total features, number of matches and the number of correct matches are all the same. The only notable discrepancy appears to be the computational times, that in the case of our re-created FREAK descriptor, is on average 0.03s, or 30ms faster than the OpenCV equivalent.

Table 3.5 - Re-created FREAK Descriptor Results

Matching Pairs (1 n^{th})	“Graffiti” Time (s)	“Bark” Time (s)	“Bikes” Time (s)	“Leuven” Time (s)
1 1	0,03	0,059	0,067	0,084
1 2	0,029	0,063	0,066	0,083
1 3	0,029	0,049	0,066	0,066
1 4	0,023	0,049	0	0,114
1 5	0,026	0,049	0	0,119
1 6	0,023	0,082	0	0,114

Upon further investigation, the discrepancies in computational times are found to be due to the fact that the OpenCV library FREAK descriptor performs various checks and conversions to ensure compatibility on a wide array of systems, and because the purposes of this study do not

require such redundancy, the algorithm had been designed for our specific system in mind, thus eliminating unnecessary bulk code. Figures 3.6 – 3.19 show the results of matches achieved on the various datasets, for various difficulty levels of transformations. In cases where no matches are observed, the images are not shown.

Figures 3.6 – 3.9 show the results of matches between the various viewpoint changes in the “Graffiti” dataset visually, as output from the simulation environment.

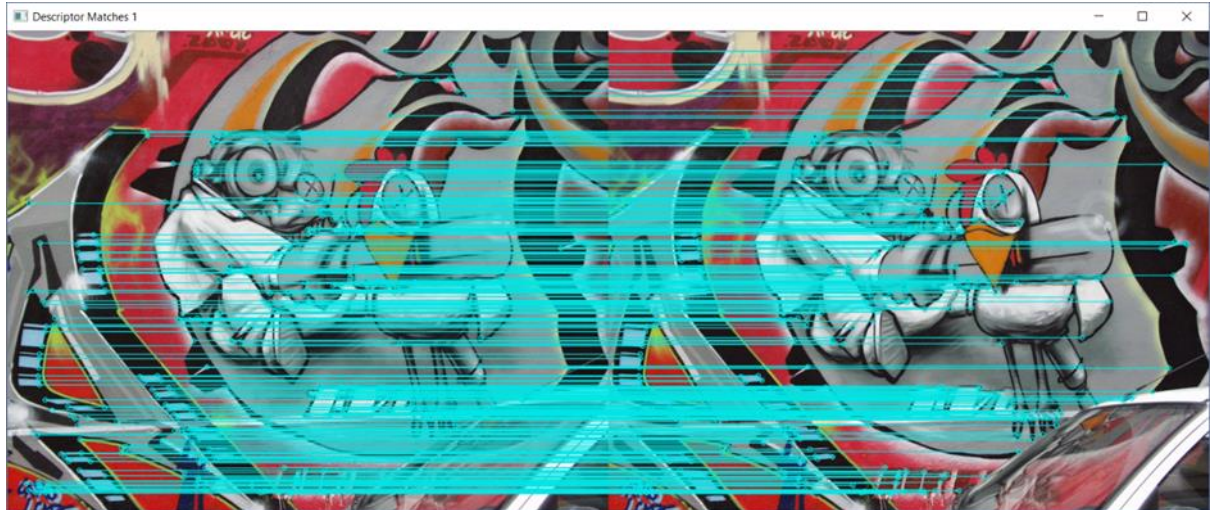


Figure 3.6 - OpenCV FREAK Descriptor "Graffiti" 1|1 Results

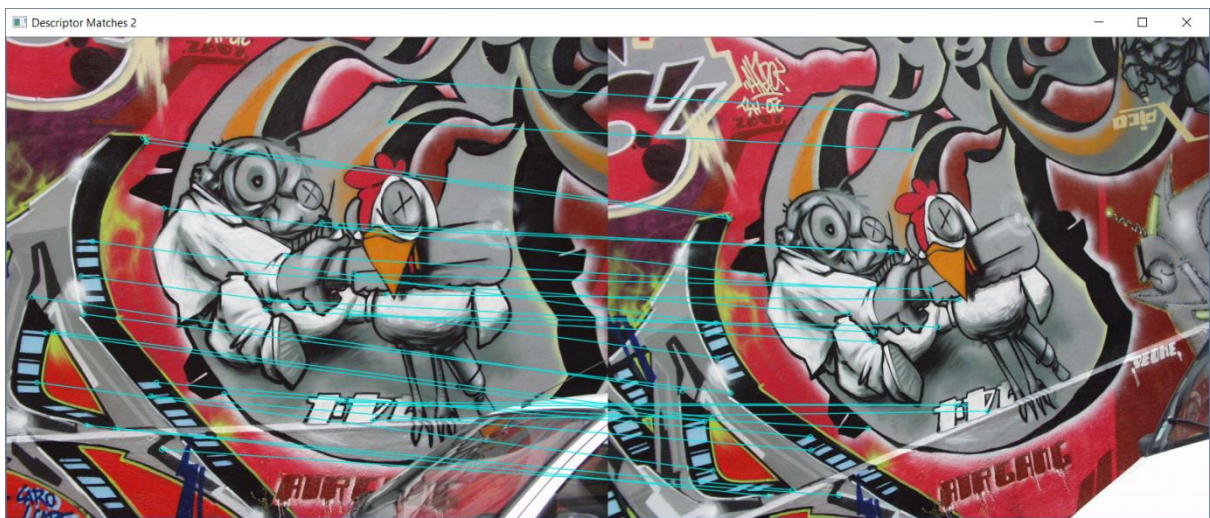


Figure 3.7 - OpenCV FREAK Descriptor "Graffiti" 1|2 Results

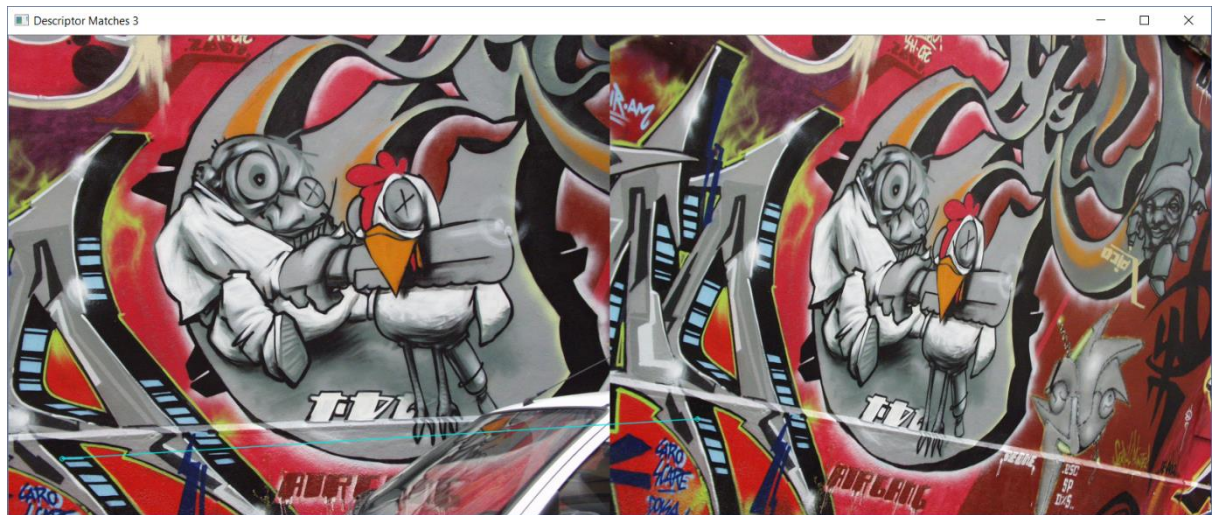


Figure 3.8 - OpenCV FREAK Descriptor "Graffiti" 1|3 Results

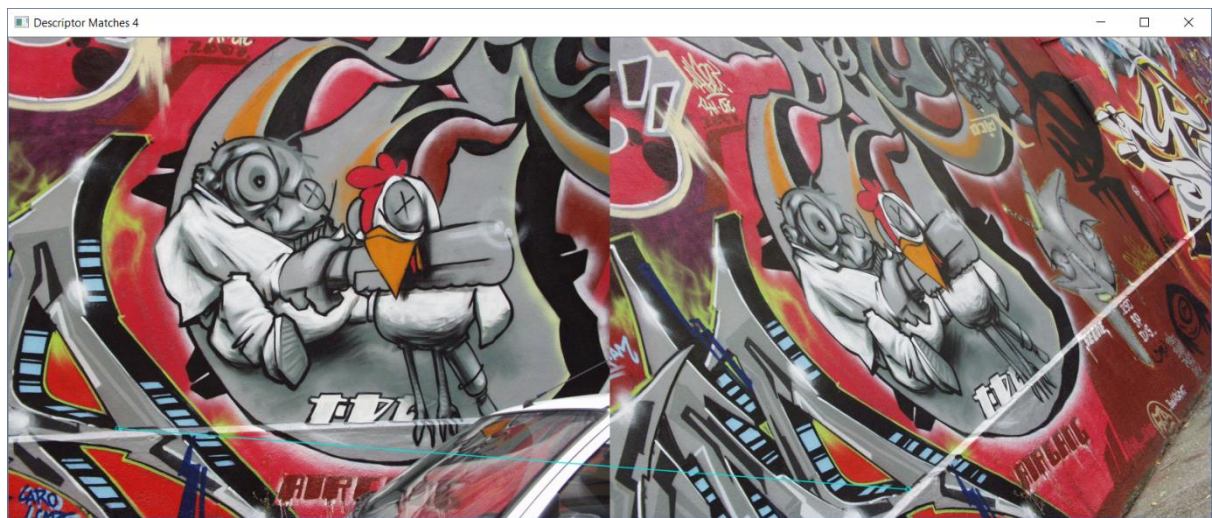


Figure 3.9 - OpenCV FREAK Descriptor "Graffiti" 1|4 Results

Figure 3.10 shows the results of matches between the rotation/ zoom changes in the "Bark" dataset visually, as output from the simulation environment. Solely the one set of results is shown, due to the "Bark" dataset only achieving matches in the first level of transformation. On more difficult transformations, no matches were found at all, deeming the displaying of visual results unnecessary.



Figure 3.10 - OpenCV FREAK Descriptor "Bark" 1|1 Results

Figures 3.11 – 3.13 show the results of matches between the various levels of blur transformations in the “Bikes” dataset visually, as output from the simulation environment.

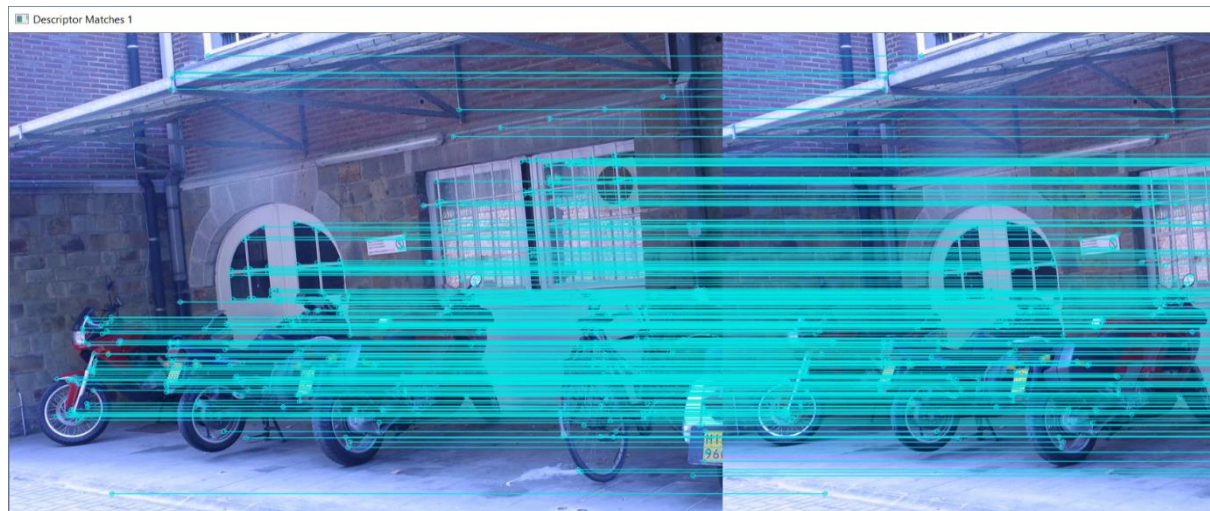


Figure 3.11 - OpenCV FREAK Descriptor "Bikes" 1|1 Results



Figure 3.12 - OpenCV FREAK Descriptor "Bikes" 1|2 Results

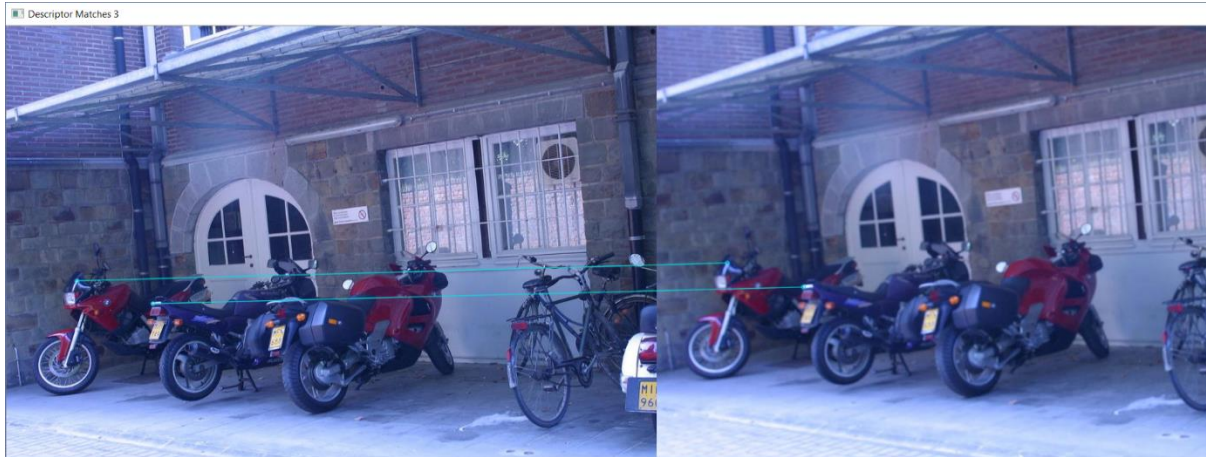


Figure 3.13 - OpenCV FREAK Descriptor "Bikes" 1|3 Results

Figures 3.14 – 3.19 show the results of matches between the various levels of luminance change in the “Leuven” dataset visually, as output from the simulation environment.

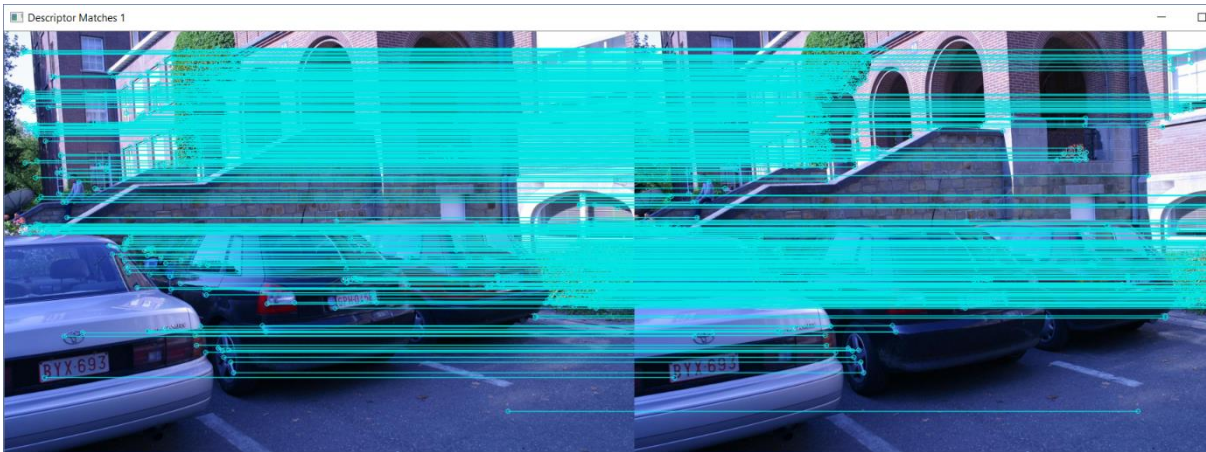


Figure 3.14 - OpenCV FREAK Descriptor "Leuven" 1|1 Results



Figure 3.15 - OpenCV FREAK Descriptor "Leuven" 1|2 Results

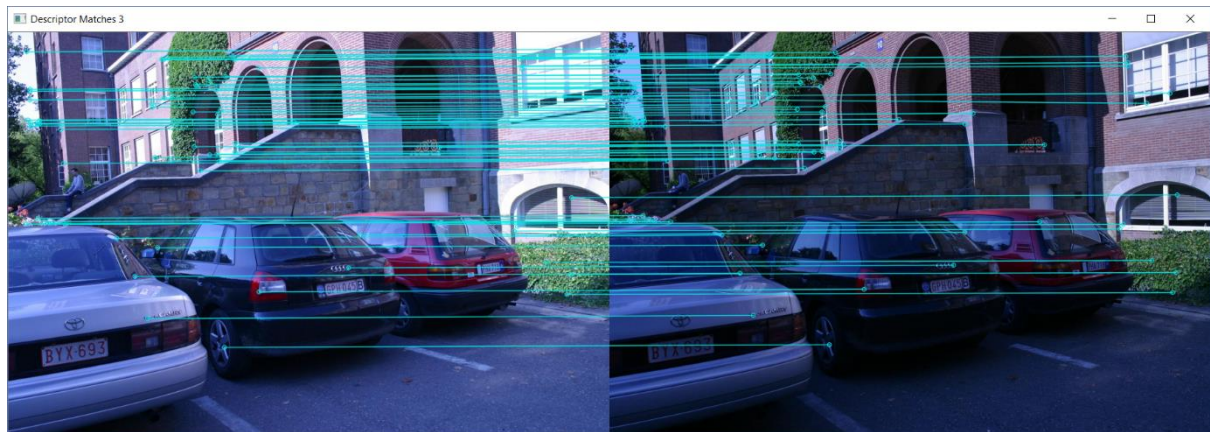


Figure 3.16 - OpenCV FREAK Descriptor "Leuven" 1|3 Results



Figure 3.17 - OpenCV FREAK Descriptor "Leuven" 1|4 Results



Figure 3.18 - OpenCV FREAK Descriptor "Leuven" 1|5 Results



Figure 3.19 - OpenCV FREAK Descriptor "Leuven" 1|6 Results

3.5.3 Re-created CREAK Results

From the results of Tables 3.1 – 3.4 and 3.5, the successful re-creation of the FREAK algorithm is confirmed, and thus the CREAK descriptor can be built upon FREAK with confidence that the foundation that has been laid is reliable. The appropriate changes have been made in order to create the CREAK descriptor and by making use of the CREAK descriptor in the same simulation environment on the "Graffiti" dataset, the following results were obtained as shown in Table 3.6.

Table 3.6 - CREAK Descriptor "Graffiti" Results

Matching Pairs (1 n^{th})	CREAK – Graffiti (Viewpoint Change)			
	Total Features on n^{th} image	# Matches between 1 and n	# Correct Matches between 1 and n	Time (s)
1 1	423	423	423	0,051
1 2	577	51	51	0,052
1 3	662	13	13	0,054
1 4	611	1	1	0,050
1 5	646	1	0	0,052
1 6	589	0	0	0,047

From Table 3.6, it can be observed that CREAK's results are very similar to that of FREAK, with the number of matches being moderately higher. CREAK does, however, present a single incorrect match at the 5th level transformation. The computational times of the CREAK descriptor are also comparable, although FREAK is on average between 10 ms and 20 ms faster than CREAK. Our re-created CREAK description appears to be slightly faster than the OpenCV library FREAK descriptor. Again, this can be accredited to the software being developed for our system alone, thus omitting the bulk of the unnecessary code that checks for

and converts data structures to formats compatible with the system on which the library is implemented. Figures 3.20 – 3.24 display the results of the CREAK descriptor on the “Graffiti” dataset, as obtained from the simulation test software.

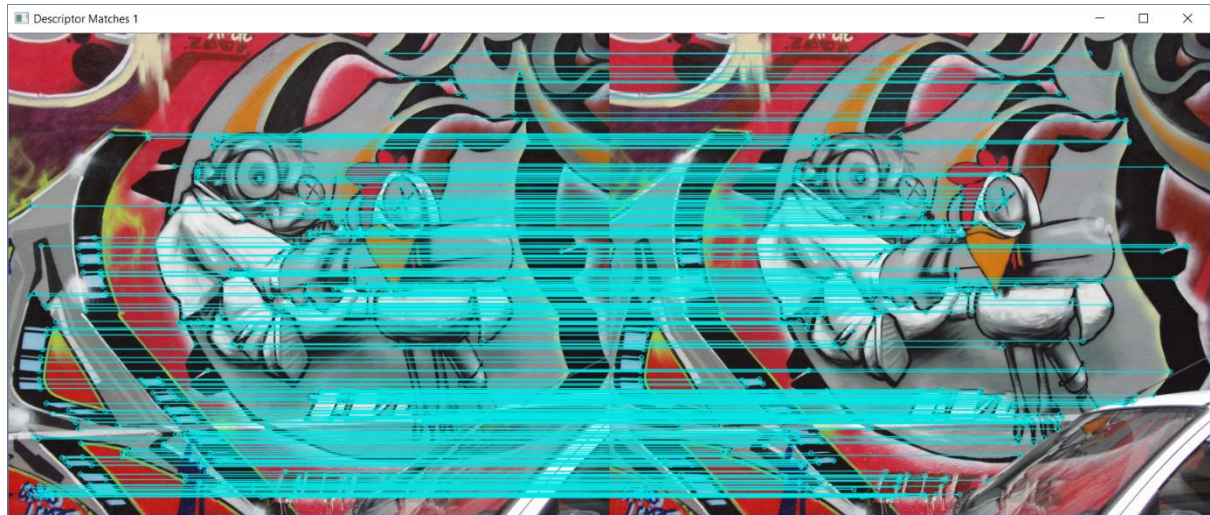


Figure 3.20 - CREAK Descriptor "Graffiti" 1|1 Results

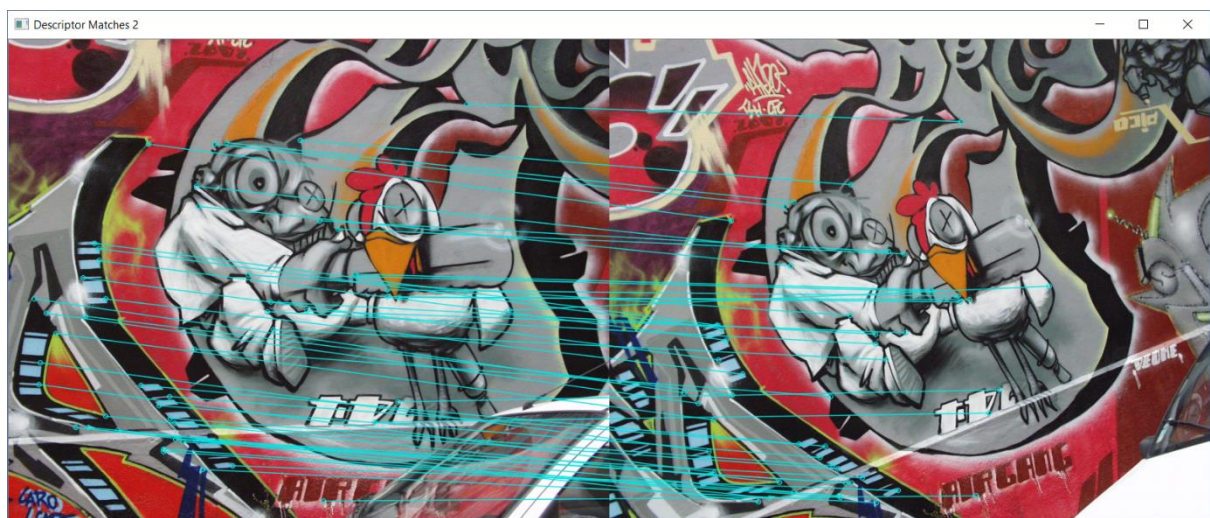


Figure 3.21 - CREAK Descriptor "Graffiti" 1|2 Results

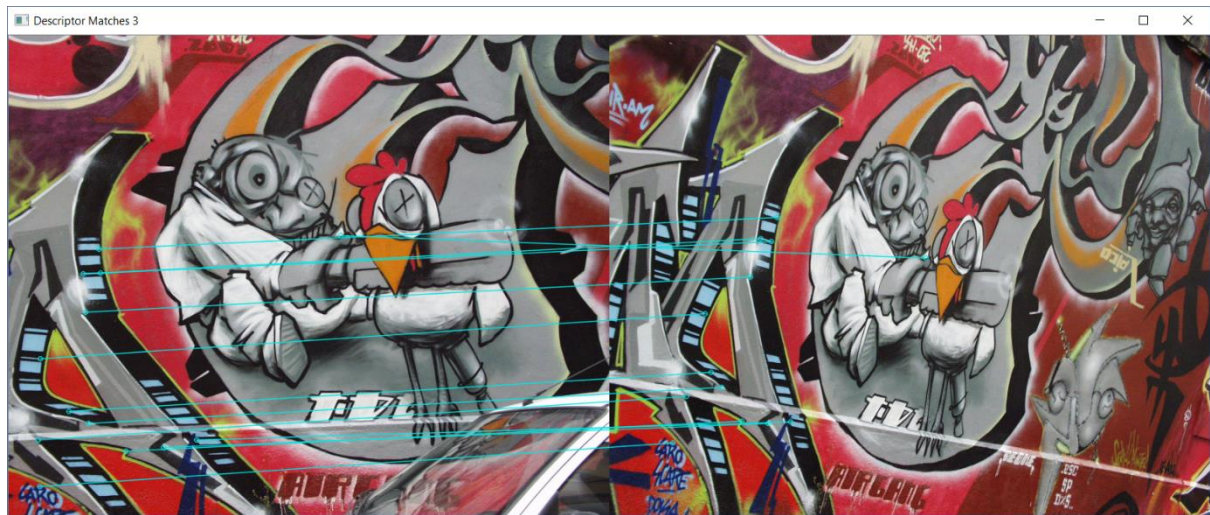


Figure 3.22 - CREAK Descriptor "Graffiti" 1/3 Results

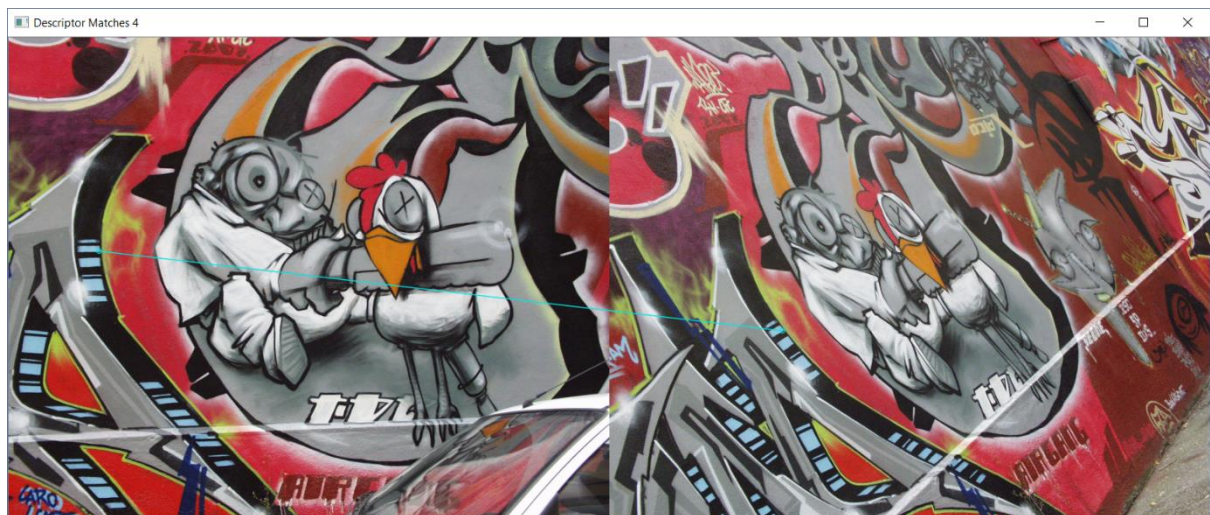


Figure 3.23 - CREAK Descriptor "Graffiti" 1/4 Results

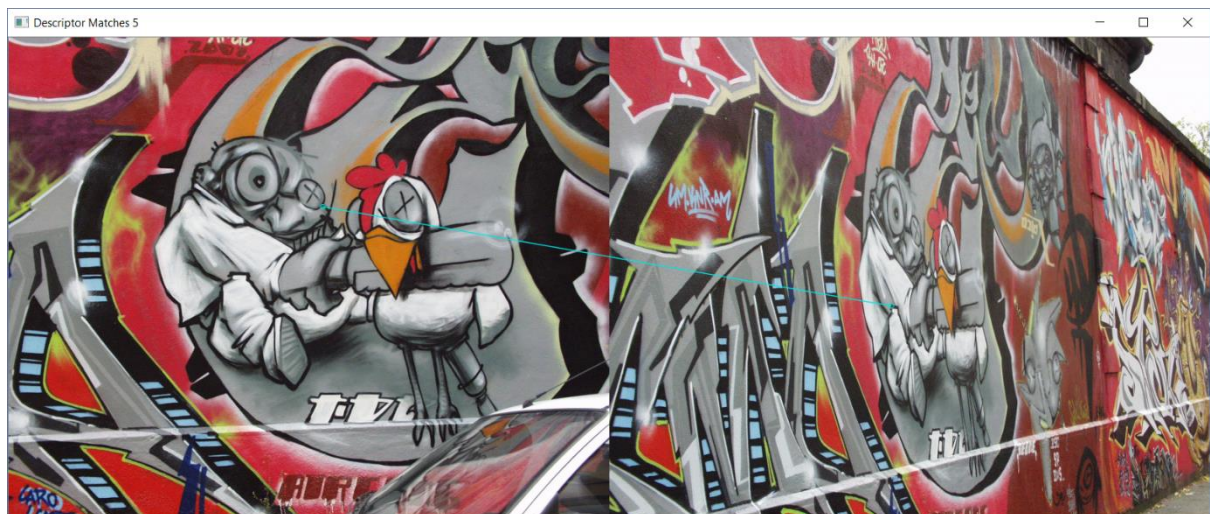


Figure 3.24 - CREAK Descriptor "Graffiti" 1/5 Results

Table 3.7 shows the results obtained on the “Bark” dataset by using the re-created CREAK descriptor.

Table 3.7 - CREAK Descriptor "Bark" Results

Matching Pairs (1 n^{th})	CREAK – Bark (Rotation/Zoom Change)			
	Total Features on n^{th} image	# Matches between 1 and n	# Correct Matches between 1 and n	Time (s)
1 1	16	16	16	0,065
1 2	4	1	0	0,067
1 3	12	0	0	0,062
1 4	79	0	0	0,068
1 5	99	0	0	0,080
1 6	143	0	0	0,099

Table 3.7 shows that, like Tables 3.2 and 3.6, that the descriptors have significant difficulty in matching features that had undergone a rotation and zoom transformation, since only the case where no rotation or zoom had been implemented, results in any matches. It is worth mentioning that CREAK has a single incorrect match at the 2nd level transformation. The computational times adhere to the same pattern as with the “Graffiti” dataset, and again show that CREAK operates at approximately 20 ms quicker than the OpenCV library FREAK descriptor. Our re-created FREAK descriptor remains the fastest by a small margin. Figures 3.25 and 3.26 display the results of the CREAK descriptor on the “Bark” dataset, as obtained from the simulation test software.



Figure 3.25 - CREAK Descriptor "Bark" 1|1 Results



Figure 3.26 - CREAK Descriptor "Bark" 1|2 Results

Table 3.8 shows the results obtained on the “Bikes” dataset by using the re-created CREAK descriptor.

Table 3.8 - CREAK Descriptor "Bikes" Results

Matching Pairs (1 n^{th})	CREAK – Bikes (Blur Transform)			
	Total Features on n^{th} image	# Matches between 1 and n	# Correct Matches between 1 and n	Time (s)
1 1	371	371	371	0,089
1 2	23	14	12	0,083
1 3	5	2	2	0,099
1 4	0	0	0	0
1 5	0	0	0	0
1 6	0	0	0	0

The “Bikes” dataset obtained very similar results from CREAK as compared to that of FREAK. Both descriptors obtained a similar number of matches, with the exception in the 2nd level transformation, where CREAK made 2 incorrect matches out of 14, as opposed to 1 incorrect match out of 12 with FREAK. The computational times adhere to the same pattern as previously discussed. Figures 3.27 - 3.29 display the results of the CREAK descriptor on the “Bikes” dataset, as obtained from the simulation test software.



Figure 3.27 - CREAK Descriptor "Bikes" 1/1 Results

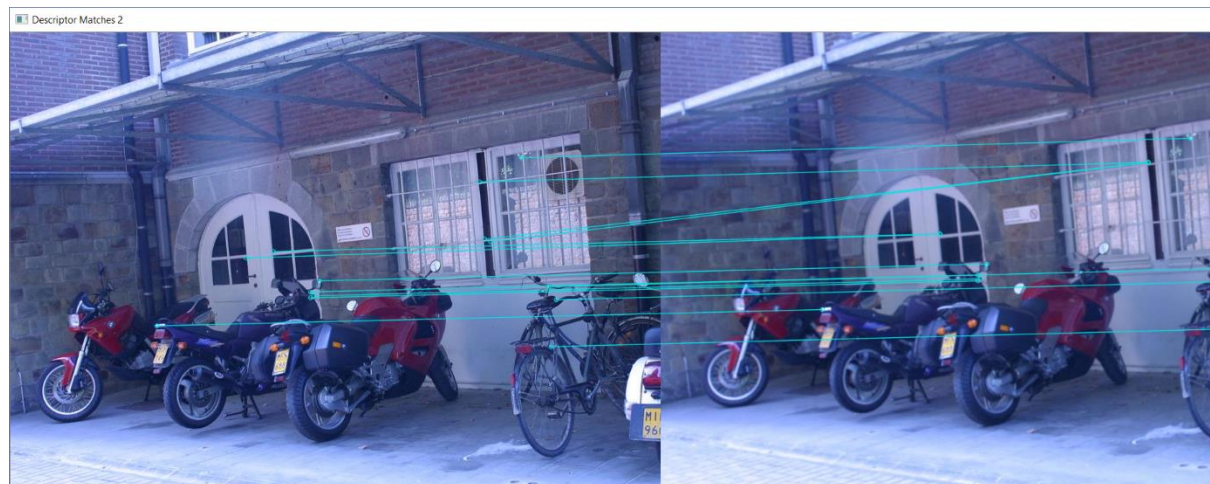


Figure 3.28 - CREAK Descriptor "Bikes" 1/2 Results

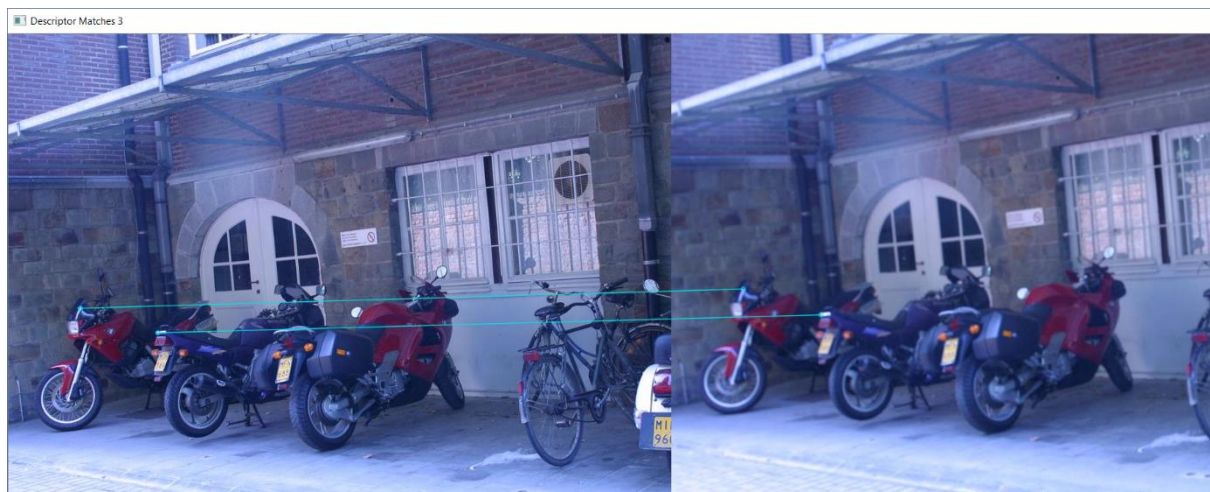


Figure 3.29 - CREAK Descriptor "Bikes" 1/3 Results

Table 3.9 shows the final results obtained on the “Leuven” dataset by using the re-created CREAK descriptor.

Table 3.9 - CREAK Descriptor "Leuven" Results

Matching Pairs (1 n^{th})	CREAK – Leuven (Luminance Change)			
	Total Features on n^{th} image	# Matches between 1 and n	# Correct Matches between 1 and n	Time (s)
1 1	629	629	629	0,122
1 2	495	143	143	0,100
1 3	419	78	78	0,100
1 4	349	56	56	0,146
1 5	253	27	27	0,132
1 6	170	12	12	0,133

As can be seen from Table 3.9, the results are once again in concordance with the results found in the “Bikes” dataset, with number of matches and computational times comparable to the FREAK descriptor. It is shown, however, that with the “Leuven” dataset, CREAK has made no incorrect matches, where FREAK did, in fact, make an incorrect match on the 6th level transformation. The number of matches in CREAK is slightly less than that of FREAK.

3.5.4 CREAK Experimental Results

For the final step in the verification of the CREAK descriptor, the testing conditions of those used by Chen *et al.* in the article proposing CREAK were re-created. For this experiment, the ORB detector is used, and the matching threshold adjusted to a more lenient value. The same dataset is used as above, more specifically image pairs 1|5 and 1|6 from the “Graffiti” dataset. The experiment is performed on both FREAK as well as CREAK, and the results are shown in Table 3.10 below:

Table 3.10 - FREAK vs CREAK "Graffiti" Experiment Results

Matching Pairs (1 n^{th})	FREAK vs CREAK “Graffiti” Experiment Results			
	FREAK Matches between 1 and n	# Correct FREAK Matches	CREAK Matches between 1 and n	# Correct CREAK Matches
1 5	6	2	7	4
1 6	3	0	3	2

Chen *et al.* found very similar results to those presented by Table 3.10 with the exception that they observed 0 correct matches with both FREAK 1|5 and FREAK 1|6 test pairs. The visual results for FREAK can be observed in Figures 3.30 and 3.31, and those of CREAK can be observed in Figures 3.32 and 3.33.

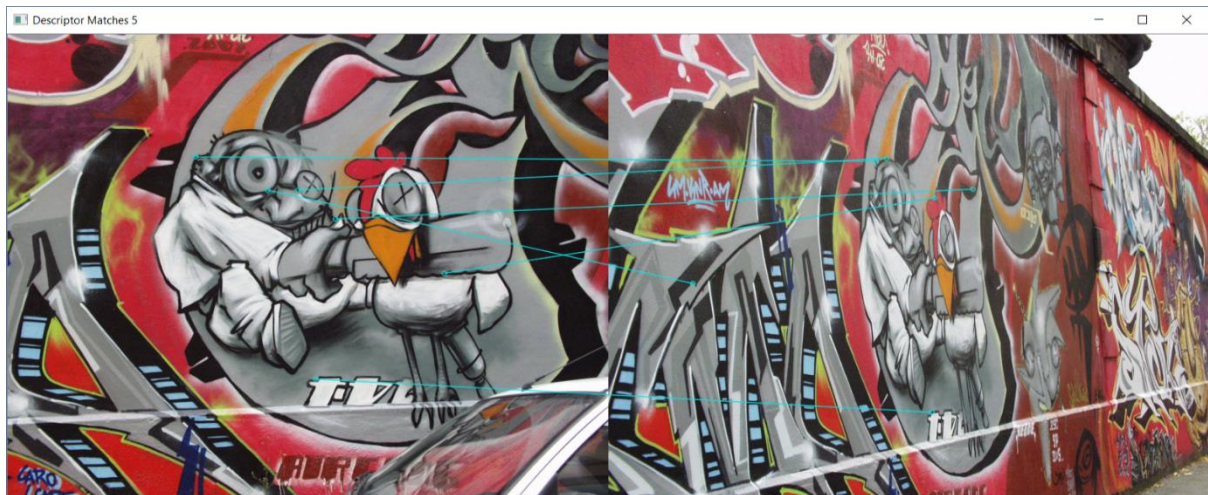


Figure 3.30 - FREAK 1/5 Experimental Results

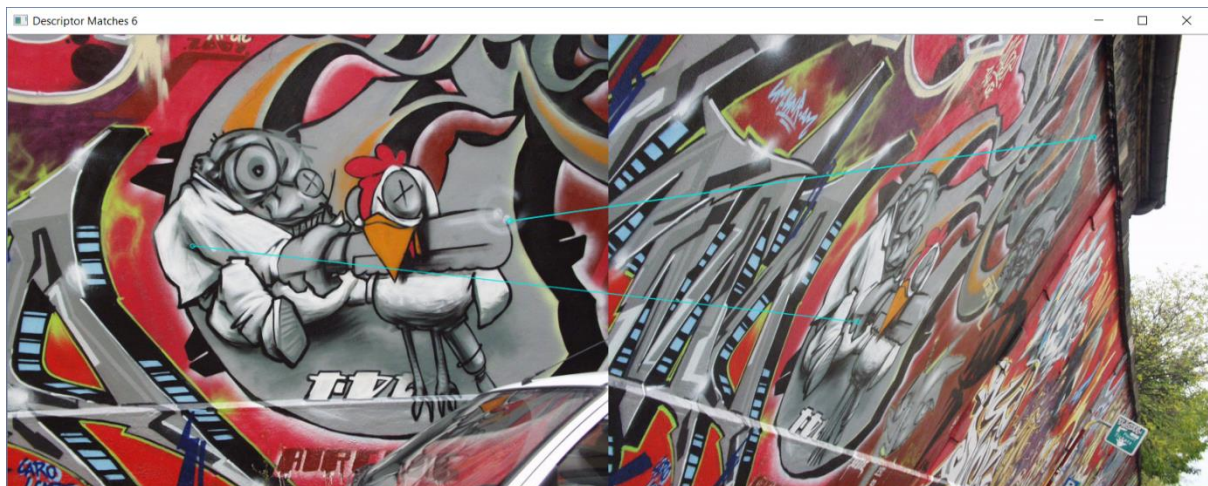


Figure 3.31 - FREAK 1/6 Experimental Results

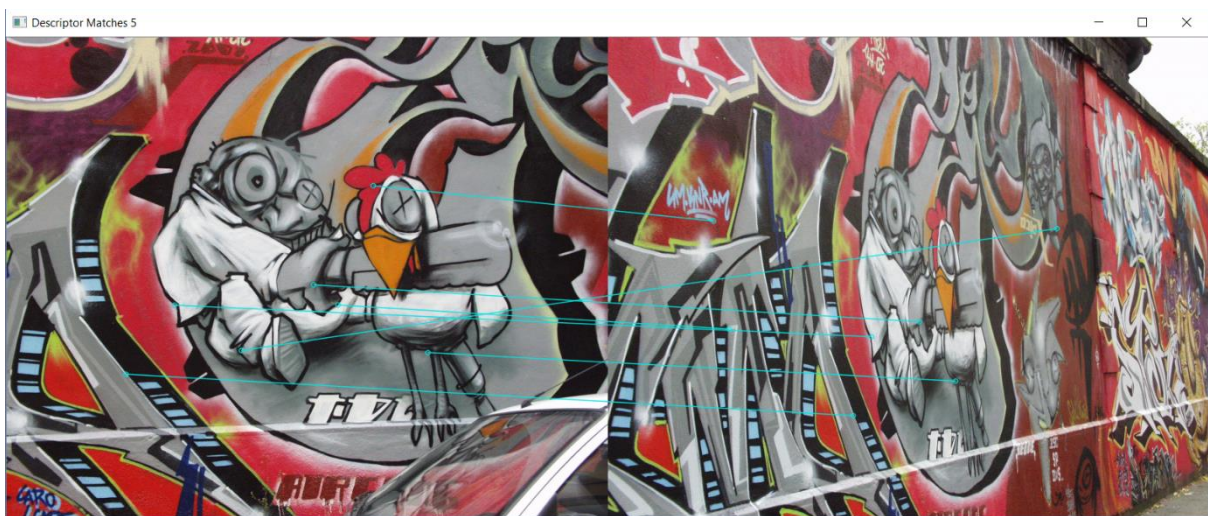


Figure 3.32 - CREAK 1/5 Experimental Results

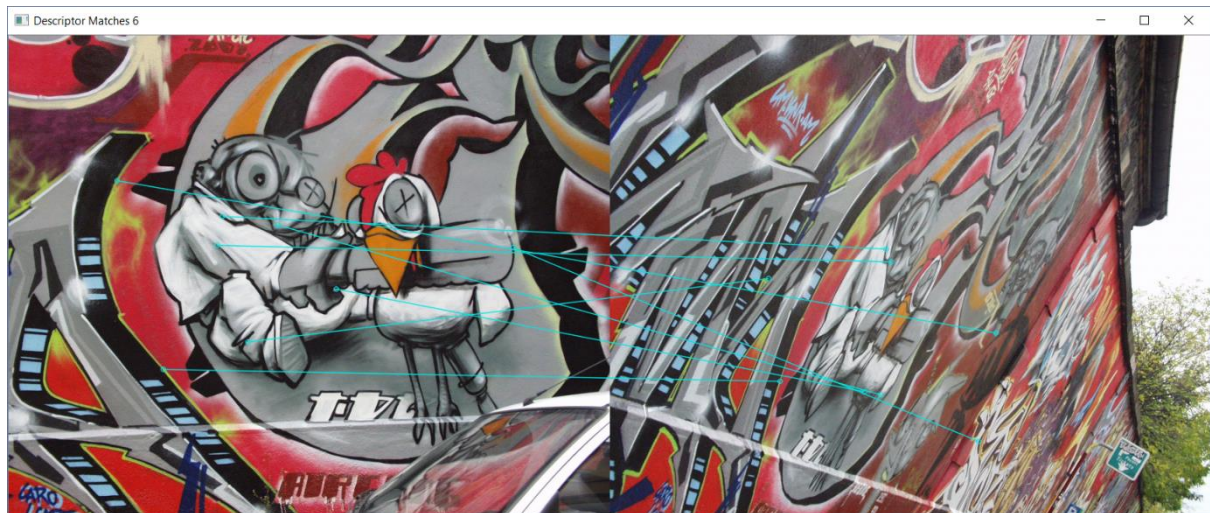


Figure 3.33 - CREAK 1|6 Experimental Results

3.6 Discussion

The CREAK descriptor achieved results that are in accordance with the expected results as documented by Chen *et al.* in [1]. The results obtained in this section show that CREAK performed, on average, 20 ms slower than FREAK, which translates to roughly 35% slower than FREAK. While the description of CREAK yielded in most cases a slightly larger number of matches as opposed to FREAK, with the exception of the luminescence test as in the Leuven dataset. This is contrary to the previously documented results demonstrated in APPENDIX A – Published Work, where it was found that CREAK consistently achieved fewer matches than FREAK, although still more accurate, which is in concordance with the experimental results found by Chen *et al.* The computational times found in this study display similar results to those in APPENDIX A. Therefore, it can be certain that the results in terms of accuracy and speed are reliable and valid.

As discussed in Chapter 1, the FREAK descriptor is effectively verified when the re-created FREAK descriptor is compared to the OpenCV implementation of FREAK. This is performed above on the University of Oxford Visual Geometry Group Dataset, and the results show that the discrepancies are negligible. In order to verify the CREAK descriptor, the results observed by CREAK's article is pursued. A similar experiment is performed as above on the two most difficult matching pairs of the "Graffiti" dataset, but by instead using the ORB detector and a more lenient matching threshold. The results obtained are once again in agreement with those found by Chen *et al.* and thus it can be confidently concluded that the re-creation of the CREAK descriptor, based on the article by Chen *et al.* had been completed successfully.

3.7 Conclusion

In this chapter, the FREAK and CREAK descriptors are re-created based on their respective articles. This chapter is dedicated to the verification of the re-created descriptors, such that they can be used later in this dissertation with confidence in their correct functionality. The results in this chapter found that both descriptors have, in fact, been implemented successfully, and the accuracy of their results is advocated by the correlation between our observed results and the results found by the respective descriptors' articles in test environments.

It was found that both FREAK and CREAK perform poorly when considering a simultaneous zoom and rotation transformation. However, their results were outstanding when there was only a luminance transformation or blur transformation. The viewpoint change is arguably the most relevant transformation in a navigation context and as such, both descriptors performed relatively well, with CREAK the apparent superior descriptor.

CHAPTER 4 - SLAM IMPLEMENTATION

4.1 Introduction

In order to effectively evaluate the CREAK descriptor in a vSLAM context, the vSLAM environment is set up to represent the environment that an unmanned vehicle might encounter. Despite not being a full vSLAM implementation due to the scale dependency of the visual odometry ignored, the implementation still closely resembles a real-world implementation of the CREAK descriptor. In this chapter, the implementation process for the vSLAM algorithm is documented along with the relevant methods used. The design and implementation methodologies are presented for each relevant aspect in pursuit of the vSLAM environment such as the monocular visual odometry and the EKF algorithm. The environment is created and the vSLAM implementation is tested with both FREAK and CREAK descriptors.

4.2 SLAM Implementation

4.2.1 Monocular Visual Odometry

Visual Odometry can be described as the process of determining a robot's location or trajectory by exclusively making use of visual sensors, as first proposed by Nistér in [32] and is an essential step in the greater SLAM problem, as odometry provides an initial estimate for the robot state. For the purpose of this dissertation, the focus lies more specifically on the use of monocular vision, such as a single camera sensor. In this section, the design for the concept of monocular visual odometry is explored by a similar approach to that by Singh in [4], where the process of attaining a trajectory from monocular visual odometry is described in Figure 4.1. The monocular visual odometry process begins by considering two successive images, taken from the perspective of the robot. Features are then extracted and described by the FREAK or CREAK descriptors as discussed in previous chapters, on each image independently. The features from the succeeding images are then matched to one another and the features corresponding to both images are considered to derive the Essential Matrix, E as discussed by Nistér in [38] where E can be defined as:

$$E \equiv \mathbf{t} \times \mathbf{R}, \quad (4.1)$$

where $\mathbf{t}$ is a 3-dimensional translation vector and $\mathbf{R}$ is a 3×3 rotation matrix. An outlier detection method, known as the Random Sample Consensus (RANSAC) [40], is introduced in order to remove any outlying matched features. With RANSAC, each time the Essential Matrix is determined, five random feature points are selected, and E estimated based on those

points. Thereafter, all the other corresponding points are tested for compliance with E . After a pre-set number of iterations, the matrix E with the most inliers is used as the final Essential Matrix. The robot's change in pose is then retrieved from E by solving for t as to determine the change in translation and solving for R to find the current change in rotation.

The new robot pose is determined by combining the change in translation and rotation with the pose the previous time step. The Essential Matrix provides a change in pose in relative units, therefore in order to convert the results into usable values corresponding to real-world coordinates, the change in pose is multiplied by a scale factor. This scale factor can be pre-determined by performing an initialization procedure or by including additional sensory information, but for the sake of simplicity in this dissertation, the scale factor will be extracted from the ground truth data included with the dataset, as it can be presumed that an actual SLAM application would have the scale factor determined. Furthermore, due to the purpose

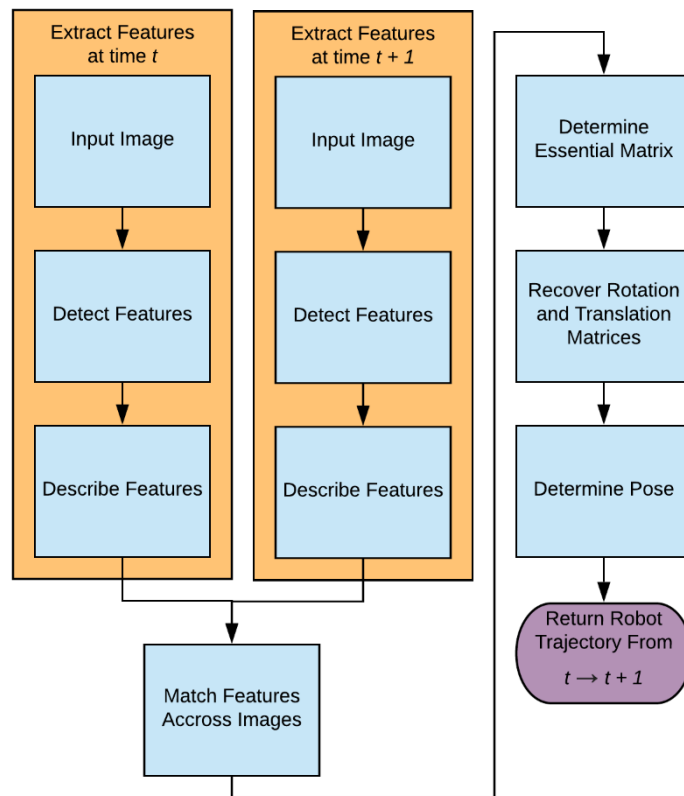


Figure 4.1 - Monocular Visual Odometry Operational Flow

of this dissertation being to evaluate the CREAK descriptor in a vSLAM context, the scale factor does not affect the evaluation of the descriptor itself, and as such the vSLAM implementation discussed is not a full SLAM solution. APPENDIX B – Visual Odometry Results shows the effects of various keypoints generated on the odometry performed by FREAK and CREAK.

4.2.2 Landmark 3D Localization

Visual odometry by making use of the Essential Matrix does a more than adequate job at determining the change in the pose of the robot based on the features detected across two succeeding frames. However, for the use of SLAM, the features detected in the aforementioned frames need to be extracted as permanent landmarks to be used as reference points in the localization to come. These landmarks are composed of the binary descriptor as described by FREAK or CREAK, as well as the ever-important 3D world coordinates at which the feature resides.

Usually, when making use of stereo camera pairs, the landmark's 3D coordinates can easily be extracted with simple depth perception. However, in the case of this dissertation, monocular vision is used, and as such a different method to determine the 3D world coordinates of each feature detected is required. For this study, the pinhole camera model is used to map features in the image to world coordinates. The operational flow is shown in Figure 4.2 and can be described as follows: once the detected features have been described and matched between the two consecutive frames, the features corresponding to both frames are considered for landmarks. In order to determine the 3D coordinate for a single feature observed, the feature's pixel location is considered in each frame. A vector is then used to describe the direction of the feature, as observed by the camera. This relative 3-dimensional vector, $\mathbf{v}$, will be determined as follows:

$$\mathbf{v} = \langle x, y, z \rangle, \quad (4.2)$$

where

$$x = p_x - pp_x, \quad (4.3)$$

and

$$y = pp_y - p_{yc}. \quad (4.4)$$

The focal length is shown in pixels as z , while p_x and p_y correspond to the x and y pixel coordinates of the feature and pp_x and pp_y are the x and y coordinates of the image's principal point, or the centre-point of the image. Figure 4.3 shows the context for determining the feature's 3-dimensional vector, $\mathbf{v}$. It is important to note, that $\mathbf{v}$ is a vector relative to the principal axis, which is the Z -axis in the image's coordinates. The orientation of the robot is known within a level of uncertainty, as it had been determined in the previous section. We can, therefore, map $\mathbf{v}$ onto world coordinates, represented by the robot's orientation vector $\mathbf{r}$, where

r_x , r_y and r_z correspond to the 3D vector orientation of the robot. This is achieved by determining a transformation, T , where

$$T[\langle 0, 0, 1 \rangle] \rightarrow \langle r_x, r_y, r_z \rangle. \quad (4.5)$$

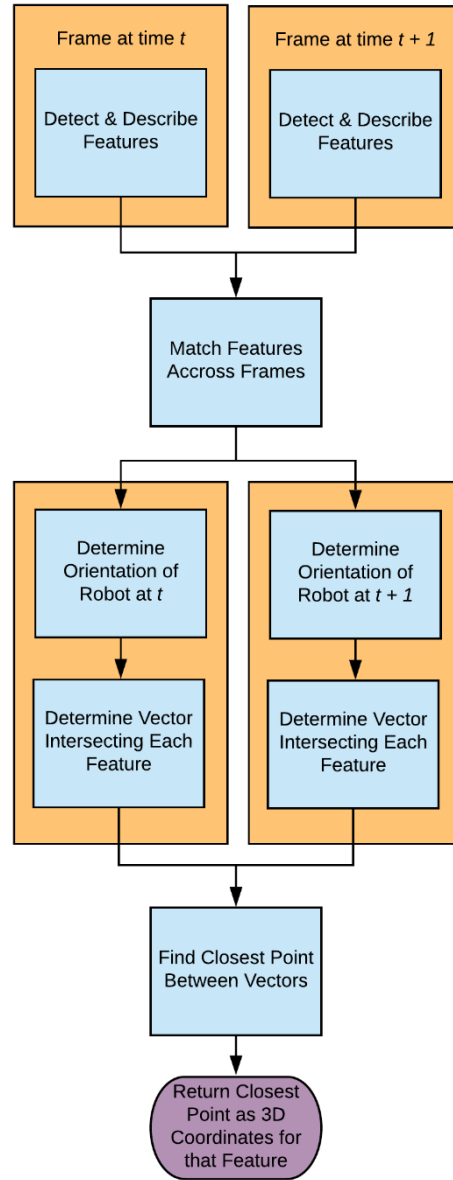


Figure 4.2 - Landmark Localization Operational Flow

The same transformation, T , can then be used to map v onto v' , as shown by:

$$T[v] \rightarrow v', \quad (4.6)$$

where v' is the vector that intercepts the feature as observed by the robot in terms of world coordinates. Thereafter, the vectors v'_i and u'_i , which correspond to the i^{th} landmark on the frame at times t and $t + 1$ respectively, are considered and the closest point between both vectors is determined, as described by Figure 4.4. Ideally, the point of intersection between the two vectors would simply be determined. However, due to possible error and tolerance, there exists a chance that the 3D vectors never intercept one another. Therefore, the point at which the vectors v'_i and u'_i are nearest to one another, can be used as the final 3D coordinate of landmark i .

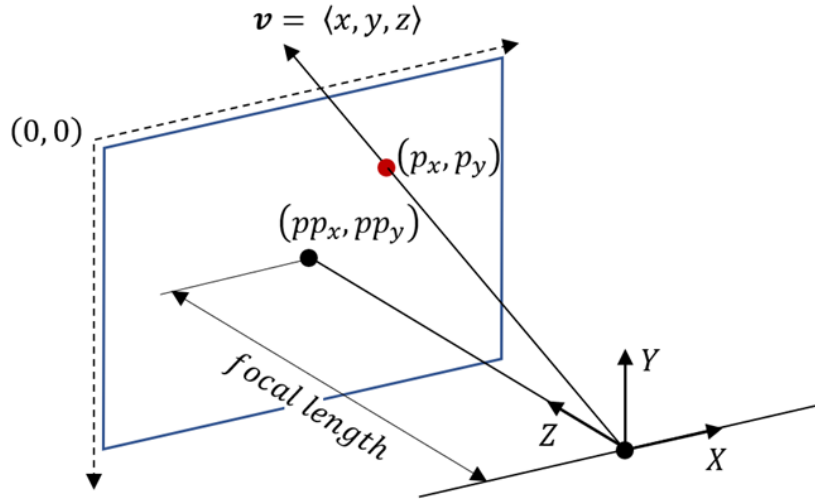


Figure 4.3 - Relative 3D Landmark Vector

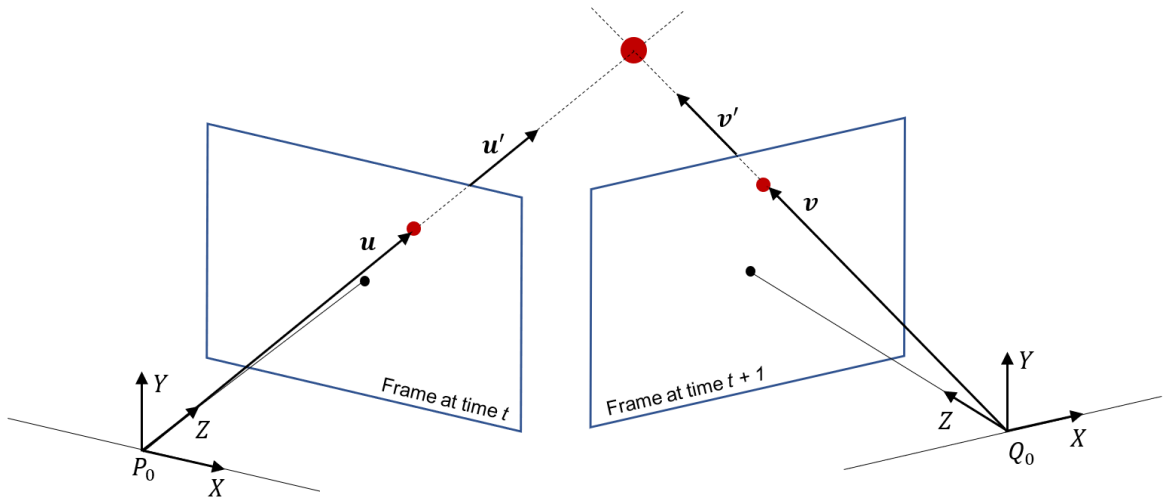


Figure 4.4 - Localizing Landmark from Different Viewpoints

The method used for determining the point at which the shortest distance between the two 3D vectors exists is similar to that discussed by Sunday in [41]. Consider two lines, L_1 with a point $P = P_0 + s\mathbf{u}'$ and line L_2 with a point $Q = Q_0 + t\mathbf{v}'$, with corresponding vectors $\mathbf{u}'$ and $\mathbf{v}'$, and scalars s and t respectively. P_0 and Q_0 are the points of origin. Assuming that P and Q are not parallel, there exists a closest point on each line $P(s_c)$ and $Q(t_c)$. For the points to be the closest possible points, the vector joining the two points, $\mathbf{w}_c$ must be as short as possible. For this to occur, it can be said that $\mathbf{w}_c$ must be perpendicular to both lines L_1 and L_2 , as shown in Figure 4.5, and can then satisfy the two equations: $\mathbf{u}' \cdot \mathbf{w}_c = 0$ and $\mathbf{v}' \cdot \mathbf{w}_c = 0$. The two equations can be solved for by substituting $\mathbf{w}_c = P(s_c) - Q(t_c) = \mathbf{w}_0 + s_c\mathbf{u}' - t_c\mathbf{v}'$, where $\mathbf{w}_0 = P_0 - Q_0$, into each other, resulting in two simultaneous linear equations:

$$(\mathbf{u}' \cdot \mathbf{v}')s_c - (\mathbf{u}' \cdot \mathbf{v}')t_c = -\mathbf{u}' \cdot \mathbf{w}_0 \quad (4.7)$$

and

$$(\mathbf{v}' \cdot \mathbf{v}')s_c - (\mathbf{v}' \cdot \mathbf{v}')t_c = -\mathbf{v}' \cdot \mathbf{w}_0. \quad (4.8)$$

Then, by setting $a = \mathbf{u}' \cdot \mathbf{u}'$, $b = \mathbf{u}' \cdot \mathbf{v}'$, $c = \mathbf{v}' \cdot \mathbf{v}'$, $d = \mathbf{u}' \cdot \mathbf{w}_0$ and $e = \mathbf{v}' \cdot \mathbf{w}_0$, s_c and t_c can be solved for by using:

$$s_c = \frac{be - cd}{ac - b^2} \quad (4.9)$$

and

$$t_c = \frac{ae - bd}{ac - b^2}, \quad (4.10)$$

where s_c and t_c can be substituted back into $P = P_0 + s_c\mathbf{u}'$ and $Q = Q_0 + t_c\mathbf{v}'$ to determine the the closest point on each line L_1 and L_2 respectively. The distance between the two points, P and Q can then be calculated, as the distance will represent a sort of reliability score. Should the distance be too large, the points are likely from mismatched features and should be discarded. If the distance between the points is within a specified threshold, the points can be accepted as reliable, and either P or Q can be selected as the 3D coordinate for landmark i .

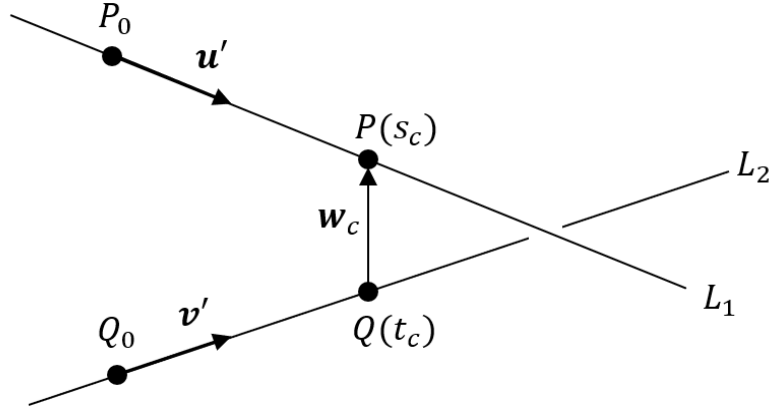


Figure 4.5 - Determining Closest Points Between Two Lines

4.2.3 EKF-SLAM

In this dissertation, the concept of Simultaneous Localization and Mapping (SLAM) is investigated. More specifically, the aim is to evaluate the performance of the FREAK descriptor against the CREAK descriptor in the context of vSLAM. Of the various existing SLAM algorithms, the Extended Kalman Filter (EKF) is arguably the most well-known filter used for state estimation. The EKF functions by linearizing motion models that estimate the current state, enabling the use of the standard Kalman Filter (KF) equations. The KF is designed to operate when the motion and measurement model can be described with linear equations, and the uncertainty pertaining to the motion and measurements are Gaussian. Due to the simplicity and reliability of the EKF, it will be used as the filtering algorithm in our SLAM implementation. In the context of this implementation, we assume the final goal to be an EKF-SLAM implementation where landmarks are not forgotten nor deleted when the robot has passed the point of observation. The reasoning is that the best evaluation of the CREAK descriptor is arguably the most extreme case that might demonstrate the strengths as well as shortcomings of the descriptor to the fullest extent. Since FREAK and CREAK are very similar in function, the evaluation is determined by exploiting the differences between the two, namely: the robustness resulting in a certain number of matches per frame, and the efficiency determining the computational time for description per frame. By keeping all observed landmarks, the more efficient descriptor should achieve better results. The operational flow for the EKF is presented in Figure 4.6.

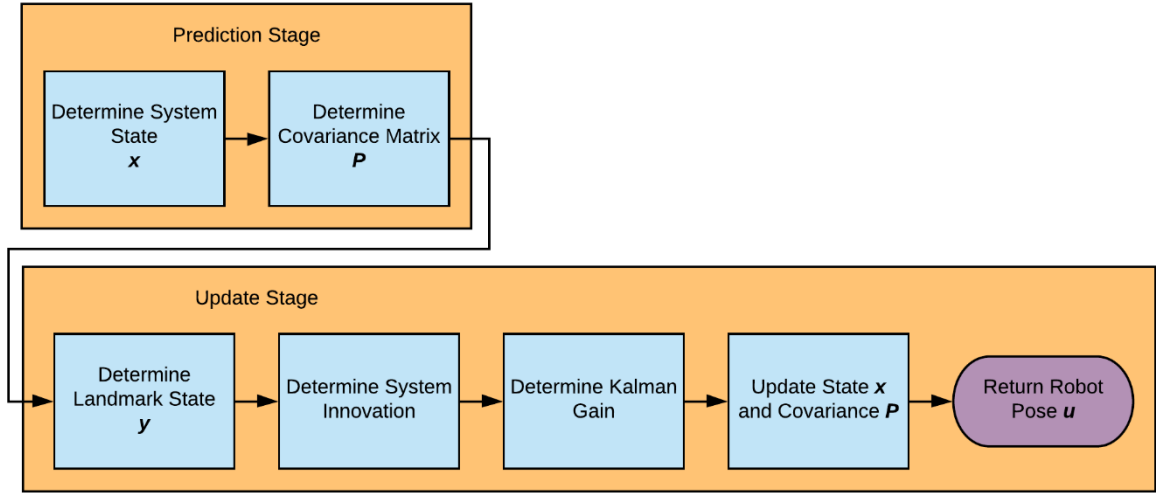


Figure 4.6 - EKF-SLAM Operational Flow

For SLAM in 3D space, the robot state $\mathbf{u}$ will be measured in 6 degrees of freedom in an Euler coordinate system, namely: $x, y, z, \theta, \phi, \omega$ which represents the x, y and z coordinates and *yaw, pitch* and *roll* respectively, shown as:

$$\mathbf{u} = [x \ y \ z \ \theta \ \phi \ \omega]^T. \quad (4.11)$$

According to Blanco in [7] and shown by the accompanying Figure 4.6, EKF-SLAM consists of two main phases, namely the prediction phase and the update phase. In the prediction phase, the current system state $\mathbf{x}_k$ is estimated at time step k as an $n \times 1$ matrix, where $n = 6 + (3 \times \mathbf{y}_m)$ where 6 represents the robot's state's 6 degrees of freedom, $\mathbf{y}_m$, represents the landmarks and m the number of landmarks. The 3 represents the landmarks' 3 degrees of freedom: x, y and z . The system state consists of the robot's state as well as each landmark's state:

$$\mathbf{x} = [\mathbf{u} \ \mathbf{y}_1 \ \mathbf{y}_2 \ \dots \ \mathbf{y}_m]^T \quad (4.12)$$

where

$$\mathbf{x}_{k|k-1} = \mathbf{f}(\mathbf{x}_{k-1|k-1}, \mathbf{u}_k). \quad (4.13)$$

Since static landmarks are assumed in the world map, the prediction stage of the EKF only changes the variables belonging to the robot's pose. Since the robot's pose is usually estimated from control data, and we have no such information, the robot is assumed to continue on its current trajectory. Thus, the previous time step's change in pose is used as the control input. In our case, the visual odometry results at time $k - 1$ are used to estimate the robot's pose at time

k . Thereafter, the covariance matrix, $\mathbf{P}$, is created of size $n \times n$, where $n = 6 + (3 \times \mathbf{y}_m)$, by using the formula below:

$$\mathbf{P}_{k|k-1} = \mathbf{G}_k \mathbf{P}_{k-1|k-1} \mathbf{G}_k^T + \mathbf{Q}_k, \quad (4.14)$$

where $\mathbf{G}$ is the $n \times n$ state transition matrix defined by the Jacobian:

$$\mathbf{G} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}, \quad (4.15)$$

with $\mathbf{f}$ the function deriving the system state and $\mathbf{Q}$ a covariance matrix for the process noise. The process noise is the uncertainty in the estimation of the change in the robot's pose in the new time step. Thereafter, the update stage iterates for each observed landmark $\mathbf{y}_i$. Each observed landmark is associated with the landmarks already in the database. If the landmark is observed for the first time, it is appended to the database, with a "times seen" counter set to 1 such that only landmarks that have been seen an acceptable number of times can be considered, thus reducing the chance of a landmark that had only been seen once and never again entering the system. If the landmark is re-observed, meaning that it is already in the database, the "times seen" counter is incremented, and if it has been seen an acceptable number of times, the innovation $\mathbf{S}_k$ at time step k is calculated as:

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k, \quad (4.16)$$

where $\mathbf{H}$ is the $n \times n$ observation matrix defined by the Jacobian:

$$\mathbf{H} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}, \quad (5.17)$$

with $\mathbf{h}$ the function deriving the landmark state $\mathbf{y}_k$ and $\mathbf{R}$ denoting the covariance matrix for the measurement noise. This is the uncertainty that appears in the position of the observed landmarks. The Kalman gain can then be calculated as:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1}, \quad (4.18)$$

and the state updated as:

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathbf{K}_k \mathbf{y}_k, \quad (4.19)$$

with the new covariance determined as:

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_{k|k-1}. \quad (4.20)$$

When all landmarks have been iterated through, the estimated state x_k and the latest covariance matrix P_k are returned, and the process is repeated at the next time step.

4.3 Simulation Environment

Microsoft Visual Studio 2017 is used to create a simulation environment for the implementation of monocular VO as well as the implementation of the EKF-SLAM simulations in C++. The GitHub repository for the code can be found at the following [link](#) [43]. The KITTI Vision Benchmark Suite [5] datasets are used, as the datasets contain a video feed of a vehicle driving around an urban setting through the streets of Karlsruhe, in Germany. The video feeds are 3-channel colour video feeds, taken at a frame rate of 10 frames per second (fps). The datasets also include reliable ground truth data, providing the pose of the robot at any given time. Figure 4.7 shows the operational flow of the simulation software.

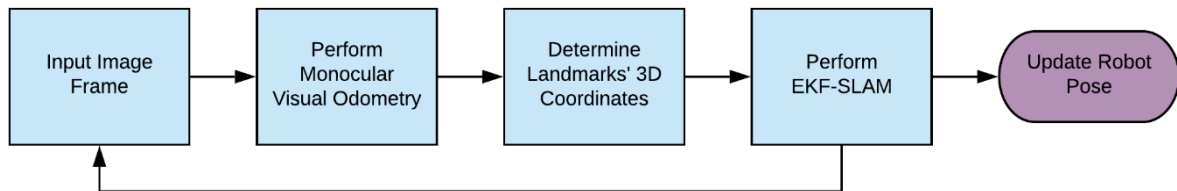


Figure 4.7 - SLAM Simulation Environment Operational Flow

The simulation environment begins by considering the first image frame from the KITTI dataset. Features are then detected by using the FAST feature detector and described on the image frame by using the FREAK and CREAK descriptors. At time step $t = 1$, there is only a single frame thus far, so the software stores the detected features with their corresponding descriptions and awaits the next frame. At time step $t = 2$ onwards, the process described thus far is repeated and the features along with their descriptions are determined for the second frame. Now that there are at least two image frames, the process can continue and monocular visual odometry is performed on the latest frame, returning an estimated pose for the robot. For each feature detector in the latest frame, the corresponding 3-dimensional coordinates are determined, and the feature is then considered as a landmark to be used in EKF-SLAM. SLAM iterates over every landmark detected and provides a more accurate estimation for the robot's current pose. The newly determined pose is returned, and the robot's trajectory plotted for a visual reference of the algorithms in action.

The simulation software is performed over the first 1000 frames of a video feed provided in the KITTI dataset, and the estimated pose is plotted along with the ground truth included in the dataset. The experiment is performed over 5 individual datasets and is performed once with the FREAK descriptor being used and repeated again with the CREAK descriptor being used

instead. The output consists of an estimated robot trajectory for the visual odometry alone and with the EKF-SLAM included, providing both a visual conclusion as well as quantitative measurements for the robot's translation and rotational information against the ground truth.

4.4 Simulation Results

By making use of the simulation environment described in the previous section, the monocular VO and the EKF-SLAM algorithms were simulated on the KITTI Vision Benchmark Suite [5] datasets, by considering the first 1000 frames from 5 individual datasets. The simulation had been performed once while using the FREAK descriptor in the odometry stage, and again while using the CREAK descriptor. Figure 4.8 shows an example of a single frame within the first dataset, 00, on which odometry is performed, demonstrating the features as matched by the preceding frame. The 'red' circles represent the landmark observed in the current frame, and the 'cyan' lines are drawn towards the feature's image position in the preceding frame.



Figure 4.8 - Monocular Visual Odometry Feature Matching in the KITTI [5] dataset 00 where Red circles represent newly observed landmarks, and Cyan lines are drawn towards the landmark's previous position

4.4.1 Threshold Sweep Results

In order to provide accurate odometry over the five datasets, which will feed into the SLAM implementation later on, we first need to determine an acceptable threshold value for matching the descriptors. The threshold value plays an important role in odometry, as there is a balance between too strict and too lenient that needs to be found. We aim to have a threshold value that is both strict enough to block out unnecessary false matches, but not too restricting as to not provide enough correct matches. An experimental environment was set up to sweep from a threshold value of 0.3 to 0.9 for both FREAK and CREAK descriptors, over all five datasets and record the translational error for results. In APPENDIX B – Threshold Sweep Results, observing the raw data reveals that in most cases, the “best” threshold was either 0.6, 0.8 or 0.9, leaving a difficult choice. However, Figures 4.8 and 4.9 visualise the findings to paint a clearer picture.

In both the FREAK and CREAK threshold sweeps, we observe a pattern that as the threshold approaches a more lenient value (approaching 1), the error gets smaller until the threshold is 0.6 or 0.8, after which the error appears to climb again.

We have thusly decided on a threshold value of 0.8, which although moderately lenient, does achieve the most consistently accurate odometry results. The trade-off of this decision, as will be discussed in greater detail later in this dissertation, is that it can now be expected to have more incorrect matches between different keypoints.

Due to the threshold value of 0.6 also demonstrating accurate odometry, albeit not as accurate as 0.8, it is still true that there are fewer incorrect matches as it is a stricter threshold. Therefore, out of the odometry context, 0.6 would be a better matching threshold. We have therefore decided that in the SLAM implementation, the association stage of SLAM will use the threshold value of 0.6 for matching keypoints because in the context of SLAM, less incorrect matches are weighted heavier than more correct matches.

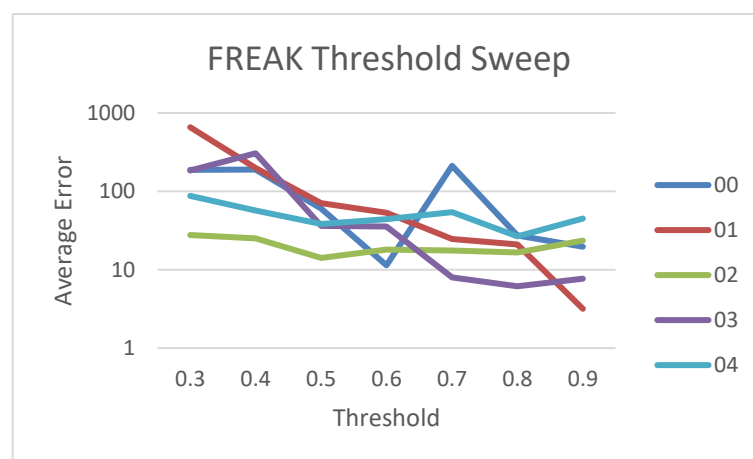


Figure 4.9 - FREAK Threshold Sweep

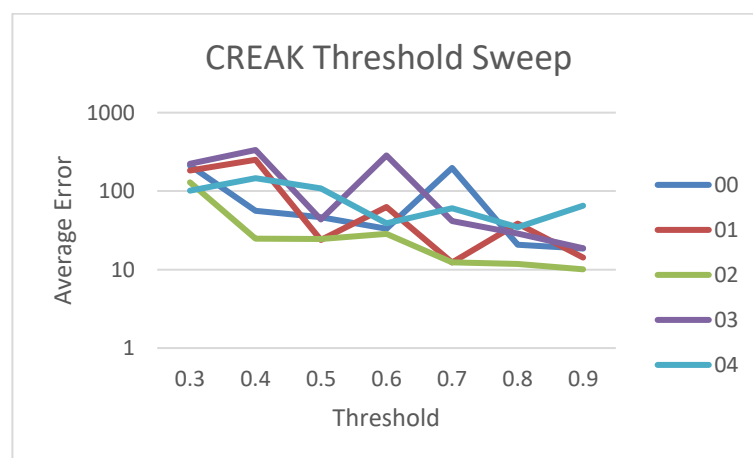


Figure 4.10 - CREAK Threshold Sweep

4.4.2 Visual Odometry Results

The simulation environment provides quantifiable results in the form of the robot's 3-dimensional position, rotational information representing the robot's orientation in terms of yaw and pitch, and finally the time (in seconds) after processing the 1000 frames from the dataset. We will use the threshold value determined in the previous section throughout the odometry process. A relevant side-note is that the dataset video frames are captured at a rate of 10 frames per second (fps) and for the results to be considered "real-time", the processing time should not be slower than 10 fps. The first pair of tests simulate the monocular VO on the KITTI datasets 00-04 by using the FREAK and CREAK descriptors, with results in terms of the position shown in Table 4.1 and rotation along with processing times shown in Table 4.2. The processing time is the absolute measured value after the entirety of the frames are processed, and the frame rate is the average frames per second as calculated from the processing time.

Table 4.1 - Monocular Visual Odometry Translation Results

Descriptor/Dataset	Monocular Visual Odometry Translation Results						
	True Position (m) (x, y, z)			Estimated Position (m) (x, y, z)			Translational Error (m)
FREAK 00	-184,8	-3,554	328,513	-209,6	-11,767	279,77	55,308
FREAK 01	463,56	-48,891	509,303	452,18	-86,644	523,38	41,869
FREAK 02	66,907	-8,643	232,934	70,691	34,956	235,34	43,829
FREAK 03	-1,768	-4,303	211,025	5,118	-14,687	209,85	12,515
FREAK 04	-382	2,739	148,241	-373,7	74,37	171,31	75,709
CREAK 00	-184,8	-3,554	328,513	-161,2	14,645	348,91	36,119
CREAK 01	463,56	-48,891	509,303	422,86	-4,264	539,39	67,481
CREAK 02	66,907	-8,643	232,934	65,563	17,239	241,14	27,185
CREAK 03	-1,768	-4,303	211,025	-4,585	-35,864	208,84	31,762
CREAK 04	-382	2,739	148,241	-380	84,082	168,83	83,929

From Table 4.1, it can be observed that FREAK performed slightly more favourably in when estimating the translation of the robot, with an average translational error of 45.846m compared to CREAK with an average error of 49.295m. Although the differences in errors are relatively small, there is a large inconsistency in the error per dataset. FREAK performed rather predictably with the exception of dataset 03, which obtained an abnormally low error. CREAK, on the other hand, ranged from obtaining a low error to a high error rather severely. When temporarily ignoring the elevation position (y -value) and its error, the results present FREAK with an average error of 21.756m and CREAK becomes more predictable with an average error of 22.875m. This points out that the odometry in the third dimension is not very accurate,

presumably due to the vehicle in the datasets experiencing a significant amount of pitch disturbance as it travels.

This alternate view of the data also explains that the visual trajectory results of CREAK appear better than that of FREAK at first glance because the trajectory is drawn in only two dimensions. Table 4.2 presents the rotational and performance results for FREAK and CREAK on the datasets.

Table 4.2 - Monocular Visual Odometry Rotation Results

Descriptor/Dataset	Monocular Visual Odometry Rotation Results							
	True Rotation (<i>yaw, pitch</i>) (radians)		Estimated Rotation (<i>yaw, pitch</i>) (radians)		Error on Rotation (<i>yaw, pitch</i>) (radians)		Processing Time (s)	Frames per Second (fps)
FREAK 00	3,062	3,103	2,824	3,122	-0,238	0,019	53,701	18,62163
FREAK 01	0,506	0,022	0,535	0,026	0,029	0,004	62,367	16,03412
FREAK 02	1,609	2,483	1,582	2,026	-0,027	-0,457	50,225	19,9104
FREAK 03	0,008	0,024	0,011	0,018	0,003	-0,006	44,971	22,23655
FREAK 04	1,143	0,036	1,306	0,453	0,163	0,417	57,581	17,36684
CREAK 00	3,062	3,103	3,117	3,027	0,055	-0,076	59,707	16,74845
CREAK 01	0,506	0,022	0,591	0,086	0,085	0,064	82,856	12,06913
CREAK 02	1,609	2,483	1,485	0,165	-0,124	-2,318	63,424	15,7669
CREAK 03	0,008	0,024	0,028	0,063	0,02	0,039	77,016	12,98431
CREAK 04	1,143	0,036	0,922	0,187	-0,221	0,151	72,62	13,77031

From Table 4.2, it is apparent that the results continue the pattern, which can be expected as the rotational error is usually very closely correlated with the translational error. When turning to the performance section of Table 4.2, we observe that FREAK has significantly better processing times, maintaining an average frame rate of 18.59 fps, whereas CREAK only achieved an average frame rate of 14.059 fps where the processing time is the absolute measured time taken to process the 1000 frames, and the frame rate is the average frames per second as calculated from the processing time.

Figures 4.11 – 4.15 show the trajectory obtained by the odometry for datasets 00-04 for both FREAK and CREAK. In each figure, the ground truth is given in ‘red’, while the estimated trajectories are plotted in ‘cyan’. APPENDIX C shows various results obtained from performing the same odometry test with various amounts of keypoints generated for the descriptors. It is clear that there is a trade-off between performance and accuracy, where more keypoints result in better results, but at a slower rate. It is shown that at approximately 5000 keypoints per

frame, both FREAK and CREAK display the best results at a reasonably low frame rate, and therefore, 5000 keypoints were used in the entirety of this study.

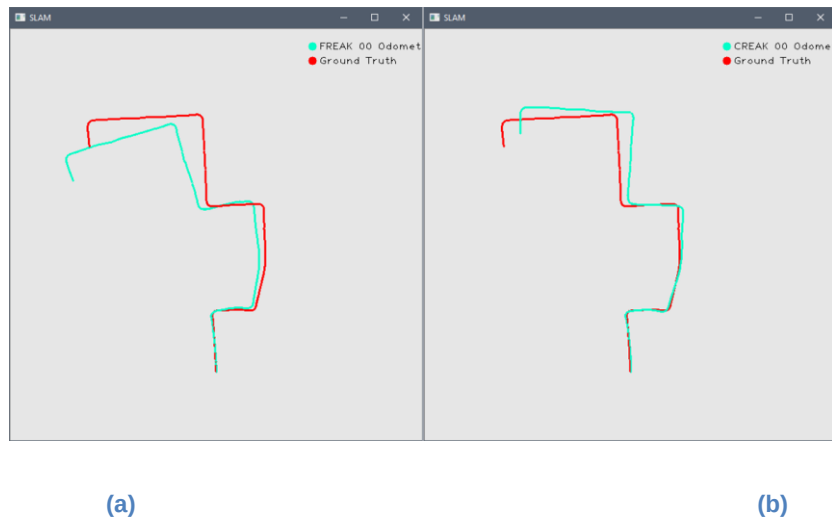


Figure 4.11 - (a) FREAK Visual Odometry and (b) CREAK Visual Odometry Trajectory for Dataset 00

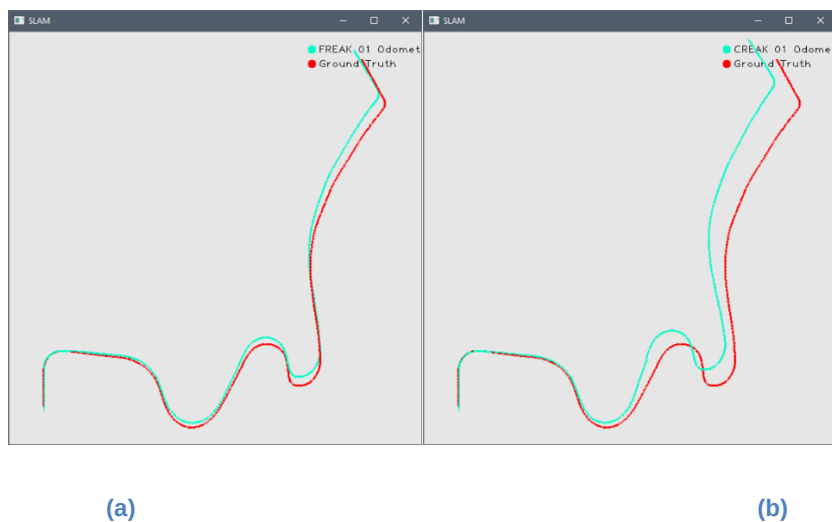


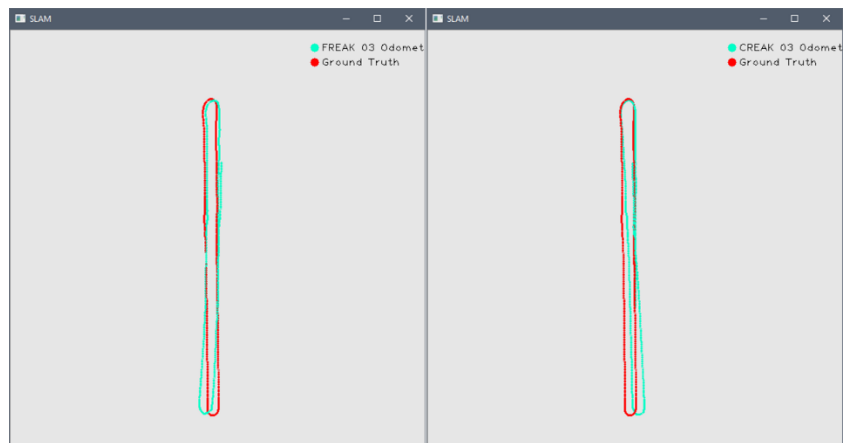
Figure 4.12 - (a) FREAK Visual Odometry and (b) CREAK Visual Odometry Trajectory for Dataset 01



(a)

(b)

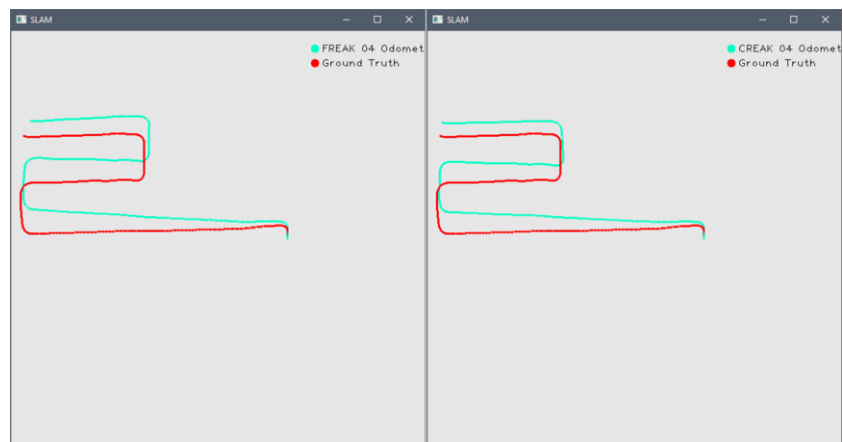
Figure 4.13 - (a) FREAK Visual Odometry and (b) CREAK Visual Odometry Trajectory for Dataset 02



(a)

(b)

Figure 4.14 - (a) FREAK Visual Odometry and (b) CREAK Visual Odometry Trajectory for Dataset 03



(a)

(b)

Figure 4.15 - (a) FREAK Visual Odometry and (b) CREAK Visual Odometry Trajectory for Dataset 04

4.4.3 EKF-SLAM Results

Building upon the results obtained by the monocular VO in the previous section, the EKF-SLAM algorithms are implemented on the same datasets. The tests that simulate the EKF-SLAM implementation on the five KITTI datasets by using the FREAK and CREAK descriptors have their results in terms of the position shown in Table 4.3 and rotation along with processing times shown in Table 4.4. As in the previous section, the processing time is the absolute measured value the program takes to process the 1000 frames, and the frame rate is the average frames per second as calculated from the processing time.

Table 4.3 - EKF-SLAM Translation Results

Descriptor/Dataset	EKF-SLAM Translation Results						
	True Position (m) (x, y, z)			Estimated Position (m) (x, y, z)			Error on Position (m) (x, y, z)
FREAK 00	-184,8	-3,554	328,513	-165,2	6,496	312,09	27,4766
FREAK 01	463,56	-48,891	509,303	465,85	-83,213	513,17	34,6152
FREAK 02	66,907	-8,643	232,934	69,786	-36,81	242,21	29,7939
FREAK 03	-1,768	-4,303	211,025	2,701	-12,24	210,13	9,15215
FREAK 04	-382	2,739	148,241	-382,7	71,551	148,01	68,8159
CREAK 00	-184,8	-3,554	328,513	-182,1	10,146	340,01	18,0873
CREAK 01	463,56	-48,891	509,303	450,36	6,372	515,29	57,1329
CREAK 02	66,907	-8,643	232,934	65,515	19,015	241,75	29,0618
CREAK 03	-1,768	-4,303	211,025	-1,996	-40,96	211,62	36,6625
CREAK 04	-382	2,739	148,241	-394,4	76,742	128,47	77,6022

As shown in Table 4.3, both FREAK and CREAK demonstrated a significant improvement upon implementing the EKF filter, as the average translational error for FREAK dropped from 45m to under 34m after an average distance travelled of 917m over the 1000 frames, while CREAK dropped from an average error of 49m to 43m. When considering only the two-dimensional x and z directions, FREAK's average error decreased from 22m to 9m, and CREAK from 23m to 11m. The y direction severely hinders CREAK's performance, with the y direction representing the vertical translation (altitude), the reason for which will be discussed further in this section.

It is apparent that (excluding the y coordinates) CREAK yields a translational error of 1.29%, where FREAK achieved an error of 0.98%. Table 4.4 displays the rotational and computational times obtained with the same simulation.

Table 4.4 - EKF-SLAM Rotation Results

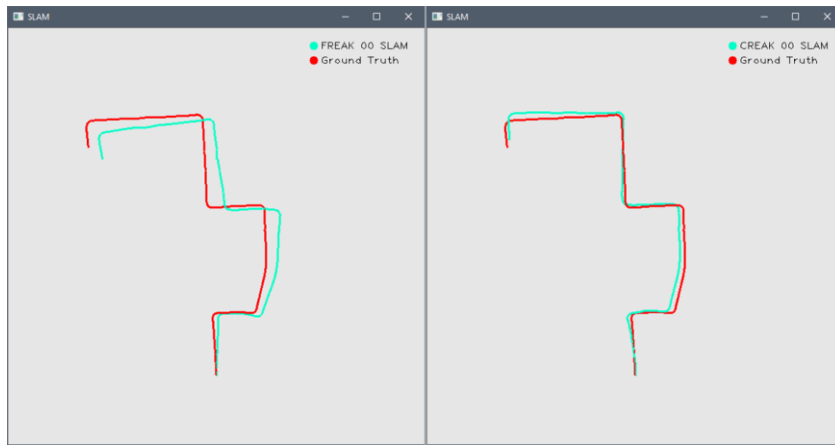
Descriptor/Dataset	Monocular Visual Odometry Rotation Results							
	True Rotation (<i>yaw, pitch</i>) (radians)		Estimated Rotation (<i>yaw, pitch</i>) (radians)		Error on Rotation (<i>yaw, pitch</i>) (radians)		Processing Time (s)	Frames per Second (fps)
FREAK 00	3,062	3,103	2,988	3,066	-0,074	-0,037	159,137	6,283894
FREAK 01	0,506	0,022	0,508	0,022	0,002	0	105,48	9,48047
FREAK 02	1,609	2,483	1,518	1,287	-0,091	-1,196	152,09	6,575054
FREAK 03	0,008	0,024	0,035	0,012	0,027	-0,012	65,35	15,30222
FREAK 04	1,143	0,036	1,369	0,552	0,226	0,516	178,43	5,604439
CREAK 00	3,062	3,103	3,11	3,04	0,048	-0,063	108,39	9,225943
CREAK 01	0,506	0,022	0,536	0,09	0,03	0,068	107,425	9,30882
CREAK 02	1,609	2,483	1,484	0,138	-0,125	-2,345	111,21	8,991997
CREAK 03	0,008	0,024	0,01	0,0397	0,002	0,0157	73,67	13,57405
CREAK 04	1,143	0,036	1,031	0,235	-0,112	0,199	123,81	8,076892

Table 4.4 further agrees with the pattern found in monocular VO results found in Table 4.2, where FREAK achieved a lower error compared to CREAK in both yaw as well as pitch angles, as given in radians. This could be expected, based on the monocular VO results obtained in the previous section. A very important result to note is that of the processing time taken to complete the simulation of 1000 frames. With monocular VO alone, FREAK is significantly faster than CREAK. However, with the included implementation of EKF-SLAM, CREAK is almost 50% faster than FREAK. Figures 4.16 to 4.20 show the trajectory obtained by FREAK and CREAK respectively with the implementation of EKF-SLAM.

Finally, the average number of detected features and matched features per frame for both FREAK and CREAK is displayed in Table 4.5.

Table 4.5 - Average Detected Features vs Average Matched Features per Frame

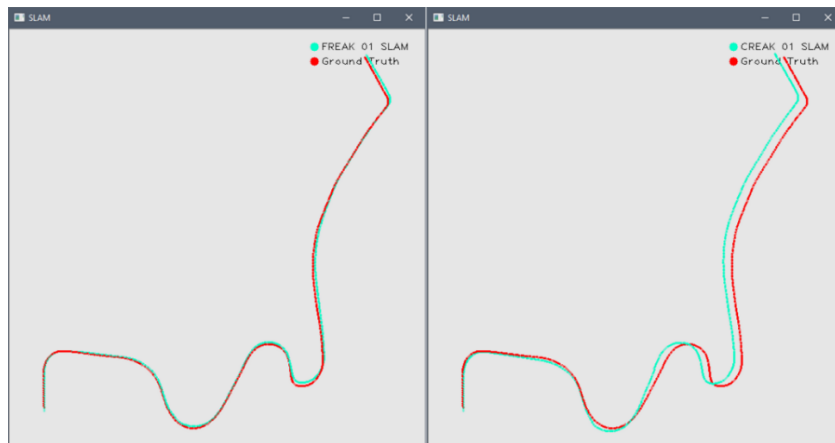
Dataset	Average Detected Features vs Average Matched Features per Frame		
	#Detected Features	#FREAK Matched Features	#CREAK Matched Features
00	1214	374	367
01	2203	432	406
02	1532	360	352
03	865	262	257
04	1466	397	378



(a)

(b)

Figure 4.16 - (a) FREAK EKF-SLAM and (b) CREAK EKF-SLAM Trajectory for Dataset 00



(a)

(b)

Figure 4.17 - (a) FREAK EKF-SLAM and (b) CREAK EKF-SLAM Trajectory for Dataset 01



(a)

(b)

Figure 4.18 - (a) FREAK EKF-SLAM and (b) CREAK EKF-SLAM Trajectory for Dataset 02



Figure 4.19 - (a) FREAK EKF-SLAM and (b) CREAK EKF-SLAM Trajectory for Dataset 03



Figure 4.20 - (a) FREAK EKF-SLAM and (b) CREAK EKF-SLAM Trajectory for Dataset 04

4.5 Discussion

The results found in this section proved interesting. With simulations regarding the monocular VO alone, the unique descriptiveness of CREAK due to including colour-space information proved to be very significant. In both the translation as well as rotational results, FREAK achieved a more accurate trajectory than that obtained by CREAK. Considering the results found in Chapter 3 regarding the number of matches, the results found in this chapter that represent a real-world scenario are somewhat to the contrary. In Chapter 3, each static image test resulted in FREAK obtaining slightly fewer matches compared to CREAK, although both descriptors appeared to be equally accurate. In the KITTI dataset, however, it is shown that CREAK instead obtains, on average, 10 fewer matches than FREAK on each frame, with the trade-off of a slightly less accurate trajectory.

This had little effect when considering the monocular VO alone in terms of computational times. However, when EKF-SLAM is implemented, the additional features added up quickly and resulted in a matrix with nearly 10,000 additional rows and columns, which is 10^8 entries more in FREAK than CREAK at the 1000th frame in the **P** matrix alone, hence the dramatic increase in computational times of FREAK as opposed to the relatively reasonable time achieved by CREAK. Both FREAK as well as CREAK obtained a vast improvement in the accuracy of the simulated trajectories with the implementation of the EKF-SLAM, although, FREAK resulted in a slightly greater improvement than that of FREAK. This can be accredited to the fact that FREAK obtains a more accurate VO by including more matches at each frame. This resulted in a much smaller error on the *y* axis which in turn had a smaller detrimental effect on the overall VO than that in CREAK. The RANSAC outlier removal implemented at the VO stage results in a case where more matches (although including a larger number of incorrect matches) is favoured above less, more accurate matches.

Finally, with the implementation of EKF-SLAM, FREAK achieved an improvement of around 26%, while CREAK achieved an improvement of nearly 12%. Regarding translation, FREAK resulted in an average error of 9mm per metre travelled, while CREAK obtained an error of 12.8mm per metre travelled. Regarding rotation, CREAK achieved a slightly higher improvement with the implementation of EKF-SLAM, with FREAK obtaining an average of 8.9% improvement, and CREAK obtaining an average of 37% improvement. The resulting error was at 0.074 radians per 1000 m travelled for FREAK, and 0.068 radians per 1000m travelled for CREAK.

4.6 Conclusion

By implementing monocular visual odometry, the results showed that FREAK performed slightly better when compared to CREAK. Furthermore, FREAK's faster algorithm resulted in FREAK achieving faster odometry, while still slightly more accurate.

The EKF-SLAM algorithm demonstrated results that show that there is a trade-off to using CREAK. Since the landmarks are kept at each time frame, instead of obsolete landmarks being removed from the system when no longer in sight, the results are extreme, yet still useful. CREAK demonstrated that consistently obtaining fewer matches in a frame while maintaining a matching accuracy on par with FREAK, was extremely favourable in the context of EKF-SLAM. Had a solution been implemented where the landmarks are removed when obsolete, the difference in computational performance might have been of a lesser effect.

The datasets used are those of a vehicle travelling in a series of real-world environments. Although still a man driven vehicle, none of the human or control inputs are used in our SLAM implementation. Therefore, the solution is valid for the context of SLAM for an unmanned vehicle.

CHAPTER 5 – CONCLUSIONS AND RECOMMENDATIONS

5.1 Introduction

Both the FREAK and CREAK descriptors have been thoroughly evaluated in this dissertation, yielding results both expected and unexpected. This chapter aims to conclude this dissertation as well as discuss and share the final thoughts about the findings obtained. Recommendations are provided for areas upon which further research should be done, and closure is given regarding the completion of each objective discussed in this dissertation.

5.2 Conclusion

Having re-created the FREAK descriptor, its performance was compared to that of the existing implementation of FREAK found in the OpenCV library. The results showed that in every case, the re-created FREAK descriptor achieved the exact same number of matches compared to the existing implementation. Regarding computational time, the re-created FREAK performs minimally faster than the existing implementation, due to the re-creation only being created with a specific application in mind, thus eliminating bulky code that ensures compatibility on any system, as with the OpenCV implementation.

CREAK is re-created upon the foundations laid by the re-created FREAK descriptor, and its performance evaluated against results shown in the proposing article of CREAK by Chen *et al.* The results obtained by the re-created CREAK were in complete agreement with those found by Chen. It was found that when compared to FREAK, CREAK achieved a more robust description yielding in more accurate matches, at the cost of an increase in computing time. With parameters replicated to those in the CREAK article [1], the results found by Chen when comparing FREAK to CREAK were re-created on the most difficult variants of the viewpoint change (Graffiti) image sets.

The implementation of visual odometry was completed on five KITTI Vision Benchmark Suite datasets and achieved expected results based on the findings obtained when verifying the FREAK and CREAK descriptor re-creations. When using FREAK for the description of features, the VO yielded a more accurate trajectory, and as could be expected, FREAK was the faster descriptor. The trajectory obtained is compared to the accurate ground truth data included in the datasets. The KITTI datasets comprise video footage of a vehicle travelling in a real-world environment, providing sensory data that is representative of those an unmanned vehicle can expect to have.

By implementing a partial EKF-SLAM solution, interesting results are obtained. Based on the findings from the implementations and visual odometry for FREAK and CREAK, it was expected that CREAK obtained fewer matches than FREAK at each frame. This was, in fact, the case, and as such, resulted in a dramatic increase in performance for CREAK. Due to the nature of the covariance matrix in the EKF being a square matrix with a size directly proportional to the number of landmarks observed, the fewer matches in the case of CREAK ended up being rather favourable. In the case of this dissertation, the EKF-SLAM implementation keeps all the landmarks indefinitely, despite most becoming obsolete once they are out of sight. This means that should a different approach be taken where obsolete landmarks are removed from the system, CREAK would not obtain such a drastic advantage over FREAK, and the difference in computational performance would be negligible. When considering the performance in terms of trajectory accuracy alone, CREAK still proves superior with a notably smaller error as compared to FREAK.

Therefore, when taking into account that the final SLAM solution might not necessarily be the ideal solution in practice, the solution is still a valid representation of any similar SLAM implementation. Therefore, the results obtained in this dissertation are considered in the evaluation of the CREAK descriptor in the context of SLAM for an unmanned vehicle. When considering basic odometry alone, the choice between FREAK and CREAK becomes situational where there is a trade-off between accuracy and speed. However, should EKF-SLAM be implemented, the CREAK descriptor achieved notably comparable results to FREAK, but with 50% faster computational times. Therefore, CREAK can be concluded as the superior descriptor in the context of SLAM in terms of speed, but when accuracy is the dominant factor, FREAK remains the ideal descriptor.

5.3 Future Work

Although this study does not find the solution to the sought after “real-time” SLAM problem, the information obtained is a great stepping-stone in the right direction.

It is believed that the use of camera sensors is part of the answer in pursuit of a real-time SLAM solution due to the low cost and nearly negligible computational requirements. The use of monocular vision in this dissertation might, admittedly, be a sub-optimal choice as opposed to the stereo vision alternative. It might be debated upon whether the decrease in computational complexity, from having half the features to process, justifies the increased complexity of trying to localize the landmarks in 3D, from monocular vision.

Therefore, a recommendation for future work is to evaluate the CREAK descriptor in the context of SLAM for an unmanned vehicle, by making use of a stereo vision setup.

Furthermore, the EKF implementation in this dissertation is only a partial solution, and although the results are relevant for most EKF-SLAM solutions, there exists room for improvement. It is therefore proposed that future work is done on the evaluation of CREAK in various SLAM solutions, with the pursuit of a real-time solution in mind.

Finally, future work is proposed on the creation of a real-world implementation of CREAK within a full vSLAM solution, where a real unmanned vehicle navigates through a previously unknown environment.

5.4 Closure

The aim of this dissertation was to *“Evaluate the CREAK descriptor against the FREAK descriptor in the context of monocular vSLAM on an unmanned vehicle”*.

In this dissertation, the FREAK and CREAK descriptors are re-created, and their performance verified against their proposing articles. The EKF is used in the SLAM implementation and is performed on a reliable dataset with accurate ground truth data. The dataset consists of a monocular video feed of a vehicle travelling in a real-world environment. Despite not being an unmanned vehicle, no control data is used, and therefore is an accurate representation of sensory data available to an unmanned vehicle.

Although the EKF-SLAM implementation discussed in this dissertation is not necessarily the most optimal choice, it is representative of any similar EKF-SLAM solution because the evaluation consists mainly of the accuracy of the matches obtained with CREAK as opposed to FREAK. Evaluation concerning the speed of CREAK is still valid in the context of SLAM because a better solution will experience either negligible or improved results in favour of CREAK.

REFERENCES

- [1] Y. Chen, C. Chan, and W. Tsai, "CREAK: Color-based Retina Keypoint Descriptor," *Proc. Int. Conf. Image Process. Comput. Vision, Pattern Recognit.*, pp. 252–258, 2016.
- [2] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast retina keypoint," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 510–517, 2012.
- [3] "University of OXFORD: Visual Geometry Group," 2004. [Online]. Available: <http://www.robots.ox.ac.uk/~vgg/data/data-aff.html>.
- [4] A. Singh, "Monocular Visual Odometry using OpenCV," *Avi Singh's blog*, 2015. [Online]. Available: <https://avisingh599.github.io/vision/monocular-vo/>.
- [5] A. Geiger, P. Lenz, and C. Stiller, "Vision meets Robotics: The KITTI Dataset," *International Journal of Robotics Research (IJRR)*, 2013.
- [6] S. Riisgaard and M. R. Blas, "SLAM for Dummies," *A Tutor. Approach to Simultaneous Localization Mapp.*, vol. 22, no. June, pp. 1–127, 2004.
- [7] J. Blanco, "Derivation and implementation of a full 6D EKF-based solution to bearing-range SLAM," *Univ. Malaga, Spain*, pp. 1–21, 2008.
- [8] W. Burgard, M. Bennewitz, D. Tipaldi, and L. Spinello, "Introduction to Mobile Robotics SLAM: Simultaneous Localization and Mapping," 2014.
- [9] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping (SLAM): part I The Essential Algorithms," *Robot. Autom. Mag.*, vol. 2, pp. 99–110, 2006.
- [10] N. Karlsson, E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. E. Munich, "The vSLAM algorithm for robust localization and mapping," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2005, vol. 2005, pp. 24–29.
- [11] W. Brink, "Stereo vision for simultaneous localization and mapping," M.S. Thesis, University of Stellenbosch, 2012.
- [12] O. S. Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, 2007.
- [13] P. C. Randall Smith, Matthew Self, "Estimating Uncertain Spatial Relationships in

Robotics * 333 Ravenswood Avenue,” *Contract*.

- [14] D. Simon, “Kalman Filtering,” *Embed. Syst. Program.*, no. June, pp. 72–79, 2001.
- [15] G. Welch and G. Bishop, “An Introduction to the Kalman Filter,” *In Pract.*, vol. 7, no. 1, pp. 1–16, 2006.
- [16] J. Solà, “Simultaneous localization and mapping with the extended Kalman Filter,” 2014.
- [17] D. G. Viswanathan, “Features from Accelerated Segment Test (FAST),” 2011, [online]. Available: [https://www.semanticscholar.org/paper/Features-from-Accelerated-Segment-Test-\(-FAST-\)-Viswanathan/cd267a4b04d835dbecf01d47fc69ed3a38c23055](https://www.semanticscholar.org/paper/Features-from-Accelerated-Segment-Test-(-FAST-)-Viswanathan/cd267a4b04d835dbecf01d47fc69ed3a38c23055). [Accessed: 20-Oct-2018]
- [18] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3951 LNCS, pp. 430–443, 2006.
- [19] G. Lowe, “SIFT - The Scale Invariant Feature Transform,” *Int. J.*, vol. 2, pp. 91–110, 2004.
- [20] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3951 LNCS, pp. 404–417, 2006.
- [21] C. Harris and M. Stephens, “A Combined Corner and Edge Detector,” *Proceedings Alvey Vis. Conf. 1988*, pp. 147–151, 1988.
- [22] O. Miksik and K. Mikolajczyk, “Evaluation of Local Detectors and Descriptors for Fast Feature Matching,” *Pattern Recognit. (ICPR), 2012 21st Int. Conf.*, no. Icp, pp. 2681–2684, 2012.
- [23] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 2564–2571, 2011.
- [24] P. L. Rosin and M. Ub, “Measuring Corner Properties.”, *Computer Vision and Image Understanding*, vol. 73 pp. 291-307, 1997
- [25] L. I. U. Feng, L. V Qiang, L. I. N. Huican, Z. Yang, and Q. I. Kexin, “An image registration algorithm based on FREAK-FAST for visual SLAM,” no. 1, pp. 6222–6226, 2016.

- [26] A. Patel, D. R. Kasat, S. Jain, M. Gwalior, and V. M. Thakare, "Performance Analysis of Various Feature Detector and Descriptor for Real-Time Video based Face Tracking," *Int. J. Comput. Appl.*, vol. 93, no. 1, pp. 975–8887, 2014.
- [27] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6314 LNCS, no. PART 4, pp. 778–792, 2010.
- [28] B. Brisk and O. R. B. Card, "CV Reading BRISK: Binary Robust Invariant Scalable Keypoints," pp. 1–8, 2011.
- [29] E. Tola, V. Lepetit, and P. Fua, "DAISY: An efficient dense descriptor applied to wide-baseline stereo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 815–830, 2010.
- [30] G. Levi and T. Hassner, "LATCH: Learned arrangements of three patch codes," *2016 IEEE Winter Conf. Appl. Comput. Vision, WACV 2016*, 2016.
- [31] R. Nave, "HyperPhysics Light and Vision." [Online]. Available: <http://hyperphysics.phy-astr.gsu.edu/hbase/vision/rodcone.html#c3>. [Accessed: 20-Oct-2018].
- [32] J. B. D. Nister, O. Naroditsky, "Visual odometry," *Proc. 2004 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition, 2004. CVPR 2004.*, vol. 1, pp. I–I, 2004.
- [33] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast Semi-Direct Monocular Visual Odometry."
- [34] D. J. Mankowitz and E. Rivlin, "CFORB: Circular FREAK-ORB Visual Odometry," 2015. [online] Available: https://www.researchgate.net/publication/278733764_CFORB_Circular_FREAK-ORB_Visual_Odometry. [Accessed: 25-Oct-2018]
- [35] I. Dryanovski, R. G. Valenti, and J. Xiao, "Fast visual odometry and mapping from RGB-D data," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2013, pp. 2305–2310.
- [36] F. S. Osório, D. F. Wolf, K. C. Branco, V. Grassi, M. Becker, and R. A. F. Romero, "Visual Odometry and Mapping for Indoor Environments Using RGB-D Cameras," in *Communications in Computer and Information Science*, 2015, vol. 507, pp. 16–31.
- [37] A. Huang and A. Bachrach, "Visual odometry and mapping for autonomous flight using an RGB-D camera," *Int. Symp.*, pp. 1–16, 2011.

- [38] D. Nistér, "An Efficient Solution to the Five-Point Relative Pose Problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 756–770, 2004.
- [39] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, pp. 133–135, 1981.
- [40] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, pp. 381–395, 1981.
- [41] D. Sunday, "Distance between 3D Lines & Segments," *iSurfer.org*, 2001. [Online]. Available: http://geomalgorithms.com/a07-_distance.html.
- [42] G. Bradski, "The OpenCV Library", Dr. Dobb's Journal of Software Tools, 2000, [Online]. FREAK Available: https://docs.opencv.org/3.4.3/df/db4/classcv_1_1xfeatures2d_1_1FREAK.html
- [43] <https://github.com/dehansmulders/Masters-Project.git>

APPENDIX A - PUBLISHED WORK

CREAK descriptor evaluation for visual odometry

Dehan Smulders^{*1}, Kenneth Uren^{*2}, George van Schoor^{*3}, Corné van Daalen^{**4} and Japie Engelbrecht^{**5}

^{*}*School of Computer and Electronic Engineering, North-West University, Potchefstroom, South Africa*

^{**}*School of Electrical, Electronic and Computer Engineering, Stellenbosch University, Stellenbosch, South Africa*

¹dehansmulders@gmail.com, ²Kenny.Uren@nwu.ac.za, ³George.vanSchoor@nwu.ac.za, ⁴cvdaalen@sun.ac.za, ⁵jengelbr@sun.ac.za

Abstract—Each year, latest state of the art technologies and algorithms arise that claim and prove to out-shine their predecessors. One such algorithm is the Colour-based Retina Key-point (CREAK) descriptor, which is based on the FAST Retina Key-point (FREAK) descriptor with the included functionality of considering colour information in its Key-point description. This paper explores the implementation of CREAK in a “real-time” visual odometry application by means of a comparative study of the more well-known FREAK algorithm. Although FREAK achieved more accurate odometry when key-points were abundant, this proved to be too computationally expensive. CREAK on the other hand outperformed FREAK when key-points are scarce due to its lack of false-positive matches.

Keywords—FREAK, CREAK, visual odometry, feature descriptor, evaluation

I. INTRODUCTION

In the modern age of automation, odometry is the prevalent field of interest. Finding a computationally efficient method to determine odometry that is accurate as well as cost effective is no simple task. Visual odometry is the method of determining location by using visual image input. In most cases, visual odometry is required by unmanned robotic vehicles to determine their current location, and therefore visual odometry is required to be in real-time. We define “real-time” as a rate no slower than 10 frames per second (FPS), since according to the KITTI Vision Benchmark Suite [1] on which the odometry is performed, the camera setup is set to capture an image every 100 ms or at a rate of 10 FPS.

In visual odometry, the input image needs to be processed to determine detected key-points. These key-points then need to be described and then matched to the same key-points in a later point in time. Thereafter, the visual odometry is determined. This paper will focus on the description aspect of the visual odometry problem. The Fast Retina Key-point (FREAK) descriptor was first introduced by Alahi *et al.* in [2], which is inspired by the human retina. This descriptor is a binary descriptor, meaning that the hamming distance can be calculated to determine matches, which is highly efficient computationally.

The FREAK descriptor is highly respected in the image processing community due to its high descriptiveness and low computational complexity. More recently, Chen *et al.* [3] proposed the Colour-based Retina Key-point (CREAK) descriptor, that, similar to FREAK, is modelled after the human retina. However, CREAK takes the colour-space information into account when computing the descriptor, resulting in higher descriptive power.

The outline of this paper is as follows: The next section will discuss the related work performed in the field of visual odometry and key-point detectors and descriptors, followed by section III describing the experimental design used for the

evaluation of the visual odometry by using FREAK and CREAK. In section IV some results will be given and discussed. Finally, the paper will conclude in section V.

II. RELATED WORK

A. Key-point Detectors

A key-point is a point in an image that can be detected and reliably re-detected by key-point detector software. Many detectors have been created throughout the years, each implementing their own variation for determining key-points from an image. The first approach for detectors include corner detection methods, that function by comparing the neighbouring pixel intensities around a pixel of interest to detect ‘corners’ within the image scene.

Another more complex approach for detectors were designed that makes use of the responses of certain filters to determine possible key-points. The Features from Accelerated Segment Test (FAST) detector [4] is a highly robust and efficient detector that makes use of the former. In FAST, a pixel of interest p is considered within the image. Thereafter, a circle of 16 pixels surrounding the pixel of interest is compared against a threshold intensity value. Should a continuous number N of surrounding pixels with their respective intensities, be within the threshold value, the pixel of interest will be labelled a key-point.

More recently, Rublee *et al.* in [6] proposed the Oriented FAST and Rotated BRIEF (ORB) detector and descriptor. ORB identified that the FAST detector does not provide a measure for cornerness, and after determining FAST key-points, a Harris corner measure in employed as in [7] to place the FAST key-points in order. Thereafter ORB picks the top N key-points. Once key-points have been detected using a detector algorithm such as FAST or ORB, they are not yet useful for the visual odometry application. For them to serve their purpose as landmarks in visual odometry, each key-point needs to be described such as to recognize them when seen in a later point in time. Such algorithms are known as descriptors.

Feature based descriptors can be categorized into two main groups, float-point based and binary descriptors. Float-point based descriptors such as the Scale Invariant Feature Transform (SIFT) [8] and Speeded Up Robust Features (SURF) [9] use a vector of histograms to describe the key-point. Therefore, the Euclidian distance between the key-points need to be calculated in the matching process.

The binary descriptor algorithms such as ORB [6], FREAK [2], and CREAK [3] makes use of a bit-set array as a descriptor. This allows for the hamming distance to be calculated between key-points in the matching process, which is significantly faster to perform.

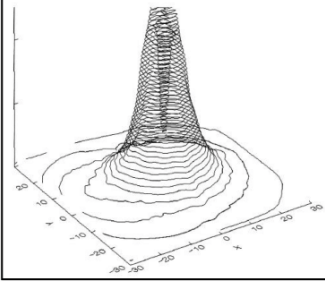


Fig. 1. Density of Ganglion Cells [2]

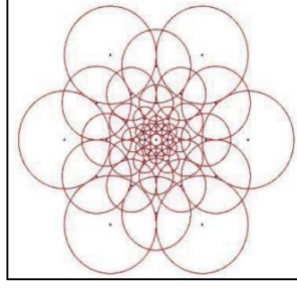


Fig. 2. FREAK Kernel Pattern [2]

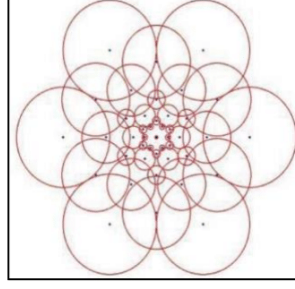


Fig. 3. CREAK Kernel Pattern [3]

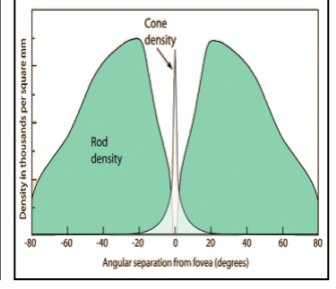


Fig. 4. Density of Cone Cells [5]

B. FREAK Descriptor

FREAK is a binary descriptor modelled after the human retina. The retina comprises of multiple concentric areas known as kernels, that each has a concentration of ganglion cells that gets denser the closer they are to the foveal (Centre of the eye) as shown in Fig. 1. The region where light influences the response of a ganglion cell is the receptive field, which increases in size with radial distance from the foveal. The special distribution of ganglion cells are distributed into four main regions, each detecting exponentially less resolution and image detail proportional to the distance from the foveal. This decrease in resolution can be interpreted as body resource optimisation.

Re-creating the retinal analogy digitally, photoreceptors determine the light intensity of pixels within an image, then the ganglion cells compare the average pixel intensities from specific areas to one another to determine the binary string descriptor. This process is described below:

- To replicate the distribution of ganglion cells, the sampling pattern seen in Fig. 2, is created around the key-point point of interest with each kernel area representing a retinal receptive field.
- Each kernel is smoothed to be less sensitive to noise and the average pixel intensity of each kernel is determined.
- The intensities of each kernel is compared to each other, and bit set depending on the difference in intensities.

Since there are 43 kernels as shown in Fig. 2, leaving a total of 903 possible matching pairs, not all pairs need to be compared as not all convey useful information. Therefore, an approach similar to ORB [6] was taken to learn the best 512 matching pairs for the descriptor from training data. The binary descriptor F is constructed by thresholding the difference between pairs of receptive fields with their corresponding Gaussian kernel. In other words, F is a binary string formed by a sequence of one-bit Difference of Gaussians (DoG):

$$F = \sum_{0 \leq a < N} 2^a T(P_a) \quad (1)$$

where P_a is a pair of receptive fields, N is the desired size of the descriptor, and $T(P_a)$ is:

$$T(P_a) = \begin{cases} 1 & \text{if } (I(P_a^{r1}) - I(P_a^{r2}) > 0, \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Since 512 matching pairs are considered, the descriptor will be 512 bits (or 64 bytes) in size.

C. CREAK Descriptor

CREAK is also a binary descriptor modelled after the human retina and is built upon the foundations laid by FREAK. Due to the nature of modern binary descriptors such as ORB [6] and FREAK [2], only the grey intensity values of images are considered. This negates a significant amount of descriptive information that can be achieved by considering the blue, red and green colour values. According to Chen *et al.* in [3], by taking the colour-space information into account will significantly aid in creating a descriptor that is more robust and discriminative.

As in FREAK, the CREAK descriptor functions with a deliberate sampling pattern, designed to mimic the functionality of the human retina and the spatial distribution of ganglion cells in receptive fields. However, the CREAK descriptor had made alterations to the design of the sampling pattern, as shown in Fig. 3.

Due to the overlapping that occurs in the innermost kernels, spatial redundancy increases and therefore promotes better discriminative power in the CREAK descriptor. Because colour information is being considered, the density of photoreceptive cone cells (which determine colour information) is higher than rods (which determine light intensity) near the foveal as shown in Fig. 4. The kernels near the centre are then decreased further to preserve colour information and to better resemble the retina model.

The CREAK descriptor algorithm is based on the foundation laid by FREAK, and from the article describing CREAK in [3], the general operational process of CREAK remains the same as in FREAK, however the colour image is split into its three components, to reveal three separate grey images, each representing the colour-spaces for blue, green and red intensities. The average kernel intensities are then determined for each of the three colour-spaces, and the kernel comparisons are performed for each.

The binary descriptor D is then constructed by thresholding the difference between pairs of receptive fields with their corresponding Gaussian kernel, but repeated once for each of the three colour-spaces. In other words, D is a binary string formed by a concatenation of sequences of one-bit Difference of Gaussians (DoG):

$$D = \sum_{i=0}^{\frac{N}{3}-1} [C_i] \quad (3)$$

and C_i is defined as:

$$C_i = 2^i T(B_i) + 2^{i + \frac{N}{3}} T(G_i) + 2^{i + \frac{2N}{3}} T(R_i), \quad (4)$$

where B , G , R represent colour channels blue, green, and red, while B_i , G_i , and R_i are colour test pairs of receptive fields with their respective corresponding channels. The form of $T(B_i)$, $T(G_i)$ and $T(R_i)$ are each functionally the same as $T(P_a)$ in FREAK, described by (2).

The complete binary descriptor D of size N is formed by concatenating three $N/3$ binary test results with Chen *et al.* in [3] claiming that $N = 192$ is the optimal descriptor size, and enlarging the size yields no apparent improvement. Therefore 64 bits are reserved for each colour-space.

D. Visual Odometry

Complex solutions to visual odometry known as Simultaneous Localization and Mapping (SLAM) [10], exist and are widely used in the robotics and automation community. Some of these SLAM solutions such as [11], [12] use complex algorithms to determine the 3D coordinates of each key-point, then stores the key-point, its 3D relative coordinates, as well as description in a database. Then from recognizing key-points from the database later in time, and from a different perspective, the SLAM algorithm can then calculate the new current position of the vehicle.

Less complex solutions exist that can take several matched key-points from two images taken from different points in time and determine the camera's current change in translation and rotation. This is done by using the Essential matrix discussed in [13]. Once the Essential matrix is known, R , t and the camera matrices can be recovered from it, where R is the rotation and t is the translation matrices. The accuracy of the rotation and translation values are solely dependent on the quality of matched key-points. Therefore, if the pixel locations of matched key-points are without error, the resulting odometry obtained will be without error as well.

The Essential matrix approach only takes two image frames into consideration. Therefore, no knowledge of prior detected key-points exist. This also implies that if an error occurs at any point in time, the error will be taken forward without any hope of restoration, unless loop closure methods are introduced. Nistér in [13] proposed that the Random Sampling Consensus (RANSAC) outlier detector be used when computing the Essential matrix.

RANSAC is an iterative algorithm that randomly samples five points from the set of matched key-points, determines the Essential matrix, then checks if all other points are inliers with the model. After a fixed number of iterations, the Essential matrix with which the maximum number of matched key-points agree, is used.

III. EXPERIMENTAL DESIGN

For this study, the CREAK descriptor will be evaluated against the FREAK descriptor and will be compared specifically in a visual odometry implementation. The

Microsoft Visual Studio 2017 environment will be used to perform all necessary tasks in C++. Chen in [3] claimed that using the ORB detector for both FREAK as well as CREAK yielded unbiased results, thus the ORB detector is used.

Before the descriptors are implemented in visual odometry, they first need to be re-created from their respective articles. The OpenCV library includes the FREAK descriptor, thus it can be used to verify the re-created FREAK algorithm. Although the OpenCV library includes a FREAK function, it is necessary to re-create the FREAK descriptor algorithm because the CREAK descriptor that is to be created, will be built upon the foundation laid by FREAK. Therefore, by re-creating the FREAK descriptor, the performance of CREAK can be more accurately determined, as any bias from lack of optimization can be negated.

The operational flow of the odometry evaluation application as is shown in Fig. 5:

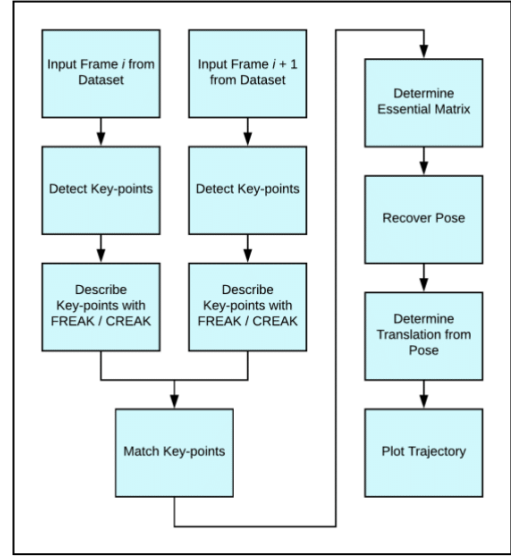


Fig. 5. Operational Flow for Odometry Evaluation Application

As described in Fig 5, the application iterates through all the frames from the KITTI Vision Benchmark Suite [1] dataset, considering two sequential frames at a time. Key-points are extracted for each individual frame, and then described using the FREAK or CREAK descriptor as desired. The described key-points are then matched to identify correlating key-points on each frame, and then used to determine the Essential matrix. From this, the pose is recovered, and the translation determined. The results are then plotted as the trajectory.

IV. RESULTS

To compare the descriptor results, the images had been taken from the University of Oxford Visual Geometry Dataset [14]. Each dataset consists of a scene, and five variations thereof. Each scene has a different type of transformation occurring, such as viewpoint change, rotation, zoom, blur and luminance. Fig. 9 to Fig. 16 show the most difficult, successful result of each dataset. TABLE I shows the results of FREAK and CREAK on the datasets. The time in *ms* for the descriptor to perform description as well as the number of matches achieved (in some cases where

applicable, incorrect matches are indicated with a forward slash in the form $\#matches/\#correct_matches$).

The compared image scenes from the dataset take the first scene as reference and iterate through the varying difficulties (the sixth scene will always have the most extreme transformation and will thus be most difficult) and is shown as *reference_image|compared_image*.

TABLE I. FREAK VS CREAK MATCHING DATASETS

	FREAK		CREAK	
	Time(ms)	Matches	Time(ms)	Matches
Graffiti (Viewpoint)				
1 2	4	150	10	127
1 3	5	55	11	49
1 4	4	16	10	8
1 5	4	4/0	11	8/7
1 6	4	5/0	9	0
Bark (Zoom/Rotation)				
1 2	3	18	8	17
1 3	2	2	8	2
1 4	3	5/2	6	2
1 5	3	0	7	0
1 6	3	0	6	0
Bikes (Blur)				
1 2	7	285	15	260
1 3	6	250	15	219
1 4	7	197	15	177
1 5	6	167/166	14	135
1 6	6	111/110	12	90
Leuven (Luminance)				
1 2	4	82	7	74
1 3	3	53	7	51
1 4	3	46/45	7	39/38
1 5	4	37	6	33
1 6	3	32	8	29

As shown by TABLE I, the FREAK descriptor was significantly faster than CREAK, however this was expected as Chen *et al.* in [3] also found that the description times were almost twice that of FREAK.

Comparing the matching results of the FREAK and CREAK descriptors in TABLE I, we can see that the results are very similar. FREAK does have slightly more matches in every scenario, however CREAK appears to be more consistent, achieving less false positive matches. In two cases, FREAK claimed to have 4 and 5 matches, of which all were incorrect. This is detrimental in a visual odometry implementation, as any false positive match can have huge effects on the odometry error. This was put to test in a visual odometry implementation, where a crude and minimalistic visual odometry algorithm was set up.

The visual material was downloaded from the KITTI Vision Benchmark Suite [1] and a simple C++ application



Fig. 6. Detected and Tracked Key-points using the CREAK descriptor

was created to display the results. Fig. 6 displays the application's visual feed as input, with all matched key-points drawn over the image. The cyan lines are connecting lines that match key-points to their location in the previous frame. Once again, the ORB detector was used in three of the experiments that were performed. The ORB detector was set to generate 500, 1000 and 2000 key-points respectively in each experiment. The odometry results obtained for 2000 frames can be found in Fig. 7, where the cyan path represents the ground truth, the magenta representing FREAK and the red representing CREAK.



Fig. 7. Odometry with ORB detecting 500, 1000 and 2000 Key-points

Thereafter, the FAST detector was used that generated roughly 5000 key-points per frame. This took significantly longer in processing time, as shown in TABLE II. The results are displayed in Fig. 8, again with cyan representing the ground truth, magenta representing FREAK and red representing CREAK.



Fig. 8. Odometry with FAST detecting roughly 5000 Key-points



Fig. 9. Graffiti 1/5 Viewpoint FREAK results (4 Matches, 0 Correct)



Fig. 11. Graffiti 1/5 Viewpoint CREAK results (8 Matches, 7 Correct)



Fig. 10. Bark 1/4 Rotation/Zoom FREAK results (5 Matches, 2 Correct)



Fig. 12. Bark 1/4 Rotation/Zoom CREAK results (2 Matches, 2 Correct)



Fig. 13. Bikes 1/6 Blur FREAK results (167 Matches, 166 Correct)



Fig. 15. Bikes 1/6 Blur CREAK results (135 Matches, 135 Correct)



Fig. 14. Leuven 1/6 Luminance FREAK results (32 Matches, 32 Correct)

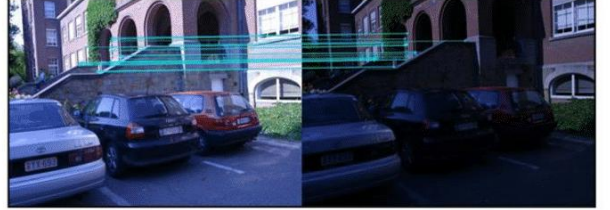


Fig. 16. Leuven 1/6 Luminance CREAK results (29 Matches, 29 Correct)

From Fig. 7 we can conclude that even though the visual odometry algorithm itself is nowhere near ideal, for the purposes of evaluating the CREAK descriptor in a visual odometry environment, it is sufficient. In each case where the ORB detector had been used, CREAK is clearly superior to FREAK in terms of descriptive power. The odometry path created by using CREAK bears closer resemblance to the ground truth, compared to FREAK in each case of 500, 1000 and 2000 key-points. Performance for more than 2000 key-points were not explored, because as TABLE II shows, more key-points would take longer to compute, and therefore would not be classified as “real-time” performance.

In Fig. 8, where the FAST detector had been used, FREAK does in fact yield better results when compared to the results of CREAK, however this is theorized to be due to the fact that using the FREAK descriptor always results in a higher number of matches, although not always correct matches, as shown in TABLE I. Due to the sheer volume of key-points generated by the FAST detector, and the 5-point relative pose method of [13] in the Essential matrix, there were enough correct matches to nullify the incorrect

matches, and therefore FREAK allowed the Essential matrix to more accurately model the odometry.

TABLE II. TIME TO PROCESS 2000 FRAMES

Detector	FREAK		CREAK	
	Time(s)	FPS	Time(s)	FPS
ORB 500	121.61	16.44	143.15	13.97
ORB 1000	135.66	14.74	158.26	12.64
ORB 2000	157.36	12.70	194.06	10.31
FAST	410.31	4.87	583.58	3.43

V. CONCLUSION

The CREAK descriptor had been implemented in a crude visual odometry algorithm, and results compared to the more well-known FREAK descriptor. Results showed that FREAK was more accurate in terms of odometry when ample key-points were supplied. However, the computation time was much slower than 10 FPS, and could thus not classify as “real-time”. In a more realistic application, where less key-points are supplied, CREAK outperforms FREAK by a notable margin, therefore we can conclude that in a visual odometry environment specifically, CREAK is not only

viable, but superior to FREAK as a rival binary descriptor, as it demonstrated more reliable descriptive power, especially in a “real-time” application.

Although not pursued in this study, CREAK is theorised to be advantageous in a SLAM environment, as its smaller bit size requires less storage when a large database of descriptors had been acquired, therefore future work will evaluate the CREAK descriptor in a “real-time” SLAM application.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of South Africa under the following grant number: TP2011073100016. Note, however, that the grant holder acknowledges that the opinions, findings, conclusions and recommendations expressed in any publication generated by the NRF supported research project are that of the author. The NRF accepts no liability in this regard.

REFERENCES

- [1] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [2] A. Alahi, R. Ortiz, and P. Vandergheynst, “FREAK: Fast retina keypoint,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 510–517, 2012.
- [3] Y. Chen, C. Chan, and W. Tsai, “CREAK : Color-based Retina Keypoint Descriptor,” *Proc. Int. Conf. Image Process. Comput. Vision, Pattern Recognit.*, pp. 252–258, 2016.
- [4] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3951 LNCS, pp. 430–443, 2006.
- [5] R. Nave, “HyperPhysics Light and Vision.” [Online]. Available: <http://hyperphysics.phy-astr.gsu.edu/hbase/vision/rodcone.html#c3>. [Accessed: 20-Oct-2018].
- [6] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 2564–2571, 2011.
- [7] C. Harris and M. Stephens, “A Combined Corner and Edge Detector,” *Proceedings Alvey Vis. Conf. 1988*, pp. 147–151, 1988.
- [8] G. Lowe, “SIFT - The Scale Invariant Feature Transform,” *Int. J.*, vol. 2, pp. 91–110, 2004.
- [9] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3951 LNCS, pp. 404–417, 2006.
- [10] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping (SLAM): part I The Essential Algorithms,” *Robot. Autom. Mag.*, vol. 2, pp. 99–110, 2006.
- [11] D. J. Mankowitz and E. Rivlin, “CFORB: Circular FREAK-ORB Visual Odometry,” 2015.
- [12] N. Karlsson, E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. E. Munich, “The vSLAM algorithm for robust localization and mapping,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2005, vol. 2005, pp. 24–29.
- [13] D. Nistér, “An Efficient Solution to the Five-Point Relative Pose Problem,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 756–770, 2004.
- [14] “University of OXFORD: Visual Geometry Group,” 2004. [Online]. Available: <http://www.robots.ox.ac.uk/~vgg/data/data-aff.html>.*

APPENDIX B - THRESHOLD SWEEP RESULTS

Table B.1 - FREAK Threshold Sweep Translation Results

Dataset	Threshold	True (x, y, z)			Estimated (x, y, z)		
0	0,3	-184,826	-3,554	328,513	188,51	129,738	272,313
0	0,4	-184,826	-3,554	328,513	184,882	-24,749	150,26
0	0,5	-184,826	-3,554	328,513	-89,351	46,916	363,437
0	0,6	-184,826	-3,554	328,513	-159,163	2,914	330,584
0	0,7	-184,826	-3,554	328,513	43,665	-161,67	80,142
0	0,8	-184,826	-3,554	328,513	-209,636	-11,767	279,769
0	0,9	-184,826	-3,554	328,513	-191,302	12,56	292,118
1	0,3	463,563	-48,891	509,303	-479,416	38,448	-432,52
1	0,4	463,563	-48,891	509,303	547,595	320,078	364,58
1	0,5	463,563	-48,891	509,303	293,246	-56,787	543,536
1	0,6	463,563	-48,891	509,303	356,982	-66,904	544,13
1	0,7	463,563	-48,891	509,303	436,41	-90,043	514,838
1	0,8	463,563	-48,891	509,303	452,183	-86,644	523,383
1	0,9	463,563	-48,891	509,303	459,96	-48,2	514,535
2	0,3	66,907	-8,643	232,934	40,924	-34,076	201,218
2	0,4	66,907	-8,643	232,934	51,174	22,905	205,143
2	0,5	66,907	-8,643	232,934	70,08	-40,34	240,478
2	0,6	66,907	-8,643	232,934	72,204	24,553	248,815
2	0,7	66,907	-8,643	232,934	44,301	-18,021	212,179
2	0,8	66,907	-8,643	232,934	70,961	34,956	235,34
2	0,9	66,907	-8,643	232,934	64,654	55,419	237,36
3	0,3	-1,768	-4,303	211,025	-352,146	153,017	167,663
3	0,4	-1,768	-4,303	211,025	562,091	-14,114	-132,326
3	0,5	-1,768	-4,303	211,025	38,307	-52,313	190,131
3	0,6	-1,768	-4,303	211,025	49,626	-45,991	224,711
3	0,7	-1,768	-4,303	211,025	-0,094	-2,222	190,809
3	0,8	-1,768	-4,303	211,025	5,118	-14,687	209,847
3	0,9	-1,768	-4,303	211,025	13,817	-11,167	211,692
4	0,3	-381,984	2,739	148,241	-305,615	-95,723	60,425
4	0,4	-381,984	2,739	148,241	-381,594	-124,797	105,959
4	0,5	-381,984	2,739	148,241	-376,295	80,659	179,554
4	0,6	-381,984	2,739	148,241	-368,549	71,66	198,092
4	0,7	-381,984	2,739	148,241	-366,097	109,557	188,849
4	0,8	-381,984	2,739	148,241	-373,698	74,37	148,241
4	0,9	-381,984	2,739	148,241	-370,47	103,651	171,024

Table B.2 - FREAK Threshold Sweep Rotation and Computational Time Results

Dataset	Threshold	True Rot		Estimated Rot		Time (s)	FPS
0	0,3	3,062	3,103	1,113	0,753	43,263	23,11444
0	0,4	3,062	3,103	0,085	0,066	44,7	22,37136
0	0,5	3,062	3,103	2,963	2,982	45,68	21,89142
0	0,6	3,062	3,103	3,062	3,095	45,67	21,89621
0	0,7	3,062	3,103	0,747	0,753	47,2	21,18644
0	0,8	3,062	3,103	2,824	3,122	49,3	20,28398
0	0,9	3,062	3,103	2,885	3,074	55,3	18,08318
1	0,3	0,506	0,022	2,911	2,978	55,98	17,86352
1	0,4	0,506	0,022	0,129	0,498	57,62	17,35509
1	0,5	0,506	0,022	0,814	0,003	58,01	17,23841
1	0,6	0,506	0,022	0,66	0,022	59,5	16,80672
1	0,7	0,506	0,022	0,615	0,022	60,3	16,58375
1	0,8	0,506	0,022	0,535	0,026	62,65	15,96169
1	0,9	0,506	0,022	0,507	0,029	79,87	12,52035
2	0,3	1,609	2,483	1,342	0,595	49,11	20,36245
2	0,4	1,609	2,483	1,493	0,29	46,47	21,51926
2	0,5	1,609	2,483	1,524	1,325	47,34	21,12379
2	0,6	1,609	2,483	1,774	3,023	48,11	20,7857
2	0,7	1,609	2,483	1,545	1,074	49,06	20,3832
2	0,8	1,609	2,483	1,582	2,026	51,06	19,5848
2	0,9	1,609	2,483	1,51	1,087	59,22	16,88619
3	0,3	0,008	0,013	1,608	1,649	44,37	22,53775
3	0,4	0,008	0,013	2,597	3,065	41,81	23,91772
3	0,5	0,008	0,013	1,376	0,316	42,88	23,3209
3	0,6	0,008	0,013	0,411	0,021	44,07	22,69117
3	0,7	0,008	0,013	0,708	0,013	45,93	21,77226
3	0,8	0,008	0,013	0,011	0,018	46,97	21,29019
3	0,9	0,008	0,013	0,017	0,012	52,2	19,15709
4	0,3	1,143	0,036	2,695	2,976	55,3	18,08318
4	0,4	1,143	0,036	1,241	0,174	52,75	18,95735
4	0,5	1,143	0,036	1,107	0,254	54,1	18,48429
4	0,6	1,143	0,036	0,74	0,044	54,49	18,35199
4	0,7	1,143	0,036	1,135	0,459	56,86	17,58706
4	0,8	1,143	0,036	1,306	0,453	57,84	17,28907
4	0,9	1,143	0,036	1,209	0,497	68,8	14,53488

Table B.3 - CREAK Threshold Sweep Translation Results

Dataset	Threshold	True (x, y, z)			Estimated (x, y, z)		
0	0,3	-184,826	-3,554	328,513	296,073	35,219	212,004
0	0,4	-184,826	-3,554	328,513	-65,097	1,149	372,755
0	0,5	-184,826	-3,554	328,513	-109,717	32,294	356,735
0	0,6	-184,826	-3,554	328,513	-161,783	51,483	306,749
0	0,7	-184,826	-3,554	328,513	148,317	8,737	83,633
0	0,8	-184,826	-3,554	328,513	-161,213	14,645	348,905
0	0,9	-184,826	-3,554	328,513	-173,371	39,835	329,326
1	0,3	463,563	-48,891	509,303	317,79	-38,116	118,862
1	0,4	463,563	-48,891	509,303	664,881	124,633	132,469
1	0,5	463,563	-48,891	509,303	508,564	-52,198	486,153
1	0,6	463,563	-48,891	509,303	557,535	-34,313	430,21
1	0,7	463,563	-48,891	509,303	483,915	-32,602	508,993
1	0,8	463,563	-48,891	509,303	422,861	-4,264	539,394
1	0,9	463,563	-48,891	509,303	443,74	-51,424	529,657
2	0,3	66,907	-8,643	232,934	-171,867	-18,398	93,273
2	0,4	66,907	-8,643	232,934	46,674	2,419	189,595
2	0,5	66,907	-8,643	232,934	42,869	-32,88	207,537
2	0,6	66,907	-8,643	232,934	76,58	-70,17	247,372
2	0,7	66,907	-8,643	232,934	81,66	9,589	228,976
2	0,8	66,907	-8,643	232,934	65,563	17,239	241,141
2	0,9	66,907	-8,643	232,934	51,477	-21,771	231,251
3	0,3	-1,768	-4,303	211,025	-198,068	25,02	-230,225
3	0,4	-1,768	-4,303	211,025	687,518	-117,802	410,435
3	0,5	-1,768	-4,303	211,025	-14,066	-120,399	209,423
3	0,6	-1,768	-4,303	211,025	663,823	-83,864	99,807
3	0,7	-1,768	-4,303	211,025	-29,555	-40,363	150,732
3	0,8	-1,768	-4,303	211,025	-51,796	-39,899	211,764
3	0,9	-1,768	-4,303	211,025	-10,148	-47,693	206,68
4	0,3	-381,984	2,739	148,241	-339,424	184,829	68,144
4	0,4	-381,984	2,739	148,241	-371,258	172,501	-109,572
4	0,5	-381,984	2,739	148,241	-387,616	-59,245	-108,011
4	0,6	-381,984	2,739	148,241	-372,293	82,999	174,751
4	0,7	-381,984	2,739	148,241	-366,269	136,775	179,244
4	0,8	-381,984	2,739	148,241	-380,043	84,082	168,825
4	0,9	-381,984	2,739	148,241	-358,486	119,601	202,64

Table B.4 - CREAK Threshold Sweep Rotation and Computational Time Results

Dataset	Threshold	True Rot		Estimated Rot		Time (s)	FPS
0	0,3	3,062	2,996	2,087	2,11	57,79	17,30403
0	0,4	3,062	2,996	2,796	2,989	58,134	17,20164
0	0,5	3,062	2,996	2,999	3,025	57,99	17,24435
0	0,6	3,062	2,996	2,864	2,996	59,216	16,88733
0	0,7	3,062	2,996	0,305	0,012	59,87	16,70286
0	0,8	3,062	2,996	3,117	3,027	62,55	15,98721
0	0,9	3,062	2,996	3,01	3,033	68,769	14,54144
1	0,3	0,506	0,022	0,962	0,187	82,365	12,14108
1	0,4	0,506	0,022	0,204	0,271	77,84	12,84687
1	0,5	0,506	0,022	0,18	0,037	79,15	12,63424
1	0,6	0,506	0,022	0,349	0,062	77,97	12,82545
1	0,7	0,506	0,022	0,529	0,024	80,38	12,44091
1	0,8	0,506	0,022	0,591	0,086	85,73	11,66453
1	0,9	0,506	0,022	0,575	0,007	97,33	10,27432
2	0,3	1,609	2,483	2,17	2,375	58,44	17,11157
2	0,4	1,609	2,483	1,703	2,603	59,41	16,83218
2	0,5	1,609	2,483	1,523	1,32	59,57	16,78697
2	0,6	1,609	2,483	1,751	2,332	61,59	16,2364
2	0,7	1,609	2,483	1,633	2,291	61,97	16,13684
2	0,8	1,609	2,483	1,485	0,165	63,91	15,647
2	0,9	1,609	2,483	1,548	1,353	72,43	13,80643
3	0,3	0,008	0,024	3,084	2,733	49,82	20,07226
3	0,4	0,008	0,024	1,135	0,827	51,608	19,37684
3	0,5	0,008	0,024	1,083	0,073	51,82	19,29757
3	0,6	0,008	0,024	1,176	0,36	53,152	18,81397
3	0,7	0,008	0,024	1,083	0,078	61,508	16,25805
3	0,8	0,008	0,024	0,044	0,013	55,68	17,95977
3	0,9	0,008	0,024	0,121	0,091	65,29	15,31628
4	0,3	1,143	0,036	0,11	1,242	67,27	14,86547
4	0,4	1,143	0,036	1,999	2,74	69,33	14,42377
4	0,5	1,143	0,036	1,909	2,934	69,29	14,4321
4	0,6	1,143	0,036	1,089	0,319	69,64	14,35956
4	0,7	1,143	0,036	0,93	0,345	71,25	14,03509
4	0,8	1,143	0,036	0,922	0,187	72,27	13,837
4	0,9	1,143	0,036	0,898	0,331	83,771	11,93731

APPENDIX C - VISUAL ODOMETRY RESULTS

Table C.1 - Visual Odometry Results for FREAK with Various Keypoints

	FREAK							
Keypoints	500	1000	2000	3000	4000	5000	10000	20000
True Position (x, y)	-184,826 -3,554 328,513	-184,826 -3,554 328,513	-184,826 -3,554 328,513	-184,826 -3,554 328,513	-184,826 -3,554 328,513	-184,826 -3,554 328,513	-184,826 -3,554 328,513	-184,826 -3,554 328,513
Odometry Position (x, y)	98,734 -275,207 88,364	251,546 161,911 -12,266	-18,828 83,459 130,605	211,033 -104,256 217,415	265,675 117,478 209,837	260,84 -58,797 367,565	21,963 -25,108 355,621	-119,608 -36,687 380,842
Error Position (x, y)	283,56 -271,653 -240,149	436,372 165,465 -340,779	165,998 87,013 -197,908	395,859 -100,702 -111,098	450,501 121,032 -118,676	445,666 -55,243 39,052	206,789 -21,554 27,108	65,218 -33,133 52,329
True Rotation	3,062 3,104	3,062 3,104	3,062 3,104	3,062 3,104	3,062 3,104	3,062 3,104	3,062 3,104	3,062 3,104
Odometry Rotation	1,959 1,757	0,148 0,649	0,866 0,292	3,041 3,007	0,875 0,676	1,478 0,599	2,821 3,063	3,088 3,106
Error Rotation	-1,103 -1,347	-2,914 -2,455	-2,196 -2,812	-0,021 -0,097	-2,187 -2,428	-1,584 -2,505	-0,241 -0,041	0,026 0,002
Time	43,645	48,541	50,095	59,472	71,336	75,341	119,361	163,822
fps	22,91213	20,60114	19,96207	16,81464	14,01817	13,27299	8,377946	6,104186
Average Error	460,2968	577,8663	272,5696	423,306	481,3355	450,7716	209,669	89,94169

Table C.2 - Visual Odometry Results for CREAK with Various Keypoints

	CREAK							
Keypoints	500	1000	2000	3000	4000	5000	10000	20000
True Position (x, y)	-184,826 -3,554 328,513	-184,826 -3,554 328,513	-184,826 -3,554 328,513	-184,826 -3,554 328,513	-184,826 -3,554 328,513	-184,826 -3,554 328,513	-184,826 -3,554 328,513	-184,826 -3,554 328,513
Odometry Position (x, y)	190,283 -80,918 114,691	251,74 -211,345 216,348	95,809 -53,983 126,537	288,639 134,783 227,015	-65,161 214,225 200,557	-21,525 87,593 218,642	-52,351 -79,579 134,679	-130,276 -83,516 196,919
Error Position (x, y)	375,109 -77,364 -213,822	436,566 -207,791 -112,165	280,635 -50,429 -201,976	473,465 138,337 -101,498	119,665 217,779 -127,956	163,301 91,147 -109,871	132,475 -76,025 -193,834	54,55 -79,962 -131,594
True Rotation	3,062 3,104	3,062 3,104	3,062 3,104	3,062 3,104	3,062 3,104	3,062 3,104	3,062 3,104	3,062 3,104
Odometry Rotation	0,525 0,49	3,03 2,359	1,554 1,414	1,5304 1,486	1,801 1,759	1,864 2,114	1,776 2,681	2,345 2,769
Error Rotation	-2,537 -2,614	-0,032 -0,745	-1,508 -1,69	-1,5316 -1,618	-1,261 -1,345	-1,198 -0,99	-1,286 -0,423	-0,717 -0,335
Time	49,441	50,439	63,167	72,433	86,278	99,347	163,086	211,445
fps	20,22613	19,8259	15,8310	13,8058	11,5904	10,0657	6,13173	4,72936
Average Error	438,6477	496,334	349,418	503,595	279,499	216,902	246,7814	163,3604

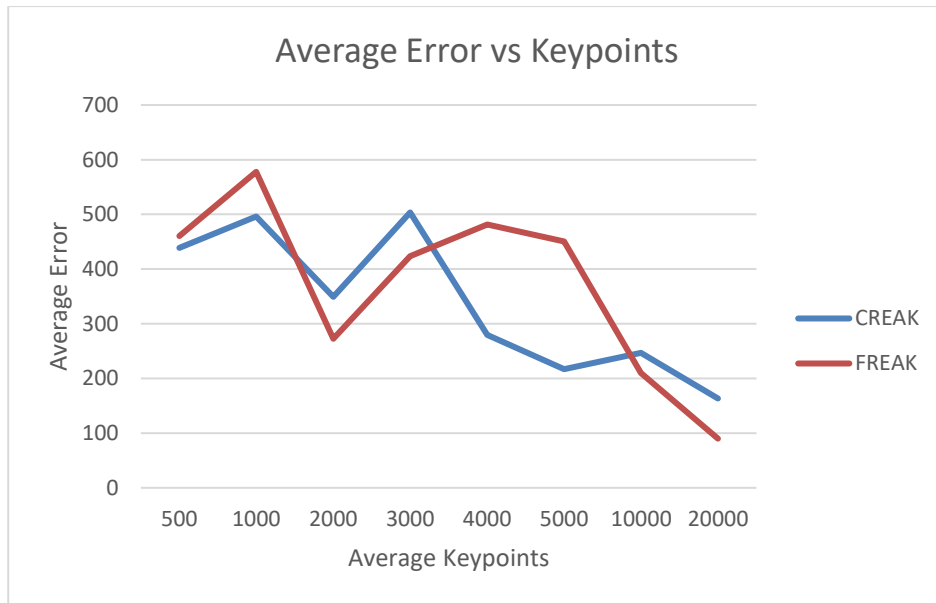


Figure C.1 - Average Error for FREAK and CREAK with Various Keypoints

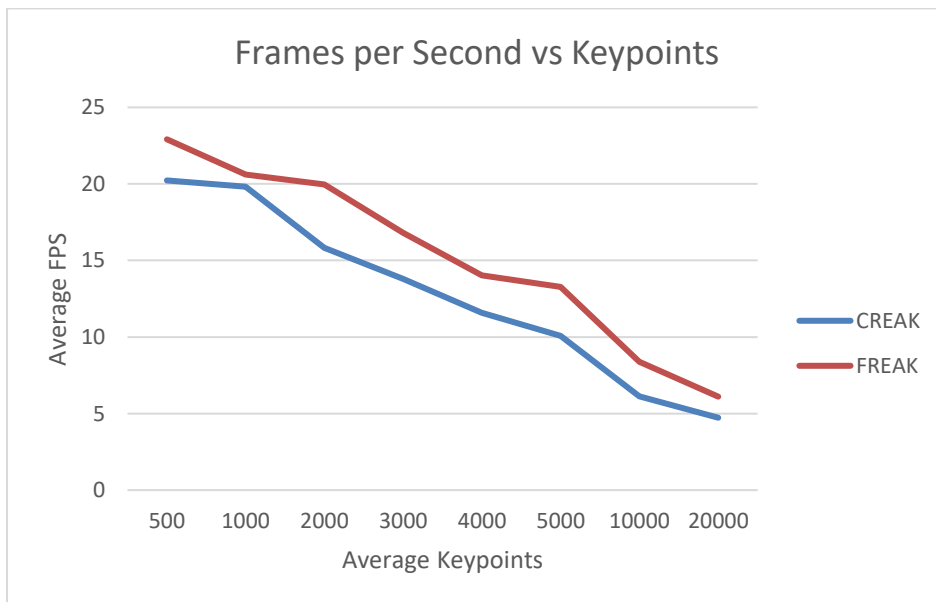


Figure C.2 - Average FPS for FREAK and CREAK with Various Keypoints