

AMERICAN UNIVERSITY OF BEIRUT

Comparative Assessment of Non-filter Based
Monocular Visual SLAM Systems

by
Georges Youssef Younes

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering
to the Department of Mechanical Engineering
of the Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
February 2016

AMERICAN UNIVERSITY OF BEIRUT

Comparative Assessment of Non-filter Based Monocular Visual SLAM Systems

by
Georges Youssef Younes

Approved by:

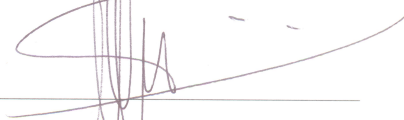
Dr. Daniel Asmar, Associate Professor
Mechanical Engineering

Advisor



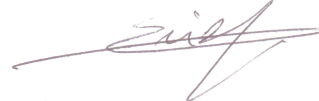
Dr. Imad Elhajj, Associate Professor
Electrical and Computer Engineering

Member of Committee



Dr. Elie Shammas, Assistant Professor
Mechanical Engineering

Member of Committee



Date of thesis defense: February 8, 2016

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Younes Georges Youssef
Last First Middle

Master's Thesis Master's Project Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

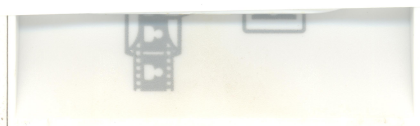
I authorize the American University of Beirut, **three years after the date of submitting my thesis, dissertation, or project**, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.



Signature

February, 12, 2016

Date



Acknowledgements

I would like to express my special appreciation and thanks to my advisor Dr. Daniel Asmar, you have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a researcher. I would also like to express my deepest appreciation to all those who provided me the possibility to complete this thesis and thank my committee members, Dr. Image Elhajj, Dr. Elie Shamma for serving as my committee members even at hardship. A special thanks to my family. Words cannot express how grateful I am to my mother and father for all of the sacrifices that you've made on my behalf. Your prayer for me was what sustained me thus far. I would also like to thank all of my friends who supported and incited me to strive towards my goal.

An Abstract of the Thesis of

Georges Youssef Younes for Master of Engineering
Major: Mechanical Engineering

Title: COMPARATIVE ASSESSMENT OF NON-FILTER BASED MONOCULAR VISUAL SLAM SYSTEMS

Monocular Visual SLAM refers to the process of determining an agent's pose using a single camera as a sensory input. Extensive research in the field for the past decade ensued a number of systems that found their ways into various applications, such as robotics and augmented reality. Although filter-based (e.g., Kalman Filter, Particle Filter) Visual SLAM systems were common at some time, non-filter based (i.e., using optimization) solutions, which are more efficient, have become the de facto methodologies for building any Visual SLAM system. The major contribution of this thesis is a comparative assessment of the state of the art in open source non-filter based monocular Visual SLAM systems, namely PTAM, SVO, DT SLAM, LSD SLAM and ORB SLAM. Detailed experiments are presented for the SLAM comparison. To motivate this comparison, we present at the beginning of the thesis a case study, of a Visual SLAM application in an outdoor scene, in which the major problems of Visual SLAM are unearthed. The second major contribution of this thesis is the development of a scaled monocular SLAM in which depth from focus is used to determine the correct scale and maintain it through a SLAM trajectory. Real experiments are also performed and the obtained results prove the viability of the proposed method.

Contents

Acknowledgements	v
Abstract	vi
List of Figures	x
List of Tables	xii
1 Introduction	1
1.1 Overview	1
1.2 Thesis purpose	2
2 Background	4
2.1 Image formation	4
2.2 Epipolar geometry	6
2.3 Structure from Motion	7
3 Case study: Visual SLAM in an AR application	10
3.1 PTAM	11
3.2 PTAMM	12
3.3 Scale estimation	13
3.4 Model anchoring	13
3.5 Miscellaneous modifications	14
3.6 Challenges and future work	14
4 A brief history	17
5 Design of Visual SLAM systems	20
5.1 Data Type	20
5.1.1 Direct methods	21
5.1.2 Indirect methods	22
5.1.3 Hybrid methods	22
5.2 Initialization	23
5.3 Data association	26

5.4	Pose estimation	29
5.5	Map generation	35
5.6	Map maintenance	40
5.7	Failure recovery	43
5.8	Loop closure	45
6	Comparative assessment of Visual SLAM systems	47
6.1	Dataset generation	47
6.1.1	Hardware setup	47
6.1.2	Hardware Configuration	48
6.1.3	Test set environment	48
6.1.4	Data processing	50
6.2	Experimental results	52
6.3	Discussion of results	53
6.3.1	PTAMM	53
6.3.2	SVO	54
6.3.3	DT SLAM	54
6.3.4	LSD SLAM	55
6.3.5	ORB SLAM	55
7	Suggested improvements	62
7.1	A. Metric scale recovery in monocular SLAM	62
7.1.1	Depth from focus	62
7.1.2	Depth from focus calibration	65
7.1.3	Depth from focus in Visual SLAM	66
7.1.4	Pose drift correction	67
7.1.5	Implementation considerations	68
7.1.6	Experiments and results	69
7.1.7	Discussion	73
7.2	A suggested visual SLAM pipeline	74
7.2.1	Data type	74
7.2.2	Initialization	75
7.2.3	Data association	75
7.2.4	Pose estimation	75
7.2.5	Map generation	76
7.2.6	Map maintenance	76
7.2.7	Failure recovery	76
7.2.8	loop closure	76
8	Conclusion	78
A	Abbreviations	79

List of Figures

2.1	Pinhole projection model.	5
2.2	digital image representation.	6
2.3	epipolar geometry.	7
2.4	Typical SfM pipeline.	8
2.5	Typical input and output of an SfM system.	9
3.1	Simplistic diagram of PTAMs pipeline	11
3.2	AR scale and anchoring issues	12
3.3	Board of highly reliable markers	14
3.4	Summary of the anchoring process.	15
5.1	Eight building blocks of a Visual SLAM system	20
5.2	Data types used by a Visual SLAM system	22
5.3	Initialization required by any Visual SLAM system	23
5.4	Data association problem	27
5.5	Pose estimation required by any Visual SLAM system	29
5.6	Map generation required by any Visual SLAM system	36
6.1	Sensor rig setup	48
6.2	AUB campus overview	49
6.3	WGS-84 Ellipsoid	49
6.4	Sample images of the generated dataset	50
6.5	reported results by PTAMM	56
6.6	LSD SLAM map after looping the dataset twice	57
6.7	LSD SLAM path aligned with INS.	57
6.8	ORB SLAM map after looping the dataset twice	58
6.9	ORB SLAM path aligned with INS.	58
6.10	effects of small baselines	59
6.11	Discontinuity and tracking failure at loop closure	59
6.12	PTAM mode of failure due to low image gradients.	60
6.13	LSD SLAM random initialization effect	60
6.14	Orb SLAM delayed initialization effects	61
7.1	Image formation process through a gauss thin lens model.	63

7.2	Focus measure profile	65
7.3	DfF calibration example	66
7.4	Conditions under which pose drift correction takes place	68
7.5	Experiment 1: System initialization	70
7.6	Reported paths after initializing using our proposed method	71
7.7	Path recorded by PTAM initialized using its default method	72
7.8	Paths recorded during the drift experiment	72
7.9	Impact of jittering and motion blur on Focus measure.	73
7.10	Impact of jittering and motion blur on Focus measure.	74

List of Tables

4.1	List of different visual SLAM systems	19
5.1	Methods used by different Visual SLAM systems	23
5.2	Initialization used by different Visual SLAM systems	26
5.3	Data association used by different Visual SLAM systems	29
5.4	Pose estimation used by different Visual SLAM systems	35
5.5	Map generation used by different Visual SLAM systems	40
5.6	Map maintenance used by different Visual SLAM systems	43
5.7	Failure recovery used by different Visual SLAM systems	45
5.8	Loop closure used by different Visual SLAM systems.	46
6.1	Summary of experimental results	53
7.1	Computational cost for initialization	69
7.2	RMSE of paths measured in experiment 3	72

Chapter 1

Introduction

1.1 Overview

Localization solutions using a single camera have been gaining considerable popularity in the past decade. Cameras are today cheap and ubiquitously found in hand-held devices such as phones and tablets. With the increase in augmented reality applications we are witnessing today, the camera is the natural sensor of choice to localize the user while projecting virtual scenes to him/her from the correct viewpoint. With their small size, cameras are the sensors of choice in localization applications where weight and power consumption are deciding factors such as for Unmanned Aerial Vehicles (UAVs). Even though there are still many challenges facing camera-based localization, it is expected that such solutions will eventually offer significant positives. Monocular Camera-based localization can be achieved via three main techniques. In the first technique, known as marker-based tracking, an artificial landmark of known dimensions is introduced into the scene. The camera localization problem then reduces to finding the camera pose with respect to the marker. Unfortunately, the requirement of having a marker always visible in the scene introduces a major limitation to this method. The second technique, known as image-based localization, the scene is processed beforehand to yield its 3D structure, scene images and corresponding camera viewpoints. The localization problem then reduces to that of matching new query images to those in the database and choosing the camera position, which corresponds to the best-matched image. In the third technique, no prior information of the scene is given; rather, map building and localization are concurrently done. This third technique is commonly referred to as Visual Simultaneous Localization and Mapping (Visual SLAM). Although image based localization and Visual SLAM are equally important, the subject of this thesis is related to the later technique.

Since the first breakthrough in 2003 with Davison's proposal of a filter-based approach to monocular Visual SLAM [1], significant improvements have been proposed to camera-based localization and mapping solutions. In 2007, Davison et al.

[2] released an open source implementation of their suggested filter-based system, also in 2007, Klein and Murray [3] introduced their groundbreaking work dubbed PTAM (Parallel Tracking and Mapping). 2014 witnessed many contributions in the Visual SLAM field as Forster et al. [4] presented SVO (Fast Semi-Direct Visual Odometry), Engel et al. [5] produced LSD SLAM (Large-Scale Direct SLAM) and Herrera et al. [6] presented DT SLAM. In 2015, Mur-Artal et al. [7] presented ORB SLAM (Oriented Fast and Rotated Binary Robust Independent Elementary Features). It is noteworthy to mention that a number of closed source systems also exist (such as RD SLAM [8], CD SLAM [9] and real-time 6Dof monocular SLAM [10]).

1.2 Thesis purpose

The purpose of this thesis is fourfold, (1) a survey on Visual SLAM systems, detailing the inner workings of the state-of-the-art in non-filter monocular-based systems beyond the information provided in their papers. (2) A comparative assessment of major open-source systems, namely PTAM, SVO, DT SLAM, LSD SLAM and ORB SLAM. Detailed experiments are presented for the SLAM comparison. To motivate this comparison, we present at the beginning of the thesis a case study, of a Visual SLAM application in an outdoor scene, in which the major problems of Visual SLAM are unearthed. (3) The development of a scaled monocular SLAM in which depth from focus is used to determine the correct scale and maintain it through a SLAM trajectory. Real experiments are also performed and the obtained results prove the viability of the proposed method. (4) An outline of a new Visual SLAM system that builds on the deductions of the comparison to address the major shortcomings of currently available systems.

This thesis is a valuable tool for any user or researcher in camera-based localization. With the many new proposed systems coming out every day, the information is daunting to the novice and one is often perplexed as to which algorithm he/she should use. Furthermore, the offered information should help researchers quickly pinpoint the shortcomings of each of the proposed techniques and accordingly help them focus their effort on alleviating these weaknesses.

The remainder of this thesis is structured as follows. Chapter II introduces preliminary theoretical computer vision knowledge. Chapter III showcases a case study of PTAM in an augmented reality application, to highlight the major shortcomings of any Visual SLAM system. Chapter IV lists a historical overview of monocular visual SLAM systems. Chapter V discusses the different components required for building a coherent visual SLAM scheme highlighting the differences between the surveyed systems. Chapter VI discusses the generation of an image dataset and details its usage to comparatively assess the performance of the surveyed systems while Chapter VII introduces our proposed solution to solving the inherent scale loss issue of monocular based systems and builds on the knowledge

presented in the context of this thesis to suggest a monocular SLAM backbone that benefits from the advantages of the surveyed systems while alleviating many of their shortcomings. Chapter VIII concludes by presenting future venues for this research.

Chapter 2

Background

To elaborate more on the visual SLAM topic, it shall prove helpful at this point to introduce some basic principles of Computer Vision techniques that form the required building blocks for the development of any visual SLAM system.

2.1 Image formation

The image formation process is comprised of a combination of two processes, a geometric transformation and a radiometric process.

Geometric model

The geometric image formation model describes the geometric mapping of a 3D scene onto a 2D image plane. For such purpose, Brunelleschi proposed, around the beginning of the fifteenth century, to approximate the geometric transformation of a 3D point to a 2D point found on the image plane by the intersection of the image plane with a ray traveling from the 3D point and through a pinhole, hence the naming pinhole projection model as depicted in Figure 2.1.

Unfortunately, for a single ray to pass through the pinhole, the latter must be of infinitesimal size, which is not the case for real cameras equipped with lenses; however, the pinhole model is mathematically convenient and provides an acceptable approximation to the real image geometric formation process. One effect of the pinhole projection model is that the end-result is an inverted representation of a 3D object onto the image plane; for convenience, a virtual image plane is considered equidistant to the pinhole such that the object representation in this plane is no longer inverted.

To fully project a 3D points to a 2D image plane, 3 coordinate frames are required; a world coordinate frame, a camera coordinate frame and a pixel coordinate frame. A 3D point $P_W = (X, Y, Z, 1)^T$ in the world's coordinate frame is mapped into the camera's coordinate frame $(X_c, Y_c, Z_c)^T$ using the homogeneous

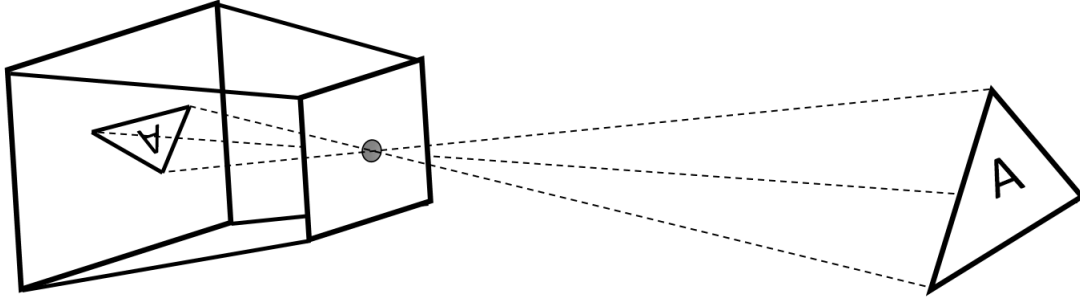


Figure 2.1: Pinhole projection model.

transformation matrix:

$$P_C = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} P_W, \quad (2.1)$$

where R is an $SO(3)$ rotation matrix and T is a (3×1) translation vector, describe the relationship between the world's coordinate frame to the camera's coordinate frame. The projected point in the camera's frame is next transformed into the pixel coordinate frame by an affine transformation matrix such that

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & -\alpha \cot \theta & x_0 \\ 0 & \frac{\beta}{\sin \theta} & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix}, \quad (2.2)$$

where $\alpha = kf$ and $\beta = lf$; f is the focal length, k and l represents the dimensions of the pixels in the sensors array and θ is the skew angle between the horizontal and vertical axis of the sensor. In theory, for orthogonal axis, θ should be 90° but due to errors in the manufacturing process of the sensor this value is slightly different. x_0 and y_0 are the coordinates in pixel units of the image center expressed in the pixel coordinate frame. The origin of the pixel coordinates is generally located at the top left corner of the image.

Radiometry formation

Light incoming from an illumination source reflects against an object, enters the imaging system and intersects the image plane. The image plane in a modern digital camera is an array of light detecting sensors. A spatial and tonal sampling of the scene takes place at the sensor level and the result is a finite, discrete pixelated representation of the scene in the form of an array of numbers. For a gray-scale image, the magnitude of the numbers found at each pixel corresponds to the intensity value of the light ray recorded at that pixel. Figure 2.2 shows the discrete array representation corresponding to a small region in the image.

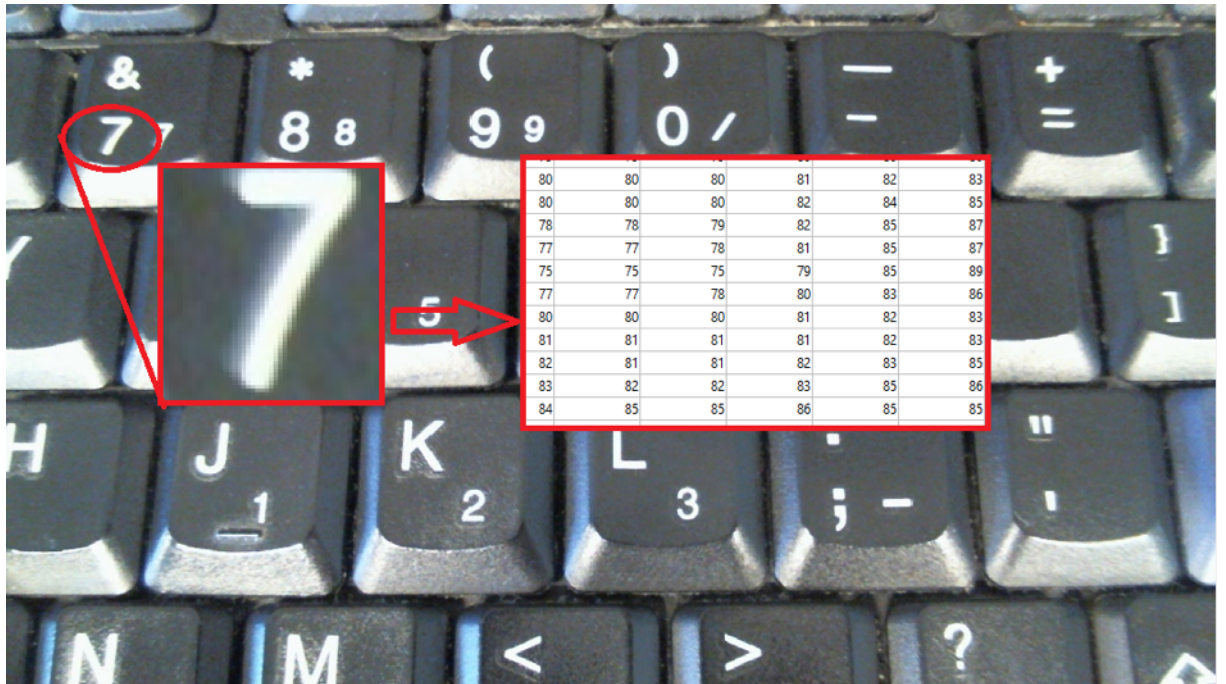


Figure 2.2: digital image representation.

Many factors contribute into the generation of the intensity values, one can list few of the major ones such as light sources (strength and direction), surface geometry, surface material and camera gain/exposure. During a Visual SLAM session, information regarding these factors is rarely available with the exception of the camera gain/exposure; hence a proper explanation of how the imaging systems operates is due.

Light reflected from an object forms what is known as the scene's radiance. Once the radiance goes through the lens and reaches the surface of the image plane it forms the image irradiance (energy arriving at a surface) and finally the response of the imaging sensor to the image irradiance results in the image brightness values. For a scene subjected to a constant light source, the radiance along a certain direction is constant, so is the image irradiance; however, the camera's response to irradiance is a function of the camera's internal systems and it can be estimated through a radiometric calibration process. For further understanding of camera's internal system, the following section describes modern camera mechanisms.

2.2 Epipolar geometry

Epipolar geometry is the projective relationship that exists between two images. It is independent of the observed scene and only depends on the camera's internal

parameters and the relative pose between the two images. To better understand epipolar geometry suppose a point X in 3D (as shown in Figure 2.3) and its projection on Image 1 is denoted by x and on Image 2 by x' . The cameras

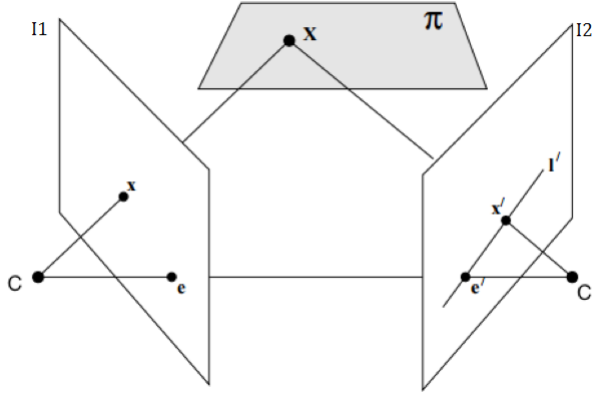


Figure 2.3: epipolar geometry.

centers (C and C'), the 3D point X and the 3D point projections x and x' on the corresponding image planes, form a plane known as the epipolar plane. Suppose now that X is not known and one is given x and seeks to find what constrains x' in the second image. Since x' must reside on the epipolar plane then its location in the second image is constrained along the line L' , that is the intersection of the second image plane with the epipolar plane. One last definition in the context of epipolar geometry is the fundamental matrix F , it is the relationship that constrains where the projection of X can occur in both Images, such that Fx forms the epipolar line L' .

2.3 Structure from Motion

Structure from motion is the process of geometrically reconstructing a scene and recovering camera motion from two or more images. SfM methods take as inputs an arbitrary number of images, taken from arbitrary poses and are able to output the 3D shape of the observed scene, in the form of point clouds, along with the relative poses of the images taken within the scene. They exploit special geometrical relationships (epipolar constraints) relating images observing the same scene to achieve their goal. Many SfM methods exist in the literature, they vary depending on the inputs at hand; some are optimized for incrementally taken images while some are optimized for randomly captured images of the scene.

The SfM pipeline summarized in Figure 2.4, starts by establishing feature correspondences across the images before estimating for each pair, that share

enough overlap, a Fundamental matrix. The Fundamental matrices are exploited to jointly recover the camera poses and the scenes structure. The algorithm loops over itself and aims to minimize the projection error of all 3D features onto all the images they appear in, by varying the entire set of cameras poses and the 3D coordinates of the structure. At each iteration, better results are produced, by using the previous ones as priors for guided feature matching and error minimizations in the estimation of the camera poses and the 3D structure.

The minimization is performed through a nonlinear minimization (such as Levenberg-Marquardt) that exploits the sparse nature of the system and is known as sparse bundle adjustment. Bundle adjustment results in an immunity to drift, as it provides a mean to compare the integrity of the camera pose to the integrity of the structure and vice versa, minimizing the errors accumulated in the estimation of both; however, the added value of drift immunity is countered by the immense increase of computational expenses. Due to inherent limitations of

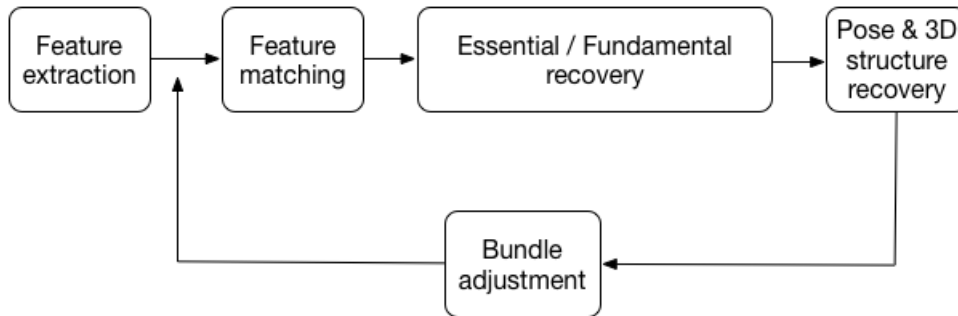


Figure 2.4: Typical SfM pipeline.

monocular visual tracking, SfM methods suffer from the loss of scale and hence the results are up to an unknown scale that is determined through other means; they produce dense point clouds (Figure 2.5) at the expense of processing time and hence they are considered offline methods and not suitable for real-time applications such as the one discussed in the upcoming chapter.

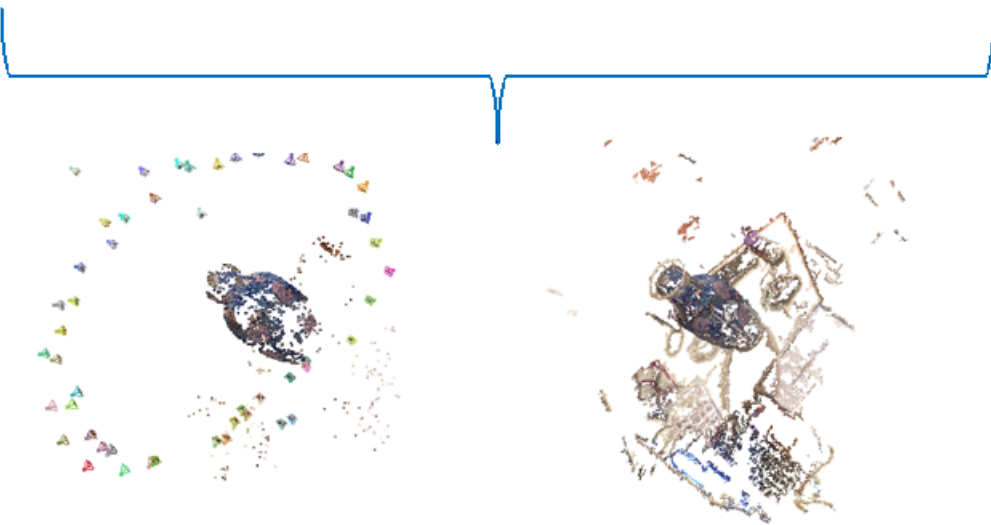
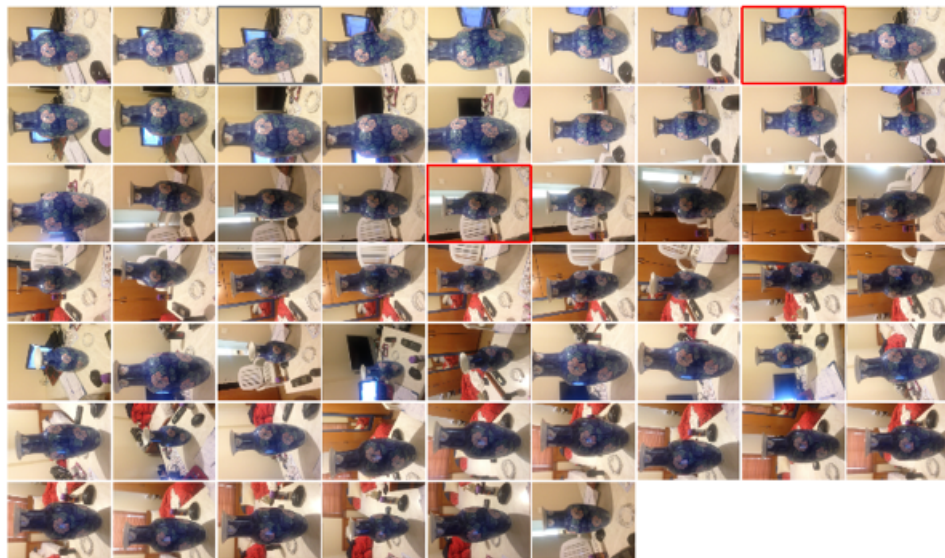


Figure 2.5: Typical input and output of an SfM system.

Chapter 3

Case study: Visual SLAM in an AR application

This chapter details a case study of a visual SLAM system integration into an augmented reality (AR) application. In contrast to virtual reality applications (VR) that works in a completely artificial, enclosed and controlled virtual environment, AR augments in real-time what the user is actually observing in a scene. To achieve a realistic augmentation, the point of view from which the user is observing the scene must be accurately estimated.

The main intuition is to use a camera to record the scene, to track the camera's pose within the scene and accordingly augment the theoretical model on top of the live video feed from the camera, before displaying it to the user. The major challenge in the above objective manifested itself in the difficulty of tracking the camera's pose in the outdoor scene without the possibility of installing an infrastructure for tracking (*e.g.*, such as a Vicon system).

The success of our augmented reality application relied heavily on an accurate user's pose estimate, hence the necessity to have a reliable motion estimation method that is capable of operating in outdoors and does not require a setup of permanent external hardware on site. The Microsoft's Kinect was considered at first, however, the Kinect would fail to operate once exposed to sunlight as its sensors would saturate with infrared readings.

The motion estimation task is then defined as the problem of estimating a camera's pose from its video feed. This problem had undertaken a significant amount of research by the computer vision community that was able, over the past two decades, to develop many solutions to it under the title of VO (visual odometry); however, visual odometry tend to drift over time and does generate a representation of the scene yielding a poor augmented reality experience, therefore our application fell under the classification of monocular SLAM (Simultaneous localization and mapping using a single camera).

3.1 PTAM

Early solutions for monocular SLAM were filter-based therefore only capable of tracking a limited number of landmarks which restricted their operations to confined spaces. PTAM was the first system to successfully separate camera tracking from scene mapping, paving the path for non-filter based solutions.

PTAM (parallel tracking and mapping) is a camera tracking system designed for augmented reality; it uses automatically extracted "Landmarks" from a video feed to estimate the ego-motion of the camera recording the feed. Its requirements are a laptop and a hand-held/head-mounted calibrated camera.

A simplistic diagram of how PTAM works is shown in Figure 3.1. A system initialization procedure is required only once at startup to kick-start the virtual map of the scene and henceforth the tracking algorithm: every incoming frame is processed to extract features, the features are matched against their precedents in the map and are used to estimate the frame's pose within the map. The estimated frame's pose is then used to refine the map and to render an augmented scene into the frame before displaying it to the user.

The virtual map in PTAM is a sparse 3D point cloud correlated with features extracted from the camera images. In this context, 2D features are a 9×9 patch of pixels extracted from a two dimensional image location with a distinctive appearance. It is noteworthy to mention that PTAM has many limitations; one

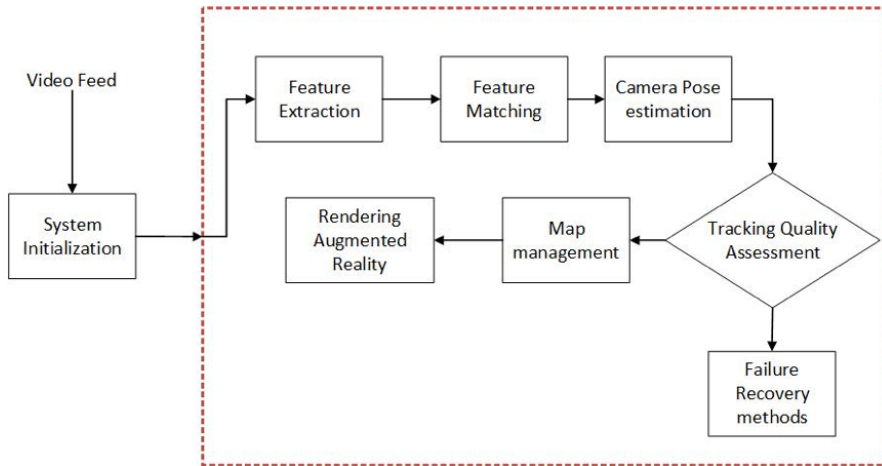


Figure 3.1: Simplistic diagram of PTAMs pipeline

drawback, of particular importance is the requirement of initialization at startup. Different initializations means different scales of maps and pose estimates (a drawback for all monocular systems). Such limitation is unacceptable in our application as the user's pose must be tracked with an SI metric scale *i.e.* in meters. A direct consequence to this limitation is the need for proper anchoring of

the augmented models in the initialization-dependent PTAM’s coordinate frame. Both issues appear in Figure 3.2

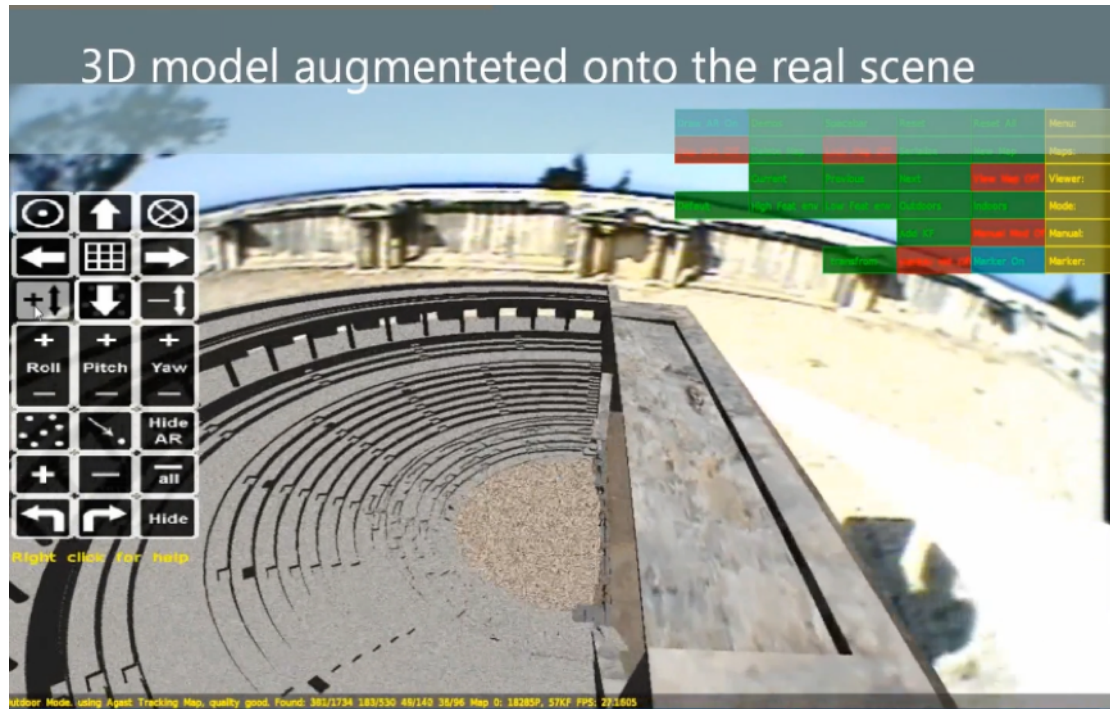


Figure 3.2: The augmented model on site before anchoring and scale are determined

3.2 PTAMM

To address the above limitations, we opted a variant of PTAM, called PTAMM [11] (parallel tracking and multiple mapping). Down to its core, PTAMM tracks motion the exact way as PTAM; however, it offers an option to serialize (save) the created map. Furthermore, by exploiting the failure recovery methods developed by PTAM, it allows the system to re-start without the need of an initialization step. Therefore, if we were to create a PTAMM compatible map of the site where the augmentation is to take place, and determine its scale and the proper anchoring of the augmented parts in it, we would be able to save the above configuration and use it to re-start the application for any user in the site without requiring them any initialization or manual anchoring procedures.

3.3 Scale estimation

To determine PTAMM’s map scale, an object of known size in the scene is sufficient. However, due to the sparse property of the map, higher perception methods are not an option, so we chose to add an artificial marker in the scene. For this, we used ARUCO [12] (a library that performs marker detection). For added robustness, the library was used to generate a board of markers; the camera’s pose is estimated, with respect to the center of the board in an SI metric scale, as the combination of the estimated camera poses with respect to each marker in the board. The generated board is shown in Figure 3.3.

Correlation between differential changes in the marker camera pose estimates and PTAMM’s camera pose estimates is used to scale the map. As the camera moves, PTAMM’s record its pose estimates in its own scale; here we define:

$$DB = \sqrt{(X_{i+1}^P - X_i^P)^2 + (Y_{i+1}^P - Y_i^P)^2 + (Z_{i+1}^P - Z_i^P)^2} \quad (3.1)$$

$$Db = \sqrt{(X_{i+1}^M - X_i^M)^2 + (Y_{i+1}^M - Y_i^M)^2 + (Z_{i+1}^M - Z_i^M)^2} \quad (3.2)$$

where P corresponds to the measurements taken in PTAM’s internal scale and M the measurements taken in the marker based coordinate frame, between 2 frames captured at time i and $i + 1$ where both, the camera tracking of PTAMM is deemed good and the marker is observed.

PTAMM’s scale is then recovered through a linear regression procedure

$$S = \min_S \sum_{i=1}^n \|Db_i - S \times DB_i\|^2, \quad (3.3)$$

that minimizes the error between a set of correlated measurements in PTAMM’s space and another in the marker’s space by varying S over a vector of multiple measurements established through a moving window of size n whenever the marker is detected in the frame.

Even though, the marker is sufficient to track the camera’s pose on its own, its presence is only used to determine the map’s scale and does not in any way contribute to the pose estimation for the augmented reality application. This is due to the major limitation of marker based tracking that requires the marker to be always fully present in the observed scene; In contrast to our application, where the marker is only required once during map creation, and can later be removed from the scene without compromising the tracking quality. Hence the marker presence is not mandatory onsite.

3.4 Model anchoring

For accurate anchoring of the augmented model into the map, we were able to exploit the markers known location in the scene to correlate between PTAMM’s

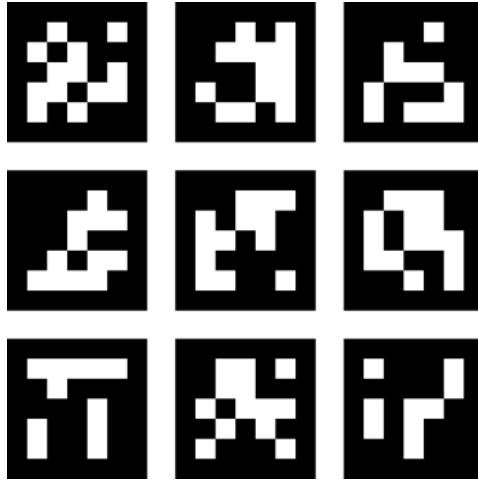


Figure 3.3: Board of HRM (highly reliable markers) generated using ARUCO library

map and the actual scene. The origin of the augmented model in PTAMM's map is set to coincide with the board of marker's origin. PTAMM's map is then aligned with the actual scene by placing the board, in the actual site, at the exact location corresponding to the origin of the augmented model. This procedure is summarized in Figure 3.4. Once the marker's pose is recorded, the marker becomes of obsolete value and hence removed.

3.5 Miscellaneous modifications

While the system behaved well in the indoor lab environment, once onsite, the algorithm failed to maintain a good tracking quality and hence internal variations and optimizations to the code were performed: A better feature extractor was added, FAST features [13] were replaced with AGAST [14] (adaptive and generic accelerated segment test) due to the fact that they are slightly faster to compute and are more repetitive in self similar structures than FAST. An outdoor mode was introduced that varies PTAMM's internal parameters to best operate in outdoors environment by reducing the number of lower level pyramid features most susceptible to lighting and view points changes.

3.6 Challenges and future work

During testing sessions of our augmented reality application, it was noted that maps created at different times of the day failed to correctly estimate the user's pose; *i.e.*, a map created at 10 am yielded poor tracking quality when invoked

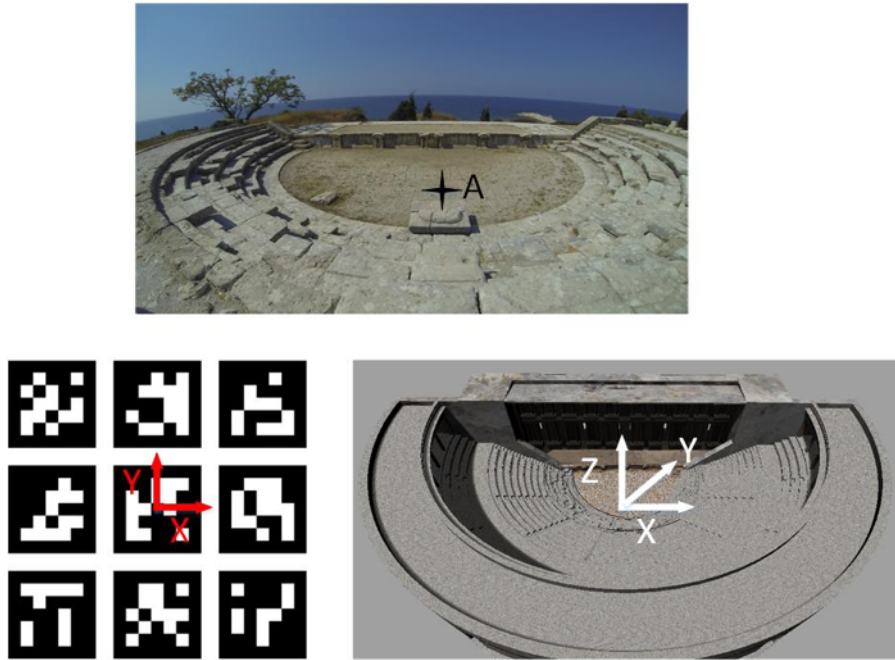


Figure 3.4: Summary of the anchoring process; the board’s center is placed at point A in the real site and its location is recorded within PTAMM’s map; this location is assumed to be the origin of the theoretical model

at 3 pm. The primary reason behind this failure is due to the changing lighting conditions throughout the day.

Another limitation manifested itself in tracking failure when the user’s POV (point of view) towards the scene varied drastically, in terms of translation and rotation between the POV of the keyframes that generated the landmarks and the currently observed POV of the scene. The primary reason behind this mode of failure is the inability of PTAM to correctly warp the landmarks generated by the distant keyframes to be able to establish reliable data associations for tracking to succeed.

While the above limitations are still challenging scenarios for state of the art monocular SLAM systems in outdoors environment, we opted to simply address them by generating multiple maps of the scene, dividing the scene into different sub-maps and have each sub-map recorded at different times of the day with 2 hours separating each session. Unfortunately, due to the vast amount of memory required to generate such database of maps, and to the fact that local variations in the lighting conditions, *i.e.*, a cloud blocking the sun or the temporal changes of the sun’s position in the sky throughout different days, rendered our solution obsolete and hence the need for an efficient lighting and point of view changes handling mechanism.

While visual SLAM systems constitutes an attractive solution to augmented reality applications, it has become evident throughout this chapter that they lack in many aspects: many milestones stand between the current implementations and a user friendly adaptation of Visual SLAM that may be used for generic AR applications especially in outdoors environment.

In an effort to unearth the origins of these shortcomings, the next chapter thoroughly investigate the different components of visual SLAM system, detailing the state of the art open source implementations.

Chapter 4

A brief history

Visual SLAM has been getting an increasing amount of interest from the scientific community in the past decade, mainly due to its importance in robotics and augmented reality applications. For such reasons, many algorithms have been developed and either released as an open-source or closed-source implementation.

Vision SLAM solutions are either filter- or non-filter based. In 2007, [2]] presented the first monocular-based SLAM solution, which was named MonoSLAM and was built around the framework of an Extended Kalman Filter. Although successful, its application in real-time was limited to very small scenes with no more than dozens of tracked features. The reason is that the computational expenses and data inference in filter based methods are high. In fact, [15]] later proved that non-filter based methods outperform filter based methods. In this paper filter based methods will not be covered in this thesis.

In 2007, Parallel Tracking and Mapping (PTAM) was released, and since then many variations and modifications of it have been proposed in [11] and [16]. PTAM was the first algorithm to successfully separate tracking and mapping into two parallel threads that run simultaneously and share information whenever necessary. This separation made the adaptation of off-line SfM methods possible within PTAM in a real-time performance. Its ideas were revolutionary in the monocular visual SLAM methods and the notion of separation between tracking and mapping became the standard backbone of almost all visual SLAM algorithms thenceforth.

In 2013, RD SLAM [8], short for Robust Monocular SLAM in Dynamic Environments, was released as a closed source algorithm, with the aim to handle occlusions and slowly varying, dynamic scenes. Alike PTAM, RD SLAM divides the tracking and mapping into two parallel threads.

In 2014, SVO [4] (semi-direct visual odometry) was published as an open-source implementation. It is a hybrid system that employs both direct and indirect methods in its proposed solution for solving the Visual SLAM task. Unlike PTAM, SVO requires a high frame rate camera. SVO was designed with the concern of operating on high end platforms as well as computationally limited ones

such as on-board hardware of a generic MAV. To achieve such resilience, SVO comes with 2 configurations, one optimized for speed and the other is optimized for accuracy.

Also in 2014, LSD SLAM [5] (Large Scale Direct monocular SLAM) was released as an open source adaptation of the visual odometry method proposed in [17]. LSD SLAM employs an efficient probabilistic direct approach to estimate semi-dense maps to be used with an image alignment scheme to solve the SLAM task. In contrast to other methods that uses bundle adjustment, LSD SLAM employs a pose graph optimization over $Sim(3)$ as in [18] which explicitly represents the scale in the system, allowing for scale drift correction and loop closure detection in real-time. A modified version of LSD SLAM was later released running on a mobile platform. LSD SLAM employs 3 parallel threads after initialization takes place: tracking, depth map estimation and map optimization.

In late 2014, short for Deferred Triangulation SLAM, DT SLAM [6] was released as an indirect method. Similar to other algorithms, it applies parallel threading to divide the visual SLAM task into 3 parallel threads: tracking, mapping and bundle adjustment. One of the main contributions in DT SLAM is its ability to estimate the camera pose from 2D and 3D features in a unified framework and suggests a bundle adjustment method that incorporates both types of features. This gives DT SLAM robustness against pure rotation movements. Another characteristic of the system, is its ability to handle multiple maps with undefined scales and merge them together once enough 3D matches are established. One important characteristic of DT SLAM is that no initialization procedure is explicitly required; rather, its embedded in the tracking thread, and is able to perform multiple initializations whenever the system is lost. Since initialization takes place automatically whenever tracking failure occurs, data can still be collected and camera tracking functions normally although at a different scale. This ability to re-initialize local sub-maps reduces the need of re-localization procedures; once enough correspondences between keyframes residing in separate sub-maps are found, the sub-maps are fused into a single map with a uniform scale throughout.

In 2015, named after the feature descriptor ORB (Oriented Fast and Rotated Binary Robust Independent Elementary Features) [19], ORB SLAM [7] was released as an indirect method for solving the visual SLAM task. Following the basic idea of PTAM, ORB SLAM also divide the problem into parallel threads, one for tracking, one for mapping, and a third for map optimization. The main contributions of ORB SLAM are the usage of ORB features in real-time, a model based initialization as suggested by Torr et al. [20], re-localization with invariance to viewpoint changes, a place recognition module using bags of words to detect loops, covisibility and Essential graphs optimization. . In Chapter V, we will further delve into the details of each of the most important Visual SLAM systems.

Table 4.1 lists the different Visual SLAM systems released to this date.

Table 4.1: List of different visual SLAM systems

Year	Name	Method	Type	Reference
2003	Real-time simultaneous localisation and mapping with a single camera	filter	indirect	[1]
2004	Simultaneous localization and mapping using multiple view feature descriptors	filter	indirect	[21]
2004	Real-Time 3D SLAM with Wide-Angle Vision	filter	indirect	[22]
2005	Real-Time Camera Tracking Using a Particle Filter	filter	indirect	[23]
2006	Scalable Monocular SLAM	filter	indirect	[24]
2007	MonoSLAM	filter	indirect	[2]
2007	Parallel Tracking and Mapping (PTAM)	non-filter	indirect	[3]
2007	Monocular SLAM as a Graph of Coalesced Observations	filter	indirect	[25]
2007	Mapping Large Loops with a Single Hand-Held Camera	filter	indirect	[26]
2007	Dimensionless Monocular SLAM	filter	indirect	[27]
2008	FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping	filter	indirect	[28]
2008	An Efficient Direct Approach to Visual SLAM	non-filter	direct	[29]
2009	Towards a robust visual SLAM approach: Addressing the challenge of life-long operation	filter	indirect	[30]
2009	Use a Single Camera for Simultaneous Localization And Mapping with Mobile Object Tracking in dynamic environments	filter	indirect	[31]
2010	On Combining Visual SLAM and Visual Odometry	filter	indirect	[32]
2010	Scale Drift-Aware Large Scale Monocular SLAM	non-filter	indirect	[33]
2011	Dense Tracking and Mapping (DTAM)	non-filter	direct	[34]
2011	Omnidirectional dense large-scale mapping and navigation based on meaningful triangulation	non-filter	direct	[35]
2012	CD SLAM - Continuous localization and mapping in a dynamic world	non-filter	indirect	[9]
2013	Robust monocular SLAM in dynamic environments (RD SLAM)	non-filter	indirect	[8]
2014	Real-time camera tracking using a particle filter combined with unscented Kalman filters	filter	indirect	[36]
2014	Semi-direct Visual Odometry (SVO)	non-filter	hybrid	[4]
2014	Large Scale Direct monocular SLAM (LSD SLAM)	non-filter	direct	[5]
2014	Deferred Triangulation SLAM (DT SLAM)	non-filter	indirect	[6]
2015	Robust large scale monocular visual SLAM	non-filter	indirect	[37]
2015	ORB SLAM	non-filter	indirect	[7]
2015	Dense Piecewise Parallel Tracking and Mapping (DPPTAM)	non-filter	direct	[38]

Chapter 5

Design of Visual SLAM systems

A generic non-filter Visual SLAM system tend to eight main issues (Fig. 5.1); namely (1) input data, (2) initialization, (3) data Association, (4) pose estimation, (5) map generation, (6) map maintenance, (7) failure recovery, and (8) loop closure. In the following sections we will detail each of these issues and critically assess how each Visual SLAM implementation addressed them.

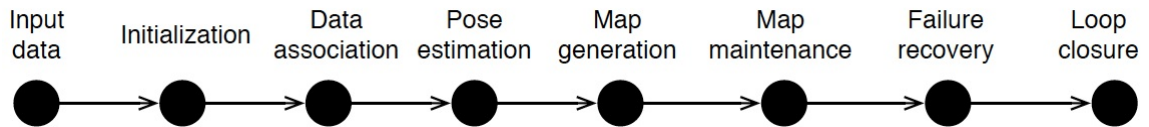


Figure 5.1: Eight building blocks of a Visual SLAM system

5.1 Data Type

Vision SLAM methods are categorized as being either direct, indirect, or hybrid. Direct methods exploit the information available at every pixel in the image (brightness values) to estimate the required parameters that fully describe the camera pose.

Indirect methods were introduced to reduce the computational complexity of processing each pixel. They process the image first to extract salient image locations known as features. A descriptor to each feature is then computed to uniquely identify the same feature in other images. Hence, the size of the processed information is reduced to include only the salient locations instead of all pixel values. Indirect methods are basically a spin-off of Structure from Motion (SfM) [39] with a goal to obtain an on-line performance, in contrast to SfM which is an off-line method. Figure 5.2 Highlights the difference between direct and indirect methods and Table 5.1 summarize the data type of the Visual SLAM systems covered in this Thesis.

5.1.1 Direct methods

The basic underlying principle for all direct methods is known as the brightness consistency constraint and is best described as:

$$J(x, y) = I(x + u(x, y) + v(x, y)), \quad (5.1)$$

where x and y are pixel coordinates; u and v denotes displacement functions of the pixel (x, y) between two images I and J of the same scene. Every pixel in the image provides one brightness constraint however it adds two unknowns (u and v) and hence the system becomes underdetermined with n equations and $2n$ unknowns (where n is the number of pixels in the image). To render eq. 5.1 solvable, Lucas and Kanade [40] suggested, in what they refer to as Forward Additive Image Alignment (FAIA), to replace all the individual pixel displacements u and v by a single general motion model, in which the number of parameters is dependent on the implied type of motion. For example, for a general 3D motion estimation, the number of required parameters are six according to the following equation:

$$W(x, y, P) = \begin{bmatrix} 1 + p_1 & p_3 & p_5 \\ p_2 & 1 + p_4 & p_6 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \quad (5.2)$$

where p_1, \dots, p_6 define the parameters of the transformation. FAIA then proceeds by iteratively minimizing the squared difference between a template and input image by changing the transformation parameters. Since then, to reduce computational complexity, other variants of the FAIA were suggested such as FCIA (Forward Compositional Image Alignment), ICIA (Inverse Compositional Image Alignment) and IAIA (Inverse Additive Image Alignment) [41]. Further discussion regarding these methods is outside the scope of this work; however, it is noteworthy to mention that they all allow the recovery of inter-frame camera motion and are referred to as visual odometry. While VO estimates the camera motion between consecutive frames only, Visual SLAM further builds a map of the environment and uses it to optimize the cameras pose estimates. As a result, VO methods tend to suffer from drift. Direct Visual SLAM methods such as LSD SLAM, employ modified versions of direct methods for visual odometry along with a map representation that is used to optimize the camera pose estimates.

Direct methods, exploit all information available in the image and are therefore more robust than indirect methods in regions with poor texture. However, calculation of the photometric error at every pixel is computationally intensive and real-time application requires heavily parallelized implementations. Direct methods are also susceptible to failure when scene illumination changes occur as the minimization of the photometric error between two frames relies on the underlying assumption of the brightness consistency constraint eq. 5.1, that corresponding pixel values remain unchanged in both frames.

5.1.2 Indirect methods

Indirect methods rely on features for matching. On one hand, feature extractors are expected to be somewhat invariant to viewpoint changes (translation, rotation and scale), and somewhat immune to illumination changes. On the other hand, it is desirable for feature extractors to be computationally efficient and fast. Unfortunately, such objectives are hard to achieve at the same time and a tradeoff between computational speed and feature quality is required. The computer vision community had developed over decades of research many different feature extractors, each exhibiting varying performances in terms of rotation invariance, scale invariance, and speed. The selection of an appropriate feature detector depends on the target platform to be used, the environment in which the visual SLAM algorithm is expected to operate in as well as the expected frame rate of the Visual SLAM algorithm. Further information regarding feature extractors is outside the scope of this work, but the reader can refer to [42] for the most recent survey on the matter.

From the list of systems that are covered in this thesis, PTAM and DTSLAM use FAST feature extractor as described in [13], while RD SLAM employ a GPU accelerated SIFT [43], and ORB SLAM uses ORB features [19].

5.1.3 Hybrid methods

Different from the direct and indirect methods, systems such as SVO are considered hybrids, which uses a combination of direct methods to establish feature correspondences and indirect methods to refine the camera pose estimates.

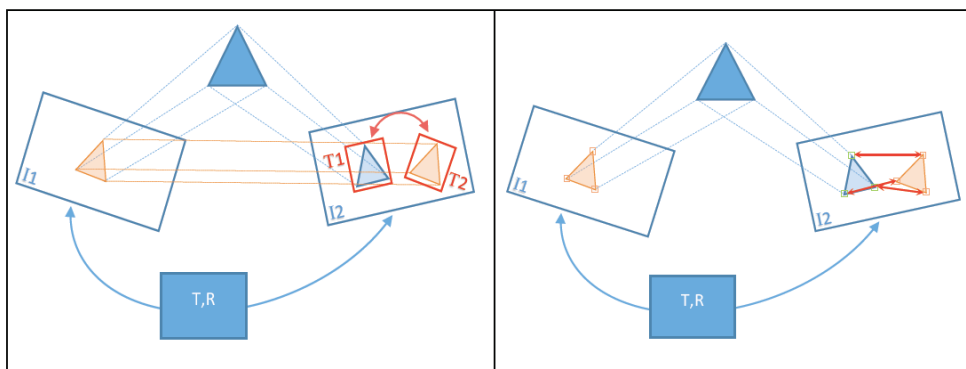


Figure 5.2: Data types used by a Visual SLAM system; (left) direct methods using all information of the triangle to match to a query.

Table 5.1: Method used by different Visual SLAM systems. Abbreviations used: indirect (i), direct (d), and hybrid (h)

	PTAM SVO		RD SLAM	DT SLAM	LSD SLAM	ORB SLAM	CD SLAM
Method	i	h	i	i	d	i	

5.2 Initialization

Figure 5.3 represents the initialization required in any Visual SLAM system. Monocular Visual SLAM systems require an initialization phase during which a map of 3D landmarks is generated. To do so, the same scene must be observed through at least two viewpoints which are separated by a significant baseline. Different solutions were proposed by different Visual SLAM systems. Table 5.2 lists the initialization solutions employed by each SLAM system.

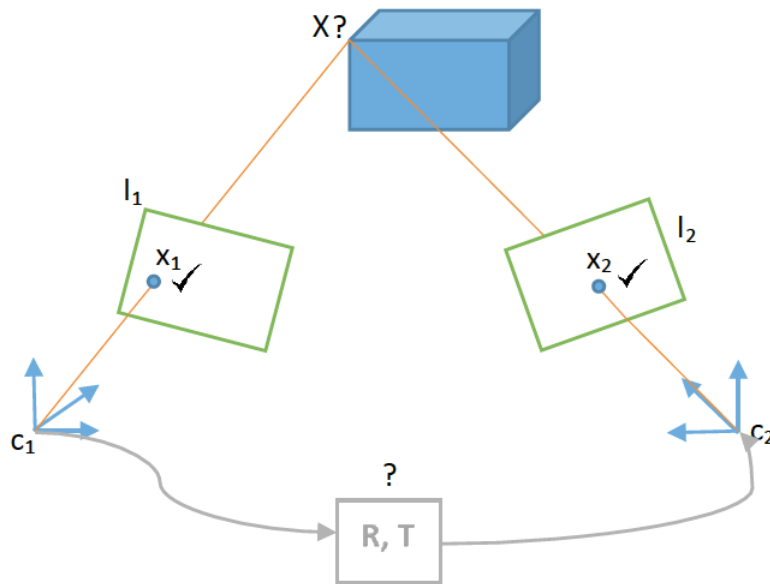


Figure 5.3: Initialization required by any Visual SLAM system

MonoSLAM initialization In early Visual SLAM systems such as in MonoSLAM [2], system initialization required the camera to be placed at a known distance from a planar scene composed of 4 corners of a square, and SLAM was initialized with the distance keyed in by the operator.

PTAM initialization To lessen the obligation of a user’s manual input of depth, PTAM’s [3] initial release suggested the usage of the five-point algorithm [44] to estimate and decompose a Fundamental matrix into an assumed to be non-planar initial scene. PTAM initialization was later changed to the usage of a Homography [45] here the scene is assumed to be composed of 2D planes. PTAM’s initialization requires the user input twice to capture the first two keyframes in the map; furthermore, it requires the user to perform, in between the first and the second keyframe, a slow, smooth and relatively significant translational motion parallel to the observed scene. FAST Features [13] extracted from the first keyframe are tracked, within a search range surrounding their position in the previous frame, in each incoming frame until the user flags the insertion of the second keyframe.

As the matching procedure takes place through the ZMSSD (zero-mean sum of squared differences) [46] metric without warping the features, establishing correct matches is susceptible to both motion blur and significant appearance changes of the features caused by camera rotations; hence the strict requirements on the user’s motion during the initialization.

To ensure minimum false matches, the features are searched for twice, once from the current frame to the previous frame and a second time in the opposite direction. If the matches in both directions are not coherent, the feature is discarded.

Finally, since PTAM’s initialization employs a Homography estimation method, the observed scene during the initialization is assumed to be planar. Once the second keyframe is successfully incorporated into the map, a MLESAC [47] loop uses the established matches to generate a Homography relating both keyframes and uses inliers to refine it before decomposing it as described in [45] into 8 possible solutions. The correct pair of camera poses is chosen such that all triangulated 3D points do not generate unreal configurations (negative depths in both frames).

The generated initial map is then scaled by assuming the magnitude of the found translation to be 0.1 units before a structure only BA (optimize only the 3D poses of the landmarks) step takes place. For augmented reality purposes, a plane is fitted to the reconstructed scene and the mean of the landmarks is selected to serve as the world coordinate frame while the positive z-direction is chosen such as the camera poses reside along its positive side.

PTAM’s initialization procedure is brittle and remains a tricky procedure to perform, especially for inexperienced users. Furthermore, it is subject to degeneracies when the planarity of the initial scenes assumption is violated or when the users motion is an appropriate, crashing the system, without means of detecting such degeneracies.

SVO initialization Similarly, Forster et al. [4] adopted in SVO, a Homography for initialization, however, SVO requires no user input and the algorithm acquires the first frame as a keyframe at startup; it extracts FAST features and tracks them with an implementation of KLT [48] (variant of direct methods) across incoming frames. To avoid a second user input, SVO monitors the median of the baseline of the features tracked between the first keyframe and the current frame; whenever this value reaches a certain threshold, the algorithm assumes enough parallax have been achieved and signals the Homography estimation to start in a RANSAC [49] scheme. The Homography is then decomposed; the correct camera poses are then selected and the landmarks corresponding to inlier matches are triangulated and used to estimate an initial scene depth. Bundle Adjustment takes place for the two frames and all their associated landmarks before the second frame is considered as the second keyframe and passed to the map management thread.

As PTAM, the initialization of SVO requires the same type of motion and is affected by sudden movement and non-planar scenes; in an attempt to automate the initial keyframe pair selection, SVO monitors the median of the baseline between features. While this method is generally successful, it could fail against degenerate cases with no means for the system of detecting such failures.

RD SLAM initialization In RD SLAM an Essential matrix decomposition is chosen [50] as their initializing procedure.

DT SLAM initialization DT SLAM did not have an explicit initialization phase; rather, it was integrated within their tracking module as an Essential matrix estimation method.

LSD SLAM initialization With the exception of MonoSLAM, all the suggested methods described above suffer from degeneracies when subjected to certain scenes; namely under low-parallax movements of the camera or when the scenes structure assumption for the corresponding method (Fundamental matrix assumption for general non-planar scenes or the Homography assumption of planar scenes) is violated. To address this issue, Engel et al. [5] suggested in LSD SLAM, a randomly initialized scenes depth from the first viewpoint that is later refined through measurements across subsequent frames. LSD SLAM uses a direct initialization method that does not require two view geometry. Instead of tracking features across two frames as the other systems do, LSD SLAM initialization procedure takes place on a single frame; pixels of interest (*i.e.*, image locations that have high intensity gradients) are initialized by default into the system with a random depth distribution and a large variance. Tracking starts directly as image alignment takes place between the first initialized keyframe and proceeding frames. Using the incoming frames, the depth measurements of the

initialized features are refined using a filter based scheme until convergence. This method does not suffer from the degeneracies of two view geometry methods (Homography and Fundamental matrix estimations), however depth estimation requires a relatively large number of processed frames before convergence takes place, resulting in an intermediate tracking phase where the generated map is not reliable.

ORB SLAM initialization To deal with the limitations arising from all the above methods, Mur-Artal et al. [7] suggested to compute in parallel, both a Fundamental matrix and a Homography in a RANSAC scheme, penalizing each model according to the symmetric transfer error of each [39], to select the appropriate model to be used. Once the best model is selected, appropriate decomposition takes place and both the scene structure and the camera poses are recovered before a bundle adjustment step optimizes the map. If the chosen model yields poor tracking quality/few feature correspondences in the upcoming frame, the initialization is quickly discarded by the system and it restarts with a different pair of frames. It is noteworthy to mention that the relationship between image coordinates and corresponding 3D point coordinates, in all the listed initialization methods aside that of monoSLAM, can only be determined up to an unknown scale.

Table 5.2: Initialization used by different Visual SLAM systems. Abbreviations used: homography decomposition (h.d.), Essential decomposition (e.d.), random depth initialization (r.d.), planar (p), non-planar (n.p.), no assumption (n.a.)

	PTAM SVO		RD SLAM	DT SLAM	LSD SLAM	ORB SLAM
Initialization	h.d.	h.d.	e.d.	e.d.	r.d.	h.d.+e.d.
Initial scene assumption	p	p	n.p.	n.p.	n.a.	n.a.

5.3 Data association

To be able to estimate the camera pose of an initialized system (given a map), data association between the current frame and the map is required. This step is inherent in systems that employ direct methods and hence data association for direct methods will be discussed with the corresponding camera pose estimation modules. On the other hand, indirect visual SLAM methods requires explicit feature matching to establish correspondences between the current frame and the map. Figure 5.4 represent the data association problem and Table. 5.3 summarizes the data association methods employed by different Visual SLAM systems.

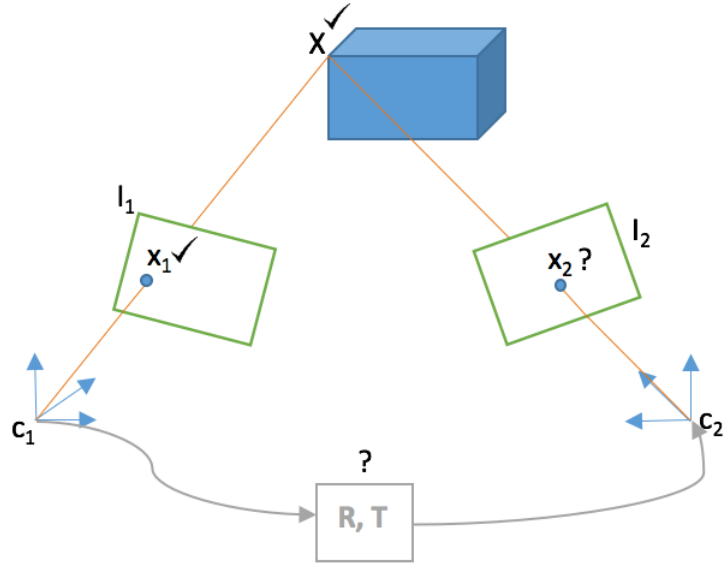


Figure 5.4: Data association problem

PTAM data association After a successful initialization of PTAM, a 4 level pyramid representation of every incoming frame is generated (e.g. level 1: 640x480, level 2: 320x240). The pyramid levels are used so that features gain robustness against scale changes and to decrease the convergence radius of the pose estimation module as shall be described later. FAST features are extracted at each level and a Shi-Tomasi score [51] for each feature is estimated; features having a Shi-Tomasi score below a threshold are removed before non-maximum suppression takes place in order to ensure high saliency of the extracted features and limit their numbers in order to remain computationally tractable. Different pyramid levels have different thresholds for Shi-Tomasi score selection and non-maximum suppression giving control over the strength and the number of features to be tracked across the pyramid levels. Reducing the thresholds for high pyramid levels gives the algorithm robustness against motion blur and rapid camera motions, while reducing the thresholds for low pyramid levels gives the tracker precision over the estimated camera pose. However, reducing the thresholds too much may lead to the addition of outliers into the system (non saliency of the features) and increased computational requirement for the increased number of features.

Different feature types require different data association procedures. In the case of PTAM, FAST features are matched using a sum of squared difference (SSD) metric over a small area of pixels surrounding the feature location and an area in the other image where the feature is suspected to be at. If the score is below a certain threshold then a match is established; however, as the two frames observing the 3D feature exhibit a significant change in viewpoint, feature match-

ing using only the above metric is expected to return poor results. Hence, further processing is required, where the feature is warped using an affine transformation that predicts how the feature would appear in the second view before attempting to match it. To gain robustness against illumination changes, the SSD metric is replaced with Zero-Mean SSD score [46].

SVO data association In a similar scheme to PTAM, SVO generates a 5 levels pyramid representation of the incoming frame; However, data association is first established through an iterative direct image alignment scheme starting from the highest pyramid level up till the third level. Preliminary data association from the previous step is then used as a prior to a FAST feature matching procedure similar to PTAM’s warping methodology with a Zero-Mean SSD score.

DT SLAM data association Relying also on FAST features, DT SLAM employs the same mechanism as PTAM to establish feature matches between frames.

LSD SLAM data association LSD SLAM does not employ an explicit data association procedure; it is inherent from the image alignment procedure employed in its camera pose estimation module.

ORB SLAM data association Also similar to PTAM’s pyramidal scheme, ORB SLAM extracts FAST corners throughout 8 pyramid levels. To ensure a homogeneous distribution along the entire image, each pyramid level is divided into cells and the parameters of the FAST detector are fine-tuned on-line to ensure a minimum of 5 corners are extracted per cell. A 256-bit ORB descriptor is then computed for each extracted feature. The higher order feature descriptor, ORB is used to establish correspondences between features. A match is established when a distance function, between the queried descriptor and its corresponding candidate in the map, scores below a certain threshold. To maintain real-time performance of such matching method, a special implementation is required: ORB SLAM discretizes and stores the descriptors into bags of words known as visual vocabulary [52]. Bags of words are used to speed up image and feature matching as the matching process is then constrained between those features that belong to the same node in the vocabulary tree.

RD SLAM data association Similar to ORB SLAM, RD SLAM employs a high order feature descriptor to establish feature matches. In contrast to other algorithms that extracts computationally cheap features using CPUs, RD SLAM employs a heavily parallelized GPU accelerated SIFT extractor. SIFT descriptors are then stored in a KD-Tree [53] that accelerates feature matching based on the nearest neighbor of the queried feature in the tree.

Table 5.3: Data association used by different Visual SLAM systems. Abbreviations used: local patch of pixels (l.p.p.)

	PTAM SVO		RD SLAM	DT SLAM	LSD SLAM	ORB SLAM
Feature type	FAST	FAST	SIFT	FAST	None	FAST
Feature descriptor	l.p.p.	l.p.p.	SIFT	l.p.p.	l.p.p.	ORB

5.4 Pose estimation

Data association for a newly acquired frame is required to estimate its pose; however, as establishing data associations blindly is computationally expensive, most Visual SLAM systems use a prior to the frames pose, which guides and limits the amount of work required for data association. Estimating this prior is generally the first task of the pose estimation module. Figure 5.5 represent the pose estimation problem and Table. 5.4 summarizes the pose estimation methods used by different Visual SLAM systems. PTAM, RD SLAM, DT SLAM and ORB

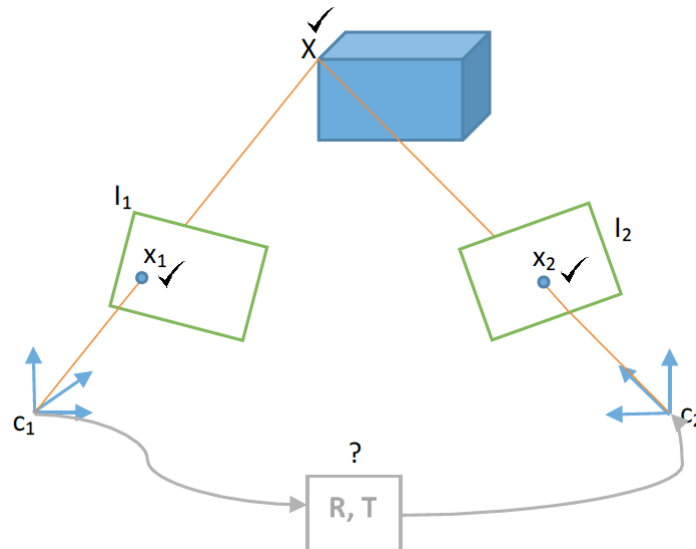


Figure 5.5: Pose estimation required by any Visual SLAM system

SLAM employ a *constant velocity* motion model that assume a smooth camera motion and uses the pose changes across the two previously tracked frames to estimate the prior of the current frame. Unfortunately, such model is prone to failure when sudden change of direction of the cameras motion occur. LSD SLAM and SVO assumes no significant change in the camera pose between consecutive

frames (such as the case in high frame rate cameras) and hence they assign the prior pose of the current frame to be the same as the previously tracked one. The prior frame pose is used to guide the data association procedure in several ways. It helps determine a potentially visible set of features from the map in the current frame, reducing computational expenses of blindly projecting the entire map; further, it helps establish an estimated feature location in the current frame, so that feature matching takes place in small search regions, instead of across the entire image. Finally, it serves as a starting point for the minimization procedure that refines the camera pose.

While direct methods estimate the camera pose by minimizing the photometric error between the current frame and the previous one over the prior pose as described before, indirect methods estimate the camera pose, by minimizing the re-projection error of landmarks from the map over the frames prior pose. The re-projection error is formulated as the distance in pixels between a projected 3D landmark onto the frame using the prior pose and its found 2-D position in the image through data association. To gain robustness against outliers (wrongly associated features), the minimization takes place over an objective function that penalizes features with large re-projection errors.

PTAM pose estimation PTAM represent the camera pose as an $SE(3)$ transformation [54] that can be minimally represented by 6 parameters. The mapping from the full $SE(3)$ transform to its minimal representation $S\xi(3)$ and vice versa can be done through logarithmic and exponential mapping in Lie algebra. The minimally represented $S\xi(3)$ transform is of great importance as it reduces the number of parameters to optimize from 12 to 6 leading to significant speedups in the optimization process.

The tracking procedure first starts by estimating a prior to the frames pose using the constant velocity motion model. The prior is then refined, using a Small Blurry Image (SBI) representation of the frame, by employing an *Efficient Second Order minimization* as described in [55]. The velocity of the prior is defined as the change between the current estimate of the pose and the previous camera pose. If the velocity is high, PTAM anticipates a fast motion is taking place and hence the presence of motion blur; to counter failure from motion blur, PTAM restricts tracking to take place only at the highest pyramid levels (most resilient to motion blur) in what is known as a coarse tracking stage only; otherwise the coarse tracking stage is followed by a fine tracking stage. However, when the tracker is stationary, the coarse stage may lead to jittering of the cameras pose and hence is turned off.

2D-3D feature-landmark correspondences, to be used in the previous step, are established by a search range in surrounding regions of their projected location in the image plane. Feature matching in PTAM is separated into two steps: coarse matching and sub-pixel refinement. For coarse level matching, a window

surrounding the features location, in the first keyframe it was observed in, is used to generate a warped representation that takes into account the viewpoint changes between the current camera pose estimate and the landmarks originating keyframe pose, however, choosing the features first appearance in their original keyframe to generate the warped matrix constitutes a weakness in PTAM’s feature matching method as largely deformed patches will fail to correctly match. If sub-pixel refinement is required, an inverse compositional matching template is generated and 8 iterations of patch alignments takes place; Established correspondences are used to formulate the re-projection error. The minimally represented initial camera pose prior is then refined by minimizing an objective function of the re-projection error that down-weights observations with large error. The objective function used is the tukey-biweight objective function as described in [56].

At this point, coarse tracking is complete and the estimated pose is passed to the map making thread for further processing; If fine tracking is to take place, features from the lowest pyramid levels are selected and a similar procedure to the above is repeated.

The tracking threads also monitors the ratio of successfully matched features in the frame against the total number of attempted feature matches to determine the tracking quality. If the tracker’s performance is deemed bad for 3 consecutive frames, failure recovery methods are initiated. Finally, the tracking thread is responsible for choosing frames to become keyframes.

A frame is labeled as keyframe in PTAM if the following conditions are met:

1. 20 frames have passed since the last keyframe was accommodated into the map.
2. The euclidean distance from the current frame’s pose to the nearest keyframe in the map has exceeded a threshold determined by the observed scene’s mean depth.
3. The queue structure of the map making thread is not full (it can contain up to 3 frames only) as processing keyframes may take time (not at frame rate).

The second condition is set so that if the tracker is close to the observed scene, landmarks are anticipated to go out of the field of view quickly and hence new landmarks must be generated in time not to suffer from tracking failure. Whereas if the observed scene is far, landmarks will not go out of view quickly and adding new keyframes will cause an increase in redundant data and bundle adjustment time.

SVO pose estimation SVO uses a sparse model based image alignment in a pyramidal scheme (from highest pyramid level to the third) in order to estimate

an initial camera pose estimate. It starts by assuming the camera pose at time t to be the same as at $t - 1$ and aims to minimize the photometric error of 2D image locations of known depth in the current frame with respect to their location at $t - 1$, by varying the camera transformation relating both frames. The minimization takes place through thirty Gauss Newton iterations of the inverse compositional image alignment method. This however introduces many limitations to SVO since the ICIA requires small displacements between frames (1pixel). This limits the operation of SVO to high frame rate cameras (typically $> 70\text{fps}$) so that the displacement limitation is not exceeded. Furthermore, the ICIA is based on the brightness consistency constraint rendering it vulnerable to any variations in lighting conditions.

SVO does not employ explicit feature matching for every incoming frame, however feature matching takes place implicitly as a byproduct of the image alignment step. Once image alignment takes place, map landmarks, that are estimated to be visible in the current frame, are projected onto the image and divided into a grid. The 2D location of the projected landmarks are fine-tuned by minimizing the photometric error between a patch extracted from the initial projected location in the current frame and a warp of the landmark generated from the nearest keyframe observing it. The minimization takes place through ICIA. Only one projected landmark per grid is used to decrease the computational complexity and to maintain only the strongest features (high saliency). This minimization however violates the epipolar constraint for the entire frame and further processing in the tracking module is required. Motion only Bundle Adjustment that refines the camera pose by minimizing the re-projection residuals caused by individual alignment of features in the previous step takes place, followed by a structure only Bundle Adjustment that refines the 3D location of the landmarks based on the refined camera pose of the previous step.

Finally, a joint (pose and structure) local bundle adjustment fine-tunes the reported camera pose estimate. The camera pose estimation module in SVO also keeps records of the tracking quality; if the number of observations in a frame is below a certain threshold or if the number of features between consecutive frames drops drastically, tracking quality is deemed insufficient and failure recovery methods are initiated. The camera pose estimation is also responsible for flagging certain frames as keyframes. A frame is selected as a keyframe in SVO if its distance (pose) to the nearest keyframe is larger than a threshold proportional to the mean of the observed scenes depth in the current frame.

DT SLAM pose estimation DT SLAM maintains a camera pose based on three tracking modes: full pose estimation, Essential matrix estimation, and pure rotation estimation. When enough 3D matches exist, a full pose can be estimated, otherwise if a sufficient number of 2D matches are established that exhibit small translations, an Essential matrix is estimated and finally if a pure

rotation is exhibited, 2 points are used to estimate the absolute orientation of the matches [57]. The tracking finally aims to minimize the error vector of both 3D-2D re-projections and 2D-2D matches in an iterative scheme. When tracking failure occurs, the system initializes a new map and continues to collect data and tracking continue in a different map; however, the map making thread continues to look for possible matches between the keyframes of the new map and the old one and once a match is established, both maps are fused together in an optimization scheme over the similarity transform linking both maps, allowing the system to handle multiple sub-maps with different scales.

LSD SLAM pose estimation The tracking thread in LSD SLAM is responsible for estimating the current frame pose with respect to the currently active keyframe in the map using the previous frame pose as a prior. The required pose is represented by an $SE(3)$ transformation and is found by an iteratively re-weighted Gauss-Newton optimization that minimizes the variance normalized photometric residual error, as described in [17], between the current frame and the active keyframe in the map. A keyframe is considered active if it is the most recent keyframe accommodated in the map. To minimize outliers effects, measurements with large residuals are down weighted from one iteration to the other.

ORB SLAM pose estimation The tracking thread in ORB SLAM is responsible for maintaining the camera pose at frame rate and makes decisions whether the processed frame is a keyframe or not. The main tracking work can be divided into two main steps: initial pose estimate and local map tracking. The initial pose estimate is first established through a constant velocity motion model similar to PTAM's and an optimization step takes place, refining the camera pose through a minimization of the re-projection error of all features that were observed in the previous frame in a narrow search range guided by the motion model. However, as this motion model is expected to be easily violated through abrupt motions, ORB SLAM detects such failure through the number of matched features, if it falls below a certain threshold, map points are projected onto the current frame using the camera pose of the previous frame and a wide range feature search takes place around the projected locations before the pose optimization takes place. If tracking fails (*i.e.*, both of the above solutions fail to establish a sufficient number of feature matches or the bundle adjustment does not converge) ORB SLAM invokes its failure recovery method to establish an initial frame pose via global re-localization.

When the initial pose estimate is established, ORB SLAM's tracking thread moves to local map tracking. To bound the complexity of large maps in an effort to make the system operate in large environments, a subset of the global map known as the local map, is defined by all landmarks corresponding to the set of

all keyframes that share edges with the current frame as well as all neighbors of this set of keyframes, taken from the covisibility graph. The keyframe from this set that shares the highest number of features with the current frame is known as K-ref to be used later for keyframe insertion checks. The selected landmarks are filtered out to keep only the features that are most likely to be matched in the current frame by monitoring the relative angle between the average norm of all keyframes observing the landmark and the ray connecting the current frames center to the landmark. Furthermore, if the distance from the cameras center to the landmark is beyond the range of the valid features scales, the landmark is also discarded. The remaining set of landmarks is then searched for and matched in the current frame before a final camera pose refinement step takes place. The tracking thread is also responsible for labeling the current frame a keyframe or not. In contrast to some of the other systems that tends to add keyframes when the cameras pose has moved a significant distance, ensuring a minimum positional change (rotational and translational), ORB SLAM, like RD SLAM, aims to ensure a minimum visual change while spawning as many keyframes as possible to increase the robustness of the tracking algorithm and maintain real-time operation. Overall, the following must be met so that the current frame is considered a keyframe:

1. 20 frames have passed since the last successful global re-localization procedure.
2. 20 frames have passed since the insertion of the previous keyframe or the local mapping thread is idle.
3. The current frame tracks successfully more than 50 features.
4. The current frame track less than 90% of K-ref.

RD SLAM pose estimation Once RD SLAM establishes enough feature matches between the incoming frame and the database of features in the KD-Tree [53], the frames pose is estimated through a motion only BA step that minimizes the re-projection error of the features over the camera pose parameters. To cope with dynamic (moving objects) in the scene and to attenuate their impact on the camera pose estimate, RD SLAM suggests a prior based adaptive RANSAC implementation, which samples (based on the outlier ratio of features in previous frames) the features in the current frame from which to estimate the camera pose and discard from the optimization the features that it suspects belong to moving objects. The tracking thread in RD SLAM is also responsible for the keyframe selection criteria as it aims to ensure enough structural change before the addition of new keyframes. Keyframe selection is divided into 2 checks, the first one ensures that the frames pose was correctly tracked and that it contains less than 80 features in common with other keyframes; if the frame passes the first

check it is considered a potential keyframe. The second check attempts to match features between the potential keyframe and the features associated with its 5 nearest keyframes; if the number of newly added landmarks is above a threshold the potential keyframe becomes a keyframe and is incorporated into the KD-Tree of the system. Since KD-tree updates are slow, the system maintains 2 trees, one active and one waiting to be updated. Once a sufficient number of newly created 3D landmarks are added into the system, it switches trees and update the inactive tree.

Table 5.4: Pose estimation used by different Visual SLAM systems. Abbreviations are as follows: constant velocity motion model (c.v.m.m), same as previous pose (s.a.p.p.), similarity transform with previous frame (s.t.p.f.), optimization through minimization of features (o.m.f.), optimization through minimization of photometric error (o.m.p.e.), Essential matrix decomposition (E.m.d.), pure rotation estimation from 2 points (p.r.e.), significant pose change (s.p.c.), significant scene appearance change (s.s.a.c)

	PTAM	SVO	RD SLAM	DT SLAM	LSD SLAM	ORB SLAM
Motion prior	c.v.m.m. +ESM	s.a.p.p.	none	s.t.p.f.	s.a.p.p.	c.v.m.m. or place recogn.
Tracking	o.m.f.	o.m.p.e.	o.m.p.e.	o.m.f.	3 modes: 1-E.m.d.; 2-o.m.f.; 3-p.r.e.	o.m.p.e.
keyframe add criterion	s.p.c.	s.p.c.	s.s.a.c.	s.s.a.c.	s.p.c.	s.s.a.c.

5.5 Map generation

The map making module in a generic visual SLAM implementation is responsible for updating the map as the camera explores new territory. At this point, a distinction between features and landmarks must be made; features are salient 2D locations belonging to the image plane whereas landmarks are 3D points belonging to the map of the scene and triangulated through the re-observation of a given feature across two+ frames separated by sufficient baseline. The map making process handles the initialization of new landmarks into the map as well as outliers detection and handling. Figure 5.6 represent the map generation task

and Table. 5.5 summarizes the landmark triangulation methods used by different Visual SLAM systems. Landmarks are triangulated via different methods;

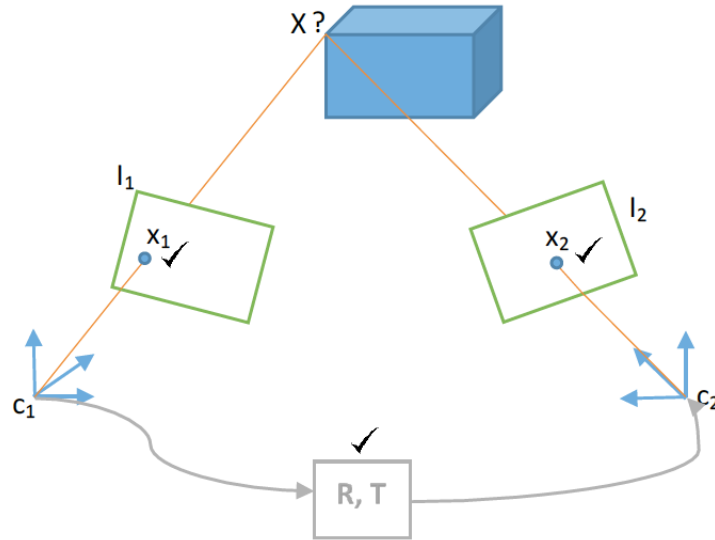


Figure 5.6: Map generation required by any Visual SLAM system

while some algorithms triangulate landmarks through features matched across two frames (PTAM, DT SLAM, RD SLAM and ORB SLAM) others employ a particle filter implementation and only integrate a landmark in the scene once enough views of the feature are established (SVO, LSD SLAM). A major limitation in all these methods is that they require a baseline between the images observing the feature for accurate 3D triangulation and hence are all prone to failure when the cameras motion is constrained to pure rotations. To counter such mode of failure, DT SLAM introduced 2D landmarks in the map that can be used for rotation estimation before their triangulation into 3D landmarks.

PTAM map generation PTAM implements Levenberg-Marquard optimization [39] that performs sparse bundle adjustment either on a local or a global magnitude. The bundle adjustment step minimizes the re-projection error of the landmarks to all keyframes they are observed in, by jointly optimizing both the 3D positions of the landmarks and the $SE(3)$ poses of the keyframes. If a new keyframe is added to the system, all bundle adjustment operations are halted and the new keyframe inherits the pose from the coarse tracking stage. The potentially visible set is then re-projected onto the new keyframe and feature matches are established (not only at the highest pyramid levels). Correctly matched landmarks are marked as Seen again; this is done to keep track of the quality of the landmarks and to allow for the map refinement step to remove corrupt data. The

correctly matched landmarks average depth from the current keyframe is used to limit the epipolar search, for new feature matches, in regions that do not contain projected landmarks, between the current keyframe and the closest to it in the database in term of position. This limits the computational cost of the search for new features and avoid adding them in regions where nearby landmarks exist as they could be the same; however, this limits the newly created features to be within the search region of the epipolar lines and hence very large variations in the scenes depth may lead to the negligence of possible landmarks that does not reside within the search region, leading in certain circumstances to tracking failure. The computational cost in PTAM scale with the map and become intractable as the number of keyframes get large and hence PTAM is designed to operate in small workspaces.

SVO map generation The map generation thread in SVO runs parallel to the tracking thread and is responsible for creating and updating the map. SVO parametrize 3D landmarks using an inverse depth parameterization model [58]. Upon insertion of a new keyframe, features extracted with highest Shi-Tomasi scores from each grid cell that does not contain any projected landmarks are chosen to initialize depth filters. These features are labeled as seeds and are initialized to be along a line passing through the camera center and the 2D location of the seed in the originating keyframe. The only parameter that remains to be solved is then the depth of the landmark. By default, the depth of the seed is initialized to the mean of the scenes depth as observed from the keyframe of origin with high uncertainty.

If no keyframe is being accommodated, the map management thread monitors and updates map seeds by subsequent observations in newly acquired frames. The seed is searched for in new frames along an epipolar search line limited by the uncertainty of the seed and the mean depth distribution observed in the current frame. As the filter converges, its uncertainty decreases and the epipolar search range decreases. If seeds fail to match frequently or diverge to infinity or a long time has passed since their initialization, they are considered as bad seeds and hence removed. Otherwise, the uncertainty of the filter is modeled as one of two types: a Gaussian distribution or a uniform model. The filter is considered to have converged when the estimated depth possesses a Gaussian distribution. Upon convergence, the filter’s depth is assigned to the seed and the latter is accommodated in the map as a landmark. This however limits SVO to operate in environments of relatively uniform depth distributions. Since the initialization of landmarks in SVO relies on many observations for the features to be triangulated, the map contains few if any outliers and hence no outliers deletion methods are required. However, this comes at the expense of delayed time before the features are turned into landmarks and added to the map to be tracked.

LSD SLAM map generation LSD SLAM’s map generation module is mainly responsible for selection and accommodation of new keyframes into the map. Its function can be divided into two main categories depending whether the current frame is deemed a keyframe or not; if it is, depth map creation takes place by keyframe accommodation, if it is not deemed a keyframe depth map refinement is done on regular frames. Since LSD SLAM is a direct method, it operates on the underlying assumption of small displacements between frames. To maintain tracking quality, LSD SLAM requires frequent addition of keyframes into the map as well as relatively high frame rate cameras.

If a frame is labeled a keyframe, the estimated depth map from the previous keyframe is projected onto it to serve as an initial depth map. Spatial regularization then takes place by replacing each projected depth value by the average of its surrounding depth values and the variance is chosen as the minimal variance value of the neighboring measurements.

An outliers detection step takes place by monitoring the probability of the projected depth hypothesis at each pixel to be an outlier or not. To make the outliers detection step possible, LSD SLAM keeps records of all successfully matched pixels during the tracking thread and accordingly increase or decrease the probability of it being an outlier.

During the spatial regularization step, if the probability that all contributing neighbors are outliers is above a given threshold, the hypothesis is removed from the system. The newly projected depth map is scaled to have a mean inverse depth value of 1. The scaling ratio is then incorporated into the $Sim(3)$ transformation that describes the new keyframe pose in the map. This is done to allow for scale optimization and correction during subsequent steps.

The $Sim(3)$ of a newly added keyframe is then estimated and refined in a direct, scale-drift aware image alignment scheme, which is similar to the one done in the tracking thread but with respect to other keyframes in the map and over the $7d.o.f.$ $Sim(3)$ transform, in contrast to the $6d.o.f.$ $SE(3)$ transform used for the regular tracking procedure. Furthermore, as the photometric error alone does not constrain the scale, the depth error integration into the minimization of the $Sim(3)$ transform is mandatory.

Due to the non-convexity of the direct image alignment method on $Sim(3)$, an accurate initialization to the minimization procedure is required; for such purpose, ESM [55] (Efficient Second Order minimization) and a coarse to fine pyramidal scheme with very low resolutions proved to increase the convergence radius of the task. The newly accommodated keyframe is flagged as the currently active keyframe and subsequent frames are tracked according to it.

If the map generation module deems the current frame as not a keyframe, depth map refinement takes place by establishing stereo matches for each pixel in a suitable reference frame. The reference frame for each pixel is determined by the oldest frame the pixel was observed in, where the disparity search range and the observation angle do not exceed a certain threshold. A 1-D search along the

epipolar line for each pixel is used with an SSD metric to establish pixel matches; the search range is limited by the prior depth value at a given pixel if available; otherwise, the entire range is scanned (if the pixel is newly observed).

To minimize computational cost and reduce the effect of outliers on the map, not all established stereo matches are used to update the depth map; instead, a subset of pixels is selected for which the accuracy of a disparity search is sufficiently large. The accuracy is determined by three criteria: the photometric disparity error, the geometric disparity error, and the pixel to inverse depth ratio. Further details regarding these criteria are outside the scope of this work, the interested reader is referred to [17]. The depth map is then propagated using the selected subset before depth map regularization takes place and outliers handling similar to the keyframe processing step.

ORB SLAM map generation ORB SLAM’s local mapping thread is responsible for keyframe insertion, map point triangulation, map point culling, keyframe culling and local bundle adjustment. The keyframe insertion step is responsible for updating the covisibility and essential graphs with the appropriate edges as well as computing the bag of words representing the newly added keyframe in the map. The covisibility graph is a pose graph that represents keyframes by nodes. Node poses within the graph represent a similarity transformation $Sim(3)$, similar to LSD SLAM, to incorporate a 7th degree of freedom (scale). Edges in a pose graph represent measurements observed in both nodes. The higher the number of features matched the higher the strength of the edge. In contrast to other methods, such as LSD SLAM, that imposes a minimum number of feature matches between keyframes to establish an edge between their corresponding nodes, ORB SLAM does not employ any thresholds on the edge creation process leading to a base graph that contains all possible connections and later, depending on the task required, extract a thresholded graph.

During keyframe insertion, ORB SLAM creates and updates a spanning tree that starts with the initial keyframe and incorporates all subsequent keyframes, that share with it the highest number of features; the result is a minimum connectivity graph. While performing graph optimization of the map, a threshold of 100 is applied to the base graph and the connectivity edges found are added to the spanning tree to form what is known as the Essential graph. In case of loop closure, the edges linking the loop ends are also included in the Essential graph. This implementation of graphs, ensures a fast optimization of the map through strongly connected network of cameras.

The map point creation module spawn new landmarks by triangulating ORB features from connected keyframes in the covisibility graph. Triangulated landmarks are tested for positive depth, re-projection error, and scale consistency in all keyframes they are observed in to be accommodated into the map.

DT SLAM map generation Similar to ORB SLAM, DT SLAM also aims to add keyframes when enough visual change has occurred. the three criteria for keyframe addition are

1. The frame to contain enough new 2D features that can be created from areas not covered by the map.
2. A minimum number of 2D features can be triangulated into 3D landmarks.
3. A given number of already existing 3D landmarks have been observed from a significantly different angle.

The map contains two types of features, 3D and 2D, the triangulation of 2D features into 3D is deferred until enough parallax between keyframes is observed hence the name of the algorithm.

Table 5.5: Map generation used by different Visual SLAM systems. Abbreviations as follows: 2 view triangulation (2.v.t.), particle filter with inverse depth parametrization (p.f.), 2D landmarks triangulated to 3D landmarks (2D.l.t.), depth map propagation from previous frame (p.f.p.f.), depth map refined through small baseline observations (s.b.o.)

	PTAMSVO		RD SLAM	DT SLAM	LSD SLAM	ORB SLAM
Map generation	2.v.t.	p.f.	2.v.t.	2D.l.t.	p.f.p.f or s.b.o.	2.v.t.

5.6 Map maintenance

Map maintenance takes care of optimizing the map through either bundle adjustment or pose graph optimization. During a map exploration phase in a generic Visual Slam session, new 3D landmarks are triangulated based on the camera pose estimates. After some time, system drift manifests itself in wrong camera pose measurements due to accumulated errors in previous camera poses (keyframes) that were used to expand the map. Table 5.6 summarizes the map maintenance procedure adopted by different Visual SLAM system.

Bundle adjustment Bundle adjustment (BA) is inherited from SfM and consists of a nonlinear optimization process of refining a visual reconstruction to jointly produce an optimal structure and coherent camera pose estimates. Bundle adjustment is computationally involved and computationally intractable if

performed on all frames and all poses. The breakthrough that enabled its application in PTAM is the notion of keyframes, where only select frames labeled as keyframes are used in the map creation and passed to the bundle adjustment process in contrast to SfM methods that uses all available frames. Different algorithms apply different criteria for keyframe labeling as well as different strategies for BA, some use jointly a local (over a local number of keyframes) LBA and global (over the entire map) GBA while others argue that a local BA only is sufficient to maintain a good quality map. To reduce the computational expenses of bundle adjustment, Strasdat et al. [59] proposed to represent the visual SLAM map by both a Euclidean map for LBA, along with a topological map for pose graph optimization that explicitly distributes the accumulated drift along the entire map.

Pose graph optimization Pose graph optimization first generates a pose graph of the map, where the keyframes are represented by nodes and edges between nodes corresponds to matched features between keyframes; the result is known as a connectivity graph. It then proceeds by minimizing the re-projection error of features onto the nodes by varying the pose of the nodes only. The Pose Graph Optimization (PGO) returns inferior results to those produced by GBA. The reason is that while PGO optimizes only for the keyframe poses and accordingly adjusts the 3D structure of landmarks GBA jointly optimizes for both keyframe poses and 3D structure. The stated advantage comes at the cost of computational time, with PGO exhibiting significant speed up compared to GBA; however, pose graph optimization requires efficient loop closure detection and may not yield an optimal result as the errors are distributed along the entire map, leading to locally induced inaccuracies in regions that were not originally wrong. Map maintenance is also responsible for detecting and removing outliers in the map due to noisy and faulty matched features. While the underlying assumption of most Visual SLAM algorithms is that the environment is static, some algorithms such as RD SLAM exploits map maintenance methods to accommodate slowly varying scenes (lighting and structural changes).

PTAM map maintenance The map making thread in PTAM runs parallel to the tracking thread and does not operate on a frame by frame basis instead it only processes select frames known as keyframes. If the map making thread is not processing new keyframes, it performs various optimizations and maintenance to the map:

- Ensure the maps local convergence through an LBA to the last added keyframe and its 4 nearest neighbors (in terms of pose) with all landmarks associated to them.
- Ensure the maps global convergence by applying a global adjustment to the

entire map.

- Applies data refinement by first searching and updating landmarks observations in all keyframes and then by removing all landmarks that have failed to successfully match features many times while marked as a potentially visible point more than 8 times.

DT SLAM map maintenance Similar to PTAM, the bundle adjustment thread of DT SLAM continuously optimizes the entire map in the background through a sparse full BA.

SVO map maintenance For runtime efficiency reasons, SVO's map management maintains only a fixed number of keyframes in the map and removes distant ones when new keyframes are added. This is performed so that the algorithm maintains real-time performance after prolonged periods of operation over large distances.

LSD SLAM map maintenance LSD SLAM runs a third parallel thread that continuously optimizes the map in the background by a generic implementation of a pose graph optimization using the g2o-framework [18]. This however leads to an inferior accuracy when compared to other methods for reasons stated above. As the pose graph optimization neglects to optimize the map against landmarks in the map and only optimizes the pose graph of the keyframes within the map and the landmarks locations are adjusted accordingly, in contrast to other methods that employ a bundle adjustment step that jointly optimize the keyframes poses as well as the landmarks measurements in each keyframe.

ORB SLAM map maintenance To maintain a good quality map and counter the effect of frequently adding features from keyframes, ORB SLAM employs rigorous landmark culling to ensure low outliers in the map. A landmark must be correctly matched 25% of the frames in which its predicted to be visible; it also must be visible from at least three keyframes after more than one keyframe has been accommodated into the map since it was spawned. Otherwise, the landmark is removed. To maintain lifelong operation and to counter the side effects (computational complexity, outliers, redundancy) of the presence of high number of keyframes in the map, a rigorous keyframe culling procedure takes place as well. Keyframes that have 90% of their associated landmarks observed in 3 other keyframes are deemed redundant and removed. The local mapping thread also performs a local bundle adjustment over all keyframes connected to the latest accommodated keyframe in the covisibility graph and all other keyframes that observe any landmark present in the current keyframe even if they're not connected through the covisibility graph.

RD SLAM map maintenance To gain immunity against slowly varying scene structures and gradual lighting changes, RD SLAM employs an on-line 3D point update mechanism at each incoming frame to remove 3D landmarks from the map that might have changed. To detect changed regions, RD SLAM monitors the histogram of colors between the current frame and the five nearest keyframes to it in the map. If enough difference between the histograms is detected then either of the following could be true: region occluded due to change in the point of view, 3D structure changed due to object removals, or scene lighting changes occurred. RD SLAM then employs a 3D point and keyframe culling mechanism to remove the points that have changed due to the latter two reasons while maintaining the points that are suspected to be only occluded by temporary obstacles.

Table 5.6: Map maintenance used by different Visual SLAM systems. Abbreviations used: Local Bundle Adjustment (LBA), Global Bundle Adjustment (GBA), Pose Graph Optimization (PGO),

	PTAM	SVO	RD SLAM	DT SLAM	LSD SLAM	ORB SLAM
Optimization type	LBA & GBA	LBA	GBA	LBA & GBA	PGO	PGO & LBA
Scene type	static & small	uniform depth	static or slow dynamic	static & small	static & small or large	static & small or large

5.7 Failure recovery

Whether due to wrong user movement (abrupt change in the camera pose and motion blur), or camera observing a featureless region or failure to match enough features, or for any other reason visual SLAM methods can eventually fail. A key module essential for the usability of any visual SLAM system is its ability to correctly recover from such failures and resume its ordinary tasks. The failure problem is described as a lost camera pose that needs to re-localize itself in the map it had previously created of its environment before failure. For such purpose different algorithms employ different methods. Table 5.7 summarizes the failure recovery mechanisms used by different Visual SLAM system.

PTAM failure recovery Upon tracking failure, PTAM’s tracker initiates a recovery procedure, where the SBI of each incoming frame is compared to the database of SBIs for all keyframes. If the intensity difference between the incoming frame and its nearest looking keyframe is below a certain threshold, the current frame’s pose is assumed to be equivalent to that of the corresponding

keyframe, ESM tracking takes place (similar to that of the tracking thread) to estimate the rotational change between the keyframe and the current frame. If converged, a PVS of landmarks from the map is projected onto the estimated pose and the tracker attempts to match the landmarks to the features in the frame. If enough features are correctly matched, the tracker resumes normally otherwise a new frame is acquired and the tracker remains lost. For successful re-localization, this method requires the lost camera's pose to be near the recorded keyframes pose and otherwise would fail when there is a large displacement between the two.

SVO failure recovery When tracking quality is deemed insufficient, SVO initiates its failure recovery method. The first procedure in the recovery process is to apply image alignment between the incoming frame and the closest keyframe to the last known correctly tracked frame. If more than 30 features are correctly matched during the image alignment step, then the re-localizer considers itself converged and continues tracking regularly; otherwise it attempts to re-localize using new incoming frames. Such a re-localizer is sensitive to any change in the lighting conditions of the scene and the lost frame location should be close enough to the queried keyframe for successful re-localization to take place.

LSD SLAM failure recovery LSD SLAM's recovery procedure first chooses randomly a keyframe from the map that has more than 2 neighboring keyframes connected to it in the pose graph. It then attempts to align the currently lost frame to it. If the outliers to inlier ratio is large, the keyframe is discarded and replaced by another keyframe at random; otherwise, all neighboring keyframes connected to it in the pose graph are then tested; If the number of neighbors, with a large inlier to outliers ratio, is larger than the number of neighbors with a large outlier to inlier ratio, or if there are more than 5 neighbors with a large inlier to outlier ratio, the neighboring keyframe, with the largest ratio is set as the active keyframe and regular tracking is accordingly resumed.

ORB SLAM failure recovery Triggered by tracking failure, ORB SLAM invokes its global place recognition module. Upon running, the re-localizer transforms the current frame into a bag of words and queries the database of keyframes for all possible keyframes that might be used to re-localize from. The place recognition module implemented in ORB SLAM, used for both loop detection and failure recovery, relies on bags of words as frames observing the same scene share a big number of common visual vocabulary. In contrast to other bag of words methods that return the best queried hypothesis from the database of keyframes, the place recognition module of ORB SLAM returns all possible hypotheses that have a probability of being a match larger than 75% of the best match. The combined added value, of the ORB features along with the bag of words implementation of the place recognition module, manifest itself in a real-time, high recall and

relatively high tolerance to viewpoint changes when performing re-localization of a lost tracker and while detecting loops. All hypotheses are then tested through a RANSAC implementation of the PnP algorithm [60] that determines the camera pose from a set of 3D to 2D correspondences. The found camera pose with the most inliers is then used to establish more matches to features associated with the candidate keyframe before an optimization over the cameras pose using the established matches takes place.

Table 5.7: Failure recovery used by different Visual SLAM systems. Abbreviations used: photometric error minimization of SBIs (p.e.m.), image alignment with last correctly tracked keyfram (i.a.l.c.k), image alignment with random keyframe (i.a.r.k.), bag of words place recognition (b.w.p.r.)

	PTAM	SVO	RD SLAM	DT SLAM	LSD SLAM	ORB SLAM
Failure recovery	p.e.m.	i.a.l.c.k.	no data	none	i.a.r.k.	b.w.p.r.

5.8 Loop closure

Since Visual SLAM is an optimization problem, its prone to drifts in camera pose estimates and returning to a certain pose after an exploration phase may not yield the same camera pose measurement as it was at the start of the run. Such camera pose drift can also manifest itself in a map scale drift that will eventually lead the system to erroneous measurements and fatal failure. To address this issue, some algorithms detect loop closures in an on-line Visual SLAM session and optimize the loops track in an effort to correct the drift and the error in the camera pose and in all relevant map data that were created during the loop. The loop closure thread attempts to establish loops upon the insertion of a new keyframe in order to correct and minimize any accumulated drift by the system over time. Table 5.8 summarizes the Loop closure mechanisms used by different Visual SLAM system.

LSD SLAM loop closure Whenever a keyframe is processed by LSD SLAM, loop closures are searched for within its ten nearest keyframes as well as through the appearance based model of openFABMAP [61] to establish both ends of a loop. Once a loop edge is detected, a pose graph optimization minimizes the similarity error established at the loops edge by distributing the error over the loops keyframes poses.

ORB SLAM loop closure Loop detection in ORB SLAM takes place via its global place recognition module that returns all hypotheses of keyframes from the database that might correspond to the opposing loop end. To ensure enough

distance change has taken place, the similarity transform between all connected keyframes to the current keyframe in the thresholded covisibility graph (threshold of 30 correspondences), is computed; the lowest score is recorded as S_{min} . All hypotheses are then tested and removed if any of the following occurs:

- It is connected to the current keyframe through the covisibility graph
- It has a similarity score less than S_{min}
- It has less than 3 other keyframe hypotheses connected to it in the covisibility graph.

To compute an accurate similarity transform, an initial set of correspondences between all ORB landmarks associated to the current keyframe are matched to the associated landmarks of queried keyframe from the database through their bags of words. Once the 3D-3D correspondences are established, they are used in a RANSAC scheme to determine a $Sim(3)$ relating both keyframes. If the similarity is supported by enough inliers, a narrow search range across the ORB descriptors associated to both keyframes is performed and the result is used to further refine the similarity transform. If enough inliers support the refined similarity transform, the queried keyframe is considered to be the other end of the loop and loop fusion takes place. The loop fusion first merges duplicate map points in both keyframes and insert a new edge in the covisibility graph that closes the loop by correcting the $Sim(3)$ pose of the current keyframe using the similarity transform. Using the corrected pose, all landmarks associated with the queried keyframe and its neighbors are projected to and searched for (in a narrow search region) in all keyframes associated with the current keyframe in the covisibility graph. The initial set of inliers, as well as the found matches are used to update the covisibility and Essential graphs, establishing many edges between the two ends of the loop. Finally, a pose graph optimization over the essential graph takes place similar to that of LSD SLAM, which minimizes and distributes the loop closing error along the loop nodes.

Table 5.8: Loop closure used by different Visual SLAM systems. Abbreviations used: Bag of Words place recognition (B.W.p.r), $sim(3)$ optimization (s.o.)

	PTAM SVO	RD SLAM	DT SLAM	LSD SLAM	ORB SLAM
Loop closure	none	none	none	OpenFabMap + s.o.	B.W.p.r. + s.o.

Chapter 6

Comparative assessment of Visual SLAM systems

To analyze the the performance of open source, state of the art visual SLAM systems, a comparative assessment is conducted in this chapter, underlining the performance of PTAM, SVO, LSD SLAM, ORB SLAM and DT SLAM. For such purpose, we developed a dataset with ground truth provided using specialized hardware as described in the following section.

6.1 Dataset generation

This section explain the AUB-dataset collection procedure, detailing the sensor setup and configuration. It also describe the test environment in which the dataset was collected, list the measurments recorded and provide tools required to align the estimated paths from different Visual SLAM systems with our dataset.

6.1.1 Hardware setup

The sensors specs used to generate our dataset can be summarized by:

- A Basler pulse camera (*puA1280 – 54uc*) with a resolution of 1.2 MP, 1/3” Aptina *AR0134* CMOS sensor equipped with a global shutter.
- A Kowa LMVZ164 1/3”, 1.6-3.4mm F1.4 manual iris vari-focal CS mount lens.
- An Xsens MTi-G-700 with an internal L1 GPS aided INS, typical 1σ RMS error values for roll/pitch/yaw/horizontal position/vertical position are $0.2^\circ/0.3^\circ/1.0^\circ/1.0m/2.0m$, pose output frequency 400Hz. . . .

The complete setup of the sensors shown in Fig 6.1 communicates with an Intel Core *i7 – 4710HQ* 2.5GHZ CPU with a 16 GB memory for data acquisition.



(a) the GPS antenna is fixed to the back of the Xsens INS which is bolted on a small chassis on top of the basler camera.

(b) The strap mounted rig.

Figure 6.1: Sensor rig setup

6.1.2 Hardware Configuration

The Basler camera was configured to acquire 640×480 colored images around the center of the CMOS sensor at 70 frames per second whereas the Xsens INS system was configured with a "GeneralMag" filter profile outputting pose estimates (rotation and position) at 400 Hz with all inertial and magnetic data enabled with a 32-bit floating point representations whereas GPS data were outputted with a 64-bit floating point representation to prevent numerical overflows.

6.1.3 Test set environment

The generated dataset contains about 16,400, colored, uncompressed **.tiff* images with a resolution of 640×480 , numbered starting from 0 corresponding to frames acquired during a loop around the green field of AUB, located in the lower campus (Figure 6.2). For each image a *.txt* file is provided and named with the same image number. It contains 7 entries as follows: Image number, Latitude, Longitude, Altitude, Roll, Pitch, Yaw. Where the Longitude, Latitude and Altitude are represented in the *WGS - 84* Ellipsoidal coordinate frame as shown in Figure 6.3, and the rotation parameters are represented in a local Earth-fixed coordinate system such as the X-axis is positive to the east, the Y-axis is positive to the north and the Z-axis positive pointing up. During data collection, the hand held sensor rig was moved such as the camera viewing direction was always parallel to the path and the loop took place along the first running lane (closest to the green field) with a trajectory of 400 meters. A sample of the collected dataset is shown in Figure 6.4 The collected dataset is publicly available at <http://feaweb.aub.edu.lb/research/cvrl/index.html> The data

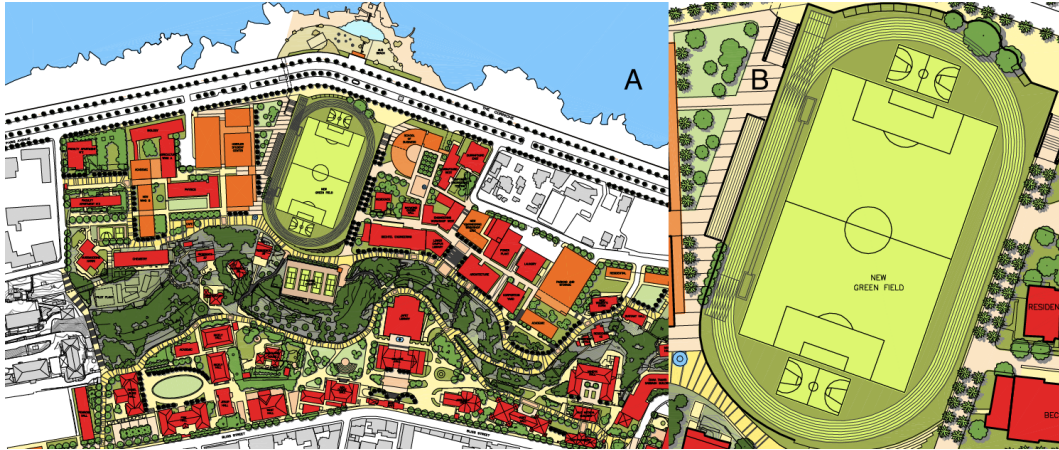


Figure 6.2: A- Overview of AUB campus. B- AUB's greenfield

λ = longitude

φ = latitude

h = altitude

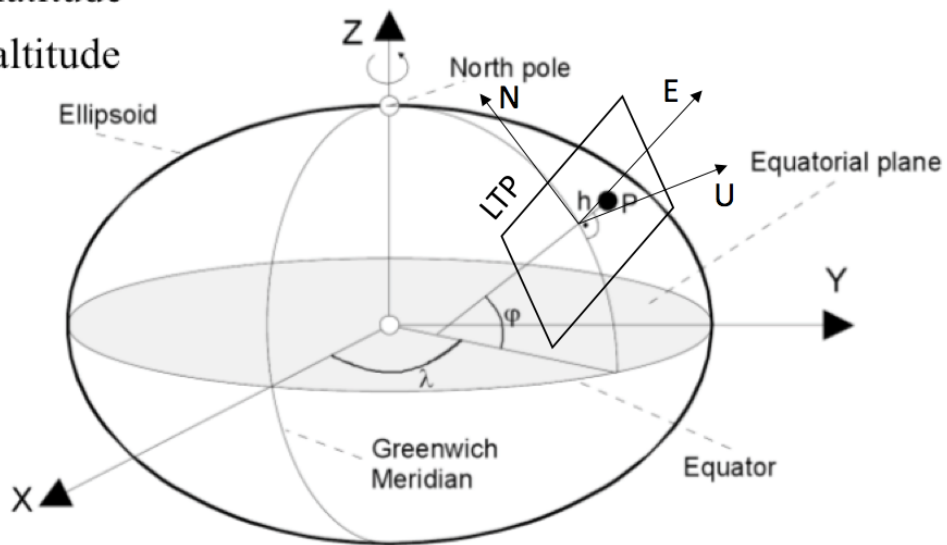


Figure 6.3: Ellipsoidal Coordinates (Latitude, Longitude, Altitude) in WGS-84 Ellipsoid

input classes for all the visual SLAM systems were modified to read our offline dataset images; whenever tracking quality of the corresponding visual SLAM system is deemed good, the camera pose is extracted and saved to a **.txt* file that contains the image number, the rotation and position estimates of the camera from the visual SLAMs internal map transformed such as the first recorded

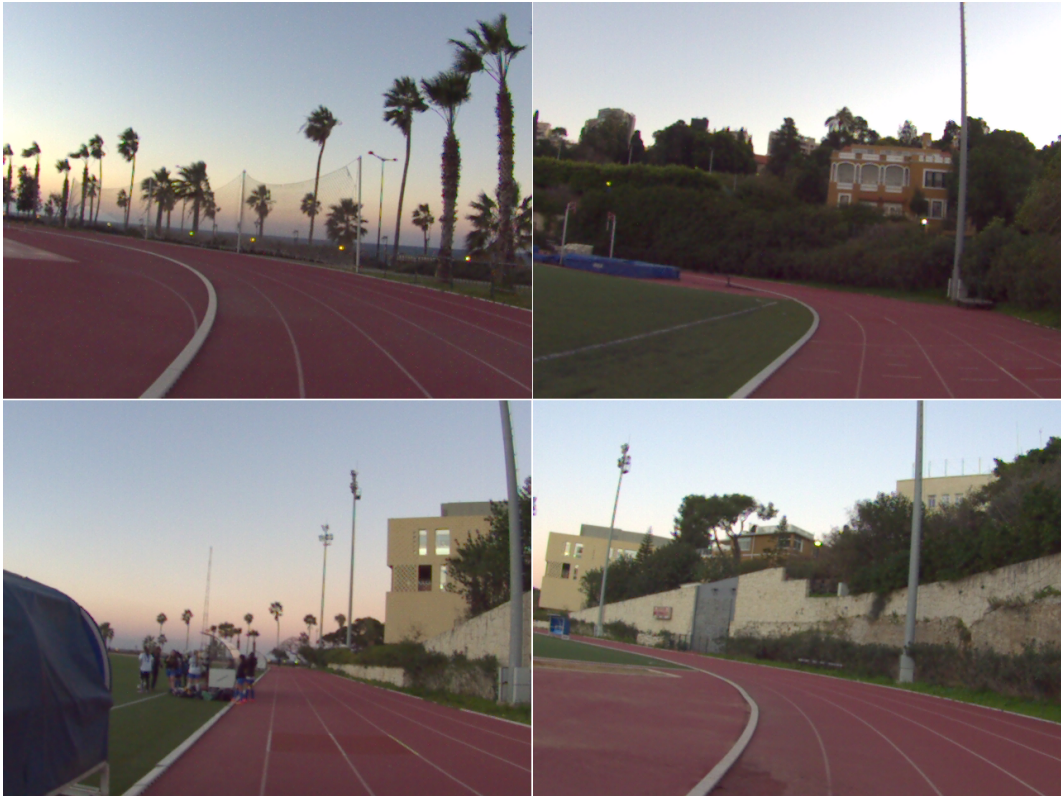


Figure 6.4: Sample images of the generated dataset

camera pose serves as the origin.

6.1.4 Data processing

To extract usable poses from the INS, to compare to the Visual SLAM measurements, data collected in the $WGS - 84$ coordinate frame are transformed to the Universal Transverse Mercator (UTM) coordinate frame. The projection from the ellipsoidal coordinate system to the 2D UTM coordinate system is carried by exploiting a locally tangent plane as shown in Figure 6.3. While further discussion regarding the conversion is outside the scope of this work, one aspect of the conversion that shall prove useful to exploit, is that the mapping through the LTP does not depend on the altitude, *i.e.* for a given Latitude and Longitude, the entire range of altitudes is mapped to a single point in the plane. This means that we can exclude altitude measurements from the estimated path without violating its integrity along the in-plane position measurements; therefore, to reduce the impacts of errors introduced by the altitude sensor of the INS, altitude measurements are set to 0. The pose estimates are then translated such that the pose estimated by the INS for the first image corresponds to $(0, 0, 0)^T$.

Due to the way the camera rig and the INS are set up, the transformation relating the camera to the INS is best described by a rotation along the Y-axis by $-\frac{\pi}{2}$ followed by a rotation along the X-axis by $\frac{\pi}{2}$. This can be interpreted as LTP plane corresponding to the X-Y plane in the INS coordinates, corresponds to the X-Z plane in the Visual SLAMs coordinate frame. Therefore, Y measurements in the Visual SLAM coordinate frame may be excluded from the scale estimation and pose error estimation without violating the integrity of the results.

The transformation required to align the paths recorded by the INS and by the different visual SLAM systems, is a rotation R , translation T and a scale Sc , relating both paths known as a similarity transform; The transformation aims to minimize the error between the paths

$$e^2(R, T, Sc) = \frac{1}{n} \sum_{i=0}^n \|P_{INS}^i - (Sc \times R \times P_{VSLAM}^i + T)\|^2, \quad (6.1)$$

by varying the parameters of the transformation. For such purpose, we exploit the knowledge of correlation between the camera poses and the INS poses through the image names by first estimating the centroids of the paths:

$$C_{INS} = \frac{1}{n} \sum_{i=0}^n P_{INS}^i, \quad (6.2)$$

$$C_{VSLAM} = \frac{1}{n} \sum_{i=0}^n P_{VSLAM}^i, \quad (6.3)$$

where $P_{INS}^i = (X, Y, 0)^T$ is the INS position estimate of the image i and $P_{VSLAM}^i = (X, 0, Z)^T$ is the Visual SLAM position estimate of the same image and n is the total number of images. Then we estimate the variance of the recorded path by the VSLAM as:

$$\sigma_x^2 = \frac{1}{n} \sum_{i=0}^n \|P_{VSLAM}^i - C_{VSLAM}\|^2, \quad (6.4)$$

Next we estimate the covariance matrix as:

$$\Sigma_{xy} = \frac{1}{n} \sum_{i=0}^n (P_{INS}^i - C_{INS}) (P_{VSLAM}^i - C_{VSLAM})^T, \quad (6.5)$$

Similar to [62] to recover the parameters of the transformation, we perform the following operations:

$$[UDV^T] = SVD(\Sigma_{xy}), \quad (6.6)$$

$$R = VU^T, \quad (6.7)$$

$$Sc = \frac{tr(D \times S)}{\sigma_x^2}, \quad (6.8)$$

$$T = C_{INS} - Sc \times R \times C_{VSLAM}, \quad (6.9)$$

where

$$S = \begin{cases} I_{3 \times 3} & \text{if } \det(U) \times \det(V) = 1 \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} & \text{if } \det(U) \times \det(V) = -1 \end{cases} \quad (6.10)$$

If $\det(R)$ was found to be negative, then R is replaced by:

$$R = V_2 \times U^T, \quad (6.11)$$

where V_2 is V with its last column multiplied by -1 . Finally, the similarity transformation is applied to the visual SLAM recorded path to align it with the INS path by:

$$P_{VSLAM,aligned}^i = Sc \times R \times P_{VSLAM}^i + T, \quad (6.12)$$

The end result is two optimally aligned paths in the X-Y coordinate frame of the INS.

6.2 Experimental results

For the purpose of our first experiment, we used our generated AUB-greenfield dataset as an input to the visual SLAM systems, and left them to loop over the image sequence once until the camera returned to the same location at which the corresponding visual SLAM system had been initialized. During the second experiment, to highlight the importance of the map optimization and map management threads, the systems were left to continue operating for another loop so that loop closure and map optimization take place.

The generated maps in the Visual SLAM's space highlighting the keyframe poses are shown for PTAMM, LSD SLAM and ORB SLAM in Fig.6.5(a) , Fig. 6.6 and Fig.6.8 respectively. While the keyframe poses are of great importance, as they ensure the integrity of the map, what is more important for the assessment of the different Visual SLAM systems is the live, local pose estimates they produce if they were to be used for an agent's localization in a scene. Therefore, the pose estimate for each tracked frame is exported and aligned with the INS measurements, using the method outlined in the previous section, such that the RMSE between the generated path of each Visual SLAM system and the path recorded by the INS is minimized for the first and second loop independently. The results obtained for PTAMM, LSD SLAM and ORB SLAM during the first loop are shown in Fig. 6.5(b), Fig. 6.7(a) and Fig.6.9(a) respectively, as for the second loop, the results of LSD SLAM are shown in Fig.6.7(b) and for ORB SLAM in Fig. 6.9(b). At the end of the second loop, the number of keyframes used to represent the map for each system, along with the number of landmarks and

their required RAM size. The experimental results are summarized in Table 6.1. SVO’s results were not reported as the system failed to track the map, no pose estimates nor map were extracted, whereas DT SLAM reported meaningless pose estimates and maps that cannot be displayed.

Table 6.1: Summary of experimental results. Abbreviations used: keyframes (KF), landmarks (LM) and not available (n.a.) RMSE values are reported in meters.

Visual SLAM	Number of KF	Number of LM	RAM usage	Loop1 RMSE	Loop2 RMSE
PTAMM	75	3973	163 MB	1.8625	1.8625
SVO	n.a.	n.a.	n.a.	n.a.	n.a.
DT SLAM	n.a.	n.a.	n.a.	n.a.	n.a.
LSD SLAM	118	6.7×10^6	1.6 GB	5.9298	4.9449
ORB SLAM	166	5570	2.7 GB	2.0265	1.7746

6.3 Discussion of results

This section analyses the previously reported results, exposing different modes of failures that challenge state of the art systems.

6.3.1 PTAMM

PTAMM’s initialization depends on the user’s choice for the first 2 keyframes. This unfortunately means that for different selection of starting keyframes, PTAMM will behave in a different manner, as internal parameters, vital to its functioning, are scaled by the ratio of the user’s translation to scene depth. To achieve optimum behavior during our experiments, the baseline separating the two initializing keyframes was roughly chosen as $1/10^{th}$ of the observed scene depth. Furthermore, if the initializing scene does not contain enough planar points, the homography estimation module falls in an endless loop, crashing the system.

This intricate initialization requirement is considered daunting especially for unexperienced users and tracking quality dependency on the user’s choice of keyframes constitutes a major drawback to the algorithm as it can cause many modes of error: system crash at startup due to the violation of the scene planarity assumption or the excessive/lack of insertion of keyframes into the map. To highlight the effect of the scene to depth ratio at startup, PTAMM was initialized with a very small baseline between the first 2 keyframes, the result was the system accommodating dozens of keyframes and triangulating hundreds of outlier landmarks in a matter of seconds as seen in Figure 6.10. An opposite scenario occurs when the baseline to depth ratio is large, if that’s the case, PTAMM will

not add new keyframes until a very large positional change had occurred, often leading to tracking failure as landmarks go out of sight and the system does not triangulate new ones.

While the GBA accompanied with a LBA yielded acceptable results, as the RMSE of PTAMM's path reflect, the system suffered from a different type of failure towards the end of the loop as highlighted in Figure 6.11. When PTAMM finished the loop, accumulated drift over time manifested itself as an erroneous camera pose, which led to tracking failure; as map landmarks, generated during the beginning of the run are being projected onto the wrong camera pose; the data association fails to correctly match the landmarks to the 2D features of the frame as they fall outside their search regions and hence tracking is deemed bad before failure recovery methods are invoked. The re-localizer quickly matches the current frame to the nearest keyframe that was generated at the beginning of the loop and the camera tracking algorithm resumes from there on, leaving behind a discontinuity in the camera pose estimate. It is noteworthy to mention that no matter how many times the system is left to loop over the dataset, the end-result is the same discontinuity at the loop closure without having any means to correct this type of failure. This may also be reflected by the RMSE of the paths generated in PTAM along the 2 loops which was identical.

Other modes of failure for PTAMM as well as most indirect Visual SLAM methods is displayed in Figure 6.12 where not enough image gradient is present to extract features leading to tracking failure.

6.3.2 SVO

One of SVOs major assumptions/drawbacks, is that it requires the observed scene to be of homogeneous depth for the tracking system to succeed. Unfortunately, it is not the case in our dataset. When SVO starts, it monitors the disparity vector between incoming frames until its large enough to initialize the system by inserting the 2nd keyframe into the map, however due to the non-homogeneous distribution of our scene, the conditions for keyframe insertion are never met; therefore, tracking quality fades and tracking failure occurs moments after the insertion of the second keyframe in the map. SVOs optimal performance is achieved when the camera is mounted downwards on a UAV, as the original paper [4] shows.

6.3.3 DT SLAM

DT SLAM's choice of not employing a failure recovery method meant that whenever tracking is lost, a new sub-map is initialized. This however lead to many sub-maps with different scales and camera poses that may not be used as a meaningful agent pose unless they're fused together with a uniform scale. For DT SLAM to achieve this fusion, it requires the tracked scene to remain in sight

for a significant amount of time so that enough correspondences between sub-maps are established. While this is a reasonable condition for small workspace environments such as a desk, its rarely the case in actual Visual SLAM applications where the camera keeps moving in a single direction and not revisiting the same scene.

Unfortunately, when benchmarked against our dataset, DT SLAM suffered from this failure and no useful camera trajectory was extracted.

6.3.4 LSD SLAM

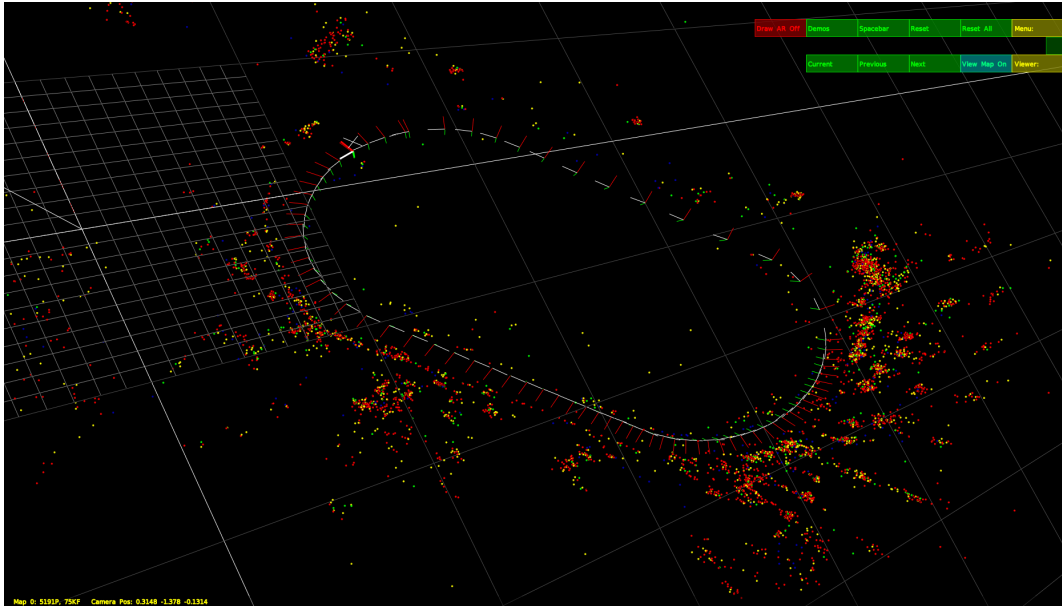
Upon the detection of the loop, LSD SLAM invoked its loop closure mechanism and the resultant map is reported in Figure 6.6

LSD SLAMs random initialization effects are clearly highlighted with a red marker, in Figure 6.13, where the camera pose estimates along the local map are erroneous and not reliable. Moreover, as the system finishes the first loop, drift manifested itself with a vast misalignment between the camera poses corresponding to the same frame, this is also reflected with the RMSE of the system for the first loop. It took the system 2000 images after the image sequence was looped for it to recognize loop closure, the equivalent of 50 meters of going through the same scenery. This can be seen by the uncorrected poses from the second loop highlighted with a green marker in Figure 6.7(b).

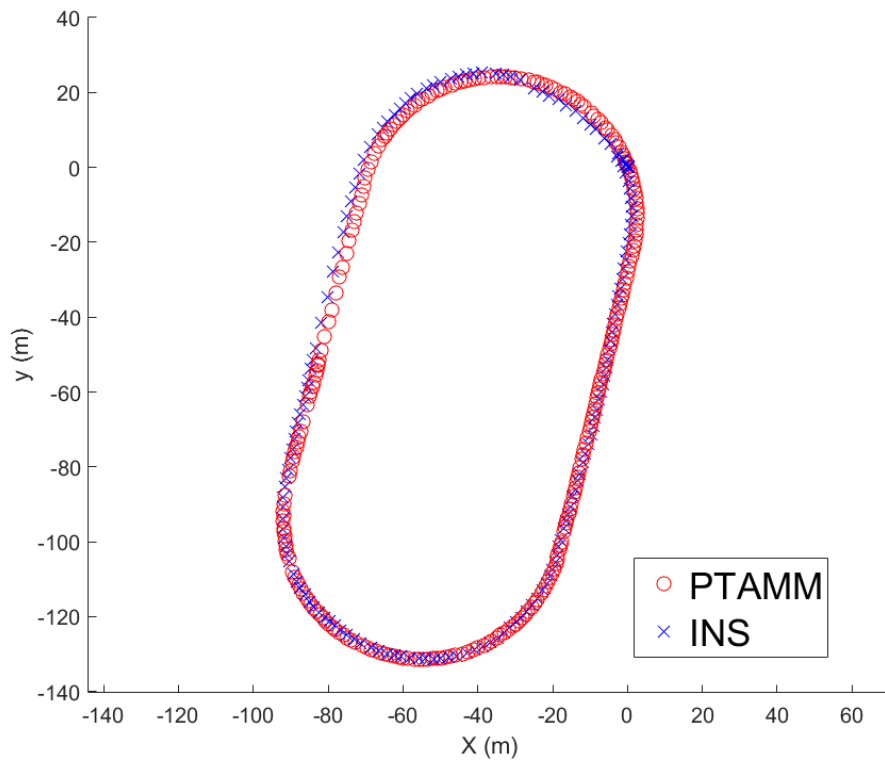
6.3.5 ORB SLAM

In contrast to PTAMM's manual initializing keyframe selection procedure, ORB SLAM employs an automatic mechanism to select a good pair for tracking to start. While this is supposed to be a fort to the system, it wasn't able to initialize until it came across a pair of frames that it considered to be good for tracking, which unfortunately didn't happen till image 8000 or the equivalent of 250 meters into the loop as shown in Figure 6.14. ORB SLAM's loop closure mechanism was instantly capable of detecting and closing the loop and the final map is shown in Figure 6.9(b). In contrast to LSD SLAM's 2nd loop that contains artifacts due to the relatively late loop closure detection.

The RMSE of the path reported by ORB SLAM clearly outperformed LSD SLAM, and lagged slightly behind PTAMM before loop closure, however by the end of the second loop it surpassed it.



(a) PTAMM map after looping the dataset twice



(b) Estimated Path in PTAMM aligned with INS measurements

Figure 6.5: reported results by PTAMM

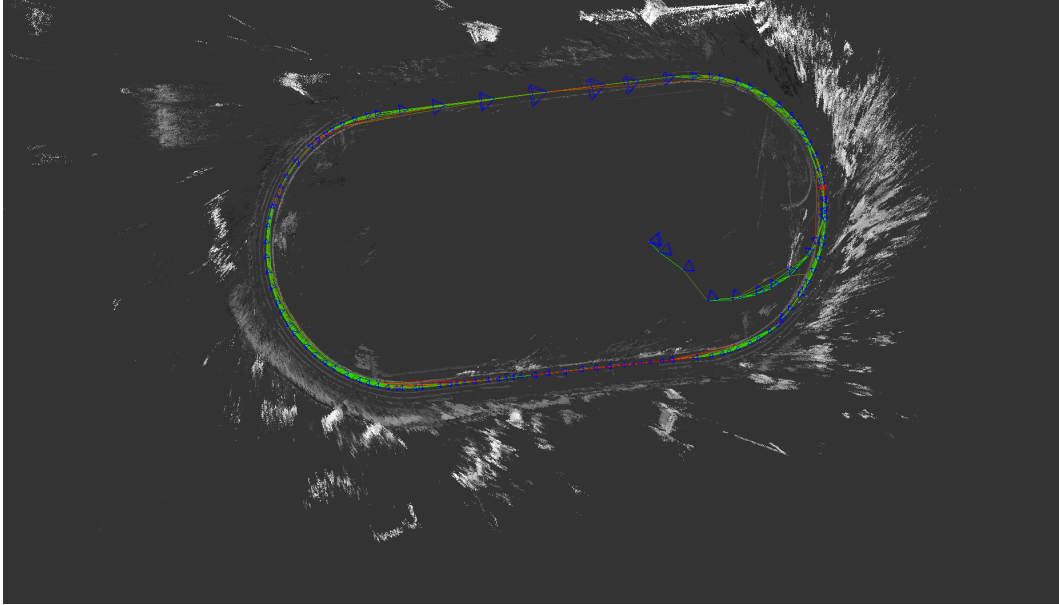
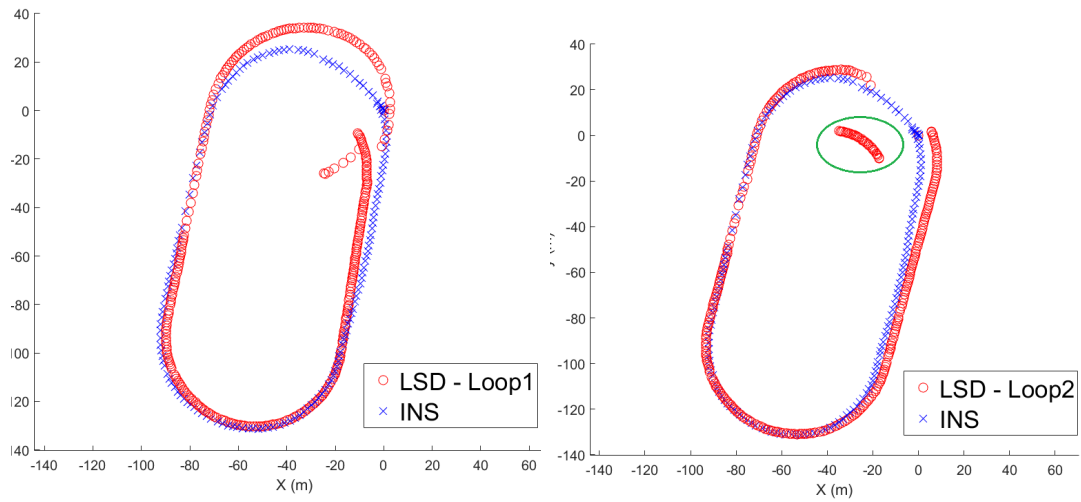


Figure 6.6: LSD SLAM map after looping the dataset twice



(a) LSD SLAM path estimate when the image sequence is looped once (before loop closure) aligned with INS measurements
 (b) The path estimated by LSD SLAM along the second loop of the image sequence aligned with the INS measurements

Figure 6.7: LSD SLAM path aligned with INS.

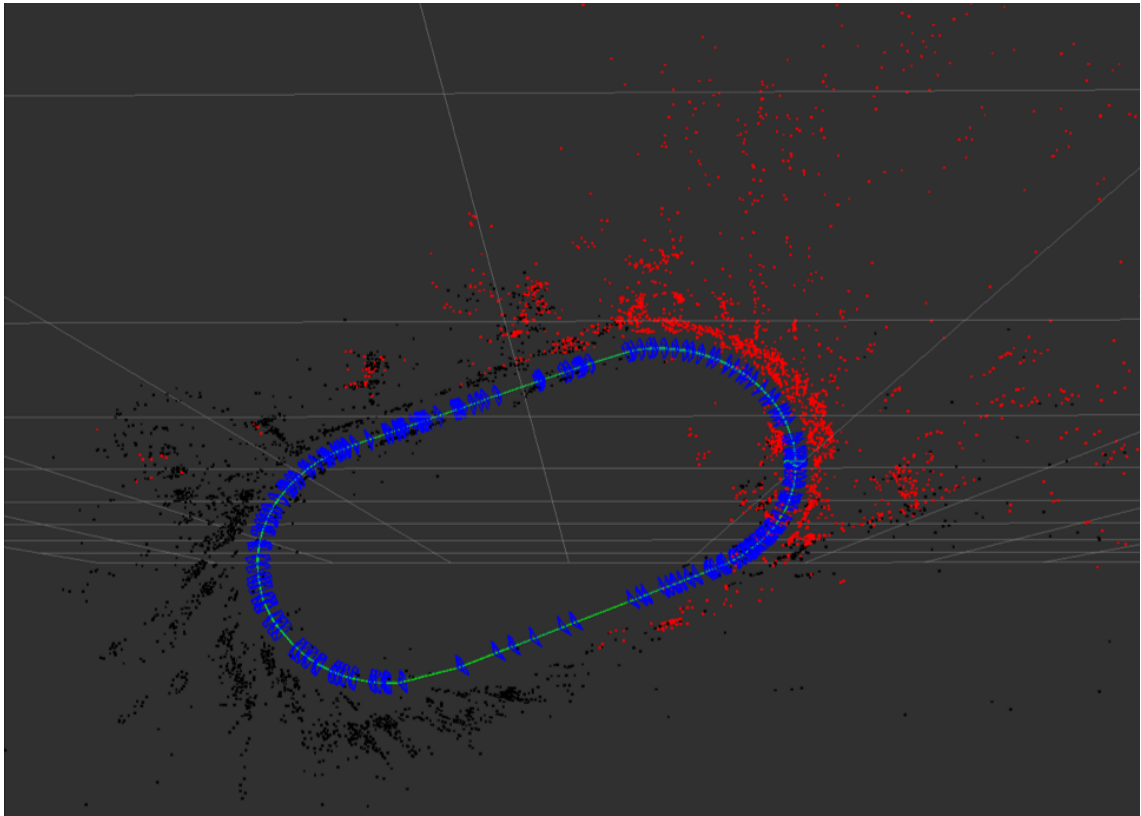
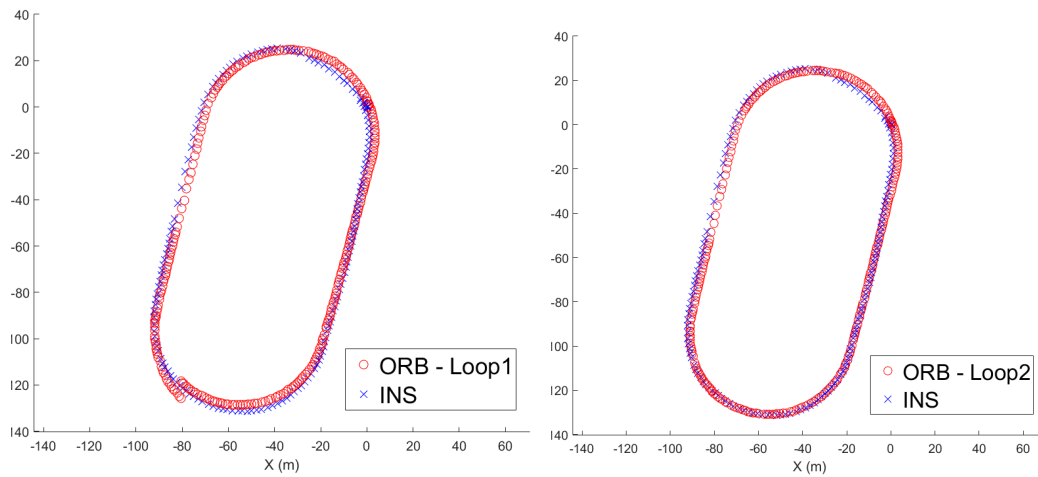


Figure 6.8: ORB SLAM map after looping the dataset twice



(a) ORB SLAM path estimate when the image sequence is looped once (before loop closure) aligned with INS measurements

(b) The path estimated by ORB SLAM along the second loop of the image sequence aligned with the INS measurements

Figure 6.9: ORB SLAM path aligned with INS.

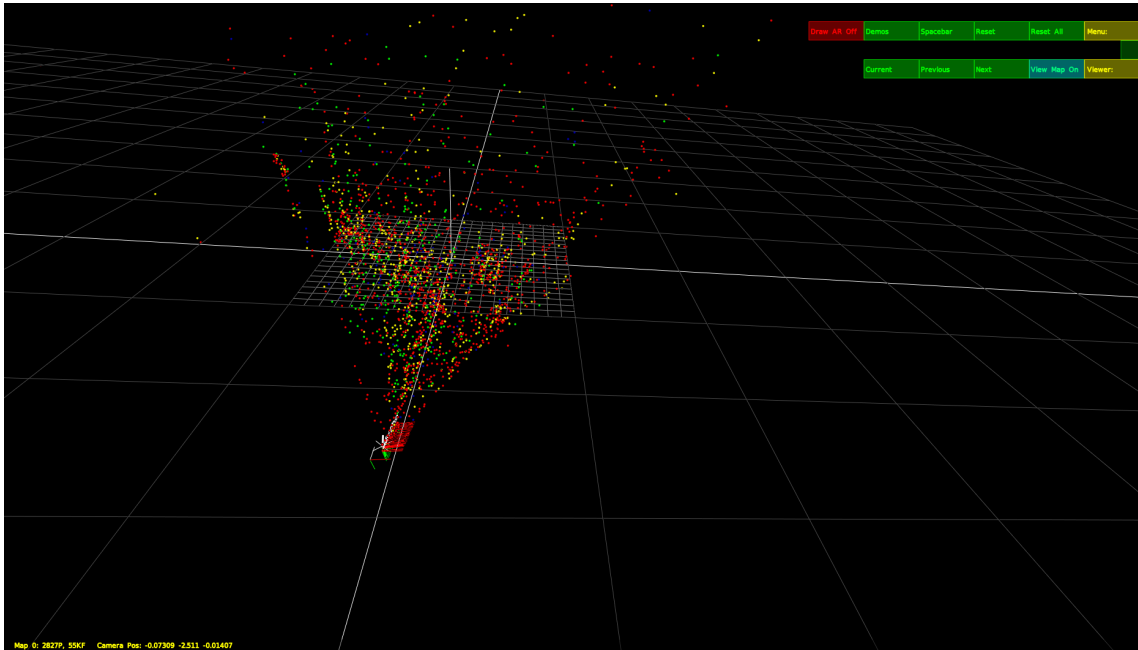


Figure 6.10: the effect of choosing a small baseline to depth ratio during the initialization procedure of PTAMM

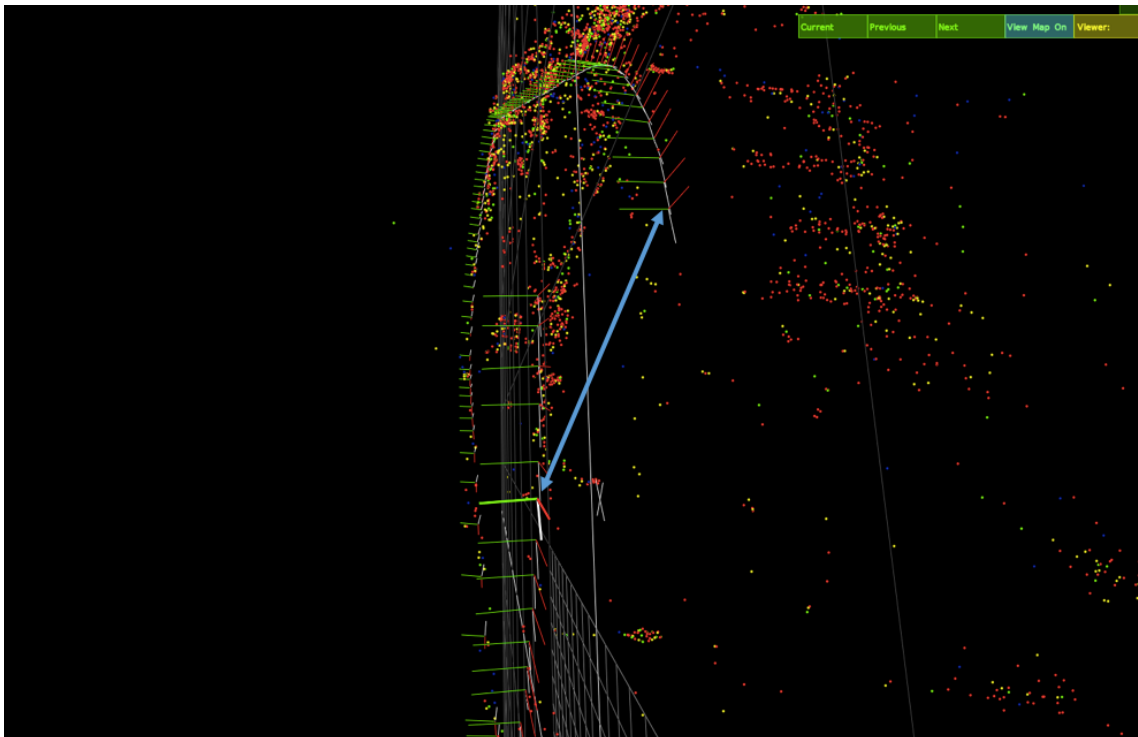


Figure 6.11: Discontinuity and tracking failure at loop closure

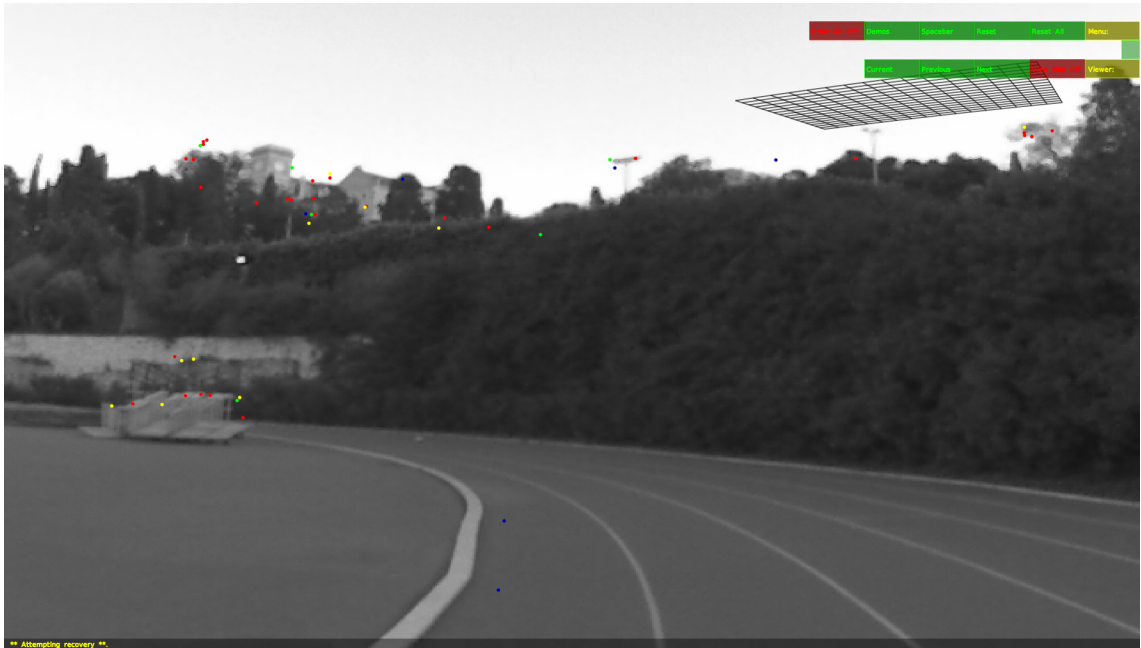


Figure 6.12: Lost camera tracking in PTAMM due to few image gradient in the scene

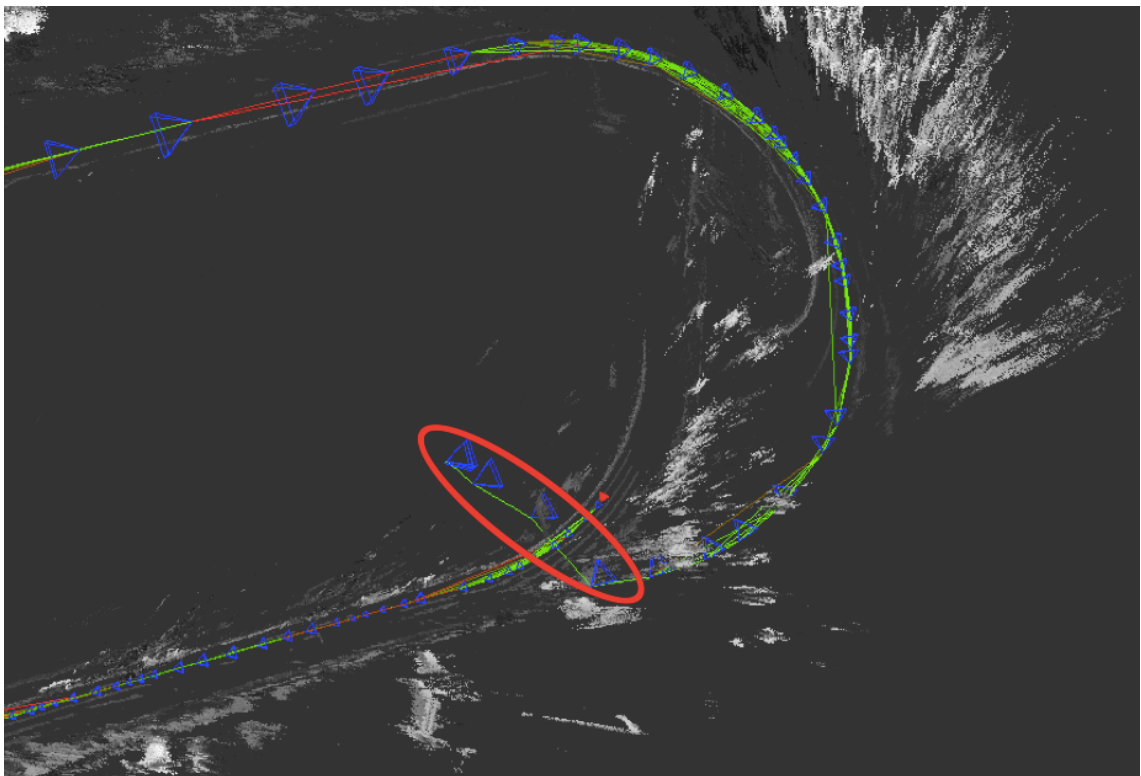


Figure 6.13: LSD SLAM random initialization effect

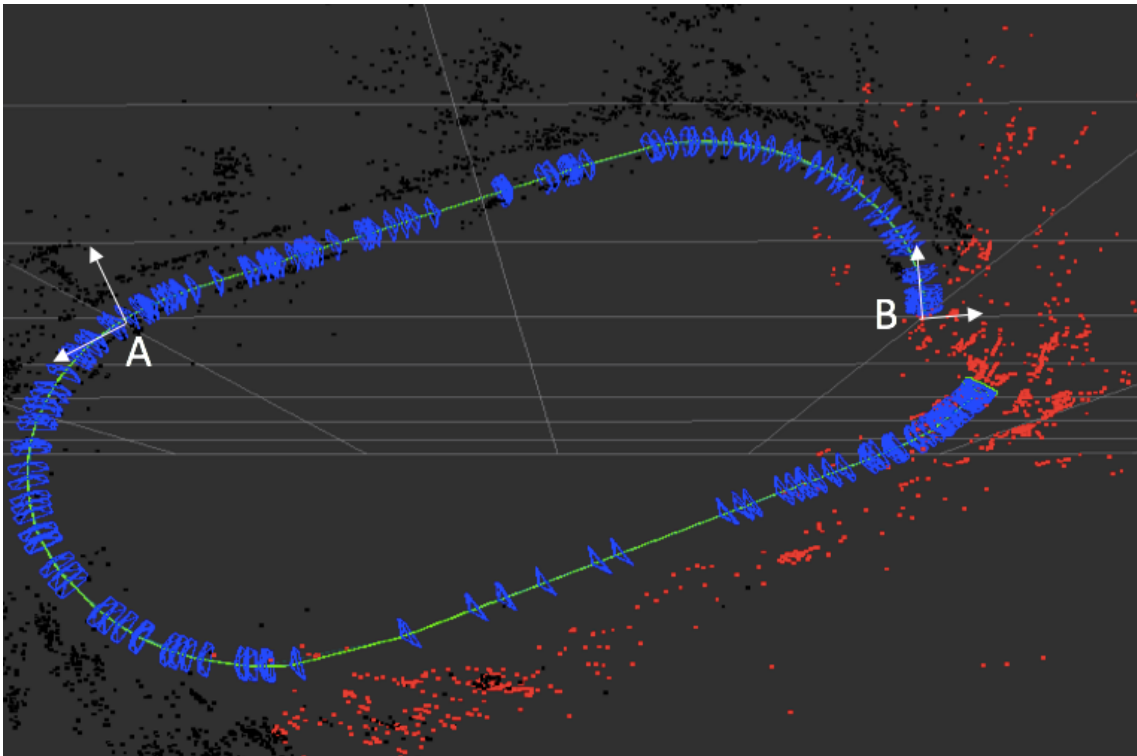


Figure 6.14: ORB SLAM map right before loop closure, point A represents the point at which the dataset sequence starts whereas point B is where the system was capable of initializing.

Chapter 7

Suggested improvements

7.1 A. Metric scale recovery in monocular SLAM

A fundamental shortcoming underlying monocular-based localization and mapping solutions (SfM, Visual SLAM) is the fact that the obtained maps and motion are solved up to an unknown scale. Yet, the literature provides interesting solutions to scale estimation using cues from focus or defocus of a camera. In this chapter, we take advantage of the scale offered by image focus to properly initialize Visual SLAM with a correct scale and show how it can be used to estimate and compensate for drift. We provide experiments showing the success of the proposed method and discuss its limitations. Our aim here is to diminish the initialization limitations discussed in chapter V, Section 2, by suggesting a novel initialization technique for Visual SLAM that requires no human input. The gist of the solution is to determine scale of a scene using depth from focus. More specifically, during initialization, the camera is moved normally to the scene in search of the image that is most focused. This is possible by performing an offline pre-calibration of the camera, where for a given camera focal distance, we determine the corresponding scene depth producing maximum image focus. Although the system is very sensitive to motion speed, motion rotation, and scene content, experiments demonstrate the success of the proposed technique. Another objective here is to reduce pose drift in Visual SLAM. For such purpose, an adaptation of the suggested initialization is employed, to provide depth information from focus; we show that the extra information can be used to accurately detect and compensate accumulated drifts within a SLAM session.

7.1.1 Depth from focus

While traditional visual SLAM concerns itself with depth estimation from parallax, research in optics suggest two methods capable of recovering depth from images that do not exhibit parallax; namely, depth from focus and depth from defocus.

Depth-from-Defocus (DfD) provides a solution to estimate the depth of a scene by measuring the blurriness of objects in an image. On the other hand, Depth-from-Focus (DfF) estimates depth by searching for the most focused scene points in a set of images that are captured while moving the camera. Figure 7.1 illustrates the effect of moving an object in front of the camera from the focal plane at distance df to a distance u . According to the well-known Gauss thin lens law:

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v}, \quad (7.1)$$

Here exists a one-to-one correspondence between an object distance u and its corresponding image distance v . Objects that are placed at the focal plane will result in maximum focus on the camera sensor. Any movement of the object away from the focal plane results in a blurred representation of the scene. What

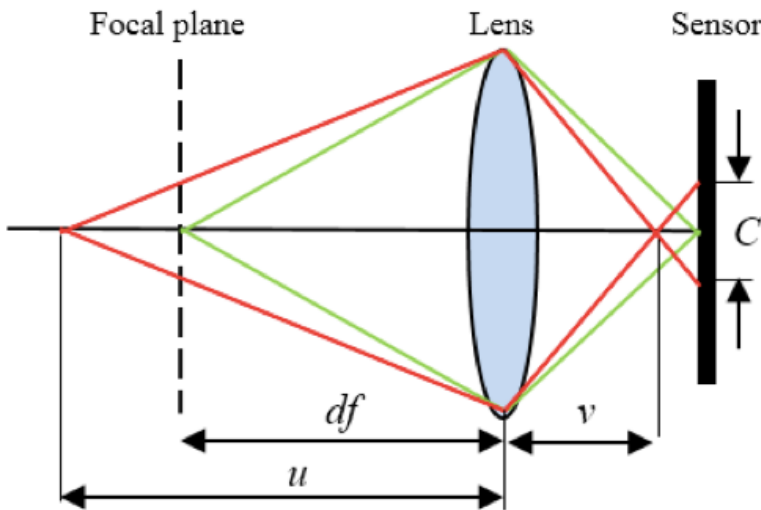


Figure 7.1: Image formation process through a gauss thin lens model.

is appealing from this theory is that that the depth of a scene in an image can be determined by measuring image focus (or defocus). Because DfD requires only one image to determine depth, whereas DfF requires several, DfD appears to be the most attractive alternative amongst the two to solve our objective. Unfortunately, DfD requires the ability to accurately measure the amount of blur in an image at different locations. While some work in the literature [63] suggests techniques to calculate this blur, our implementation of their techniques did not provide acceptable results.

As an alternative, DfF methods recover depth by searching for the state of the imaging system for which the object is in-focus in the image plane. This can be achieved by either varying the distance v between the lens and the sensor or by varying the distance from the lens to the object u . In the first technique,

for each scene the focus of camera is varied until the image is in focus. Pucihar and Coulton [64], provide such a solution, which requires an offline calibration resulting in a lookup table relating focus-to-depth. Allowing the camera to change its focus during a visual SLAM session is ruinous because the intrinsic camera calibration parameters, vital to achieve acceptable tracking performance, varies with the focus and would lead to tracking failure. Furthermore, not all cameras retain an auto-focusing mechanism nor allow access to their drives. Suwajanakorn et al. [65] recently suggested an algorithm capable of recovering depth from focus using an uncalibrated camera; however, their suggested pipeline requires twenty minutes of processing, which is intractable for real-time operations.

In this thesis, we adopt a different approach by moving the camera during initialization in the direction normal to the scene and estimate depth corresponding to the most in-focus image. For this method to succeed, an appropriate focus measure operator is a critical in ensuring accurate depth estimation. A wide variety of algorithms [66] have been used to measure the degree of focus of image patches or the image as a whole. Operators depending on the derivatives like the Gradient and Laplacian-based operators have been used to estimate sharpness in an image by measuring the density of edges. Wavelet [67] and direction cosine-based [68] operators measure the frequency components of the image as a basis for extrapolating focus. Statistical-based operators have also been used taking advantage of statistical measure to compute the focus level.

Given their real-time requirements, Visual SLAM implementations pose constraints on the amount of allowable processing time for each frame, Therefore, a simple, fast, and relatively accurate focus operator is used, consisting of first extracting the Laplacian:

$$\nabla^2 I(x, y) = \frac{\partial^2 I(x, y)}{\partial x^2} + \frac{\partial^2 I(x, y)}{\partial y^2}, \quad (7.2)$$

and then summing the Laplacian over a window; a step necessary to help deal with poorly-textured surfaces:

$$FM(x_0, y_0) = \sum_{(x,y) \in \Omega(x_0, y_0)} \nabla^2 I(x, y), \quad (7.3)$$

where $\Omega(x_0, y_0)$ is the support window chosen as the 24×24 pixel patch centered at $I(x_0, y_0)$. The total focus measure of the image is then found as:

$$F = \frac{1}{n} \sum_{(x,y) \in I} FM(x, y)^2, \quad (7.4)$$

where n is the number of pixels in the image. While the theory for finding depth from focus is straightforward, during implementation it was found to be extremely sensitive to different parameters, which will be discussed in detail in the following sections.

7.1.2 Depth from focus calibration

Falling back on the image formation process in Figure 7.1, it is deducible that a camera moving longitudinally along a path toward a planar scene exhibits maximum focus when the camera is positioned at the focal plane. Figure 7.2 illustrates how the focus measure of an image is expected to vary with the distance from the scene. What is sought is an algorithm that is capable of determining,

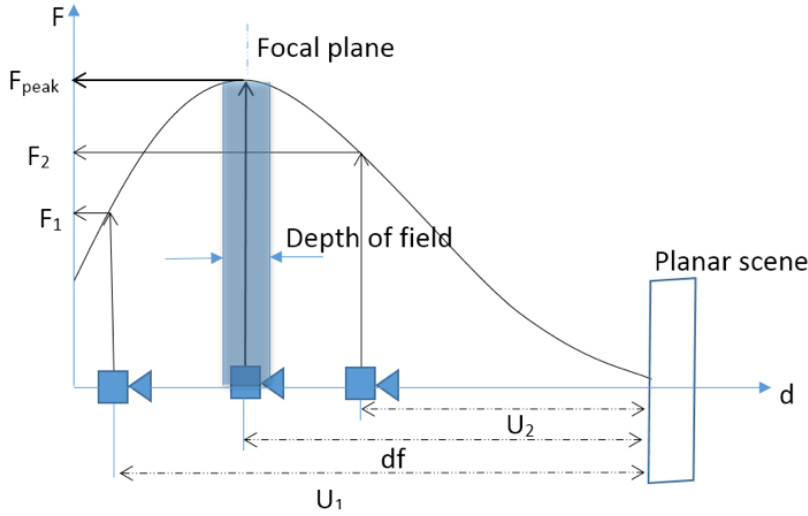


Figure 7.2: Focus measure profile vs change in distance between camera and scene.

among a range of images, the image exhibiting maximum focus F_{peak} for a given scene. To make this possible, before initializing Visual SLAM, the system would have to be calibrated in order to determine, for a given focal length, the depth corresponding to maximum focus.

We propose a very simple calibration process that determines, for fixed intrinsic parameters, the distance separating the camera from a planar scene for which the scene appears to be sharp and in focus in the image plane. To quantify "focus" in the image, we employ the "Energy of the Laplacian" focus measure operator described in eq. 7.4.

The calibration procedure then takes place by fixing either the camera or the planar scene and moving the other longitudinally away, while monitoring the focus measure response. When a peak in the focus measure response is recorded, the corresponding distance from the camera is registered as the focal plane's distance. Figure 7.3 shows the actual focus measure recorded during one calibration step, which reveals the camera's focal plane at a distance of 23 cm from the lens.

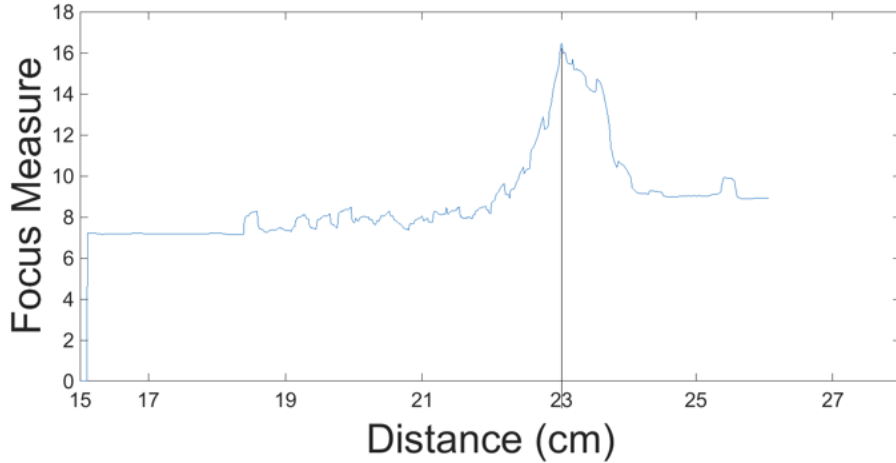


Figure 7.3: Example DfF calibration of a camera. Note that in this case the focal plane is determined at 23 cm from the lens.

7.1.3 Depth from focus in Visual SLAM

Once the focal plane’s distance is known through the calibration process, it can be used to initialize any Visual SLAM system, assuming the observed scene during initialization is planar as is the case in most Visual SLAM implementations. In this thesis, we test the technique on PTAM as a replacement for its default initialization using Homographies.

Once the system is started, tracking and mapping are set to an idle state, while the user is asked to move forward and backward towards a planar scene. The focus measure is then recorded automatically at every frame and the one corresponding to the peak focus measurement is registered as the first keyframe in the map. The initialization here is considerably different from the traditional methods of Visual SLAM, in which the user is required to trigger the system, move the camera a distance that is ad-hoc, and then trigger the system again once sufficient parallax is achieved. In what we are proposing, no human intervention is required except for the initial trigger and then the camera is moved tangent to its optical axis until the system automatically initializes. This type of motion is more natural than a lateral one, especially for mobile platforms such as Unmanned Aerial Vehicles (UAV) with a downward looking camera, or for nonholonomic land vehicles equipped with forward looking cameras.

During initialization, the pose of each keyframe is represented as a rigid body transformation $\in SE(3)$. Initially, the pose $E_{1,w}$ belonging to the first KF is assigned the 4×4 identity matrix. FAST features [13] from the 0^{th} pyramid level are then extracted and their 3D coordinates are initialized as:

$$\begin{bmatrix} X' & Y' & Z' & 1 \end{bmatrix} = E_{1,w} \begin{bmatrix} \frac{P_x}{D} & \frac{P_y}{D} & \frac{P_z}{D} & 1 \end{bmatrix}^T, \quad (7.5)$$

where $D = \frac{1}{focalplannedistance}$, hardcoded into the system beforehand through the camera calibration process. P_x and P_y are pixel coordinates of the extracted features projected onto a normalized image plane using the radial distortion model of [69].

Similar to PTAM, the mean of the 3D features is then elected to serve as the world coordinate frame and the entire map is then transformed accordingly. Once the initialization procedure is complete, the visual SLAM system resumes its regular tasks, performing camera tracking and scene mapping. In addition to using DfF for the initialization phase of Vision SLAM, it can also be used to correct drift in the map and pose throughout the SLAM session.

7.1.4 Pose drift correction

During the map exploration phase of a generic Visual SLAM session, new 3D landmarks are triangulated based on the tracker’s camera pose estimates. After some time, the error in the camera pose estimates accumulates and propagates throughout the map because the erroneous camera poses are used to triangulate new 3D landmarks; this is known as drift.

Our proposed system makes use of the extra information provided by the focus measure to correct drifts under special conditions. In contrast to the initialization procedure, where both the SLAM map and the camera’s pose are not available, during regular operations both are explicitly present and may be exploited to our advantage. The camera pose estimate is continuously monitored, and if the camera is found, at a distance near its focal plane, from a planar scene, the focus measure is monitored as the camera approaches the scene.

Figure 7.4 illustrates, the configuration required for the system to account for drift. Here, the observed scene is theorized to be planar and the camera is expected to be directed and moving along the normal to the scene. To ensure that these conditions are met, the 3D landmarks of the observed scene are projected onto every incoming frame. The mean and standard deviation of their coordinates expressed in the camera frame along the z -direction are recorded. For the above conditions to be satisfied, the recorded standard deviation should be 0; if that’s the case, the distance from the camera to the scene is then monitored until the camera goes from one side of the band into the other, where the band is defined by the focal plane’s distance $\pm \frac{\delta}{2}$ and δ is a user defined parameter proportional to the depth of field of the lens in use. For our system we empirically select a δ of 10 cm. The depth distribution of the features in the frame at which the peak focus measure was found is then compared to the focal plane distance in order to estimate the drift.

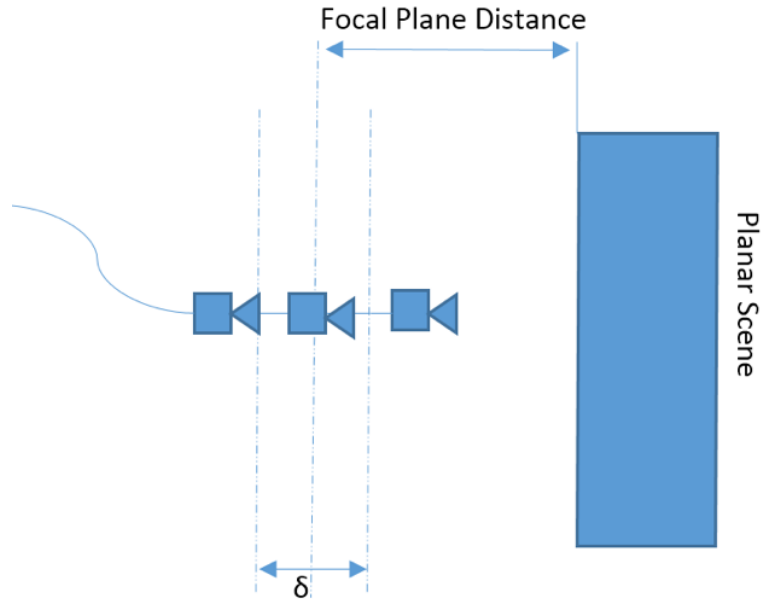


Figure 7.4: Conditions under which pose drift correction takes place

7.1.5 Implementation considerations

Although in theory, the steps for implementing DfF in the initialization of SLAM are relatively straightforward, additional care must be taken during implementation as will be discussed next.

To ensure few outliers and good features to track in the map, extracted fast features from the first keyframe are sorted according to their Shi-Tomasi score, and only the strongest 400 features are used in the map initialization phase. Once the first keyframe is successfully inserted into map, tracking starts using features corresponding only at 0th pyramid level, rendering tracking quality fragile. To alleviate this drawback, the criteria tested for keyframe insertion are relaxed so that the second keyframe is quickly inserted in the map as soon as enough parallax is observed, thereby ensuring the population of features at different pyramid levels. A moving average filter is employed to attenuate the effect of noisy measurements in the focus measure operator. To account for various noise in the system, originating from outlier landmarks, the standard deviation constraint in the drift correction module is relaxed to accommodate scenes with standard deviations of up to 2 cm.

To ensure quality of the focus measurements during drift correction, the recorded focus measures are not used if:

- The camera spends too much time within the band of interest.
- The camera exits from the same side of the band.

- The camera’s heading deviates from the scene’s normal. . . .

7.1.6 Experiments and results

Our proposed method was implemented in PTAM and tested on a laptop with an Intel Core *i7 – 4710HQ* 2.5GHZ CPU, 16 GB memory; no GPU acceleration was used.

As table 7.1 shows, the computational cost for each focus measurement required at every frame is 5.4 ms; once the focused frame is found, our proposed initialization requires 8.5 ms to kick-start the system, in contrast to the default Homography initialization of PTAM that requires on average 250 ms. Furthermore, our proposed initialization does not yield multiple solutions in contrast to the Homography estimation that in some cases may be degenerate or return ambiguous results. To test our system, three experiments were conducted. Dur-

Table 7.1: Computational cost for initialization

Operation	Time(ms)
Focus measurement	5.4
Our initialization module	8.5
Homography initialization	250

ing our experiments, motion was only in the vertical direction, therefore ground truth was collected manually, by measuring the actual distance between the lens and the scene.

Experiment 1

Experiment 1 consisted of fixing the camera on a rig that can move in a single direction normal to a planar scene. This was necessary to validate the theory and test its application to PTAM in a controlled environment. The camera was focused beforehand at a distance of 23 cm. Within the controlled confines of a fixed rig, the experiment was repeated 31 times, either moving towards the planar scene from a starting position of 30 cm or moving away from the scene with a starting position of 10 cm, at a constant rate. The actual distance between the camera and the scene, at which our proposed method initialized the system, was then recorded.

The obtained results are shown in Figure 7.5 as circles; with a mean of 22.93 cm and a standard deviation of 0.2 cm, they demonstrate the accuracy and precision of our initialization module. The same experiment was then repeated but this time the hand-held camera was free to move in 6D. Nevertheless, the user was asked to avoid high acceleration movements and tilting as much as possible. In this experiment, the objective was to study the impact of factors such as

camera orientation changes and motion blur, induced by human interference, on the initialization quality of PTAM. The obtained results, shown in Figure 7.5 as

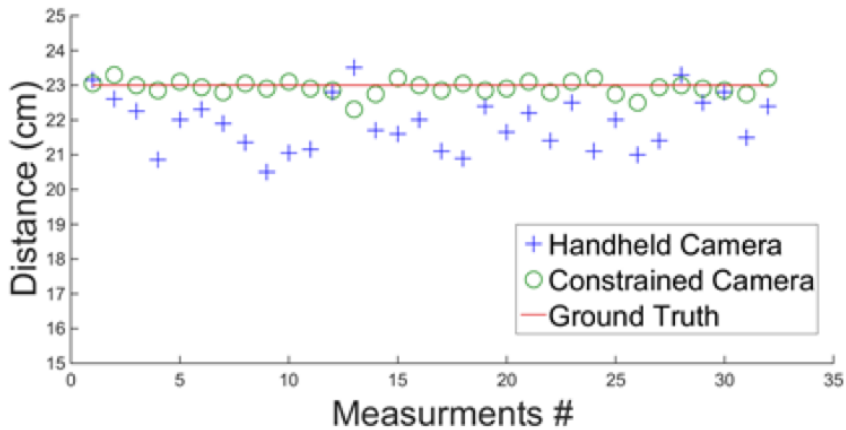


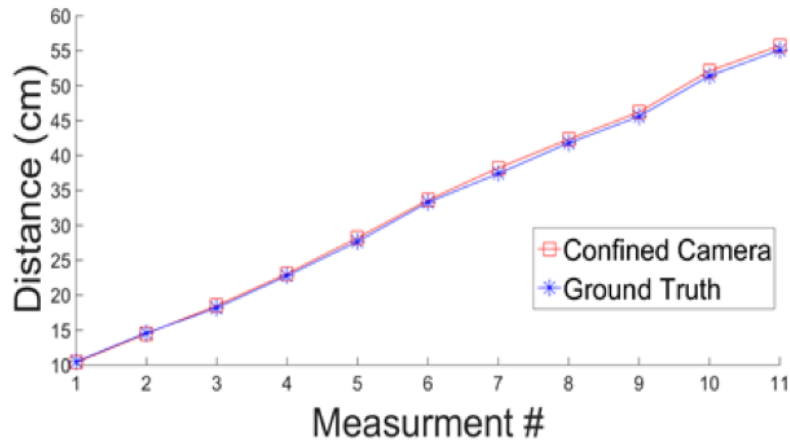
Figure 7.5: Experiment 1: System initialization

'+' yielded a mean of 21.9 cm and a standard deviation of 0.8 cm in contrast to the actual focal plane's distance of 23 cm. The decrease in precision and accuracy of the initialization are accredited to human induced errors such as positioning of the camera at a tilted angle during the initialization procedure or by moving too fast, inducing motion blur into the image frames therefore affecting the focus measurements.

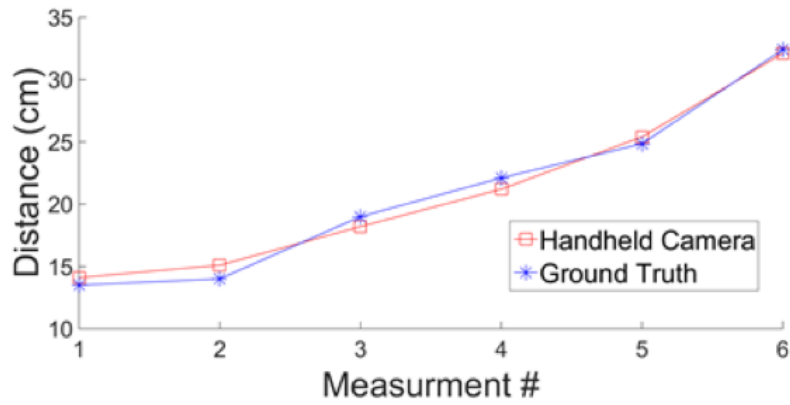
Experiment 2

The second experiment consists of initializing PTAM using our method and then recording its camera pose measurements and compare them to the ground truth for both the fixed rig and hand-held cases. While this experiment can be easily performed in 6D, for visualization purposes, it was conducted along a single dimension. After initialization, the scene was explored, by inserting keyframes with enough parallax between them, to ensure a good baseline for feature triangulation to take place, before returning to the experiment's configuration of motion along a single direction.

The recorded paths are shown in Figures 7.6(a) and 7.6(b). They show that our initialization module was able to snap to the actual scene's scale by yielding pose estimates with an RMSE of 0.49 cm in the confined camera case and 0.62 cm in the hand held version of the experiment. To contrast our obtained results, the same experiment was repeated, only this time using the default initialization procedure of PTAM. Figure 7.7 contrasts the scale differences between the path estimate extracted from PTAM and the ground truth. It is noteworthy to mention that, if the experiment were to be repeated, using a different baseline between the initializing keyframes, the scale would have been significantly different since



(a) Camera fixed on a rig



(b) Hand-held camera

Figure 7.6: Reported paths after initializing using our proposed method

the estimated scale is inversely proportional to the baseline to depth ratio.

Experiment 3

For the sake of testing our system in correcting drift, the following experiment took place within a single SLAM session. First PTAM was initialized through our proposed method and the camera was moved along the rig while the camera's path was recorded. A case of drift was then simulated by shifting PTAM's map by 5 cm and the trajectory was recorded again. Our drift detection routine was then initiated, as the camera spanned the trajectory of the rig. Once our system detected and corrected the drift, the camera's path was recorded again. Our module detected a drift of 5.1 cm and corrected the map accordingly. Figure 7.8

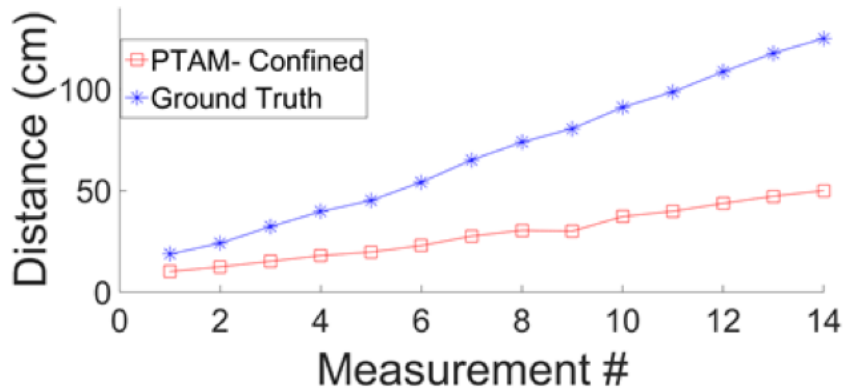


Figure 7.7: Path recorded by PTAM initialized using its default method shows the recorded paths and Table 7.2 reports the RMSE values for each path.

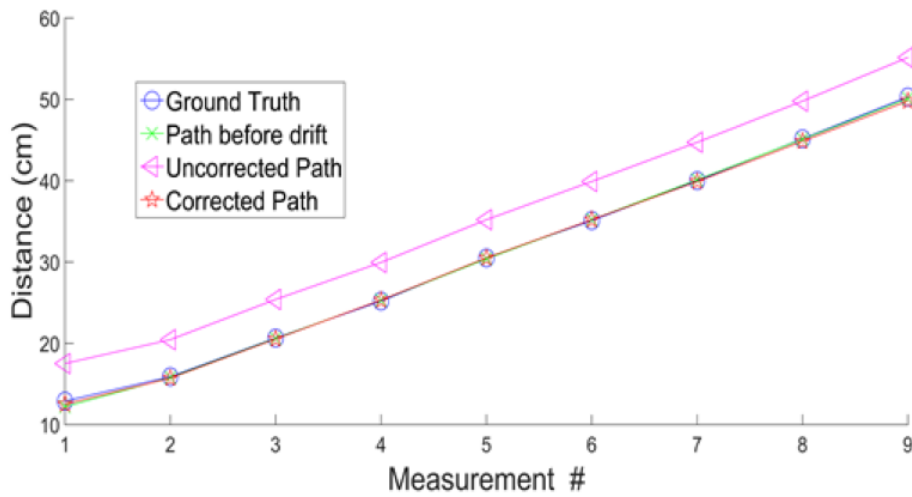


Figure 7.8: Paths recorded during the drift experiment

Table 7.2: RMSE of paths measured in experiment 3

Path	RMSE(cm)
Before Drift	0.255
After drift uncorrected	4.7086
After drift corrected	0.2502

7.1.7 Discussion

The accuracy and repeatability reported in Experiment 1 proves the viability of the suggested initialization under the constraints, where the camera moves normally to the planar scene; furthermore, our method demonstrated its strength in drift detection and correction through Experiment 3, where the system was able to fully recover from the forcibly induced drift under the proper constraints. However, as the constraints are relaxed through the hand-held version of Experiment 1, where the camera's motion is subject to human interference, the decrease in accuracy and repeatability can be traced back to several factors namely, (1) motion blur caused by jittering and fast motions of the camera, (2) rotation deviations from the normal of the scene due to camera handling errors.

Figure 7.9 illustrates the impact of motion blur and noise on the focus measure; it emphasizes the effect of subjecting the camera to small abrupt motions, inducing false peaks in the focus measurements. To reduce the effects of jittering

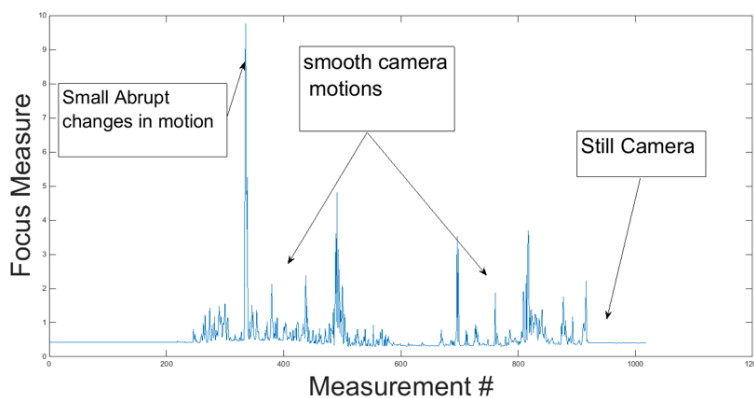


Figure 7.9: Impact of jittering and motion blur on Focus measure.

and motion blur on the system, a down sampled representation of the image may be used to estimate the focus measure, at the expense of decreasing its sensitivity.

Figure 7.10 illustrates the repercussions of violating the motion along the normal to the plane constraint. As the camera is tilted at an angle with the normal to the plane, the reported distance at which the peak focus measure would fall shorter than the actual focal plane distance, which explains why the mean of the hand held version of Experiment 1 was shifted below the actual value of the focal plane. Furthermore, as the camera translates, its field of view increases/decreases; as such, many objects come into and out of the image, interfering with the globally estimated focus measure and leading to difficulties in estimating the correct focus measure peak.

Future venues of this work include an automatic region of interest selection and tracking criteria to perform selective focus measurements over subsets of the

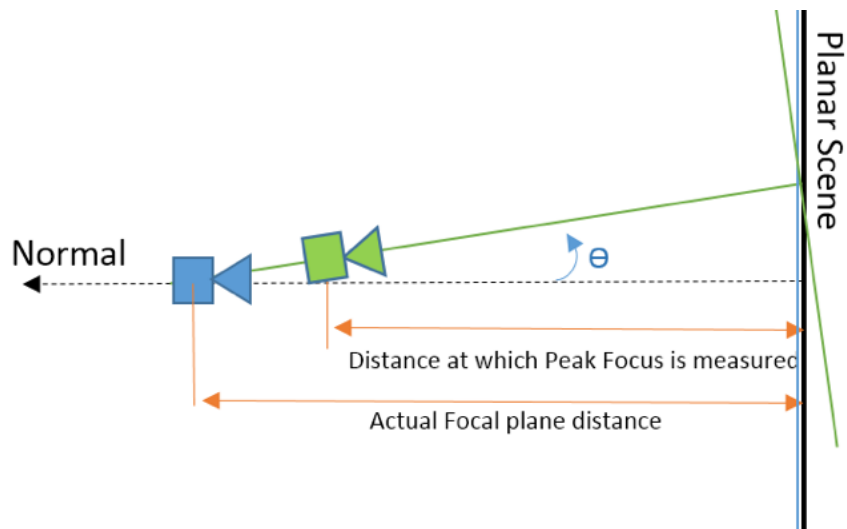


Figure 7.10: Impact of jittering and motion blur on Focus measure.

images, reducing the impact of increasing/decreasing field of view on the focus measure operator. Furthermore, it is interesting to test if obtaining few focus measures at their associated regions of interest may be extrapolated to yield accurate depth estimates without the requirement of the camera's focal plane to sweep through them and hence relaxing the initial scene's planarity constraint.

7.2 A suggested visual SLAM pipeline

During the course of this work we outlined, detailed and discussed state of the art in monocular SLAM. We laid out the fortes and the flaws of various systems and at this point, a combination of all the different modules, presented by various systems, can be put together in an effort to establish a new, superior Visual Slam algorithm. An ideal monocular Visual SLAM is expected to be immune to lighting changes and dynamic scenes while accurately providing pose estimates that does not drift over time along with a reliable map representation that is both computational and memory efficient to handle large-scale environments.

7.2.1 Data type

When choosing the input data type, one must bare in mind the computational efficiency of indirect methods vs. the systems robustness when subjected to low gradient scenes. For this purpose, we suggest an alternating direct and indirect data types similar to SVOs implementation.

7.2.2 Initialization

Our proposed DfF initialization in the previous section would constitute a viable solution to kick start a visual SLAM system; however, it requires a planar scene and a camera motion along the scenes normal; one could argue that these requirements have important degenerative ramifications when violated and the method require more research for it to become a reliable initialization procedure. In the shadow of the stated argument and till our DfF initialization matures, a dual initialization procedure that estimates both a fundamental and a Homography, similar to the ORB SLAM's implementation is suggested, where the system does not suffer from degeneracies when subjected to planar or non planar scenes and does not initialize a non-reliable map that requires dozens of frames before converging.

7.2.3 Data association

Data association methods using feature descriptors are somehow robust against rotations and illumination changes, however they suffer when the scene changes considerably, leading to erroneous camera pose estimates hence failure. To address these modes of failure, the data association mechanism suggested in ORB SLAM (BoW matching of high order descriptors) is suggested to be used, where a features descriptor is updated to the latest descriptor it was successfully matched, in conjunction with the sampling mechanism employed in RD SLAM, based on prior outlier measurements, to flag features that belong to dynamic objects.

7.2.4 Pose estimation

While DT SLAMs tracker offer interesting properties, we have no proof that they constitute a reliable camera tracking mechanism; whenever tracking is lost, the system re-initialized a new map and tracking continued in the newly spawned map so that no data is lost and no failure recovery methods are required; however, as the tracking module failed plenty of times, the result was dozens of meaningless sub-maps with no coherent camera pose estimates in the process. For such purpose, we propose a variant of DT SLAM's tracker, that incorporates both 2D and 3D landmarks in a unified framework to estimate the camera pose, however re-initialization of a new map is deferred until a re-localization mechanism fails after a significant number of attempts. Furthermore, to overcome the limitations of tracking methods that employs alternating pose estimate schemes (direct and indirect), we suggest to separate the two methods, resorting to direct methods when few indirect features are observable.

7.2.5 Map generation

To ensure a minimum number of outliers in the map as well as spawning enough landmarks to maintain tracking quality, the map generation module is suggested to have a lean appearance change based keyframe selection criteria where new 3D landmarks in the map are triangulated if and only if the 2D feature was successfully observed in at least 3 keyframes that exhibit a significant baseline between them.

Furthermore, if the operator desires, an option for a dense map representation could be enabled through an independent parallel thread that recovers a dense representation of the map based on the camera pose estimates anchored by the sparse, feature based tracking module similar to [70].

7.2.6 Map maintenance

Similar to ORB SLAM, a rigorous map point and KF culling criteria is recommended to ensure the health of the map, coupled with an appearance change detection mechanism, similar to the one employed in RD SLAM, that updates keyframes corresponding to significantly changed regions; allowing the system to gain robustness against slowly varying scenes.

To handle sub-maps, the map maintenance module should be equipped, similar to DT SLAM, with a mechanism that allows it to fuse multiple, separate maps into one when enough correspondences are established between keyframes residing in different maps.

7.2.7 Failure recovery

For a reliable failure recovery procedure, a combination of ORB & LSD SLAM is desired: upon tracking failure, the system attempts to re-localize using all keyframes connected to the last accurate camera pose estimate through the co-visibility graph. If it fails, the entire map is queried, similar to ORB SLAM, to generate multiple recovery candidates throughout the map. If the system fails to recovery after the above procedure was repeated many times, the system re-initializes a new map and tracking resumes in the new map until enough correspondences between different sub-maps are established through the map maintenance module and the sub maps are merged into one in a scheme similar to DT SLAM's.

7.2.8 loop closure

A robust loop closure mechanism is mandatory for the system to correct for drifts accumulated during large scale operations. For such purpose an implementation

similar to the one presented in ORB SLAM is suggested to complete the suggested system.

Chapter 8

Conclusion

Throughout this thesis, we have outlined the fundamental building blocks of a generic monocular visual SLAM system, thoroughly analyzed the state of the art in monocular visual SLAM implementations uncovering what gives each system its individualities, before comparatively assessing the performance of ORB SLAM, PTAM and LSD SLAM using our generated dataset dubbed AUB dataset. Our analysis of all available systems unveiled the skeleton of a new monocular visual SLAM system that combines the added value of previously released systems while diminishing their shortcomings. Furthermore, we have suggested a novel Visual SLAM initialization procedure capable of recovering an accurate metric scale and proved the viability of the method under the proper constraints. We have also showcased the implementation of PTAM, a visual SLAM system, in an augmented reality application for outdoor environments.

Future venues and continuation of this work include the development of our suggested depth from focus initialization method to overcome the limitations outlined in chapter VII as well as the implementation of the suggested visual SLAM system and finding new approaches to address lighting impact on Visual SLAM system.

Visual SLAM methods have noticeably advanced during recent years; nevertheless, one major problem that needs to be addressed by the Visual SLAM community is that of changing lighting conditions. The inputs, upon which Visual SLAM heavily rely on, whether direct or indirect, becomes obsolete as lighting conditions vary, causing failure due to their inability to establish correspondences with the map.

Appendix A

Abbreviations

FAIA	Forward additive image alignment
FCIA	Forward compositional image alignment
ICIA	Inverse compositional image alignment
IAIA	Inverse additive image alignment
SLAM	Simultaneous localization and mapping
VSLAM	Visual SLAM
SfM	Structure from Motion
PTAM	Parallel tracking and mapping
PTAMM	Parallel tracking and multiple mapping
RD SLAM	Robust Dynamic SLAM
SVO	Semi-direct visual odometry
MAV	Micro aerial vehicle
LSD	Large scale direct monocular slam
DT SLAM	Deferred triangulation SLAM
ORB	oriented FAST and rotated binary robust independent elementary features
VO	Visual odometry
FAST	Feature from accelerated segment test
GPU	Graphics processing unit
SIFT	Scale invariant feature transform
ZMSSD	Zero mean sum of squared difference
MLESAC	Maximum likelihood estimation sample consensus
BA	Bundle adjustment
KLT	Lucas-kanade tracker
KF	Keyframe
RANSAC	Random sample consensus
SSD	Sum of squared difference
SE	Special Euclidean group
SO	Special rotation group
Sim	Similarity transform
BoW	Bags of Words

K-d tree	K-dimensional tree
CPU	Central processing unit
SBI	Small blurry image
ESM	Efficient second order minimization
LBA	Local bundle adjustment
GBA	Global bundle adjustment
PGO	Pose graph optimization
AR	Augmented reality
VR	Virtual reality
RMSE	Root mean squared error
DfF	Depth from focus
DfD	Depth from Defocus
INS	Inertial navigation system
UTM	Universal Transverse Mercator
GPS	Global positioning system
AGAST	Adaptive and generic accelerated segment test
ABO	Array buffer object
Fps	Frames per second
RAM	Random access memory
HRM	Highly reliable markers
HD	Homography decomposition
ED	Essential decomposition
RD	Random depth
PEM	Photometric error minimization
Ialck	Image alignment with last correctly tracked frame
Iark	Image alignment with random keyframe
Bwpr	Bags of words place recognition
Cvmm	Constant velocity motion model
Sapp	Same as previous pose
Stpf	Similarity transform with previous frame
Omf	Optimization through minimization of features
Ompe	Optimization through photometric error
Pre	Pure rotation estimation
Spc	Significant pose change
Ssac	Significant scene appearance change
2vt	2 view triangulation
2mvt	2 or more view triangulation
Pf	Particle filter
2Dlt	2D landmarks triangulated to 3D landmarks
Pfpf	Depth map propagation from previous frame
Sbo	Small baseline observation
POV	Point of view

Bibliography

- [1] A. Davison, “Real-time simultaneous localisation and mapping with a single camera,” in *Ninth IEEE International Conference on Computer Vision*, pp. 1403–1410 vol.2, 2003.
- [2] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, “MonoSLAM: real-time single camera SLAM,” *Pattern Analysis and Machine Intelligence (PAMI), IEEE Transactions on*, vol. 29, pp. 1052–67, June 2007.
- [3] G. Klein and D. Murray, “Parallel Tracking and Mapping for Small AR Workspaces,” *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 1–10, November 2007.
- [4] C. Forster, M. Pizzoli, and D. Scaramuzza, “SVO : Fast Semi-Direct Monocular Visual Odometry,” in *Robotics and Automation (ICRA), IEEE International Conference on*, 2014.
- [5] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-Scale Direct Monocular SLAM,” in *Computer Vision – ECCV 2014 SE - 54* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), vol. 8690 of *Lecture Notes in Computer Science*, pp. 834–849, Springer International Publishing, 2014.
- [6] D. Herrera, J. Kannala, K. Pulli, and J. Heikkila, “DT-SLAM: Deferred Triangulation for Robust SLAM,” in *3D Vision, 2nd International Conference on*, vol. 1, pp. 609–616, IEEE, December 2014.
- [7] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “ORB-SLAM: A Versatile and Accurate Monocular SLAM System,” *IEEE Transactions on Robotics*, vol. PP, no. 99, pp. 1–17, 2015.
- [8] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao, “Robust monocular SLAM in dynamic environments,” *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 209–218, 2013.
- [9] K. Pirker, M. Rütger, and H. Bischof, “CD SLAM - Continuous localization and mapping in a dynamic world.,” in *IEEE International Conference on Intelligent Robots Systems (IROS)*, pp. 3990–3997, IEEE, 2011.

- [10] H. Lim, J. Lim, and H. J. Kim, “Real-time 6-DOF monocular visual SLAM in a large-scale environment,” in *Robotics and Automation (ICRA), IEEE International Conference on*, pp. 1532–1539, May 2014.
- [11] R. O. Castle, G. Klein, and D. W. Murray, “Video-rate Localization in Multiple Maps for Wearable Augmented Reality,” in *Proc 12th {IEEE} Int Symp on Wearable Computers*, pp. 15–22, 2008.
- [12] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, “Automatic generation and detection of highly reliable fiducial markers under occlusion,” *Pattern Recognition*, vol. 47, no. 6, pp. 2280 – 2292, 2014.
- [13] E. Rosten and T. Drummond, “Machine Learning for High-speed Corner Detection,” in *9th European Conference on Computer Vision - Volume Part I, Proceedings of the, ECCV’06*, (Berlin, Heidelberg), pp. 430–443, Springer-Verlag, 2006.
- [14] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, “Adaptive and Generic Corner Detection Based on the Accelerated Segment Test,” in *Proceedings of the European Conference on Computer Vision (ECCV’10)*, sep 2010.
- [15] H. Strasdat, J. M. M. Montiel, and A. J. Davison, “Real-time monocular SLAM: Why filter?,” in *Robotics and Automation (ICRA), IEEE International Conference on*, pp. 2657–2664, May 2010.
- [16] G. Klein and D. Murray, “Improving the Agility of Keyframe-Based {SLAM},” in *Proc. 10th European Conference on Computer Vision (ECCV)*, (Marseille), pp. 802–815, October 2008.
- [17] J. Engel, J. Sturm, and D. Cremers, “Semi-dense Visual Odometry for a Monocular Camera,” in *Computer Vision (ICCV), IEEE International Conference on*, pp. 1449–1456, IEEE, dec 2013.
- [18] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “G2o: A general framework for graph optimization,” in *Robotics and Automation (ICRA), IEEE International Conference on*, pp. 3607–3613, IEEE, May 2011.
- [19] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” in *International Conference on Computer Vision (ICCV)*, pp. 2564–2571, November 2011.
- [20] P. Torr, A. Fitzgibbon, and A. Zisserman, “The Problem of Degeneracy in Structure and Motion Recovery from Uncalibrated Image Sequences,” *International Journal of Computer Vision*, vol. 32, no. 1, pp. 27–44, 1999.

- [21] J. Meltzer, R. Gupta, M.-H. Yang, and S. Soatto, “Simultaneous localization and mapping using multiple view feature descriptors,” in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 2, pp. 1550–1555 vol.2, Sept 2004.
- [22] A. Davison, Y. G. Cid, and N. Kita, “Real-time 3D SLAM with wide-angle vision,” in *Proc. IFAC Symposium on Intelligent Autonomous Vehicles, Lisbon*, July 2004.
- [23] M. Pupilli and A. Calway, “Real-time camera tracking using a particle filter,” in *In Proc. British Machine Vision Conference*, pp. 519–528, 2005.
- [24] E. Eade and T. Drummond, “Scalable monocular slam,” in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 469–476, June 2006.
- [25] E. Eade and T. Drummond, “Monocular slam as a graph of coalesced observations,” in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1–8, Oct 2007.
- [26] L. Clemente, A. Davison, I. Reid, J. Neira, and J. Tardós, “Mapping Large Loops with a Single Hand-Held Camera,” (Atlanta, GA, USA), June 2007.
- [27] J. Civera, A. J. Davison, and J. M. Montiel, “Dimensionless monocular slam,” in *Proceedings of the 3rd Iberian Conference on Pattern Recognition and Image Analysis, Part II, IbPRIA '07*, (Berlin, Heidelberg), pp. 412–419, Springer-Verlag, 2007.
- [28] K. Konolige and M. Agrawal, “Frameslam: From bundle adjustment to real-time visual mapping,” *Robotics, IEEE Transactions on*, vol. 24, pp. 1066–1077, Oct 2008.
- [29] G. Silveira, E. Malis, and P. Rives, “An efficient direct approach to visual slam,” *Robotics, IEEE Transactions on*, vol. 24, pp. 969–979, Oct 2008.
- [30] S. Hochdorfer and C. Schlegel, “Towards a robust visual slam approach: Addressing the challenge of life-long operation,” in *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pp. 1–6, June 2009.
- [31] D. Migliore, R. Rigamonti, D. Marzorati, M. Matteucci, and D. G. Sorrenti, “Use a single camera for simultaneous localization and mapping with mobile object tracking in dynamic environments,” 2009.
- [32] B. Williams and I. Reid, “On combining visual slam and visual odometry,” in *Proc. International Conference on Robotics and Automation*, 2010.

- [33] H. Strasdat, J. Montiel, and A. Davison, “Scale drift-aware large scale monocular slam,” The MIT Press, 2010.
- [34] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, “Dtam: Dense tracking and mapping in real-time,” in *Proceedings of the 2011 International Conference on Computer Vision, ICCV '11*, (Washington, DC, USA), pp. 2320–2327, IEEE Computer Society, 2011.
- [35] A. Pretto, E. Menegatti, and E. Pagello, “Omnidirectional dense large-scale mapping and navigation based on meaningful triangulation,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3289–3296, May 2011.
- [36] S.-H. Lee, “Real-time camera tracking using a particle filter combined with unscented kalman filters,” *Journal of Electronic Imaging*, vol. 23, no. 1, p. 013029, 2014.
- [37] G. Bourmaud and R. Megret, “Robust large scale monocular visual slam,” in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pp. 1638–1647, June 2015.
- [38] A. Concha and J. Civera, “Dpptom: Dense piecewise planar tracking and mapping from a monocular sequence,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 5686–5693, Sept 2015.
- [39] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.
- [40] B. D. Lucas and T. Kanade, “An Iterative Image Registration Technique with an Application to Stereo Vision,” in *International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'81*, (San Francisco, CA, USA), pp. 674–679, Morgan Kaufmann Publishers Inc., 1981.
- [41] S. Baker and I. Matthews, “Lucas-Kanade 20 Years On: A Unifying Framework,” *International Journal of Computer Vision*, vol. 56, pp. 221–255, feb 2004.
- [42] J. Hartmann, J. H. Klussendorff, and E. Maehle, “A comparison of feature descriptors for visual SLAM,” in *Mobile Robots (ECMR), 2013 European Conference on*, pp. 56–61, sep 2013.
- [43] D. Lowe, “Object recognition from local scale-invariant features,” in *International Conference on Computer Vision (ICCV), the seventh IEEE*, vol. 2, pp. 1150–1157 vol.2, IEEE, 1999.

- [44] D. Nistér, “An efficient solution to the five-point relative pose problem.,” *Pattern Analysis and Machine Intelligence (PAMI), IEEE Transactions on*, vol. 26, pp. 756–77, June 2004.
- [45] O. Faugeras and F. Lustman, “Motion and structure from motion in a piecewise planar environment,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 02, pp. 485–508, September 1988.
- [46] J. L. C. Jérôme Martin, “Experimental Comparison of Correlation Techniques,” in *IAS-4, International Conference on Intelligent Autonomous Systems*, 1995.
- [47] P. H. S. Torr and A. Zisserman, “MLE-SAC,” *Computer Vision and Image Understanding*, vol. 78, pp. 138–156, April 2000.
- [48] C. Tomasi and T. Kanade, “Detection and Tracking of Point Features,” tech. rep., *International Journal of Computer Vision*, 1991.
- [49] M. A. Fischler and R. C. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,” *Communications ACM*, vol. 24, pp. 381–395, June 1981.
- [50] R. I. Hartley, “In Defense of the Eight-Point Algorithm,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 19, pp. 580–593, June 1997.
- [51] J. Shi and C. Tomasi, “Good features to track,” in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pp. 593–600, June 1994.
- [52] D. Galvez-López and J. D. Tardos, “Bags of Binary Words for Fast Place Recognition in Image Sequences,” *Robotics, IEEE Transactions on*, vol. 28, pp. 1188–1197, oct 2012.
- [53] J. L. Bentley, “Multidimensional Binary Search Trees Used for Associative Searching,” *Communications ACM*, vol. 18, pp. 509–517, September 1975.
- [54] B. C. Hall, *Lie Groups, Lie Algebras, and Representations*, vol. 222. Springer-Verlag, number 102 ed., 2003.
- [55] S. Benhimane and E. Malis, “Homography-based 2D Visual Tracking and Servoing,” *International Journal of Robotics Research*, vol. 26, pp. 661–676, July 2007.
- [56] R. a. Moranna, R. D. Martin, and V. J. Yohai, *Robust Statistics*. Wiley, 2006.

- [57] L. Kneip, R. Siegwart, and M. Pollefeys, *Finding the Exact Rotation between Two Images Independently of the Translation*, ch. Finding th, pp. 696–709. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.
- [58] J. Civera, A. Davison, and J. Montiel, “Inverse Depth Parametrization for Monocular SLAM,” *IEEE Transactions on Robotics*, vol. 24, pp. 932–945, October 2008.
- [59] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, “Double Window Optimisation for Constant Time Visual SLAM,” in *International Conference on Computer Vision, Proceedings of the, ICCV ’11*, (Washington, DC, USA), pp. 2352–2359, IEEE Computer Society, 2011.
- [60] V. Lepetit, F. Moreno-Noguer, and P. Fua, “EPnP: An Accurate $O(n)$ Solution to the PnP Problem,” *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, 2009.
- [61] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth, “OpenFABMAP: An open source toolbox for appearance-based loop closure detection,” in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4730–4735, IEEE, May 2012.
- [62] S. Umeyama, “Least-squares estimation of transformation parameters between two point patterns,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 13, pp. 376–380, apr 1991.
- [63] C.-H. Shen and H. H. Chen, “Robust focus measure for low-contrast images,” in *Consumer Electronics, 2006. ICCE ’06. 2006 Digest of Technical Papers. International Conference on*, pp. 69–70, jan 2006.
- [64] K. Čopič Pucihar and P. Coulton, “Estimating Scale Using Depth from Focus for Mobile Augmented Reality,” in *Symposium on Engineering Interactive Computing Systems, the 3rd ACM SIGCHI, EICS ’11*, (New York, NY, USA), pp. 253–258, ACM, 2011.
- [65] S. Suwajanakorn, C. Hernandez, and S. M. Seitz, “Depth From Focus With Your Mobile Phone,” in *Computer Vision and Pattern Recognition (CVPR), the IEEE Conference on*, June 2015.
- [66] S. Pertuz, D. Puig, and M. A. Garcia, “Analysis of focus measure operators for shape-from-focus,” *Pattern Recognition*, vol. 46, no. 5, pp. 1415–1432, 2013.
- [67] G. Yang and B. J. Nelson, “Wavelet-based autofocusing and unsupervised segmentation of microscopic images,” in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 3, pp. 2143–2148 vol.3, oct 2003.

- [68] J. Jeon, J. Lee, and J. Paik, “Robust focus measure for unsupervised auto-focusing based on optimum discrete cosine transform coefficients,” *Consumer Electronics, IEEE Transactions on*, vol. 57, pp. 1–5, February 2011.
- [69] F. Devernay and O. Faugeras, “Straight Lines Have to Be Straight: Automatic Calibration and Removal of Distortion from Scenes of Structured Environments,” *Machine Vision Applications*, vol. 13, pp. 14–24, aug 2001.
- [70] R. Mur-Artal and J. D. Tardós, “Probabilistic Semi-Dense Mapping from Highly Accurate Feature-Based Monocular {SLAM},” in *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*, 2015.