A Unified Hybrid Formulation for Visual SLAM

by

Georges Youssef Younes

A thesis
presented to the University of Waterloo
and the American University of Beirut
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering
and
Mechanical Engineering
under a cotutelle agreement.

Waterloo, Ontario, Canada, 2021

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner: José María Martínez Montiel
Professor, Dept. of Informática e Ingeniería de Sistemas,
University of Zaragoza

Supervisor(s): John Zelek
Associate Professor, Dept. of Systems Design Eng.,
University of Waterloo

Daniel Asmar
Associate Professor, Dept. of Mechanical Eng.,
American University of Beirut

Internal Members: Alexander Wong
Associate Professor, Dept. of Systems Design Eng.,
University of Waterloo

Bryan Tripp
Associate Professor, Dept. of Systems Design Eng.,
University of Waterloo

Elie Shammas
Associate Professor, Dept. of Mechanical Eng.,
American University of Beirut

Kamel Abu Ghali
Professor, Dept. of Mechanical Eng.,
American University of Beirut

Internal-External Member: Stephen Smith
Associate Professor, Dept. of Electrical and Computer Eng.,
University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

I hereby declare that I am the sole author of all chapters in this thesis except Chapter 7.
This thesis consists in part of eight contributions written for publication. Exceptions to sole authorship of material are as follows:

Research presented in chapter 7 The research presented in this chapter was co-authored between Karim Smaha, and myself. Karim and I contributed equally to the code development, dataset generation, experiments, and the writing of the manuscript.

Abstract

Visual Simultaneous Localization and Mapping (*Visual SLAM (VSLAM)*), is the process of estimating the six degrees of freedom ego-motion of a camera, from its video feed, while simultaneously constructing a 3D model of the observed environment. Extensive research in the field for the past two decades has yielded real-time and efficient algorithms for *VSLAM*, allowing various interesting applications in augmented reality, cultural heritage, robotics and the automotive industry, to name a few.

The underlying formula behind *VSLAM* is a mixture of image processing, geometry, graph theory, optimization and machine learning; the theoretical and practical development of these building blocks led to a wide variety of algorithms, each leveraging different assumptions to achieve superiority under the presumed conditions of operation. An exhaustive survey on the topic outlined seven main components in a generic *VSLAM* pipeline, namely: the matching paradigm, visual initialization, data association, pose estimation, topological/metric map generation, optimization, and global localization.

Before claiming *VSLAM* a solved problem, numerous challenging subjects pertaining to robustness in each of the aforementioned components have to be addressed; namely: resilience to a wide variety of scenes (poorly textured or self repeating scenarios), resilience to dynamic changes (moving objects), and scalability for long-term operation (computational resources awareness and management). Furthermore, current state-of-the art *VSLAM* pipelines are tailored towards static, basic point cloud reconstructions, an impediment to perception applications such as path planning, obstacle avoidance and object tracking.

To address these limitations, this work proposes a hybrid scene representation, where different sources of information extracted solely from the video feed are fused in a hybrid *VSLAM* system. The proposed pipeline allows for seamless integration of data from pixel-based intensity measurements and geometric entities to produce and make use of a coherent scene representation. The goal is threefold: 1) Increase camera tracking accuracy under challenging motions, 2) improve robustness to challenging poorly textured environments and varying illumination conditions, and 3) ensure scalability and long-term operation by efficiently maintaining a global reusable map representation.

Acknowledgements

I would like to thank both of my supervisors John and Daniel, for their guidance, consistent support, and for being there during the dark periods of research, without you guys I would not be submitting this document! Thank you for everything.

Dedication

To Feyrouz, Youssef, Johnny, and Magdalena.

Table of Contents

List of Tables xiii

List of Figures xv

Abbreviations xvii

List of Symbols xix

1 Introduction 1

1.1 Premise 1

1.2 Motivation 2

1.3 Scope 2

1.4 Publications 3

2 Background 5

2.1 3D Rigid body transformations 5

2.2 Camera geometric model 6

2.3 Residual vector 6

2.3.1 Geometric re-pojection error 6

2.3.2 Photometric error 7

2.3.3 Robust cost functions 7

2.4 Optimization: Non Linear Least Squares 7

2.4.1 Gradient Descent 8
2.4.2 Gauss-Newton 9
2.4.3 Levenberg-Marquardt (LM) 10

3 VSLAM Architecture: Design, Survey and future directions 12
3.1 Matching paradigm: Direct/Feature-based 14
3.1.1 Direct methods 14
3.1.2 Feature-based methods 15
3.1.3 Hybrid methods 15
3.2 Data association 16
3.2.1 2D-2D 16
3.2.2 3D-2D 17
3.2.3 3D-3D 17
3.3 Visual Initialization 17
3.4 Pose estimation 19
3.5 Topological/metric map generation 21
3.6 BA - PGO - Map Maintenance 24
3.7 Global Localization 26
3.7.1 Failure Recovery 26
3.7.2 Loop Closure 27
3.8 Traits of different VSLAM components 28
3.8.1 Traits of the matching paradigms 28
3.8.2 Traits of the different data association types 31
3.8.3 Traits of initialization 31
3.8.4 Traits of pose estimation 32
3.8.5 Traits of map generation 33
3.8.6 Traits of BA/PGO/map maintenance 34
3.8.7 Traits of global localization 34

3.9 VSLAM: Towards a Hybrid Scene Representation, Feature-based - direct maps interactions . . . 35
3.10 Survey update . . . 35

4 Feature Assisted Direct Monocular Odometry 38
4.1 FDMO . . . 41
4.1.1 Notation . . . 41
4.1.2 Odometry . . . 42
4.1.3 Mapping . . . 44
4.1.4 Feature-based map maintenance . . . 46
4.2 FDMO-F . . . 47
4.2.1 Feature Extraction . . . 47
4.2.2 Pose Estimation . . . 48
4.2.3 Local Mapping . . . 49
4.3 Experimental Setup and Procedure . . . 50
4.3.1 Computational Cost . . . 50
4.3.2 Datasets . . . 51
4.3.3 Quantitative evaluation . . . 52
4.3.4 Qualitative assessment . . . 57

5 UFVO: A Unified Formulation for Visual Odometry 60
5.1 RelatedWork . . . 62
5.2 Nomenclature . . . 63
5.3 Proposed System . . . 63
5.3.1 Feature types . . . 63
5.3.2 System Overview . . . 64
5.3.3 Corner and pixel features sampling and activation . . . 65
5.3.4 Joint Multi-Objective Image alignment . . . 66
5.3.5 Utility function . . . 70

5.3.6 Mapping . . . 71
5.4 Evaluation . . . 72
5.4.1 Computational Cost . . . 72
5.4.2 Quantitative Evaluation . . . 73
5.5 Discussion . . . 74
5.6 Conclusion . . . 76

6 UFVS: A Unified Formulation for Visual SLAM 77

6.1 Loop Closure Essentials . . . 77
6.2 Literature Review . . . 78
6.3 Proposed System . . . 81
6.3.1 Pose Estimation . . . 82
6.3.2 Map representation . . . 83
6.3.3 Mapping . . . 83
6.3.4 Loop Closure . . . 84
6.4 Results and Discussion . . . 86
6.4.1 Qualitative Results . . . 86
6.4.2 Quantitative Results . . . 87
6.5 Conclusion . . . 90

7 Deep Camera Intrinsic Calibration from Natural Scene Images 91

7.1 Camera Calibration: An Introduction . . . 91
7.2 Related Work . . . 93
7.3 Background . . . 95
7.3.1 Kruppa’s equations . . . 96
7.4 Proposed System . . . 97
7.4.1 Solving Kruppa’s equations . . . 97
7.4.2 Network Architecture . . . 99

7.4.3 Dataset 99
7.5 Experiments and Results 103
7.5.1 Error Distribution 104
7.5.2 Comparison to Literature 105
7.5.3 Effect of input size 106
7.5.4 Limitations 107
7.6 Conclusion 108

8 Conclusion and future directions **109**
8.1 Sparse metric maps and semantic labels interactions 110
8.2 Semi-dense metric map and semantic labels interactions 110
8.3 Camera Calibration 111
8.3.1 Photometric Calibration 111
8.3.2 Geometric Calibration 111

References **113**

APPENDICES **131**
.1 Monocular VSLAM Survey 131
.2 Alignment error on the TumMono dataset. 134

List of Tables

3.1 Architecture used by different monocular SLAM systems 16
3.2 Feature extractors and descriptors employed by open source systems 17
3.3 Initialization methods 19
3.4 Pose estimation methods 21
3.5 Map generation methods 22
3.6 Map maintenance 24
3.7 Failure recovery 27
3.8 Loop closure. 28

4.1 Comparison between Direct and Indirect methods. 39
4.2 Computational time (ms) for processes in DSO, FDMO, FDMO-f and ORB-SLAM2. 51
4.3 Computational time (ms) breakdown for every processes in FDMO-f. 51
4.4 Measured drifts after finishing one and two loops over various sequences from the TumMono dataset. 57

5.1 Computational cost of different components of UFVO in ms. 73

6.1 Average computational cost (ms) associated with tracking and mapping threads for various VSLAM systems and DSO as odometry system. 87
6.2 Memory cost associated with the various VSLAM systems. 88
6.3 Localization error (meters) on the Euroc dataset (left images) 89

7.1 Parameter distribution used in generating the synthetic dataset. 101

7.2 New camera intrinsics as a function of the ground truth ones 102
7.3 Percentage Mean absolute error in intrinsics 106
7.4 Mean absolute error of the various developed models 107

1 VSLAM survey summary from 2003 to 2016. 131
2 Translational drift. 134
3 Rotational drift. 135
4 Scale drift. The closer to 1 the better. 136

List of Figures

3.1 Architecture of a Keyframe-based monocular SLAM system 13

3.2 Generic initialization pipeline. 18

3.3 Generic pose estimation pipeline. 19

3.4 Generic map generation pipeline. 22

3.5 Landmark triangulation by optimization. 23

3.6 Landmark estimation using filter based methods. 24

3.7 Generic map maintenance pipeline. 25

3.8 Map maintenance effects on the map. 26

3.9 Comparison between the different traits of direct vs. feature-based KSLAM systems. 30

4.1 Front-end flowchart of FDMO. 42

4.2 FDMO Tracking Recovery flowchart. 44

4.3 FDMO mapping flowchart. 45

4.4 FDMO-f feature detection. 48

4.5 FDMO-f pose estimation flow chart. 49

4.6 FDMO-f mapping flow chart. 50

4.7 Frame drop experiment depiction. 53

4.8 Backward frame drop experiment depiction. 54

4.9 Frame drop experiment results. 55

4.10 Sample paths estimated by various VO systems. 58

4.11 Features matched of FDMO and ORB-SLAM2 in a feature deprived environment. 59

5.1 UFVO 3D recovered map. 61

5.2 Summary of the feature types, their associated residuals, and their usage in UFVO. 64

5.3 Overview of UFVO's tracking and mapping threads. 67

5.4 Occupancy grid (left), and its associated keyframe's image (right). 67

5.5 The alignment error e_{align} UFVO (ours), LCSD, LDSO, DSO, and ORB SLAM 2 for all sequences of the TUM Mono dataset[45]. 74

5.6 The number of runs for which the alignment error was lower than the alignment error shown along the x-axis. The steeper and closer to the point (0,500) the better. 75

6.1 UFVS flowchart showing the various processes in the three main threads. 81

6.2 Density control and global/local reconstructed maps. 82

6.3 Global Bundle Adjustment constraints. 85

6.4 Constraints available for Pose Graph Optimization in various VSLAM systems. . 86

7.1 Summary of the proposed camera calibration 92

7.2 Sample image sets from the synthetically generated image dataset 100

7.3 Real Dataset sequence generation process 101

7.4 Cumulative error distributions of the estimated intrinsics using the different variations of the proposed model 104

7.5 Box-plot of the error distributions of the proposed model against the ground truth f_x, f_y, c_x, c_y 105

Abbreviations

BA Bundle Adjustment 24, 27

BoVW Bag of Visual Words 27, 34, 36, 37

FAIA Forward Additive Image Alignment 14

FCIA Forward Compositional Image Alignment 14

FDMO Feature Assisted Direct Monocular Odometry 3, 38–40

GBA Global Bundle Adjustment 25

GPS Global Positioning System 1

IAIA Inverse Additive Image Alignment 14

ICIA Inverse Compositional Image Alignment 14

IMU Inertial Measurement Unit 1

KSLAM KeyFrame based monocular SLAM 12, 14, 17–19, 21, 27

LBA Local Bundle Adjustment 25, 26

LIDAR Light Detection and Ranging 1

PGO Pose Graph Optimization 24, 26, 27

SLAM Simultaneous Localization and Mapping 1, 2

UFVO Unified Formulation for Visual Odometry 3, 60, 62, 76, 111

VIO Visual Inertial Odometry 2, 3

VO Visual Odometry 2, 3, 38–40, 53

VSLAM Visual SLAM v, 1–3, 21, 35–37, 109, 111, 112, 131

List of Symbols

$\mathbf{SE}(\mathbf{3})$ Special Euclidean transform 5, 6

$\mathbf{SIM}(\mathbf{3})$ Similarity transform 5

$\mathbf{SO}(\mathbf{3})$ Special Rotation Group 5

$\mathbf{se}(\mathbf{3})$ Special Euclidean transform tangent space 5

$\mathbf{sim}(\mathbf{3})$ Similarity transform tangent space 5, 27

Chapter 1

Introduction

1.1 Premise

Simultaneous Localization and Mapping (SLAM) refers to the process of extracting a scene representation of an agent exploring an unknown environment, while simultaneously localizing the agent in that environment.

SLAM solutions are considered indispensable for various tasks including navigation, surveillance, manipulation, and augmented reality applications to name a few. SLAM can be performed with various sensors and/or combination of sensors using cameras, *Light Detection and Ranging (LIDAR)*, range finders, *Global Positioning System (GPS)*, *Inertial Measurement Unit (IMU)*, etc. The more information such as depth, position, and speed are available at the sensory level, the easier the problem. If only a single camera is used, the problem is more challenging: single cameras are bearing only sensors; however, the rewards are great because a single camera is passive, consumes low power, is of low weight, needs a small physical space, is inexpensive, and ubiquitously found in hand-held devices. Cameras can operate across different types of environments, both indoor and outdoor, in contrast to active sensors such as infrared based RGB-D sensors that are sensitive to sunlight. Cameras also encode a relatively rich amount of information including colors and textures, in contrast to range finders and *LIDAR*, that can only capture depth measurements.

For the aforementioned reasons, it comes to no surprise that there has been extensive research in *VSLAM* for the past two decades yielding various solutions, each leveraging different assumptions to achieve superiority under some presumed condition of operation.

1.2 Motivation

In their basic form, most of the proposed systems evolve around generating a geometric 3D representation of the environment using a set of landmarks (keypoints, pixels, edges, etc.)[83, 84], in what is referred to as *metric maps*. Realizing that a metric representation becomes computationally intractable in large environments, researchers attempted to forfeit geometric information in favor of connectivity information through *Topological maps*[10]; however, metric measurements were still needed to localize in a meaningful way, *i.e.* distance measurement, low-level motor control, obstacle avoidance, etc. Unfortunately, the conversion from topological to metric maps is not a trivial process, and hybrid maps were introduced [124, 44, 99], where a scene is both metric and topological.

The use of the different scene maps splits most proposed approaches in the literature into one of two categories, either *Visual Odometry (VO)* or *VSLAM*, where *VO* maintains and operates a strictly local map while *VSLAM* makes use of both local and global map representations. However, the ability to extend a *VO* system to a *VSLAM* system is not a trivial process since it is limited at the lowest level, by the choice of landmarks extracted from the image.

Landmarks are in turn categorized as either *Direct*, *feature-based* (also referred to as *Indirect*), or a hybrid of both [173]. Direct landmarks refer to photometric measurements (pixel intensities) and are mostly restricted to *VO* systems, as opposed to feature-based landmarks that are extracted as a sparse intermediate image representation and can be used for both local and global localization. The choice of feature-based or direct has important ramifications on the overall design, ability and performance of the overall system, with each type exhibiting its own challenges, advantages, and disadvantages. In fact, as it will be shown later, the properties of Direct and Indirect systems are of complementary nature, and hybrid systems that makes use of both is undoubtedly the next step in the right direction.

This work explores the different modes of interactions between two types of landmarks that can be extracted for a video feed to develop a *VSLAM* framework that leverages their complementary nature in a hybrid scene representation, hence the title *A Unified Hybrid formulation for VSLAM*.

1.3 Scope

While the scope of the work is limited to *VSLAM* solutions using a single camera, hereafter referred to as monocular *SLAM*, the methods applied/developed for monocular systems are valid and applicable, with some modification, to stereo rigs, other vision based sensors (RGBD cameras) and Visual Inertial Systems *Visual Inertial Odometry (VIO)*.

On another note, the Deep Learning research community have made significant strides over the past few years to port the AI revolution to VSLAM; nevertheless, traditional methods still reign supreme: from applicability perspectives, traditional *VSLAM* or odometry systems are easily extended to new sensors (*e.g. VIO*, stereo, RGBD systems), they do not suffer from unpredictable modes of failures, and allow for intricate tuning of various components to particular environments or tasks. In contrast, most state of the art end-to-end neural networks applied to *VSLAM*, fail to generalize, and their training and evaluations are limited to the same few publicly available datasets. For this reason the scope of this work is limited to traditional methods. That being said, advancements in neural network capabilities have shown merit in solving various sub-tasks of the general *VSLAM* problem, one of which is the camera calibration task as will be explored in chapter 7.

1.4 Publications

Multiple systems were developed along the way, each leveraging the lessons learned from its predecessor. Starting from **Keyframe-based monocular SLAM: design, survey, and future directions** [173], a set of guidelines and *future directions* were identified, and were summarized in **Towards a Hybrid Scene Representation in *VSLAM*** [174] leading to the first proposed system ***Feature Assisted Direct Monocular Odometry (FDMO)*** [177]. *FDMO* features a loosely coupled integration between traditional Direct and Indirect odometry systems to improve resilience against challenging erratic motions. It operates by monitoring the pose estimation of a photometric alignment process and invokes a feature-based recovery procedure when failure is detected. *FDMO* was further refined in **Robustifying Direct *VO* to large baseline motions** [176] where various modifications were introduced, allowing for an efficient loose coupling of both Direct and Indirect frameworks concurrently at frame-rate. A thorough experimental procedure was then established in **Robustness of VO Systems to Subsampled Motions** [59] which paved the way for ***Unified Formulation for Visual Odometry (UFVO)*** [178], where a tightly coupled joint optimization is explored, leading to an overall system robustness improvement and accuracy gains over the state of the art *VO* systems.

The various properties of *UFVO* were thoroughly analyzed in **The Advantages of a Joint Direct and Indirect VSLAM in AR** [175] and used as blueprints for the latest system (currently unpublished) **UFVS: a Unified Formulation for Visual SLAM** where a global map is efficiently maintained and reused for loop closure.

Finally, the knowledge of the camera calibration matrix is an essential requirement for all *VSLAM* and *VO* systems, including the aforementioned contributions. However, recovering the camera calibration requires an offline process where a calibrating pattern is observed from various point of views. In an effort to circumvent this need for an offline calibration process, **Deep Camera Intrinsic Calibration from Natural Scene Images** (currently under review) explores a deep learning approach to recover the camera's intrinsic calibration directly from the video feed without the need of a calibrating pattern.

Chapter 2

Background

2.1 3D Rigid body transformations

A point X in 3D is represented by a vector of homogeneous coordinates: $X = (x \quad y \quad z \quad 1)^T$. Rigid transformations $\in \mathbf{SE(3)}$ are used to transform the coordinates of the point X from one coordinate frame to another:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \left(\begin{array}{ccc|c} & R & & T \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2.1}$$

where $R \in \mathbf{SO(3)}$ is a 3×3 rotation matrix and T is a $3D$ translation vector $\in \Re^3$. The $\mathbf{SE(3)}$ Lie group is a 4×4 matrix, minimally represented, in the tangent space $\mathbf{se(3)}$, by a $6D$ vector. An $\mathbf{se(3)}$ vector is mapped to an $\mathbf{SE(3)}$ via the *exponential* map and vice versa via the *log* map. An extension of $\mathbf{SE(3)}$ is the group of similarity transforms which also include a scale $s \in \Re^+$ and are denoted as $\mathbf{SIM(3)}$:

$$SIM(3) = \left(\begin{array}{ccc|c} & sR & & T \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{2.2}$$

In turn, $\mathbf{SIM(3)}$ are minimally represented in the tangent space $\mathbf{sim(3)}$, by a 7D vector via the exponential map and vice versa via the logmap.

The inverse depth parametrization X' of a 3D point, in the camera coordinate frame, associated with a pixel p at a given depth d, is defined by:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} P_x/d \\ P_y/d \\ 1/d \\ 1 \end{pmatrix} \tag{2.3}$$

2.2 Camera geometric model

A 3D point X $\begin{pmatrix} x & y & z & 1 \end{pmatrix}^T$ expressed in the camera coordinate frame is projected onto the image's frame using the intrinsic camera calibration parameters. One of the popular camera intrinsics models that accounts for radial distortion is the rad-tan model and is best described by:

$$\pi(X) = \begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} + \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} \times \frac{\frac{1}{\omega}\arctan(2\sqrt{\frac{x^2+y^2}{z^2}}\tan\frac{\omega}{2})}{\sqrt{\frac{x^2+y^2}{z^2}}} \times \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix}, \tag{2.4}$$

where $u_0, v_0, f_x, f_y, \omega$ are camera specific and found in an off-line calibration step. When there is no camera distortion (*e.g.*, synthetic images) a simple pinhole projection model can be used and is best described by

$$\pi(X) = KX = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \end{pmatrix}. \tag{2.5}$$

2.3 Residual vector

2.3.1 Geometric re-pojection error

The geometric re-projection error vector $\in \Re^2$ is found on a per feature i basis as

$$e_i = \left| \begin{pmatrix} u'_i \\ v'_i \end{pmatrix} - \pi(PX) \right|, \tag{2.6}$$

where X is a 3D landmark, P $\in \mathbf{SE(3)}$ is a transformation that maps from world frame to camera frame, and $\begin{pmatrix} u'_i & v'_i \end{pmatrix}^T$ are the 2-d coordinates of the corresponding feature match found through data association.

2.3.2 Photometric error

The photometric residual $\in \Re^1$ is defined over a window of pixels η surrounding the pixel of interest j as:

$$e_j = \sum_{p \in \eta} \left(I_i[p] - I_j[\pi(R\pi^{-1}(p, d_p) + T)] \right), \tag{2.7}$$

where (R, T) are priors over the transformation between the frames I_i and I_j and d_p is the inverse depth value associated with the pixel p. However, this formulation is sensitive to slight brightness variations from one frame to another; therefore, [40] introduced a brightness transfer function to gain robustness against such variations and became the de-facto formulation for direct alignment in the form:

$$e_j = \sum_{p \in \eta} \left[(I_i[p] - b_i) - \frac{t_i e^{a_i}}{t_j e^{a_j}} (I_j[\pi(R\pi^{-1}(p, d_p) + T)] - b_j) \right], \tag{2.8}$$

where t_i and t_j are frame exposure times (if available, otherwise set to 1) and a_n, b_n are common for all points belonging to frame n and are estimated in the pose optimization step.

2.3.3 Robust cost functions

To account for outliers, the Huber norm $|.|_\gamma$ is applied to the photometric or geometric residual vectors and is defined by:

$$\rho(t) = \begin{cases} \frac{1}{2}t^2, & if|t| \leq \gamma \\ \gamma|t| - \frac{1}{2}\gamma^2, & if|t| > \gamma \end{cases}, \tag{2.9}$$

where γ (the outlier threshold) is a tuning parameter, below which the Huber norm is quadratic and above which it becomes linear, thereby reducing the effect of outliers on the final objective function.

2.4 Optimization: Non Linear Least Squares

For the sake of generalization, the optimization in this section will be discussed in terms of a generic objective function

$$F(x) = \frac{1}{2} \sum_{i}^{n} f_i(x)^2, \tag{2.10}$$

where f(x) is some residual function. The optimization problem is then defined as:

$$x^* = \underset{x}{\operatorname{argmin}} \frac{1}{2} f(x)^T f(x). \tag{2.11}$$

In some applications, different residuals can contribute differently to the objective function, as such a weighting scheme is incorporated into the optimization, and the problem becomes:

$$x^* = \underset{x}{\operatorname{argmin}} \frac{1}{2} f(x)^T W f(x). \tag{2.12}$$

The Weighting matrix $W \succcurlyeq 0$ is typically a diagonal matrix and initialized with an external source of information such as the octave level at which a key-point was detected or some a-priory source of information.

There are many ways to solve for the update step, we will limit the discussion in this section to the gradient descent, Gauss-newton and Levenberg-Marquardt.

2.4.1 Gradient Descent

The rationale behind Gradient Descent is to follow the negative direction of the gradient (steepest descent), evaluated at the current estimate *x*, until convergence.
The gradient of F is given by the vector

$$\nabla F(x) = \nabla(\frac{1}{2} f(x)^\intercal f(x)) = \frac{1}{2} 2J(x)^\intercal f(x), \tag{2.13}$$

where J(x) is the Jacobian matrix defined as:

$$J(x) = \left[\frac{\partial f_i}{\partial x_j}\right]_{i=1,\dots m, j=1,\dots,n} = \begin{bmatrix} \nabla f_1(x)^\intercal \\ \nabla f_2(x)^\intercal \\ \vdots \\ \nabla f_i(x)^\intercal \end{bmatrix}_{i \times j} \tag{2.14}$$

where *i* spans the number of observed residuals, j is the number of variables, and

$$\nabla f(x)^\intercal = \left[\frac{\partial f_i}{\partial x_1}, \frac{\partial f_i}{\partial x_2}, \dots, \frac{\partial f_i}{\partial x_j}\right]_{1 \times j}. \tag{2.15}$$

It follows that the gradient can be re-written as:

$$\nabla F(x) = J^{\intercal}(x) f(x). \tag{2.16}$$

The optimization is solved in an iterative fashion starting with an initial guess for x, and by estimating an update step $\Delta(x)$ along the negative gradient direction, that is:

$$x_{k+1} = x_k + \alpha \Delta x, \quad \text{where} \quad \Delta x = -J^{\intercal}(x) f(x), \tag{2.17}$$

and α is the step size along the gradient. In the case of weighted gradient descent the step update becomes $\Delta x = -J^{\intercal}(x) W f(x)$ Typical gradient descent behaviour is that it quickly approaches the solution from a distance, however its convergence slows down as it gets closer to the optimal point.

2.4.2 Gauss-Newton

Gauss-Newton approximates the objective function with a quadratic function by performing a Taylor series approximation of F(x) around x. The objective function is re-written as

$$F(x + \Delta x) \approx F(x) + \nabla F(x)^{\intercal} \Delta x + \frac{1}{2} \Delta x^{\intercal} \nabla^2 F(x) \Delta x, \tag{2.18}$$

where $\nabla F(x)$ is defined in 2.16, and the Hessian $\nabla^2 F(x)$ is found using:

$$\nabla^2 F(x) = J(x)^{\intercal} J(x) + \sum_{i=1}^{m} (f_i(x) \nabla^2 f_i(x)). \tag{2.19}$$

Computing the full Hessian is computationally expensive since it involves computing the Hessian of the residuals as well. Both Gauss-Newton and Levenberg-Marquardt assume that the initializing point is close enough to the final solution, so that the residual term in the Hessian is negligible and the objective function Hessian can then be approximated by:

$$\nabla^2 F(x) \approx J(x)^{\intercal} J(x). \tag{2.20}$$

Since $J(x)^{\intercal} J(x)$ is quadratic, it follows that it is also positive semi-definite, thereby satisfying the necessary condition for local optimality. The only remaining necessary condition for Gauss-Newton method is that J(x) is not rank deficient, otherwise Gauss-Newton will not be effective.

Finally the taylor approximation of the objective function can then be re-written as:

$$F(x+\Delta x) \approx F(x) + f(x)^T J\Delta x + \frac{1}{2}\Delta x^T J^T J\Delta x. \tag{2.21}$$

A local minimum can then be found by setting the first derivative of the taylor approximation to 0, that is:

$$\frac{\partial F(x+\Delta x)}{\partial \Delta x} = f(x)^{\intercal} J + \frac{1}{2} 2\Delta x^{\intercal} J(x)^{\intercal} J(x) = 0, \tag{2.22}$$

resulting in what is known as the *normal equations* defined by:

$$J^T J\Delta x = -J^T f(x). \tag{2.23}$$

In the case of weighted optimization, the *normal equations* becomes

$$J^{\intercal} W J\Delta x = -J^{\intercal} W f(x). \tag{2.24}$$

Solving Gauss-Newton then boils down to iteratively solving the normal equations and applying the increments to the variable vector x:

$$x_{k+1} = x_k + \alpha\Delta x, \quad \text{where} \quad \Delta x = -(J(x)^{\intercal} W J(x))^{-1} J^{\intercal}(x) W f(x), \tag{2.25}$$

and α is the step size. Unlike the gradient-descent, Gauss-Newton is notable for its convergence close to the solution, but is relatively slow to approach it from a distance.

2.4.3 Levenberg-Marquardt (LM)

Unlike the previous line search methods, LM is a trust region method that aims to solve

$$\begin{gathered} \underset{\Delta x}{\operatorname{argmin}} \frac{1}{2} \|f(x) + J(x)\Delta(x)\|^2 \\ s.t. \|\Delta x\| < K \end{gathered}, \tag{2.26}$$

where $K > 0$ is the trust region radius (spherical trust region). Further details regarding the derivation of LM are outside the scope of this review, but it is noteworthy to mention its behaviour. In particular LM can be interpreted as a strategy that switches between gradient descent and Gauss-Newton variant when necessary. It does so by introducing a dampening factor λ, which controls the behavior of the descent. Large values of the residual causes the algorithm to behave like gradient descent and therefore quickly approaches the solution from a distance. As the residuals decrease, LM switch to a trust-region variant of Gauss-Newton, thereby not slowing down as it approaches the optimal point. It is also superior to Gauss-Newton as it accounts for the approximated Hessian with λ, and is well behaved even when J(x) is rank deficient.

Finally the step update rule for LM is $x_k = x + \Delta x$ and the weighted and unweighted normal equations for LM can be respectively found using:

$$\begin{aligned} \Delta x &= -((J(x)^{\intercal}J(x) + \lambda I))^{-1}J(x)^{\intercal}f(x) \\ \Delta x &= -((J(x)^{\intercal}WJ(x) + \lambda I))^{-1}J(x)^{\intercal}Wf(x) \end{aligned}. \tag{2.27}$$

The update step is then applied to the current estimate x until the convergence criteria is met or a maximum number of iterations is reached.

Chapter 3

VSLAM Architecture: Design, Survey and future directions

Monocular SLAM architecture can be either filter-based, such as using a Kalman filter[33], or Keyframe-based, relying on numerical optimization methods. In a filter-based framework, the camera pose and the entire state of all landmarks in the map are tightly joined and need to be updated at every frame, limiting the tractability of the filter to small environments. On the other hand, in a Keyframe based approach, the problem can be split into two parallel processes: pose tracking on the front-end, and mapping on the backend. The frontend is typically responsible for image processing, frame-to-frame pose tracking and Keyframe selection, where Keyframes are frames that were flagged according to some predefined criteria, and used to propagate the scene reconstruction in the backend. The parallel process approach was first introduced in [83] where the backend was no longer constrained by real-time requirements; it only ran at Keyframe rate, which allowed for intricate computationally expensive optimizations to take place for the scene reconstruction. As a consequence, Strasdat *et al*. in 2010 [151] showed that Keyframe methods outperform filter-based ones; it is therefore not surprising to note (as the list in Appendix .1 shows) that most new releases of monocular SLAM and odometry systems are Keyframe-based; therefore, the remainder of this work will be oriented towards Keyframe approaches.

Designing a generic *KeyFrame based monocular SLAM (KSLAM)* system requires the treatment of seven main components, summarized in Fig. 3.1): (1) matching paradigm, (2) data association, (3) visual initialization, (4) pose estimation, (5) Topological/metric map generation, (6) BA/PGO/map maintenance, and (7) global localization.

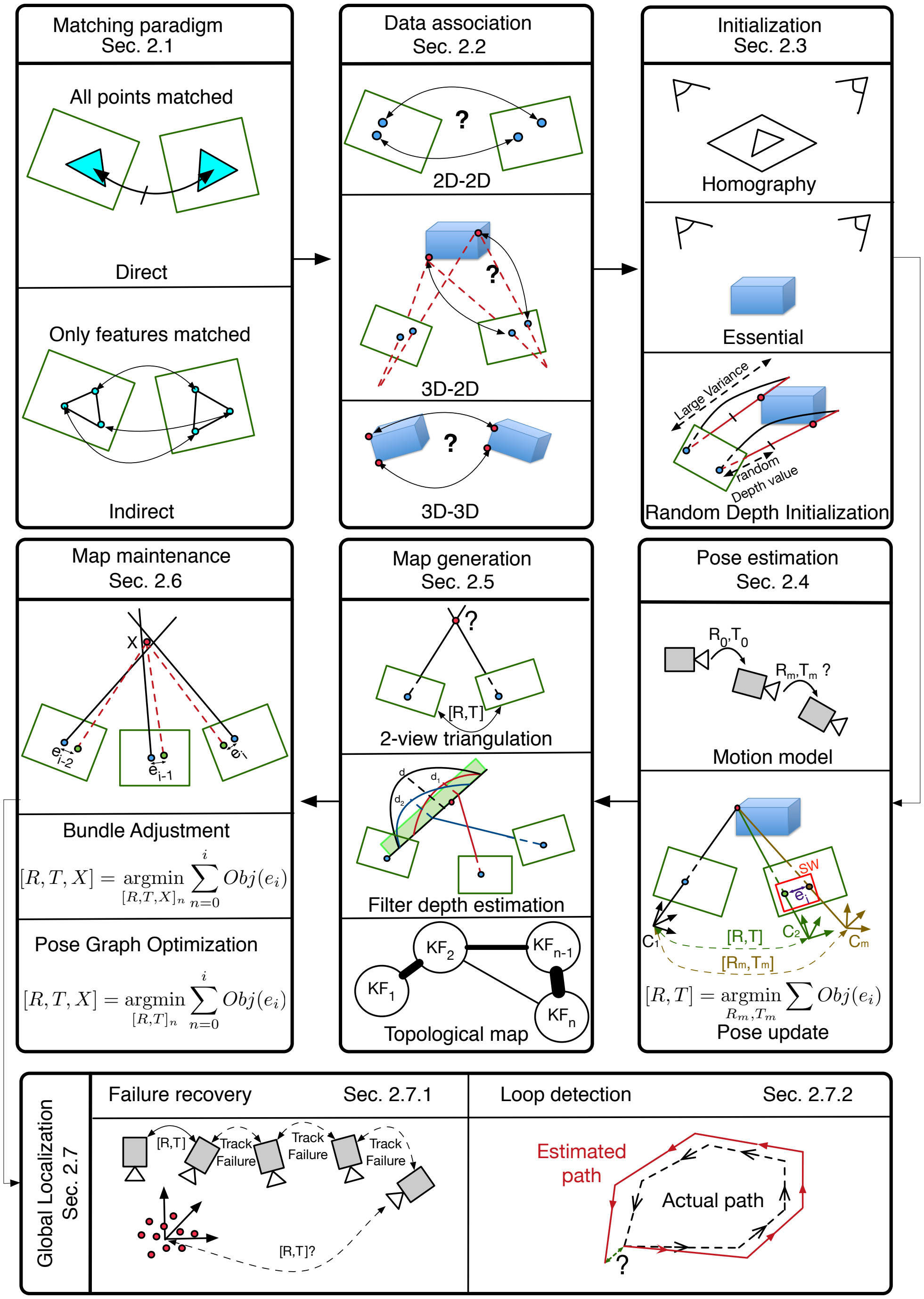


Figure 3.1: Architecture of a Keyframe-based monocular SLAM system

3.1 Matching paradigm: Direct/Feature-based

Landmarks extracted from images can be categorized into two paradigms: feature-based (Indirect) or pixel-based (Direct) methods. While both operate on data extracted from a video feed, each paradigm processes the input differently, leading to a different but often complementary set of properties. Table 3.1 summarizes the matching paradigms of open source *KSLAM* systems.

3.1.1 Direct methods

Direct methods use raw pixel intensities as inputs: no intermediate image representation is computed, hence the naming *Direct*. Direct methods can be dense, semi-dense, or sparse. Dense methods exploit the information available at every pixel, semi-dense methods exploit the information from pixels at which the gradient of image brightness is significant, and sparse methods use a relatively small set of sampled pixels with strong response to some metric such as Harris corner detector[66], FAST [140], etc.

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

$$J(x,y,t) = I(x+u(x,y), y+v(x,y), t+1), \tag{3.1}$$

where x and y are pixel coordinates; u and v denote displacement functions of the pixel (x,y) between two images I and J of the same scene taken at time t and $t+1$ respectively. [109] suggested, in what they referred to as *Forward Additive Image Alignment (FAIA)*, to replace all the individual pixel displacements u and v by a single general motion model $W(x,y,p)$, in which the number of parameters is dependent on the implied type of motion. For example, if the number of parameters is 2, the system reduces to computing optical flow. FAIA iteratively minimizes the squared pixel-intensity difference between the two images over the transformation parameters p:

$$\underset{p}{\operatorname{argmin}} \sum_{x,y} [I(W(x,y,p)) - J(x,y)]^2, \tag{3.2}$$

where $W(.,.,p)$ is a warping transform that encodes the relationship relating the two images and p corresponds to the parameters of the transform. Equation (3.2) is non-linear and requires an iterative non-linear optimization process, solved using either Gauss-Newton or LM optimizations. Since then, other variants of the *FAIA* were suggested such as *Forward Compositional Image Alignment (FCIA)*, *Inverse Compositional Image Alignment (ICIA)* and *Inverse Additive Image Alignment (IAIA)*, each with different computational complexities.

To account for brightness variation across frames, most modern formulations replace (3.2) with

(2.8) without significant modifications to the iterative optimization process.

3.1.2 Feature-based methods

Feature-based methods process 2D images to extract *salient* geometric primitives such as Keypoints (corners), edges, lines, etc. The pixel patterns surrounding these features are manipulated to generate descriptors as a quantitative measure of similarity to other features, after which, the image itself becomes obsolete. Extracted features are expected to be distinctive and invariant to viewpoint and illumination changes, as well as resilient to blur and noise. On the other hand, it is desirable for feature extractors to be computationally efficient and fast. Unfortunately, such objectives are hard to achieve simultaneously, a trade-off between computational speed and feature quality is required.

The computer vision community has developed, over decades of research, many feature extractors and descriptors, each exhibiting varying performances in terms of rotation and scale invariance, as well as speed [87]. The selection of an appropriate feature detector depends on the platform's computational power, the environment in which the monocular SLAM algorithm is due to operate in, as well as its expected frame rate. Feature detector examples include the Hessian corner detector [6], Harris detector [66], Shi-Tomasi corners [146], Laplacian of Gaussian detector [101], MSER [114], Difference of Gaussian [107] and the accelerated segment test family of detectors (FAST, AGAST, OAST) [111].

To minimize computational requirements, most feature-based systems use *FAST* [140] for their feature extractor, coupled with a feature descriptor for data association. Feature descriptors include, and are not limited to, BRIEF [17], BRISK [97], SURF [4], SIFT [108], HoG [31], FREAK [3], ORB [141], and a low level local patch of pixels. Further information regarding feature extractors and descriptors is outside the scope of this work, but the reader can refer to [120], [69], [137], or [72] for comparisons.

3.1.3 Hybrid methods

Different from the direct and feature-based methods, some systems such as SVO[52] are considered hybrids; they use a combination of both direct and feature-based methods to refine the camera pose estimates, or to generate a dense/semi-dense map.

Table 3.1: Architecture used by different monocular SLAM systems. Abbreviations used: indirect (i), direct (d), and hybrid (h)

	PTAM	SVO	DT SLAM	LSD SLAM	ORB SLAM	DPPTAM	DSO
Method	I	H	I	D	I	D	D

3.2 Data association

Data association is defined as the process of establishing measurement correspondences across different images; while it is implicit for direct methods, it is explicitly done in feature-based methods using either 2D-2D, 3D-2D, or 3D-3D correspondences.

3.2.1 2D-2D

In 2D-2D correspondence, the 2D feature's location in an image I_2 is sought, given its 2D position in a previously acquired image I_1. Depending on the type of information available, 2D-2D correspondences can be established in one of two ways: when a map is not available, and neither the camera transformation between the two frames nor the scene structure is available (*i.e.*, during system initialization), 2D-2D data association is established through a large search window surrounding the feature's location from I_1 in I_2. On the other hand, when the transformation relating I_1 and I_2 is known (*i.e.*, the relative camera pose is known), 2D-2D data correspondences are established through Epipolar geometry, where a feature in I_1 is mapped to a line in I_2, and the two dimensional search window collapses to a one dimensional search along a line. This latter case often occurs when the system is triangulating 2D features into 3D landmarks during map generation. To limit the computational expenses, a bound is imposed on the search region along the Epipolar line.

In both methods, each feature has associated with it a descriptor, which can be used to provide a quantitative measure of similarity to other features. The descriptor similarity measure varies with the type of descriptors used; for example, for a local patch of pixels descriptor, it is typical to use the Sum of Squared Difference (SSD), or a Zero-Mean SSD score (ZMSSD) to increase robustness against illumination changes, as is done in [82]. For higher order feature descriptors—such as SIFT or SURF—the L1-norm, the L2-norm can be used, and for binary descriptors the Hamming distance is employed. Establishing matches using these measures is computationally intensive and may, if not carefully applied, degrade real-time performance. For such purpose, special implementations that sort and perform feature matching in K-D trees [122] or bag of words[54] are usually employed.

3.2.2 3D-2D

In 3D-2D data association, one seeks to estimate correspondences between 3D previously triangulated landmarks and their 2D projections onto a newly acquired frame, without the knowledge of the new camera pose. This type of data association is typically used during the pose estimation phase of *KSLAM*. To solve this problem, previous camera poses are exploited to yield a hypothesis on the new camera pose, in what is referred to as motion models, and accordingly project the 3D landmarks onto that frame. 3D-2D data association then proceeds similarly to 2D-2D feature matching, by defining a search window surrounding the projected location of the 3D landmarks.

3.2.3 3D-3D

3D-3D data association is typically employed to estimate and correct accumulated drift along loops: when a loop closure is detected, descriptors of 3D landmarks, visible in both ends of the loop, are used to establish matches among landmarks that are then exploited—as explained in [158]—to yield a similarity transform between the frames at both ends of the loop.

Table 3.2 summarizes the feature types and descriptors employed by various open-source *KSLAM* systems.

Table 3.2: Feature extractors and descriptors. Abbreviations:local patch of pixels (L.P.P.)

	PTAM	SVO	DT SLAM	LSD SLAM	ORB SLAM	DPPTAM	DSO
Feature type	FAST	FAST	FAST	None	FAST	None	None
Feature descriptor	L.P.P.	L.P.P.	L.P.P.	L.P.P.	ORB	L.P.P.	L.P.P.

3.3 Visual Initialization

Monocular cameras are bearing-only sensors, that is, they cannot directly perceive depth from a single image; nevertheless, up to scale depth can be estimated via temporal stereoscopy after observing the same scene through at least two different viewpoints.

During initialization, neither pose nor structure are known, and *KSLAM* requires a special initialization phase, during which both a map of 3D landmarks, and the initial camera poses are generated. After *KSLAM* is initialized, camera pose and 3D structure build on each other, in a heuristic manner, to propagate the system in time by expanding the map to previously unobserved scenes, while keeping track of the camera pose in the map.

To initialize a generic *KSLAM* system, researchers adopted the methods developed by [103] to simultaneously recover the camera pose and the 3D scene structure. The intuition behind [103] was to algebraically eliminate depth from the problem, yielding either the Essential matrix or a Homography matrix. However, the elimination of depth has significant ramifications on the recovered data, since the exact camera motion between the two views cannot be recovered: the camera translation vector is recovered up to an unknown scale λ. Since the translation vector between the two views defines the baseline used to triangulate the 3D landmarks, scale loss also propagates to the recovered 3D landmarks, yielding a scene that is also scaled by λ. Furthermore, the decomposition of the Essential or Homography matrix yields multiple solutions, of which only one is physically feasible and is disambiguated from the others through Cheirality checks. The solutions are also degenerate in certain configurations, such as when the observed scene is planar in the case of Essential matrix, and non-planar in the case of a Homography.

Figure 3.2 shows the flowchart of a generic model-based initialization procedure. To mitigate degenerate cases, *random depth* initialization, as its name suggests, initializes a *KSLAM* by randomly assigning depth values with large variance to a single initializing Keyframe. The random depth is then iteratively updated over subsequent frames until the depth variance converges. Random depth initialization is usually employed in the direct framework, however they are generally brittle and require slow translation-only motion while observing the same scene to converge.

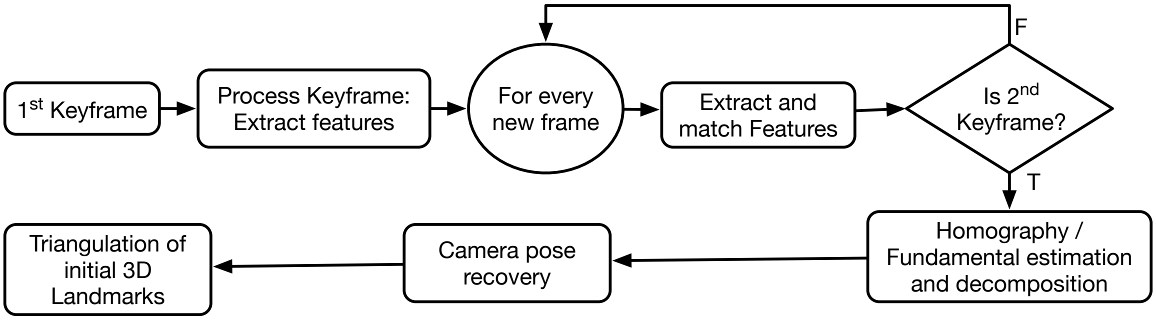

1st Keyframe
Process Keyframe: Extract features
For every new frame
Extract and match Features
Is 2nd Keyframe?
F
T
Homography / Fundamental estimation and decomposition
Camera pose recovery
Triangulation of initial 3D Landmarks

Figure 3.2: Generic initialization pipeline.

Table 3.3 summarizes the initialization methods employed by different monocular *KSLAM* systems, along with the assumption they make about the configuration of the scene at startup.

Table 3.3: Initialization. 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	DT SLAM	LSD SLAM	ORB SLAM	DPPTAM	DSO
Initialization	h.d.	h.d.	e.d.	r.d.	h.d.+e.d.	r.d.	r.d.
Initial scene assumption	p	p	n.p.	n.a.	n.a.	n.a.	n.a.

3.4 Pose estimation

Because data association is computationally expensive, most monocular SLAM systems assume for the pose of each new frame a prior, which guides and limits the amount of work required for data association. Estimating this prior is generally the first task in pose estimation: a map and data association between two frames are known, and one seeks to estimate the pose of the second frame given the pose of the first. Table. 3.4 summarizes the pose estimation methods used by different *KSLAM* systems.

Most systems employ a *constant velocity* motion model that assumes a smooth camera motion across the recently tracked frames to estimate the prior for the current frame. Some systems assume no significant change in the camera pose between consecutive frames, and hence they assign the prior for the pose of the current frame to be the same as the previously tracked one. Figure 3.3 presents a generic pipeline for pose estimation. The pose of the prior frame is used to guide the data association procedure in several ways. It helps determine a potentially visible

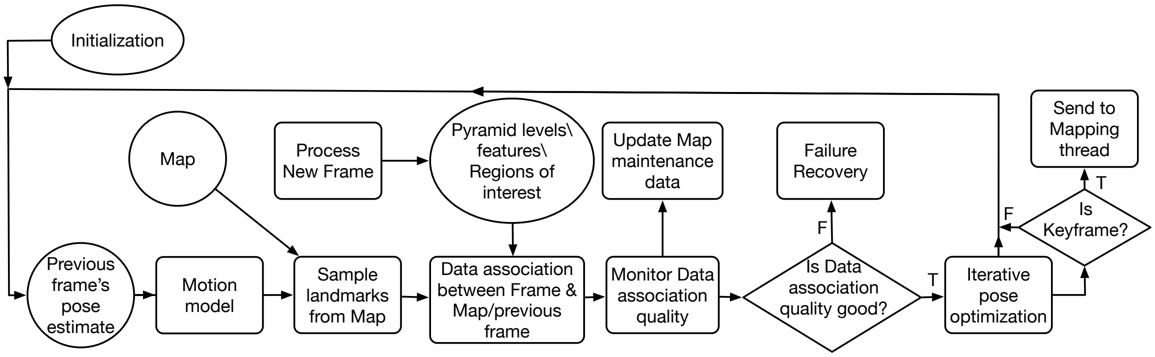

Initialization
Map
Process New Frame
Pyramid levels\ features\ Regions of interest
Update Map maintenance data
Failure Recovery
Send to Mapping thread
T
Is Keyframe?
F
F
Previous frame's pose estimate
Motion model
Sample landmarks from Map
Data association between Frame & Map/previous frame
Monitor Data association quality
Is Data association quality good?
T
Iterative pose optimization

Figure 3.3: Generic pose estimation pipeline.

set of features from the map in the current frame, thereby reducing the computational expense of blindly projecting the entire map. Furthermore, it helps establish an estimated feature location in the current frame, such 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, which refines the camera pose.

Direct and feature-based methods estimate the camera pose by minimizing a measure of error between frames; direct methods measure the photometric error, modeled as the intensity difference between pixels; in contrast, indirect methods measure the re-projection error of landmarks from the map over the frame's prior pose. The re-projection error is formulated as the distance in pixels between a projected 3D landmark onto a frame, and its corresponding 2-D position in the image.

A motion model is used to seed the new frame's pose at C_m (Figure 3.1), and a list of potentially visible 3D landmarks from the map are projected onto the new frame. Data association takes place in a search window S_w surrounding the location of the projected landmarks. The system then proceeds by minimizing the re-projection error e_j over the parameters of the rigid body transformation. To gain robustness against outliers, the minimization takes place over an objective function that penalizes features with large re-projection errors. The camera pose optimization problem is then defined as:

$$T_i = \underset{T_i}{\operatorname{argmin}} \sum_j Obj(e_j), \tag{3.3}$$

where T_i is a minimally represented Lie group of either $S\xi(3)$ or $sim(3)$ camera pose, $Obj(.)$ is an objective function and e_j is the error defined through data association for every matched feature j in the image.

Finally the tracking algorithm decides whether the new frame should be flagged as a Keyframe or not. A Keyframe is a special frame used for expanding the map. The decisive criteria can be categorized as either *significant pose change* or *significant scene appearance change*; A decision is usually made through a weighted combination of different criteria; examples of such criteria include: a significant change in the camera pose measurements (rotation and/or translation), the presence of a significant number of 2D features that are not observed in the map, a significant change in what the frame is observing (by monitoring the intensity histograms or optical flow), the elapsed time since the system flagged its latest Keyframe.

Table 3.4: Pose estimation. 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 geometric error(o.m.g.e.), 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	DT SLAM	LSD SLAM	ORB SLAM	DPPTAM	DSO
Motion prior	c.v.m.m. +ESM	s.a.p.p.	s.t.p.f.	s.a.p.p.	c.v.m.m. or place recogn.	c.v.m.m. or s.a.p.p.	multiple c.v.m.m.
Tracking	o.m.g.e.	o.m.p.e.	3 modes: 1-e.m.d.; 2-o.m.g.e.; 3-p.r.e.	o.m.p.e.	o.m.g.e.	o.m.p.e.	o.m.p.e.
Keyframe add criterion	s.p.c.	s.p.c.	s.s.a.c.	s.p.c.	s.s.a.c.	s.p.c.	s.s.a.c. or s.p.c.

3.5 Topological/metric map generation

Typical *VSLAM* systems employ two different types of map representations, namely *metric* and *topological* representations.

a. Metric map

The map generation module is responsible for generating a representation of the previously unexplored, newly observed environment. Table 3.5 summarizes map generation methods employed by different monocular *KSLAM* systems; Typically, the map generation module represents the world as a dense/semi-dense (for direct) or sparse (for feature-based methods) cloud of points. Figure 3.4 presents the flowchart of a map generation module. As different viewpoints of an unexplored scene are registered with their corresponding camera poses through the pose tracking module, the map generation module triangulates 2D points into 3D landmarks; also it keeps track of their 3D coordinates, and expands the map within what is referred to as a *metric* representation of the scene. In a metric map, the structure of new 3D landmarks is recovered from a known transformation between two frames at different viewpoints, using their corresponding data associations [68]. Since data association is prone to erroneous measurements, the map generation module is also responsible for the detection and handling of outliers, which can potentially de-

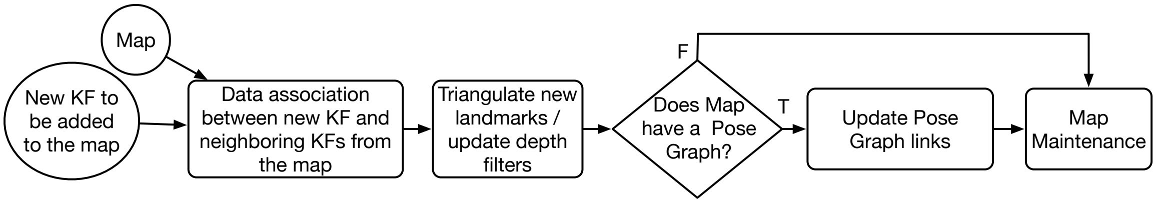

Map
New KF to be added to the map
Data association between new KF and neighboring KFs from the map
Triangulate new landmarks / update depth filters
Does Map have a Pose Graph?
F
T
Update Pose Graph links
Map Maintenance

Figure 3.4: Generic map generation pipeline.

Table 3.5: Map generation. Abbreviations: 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.), multiple hypotheses photometric error minimization (m.h.p.m.)

	PTAM	SVO	DT SLAM	LSD SLAM	ORB SLAM	DPPTAM	DSO
Map generation	2.v.t.	p.f.	2D.l.t.	p.f.p.f and s.b.o.	2.v.t.	m.h.p.m	s.b.o.
Map type	metric	metric	metric	hybrid	hybrid	metric	metric

stroy the map.

Due to noise in data association and pose estimates of the tracked images, projecting rays from two associated features will most probably not intersect in 3D space. To gain resilience against outliers and to obtain better accuracy, the triangulation is typically performed over features associated across more than two views. Triangulation by optimization as described by

$$X = \underset{[x,y,z]}{\operatorname{argmin}} \sum_{n} e_n, \tag{3.4}$$

and shown in Fig. 3.5, aims to estimate a landmark position $[x, y, z]$ from its associated 2D features across n views, by minimizing the sum of its re-projection errors e_n in all Keyframes I_n observing it. Filter based landmark triangulation, as shown in Fig. 3.6, recovers the 3D position of a landmark by first projecting into the 3D space, a ray joining the camera center of the first Keyframe observing the 2D feature and its associated 2D coordinates. The projected ray is then populated with a filter having a uniform distribution (D_1) of landmark position estimates, which are then updated as the landmark is observed across multiple views. The Bayesian inference framework continues until the filter converges from a uniform distribution to a Gaussian featuring a small variance (D_3) [73]. Filter-based triangulation results in a delay before an observed landmark's depth has fully converged, in contrast to triangulation by optimization methods, that

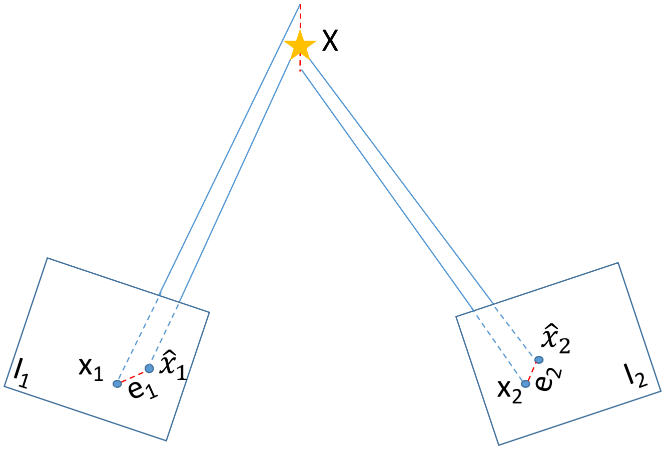


Figure 3.5: Landmark triangulation by optimization.

can be used as soon as the landmark is triangulated from two views. To overcome this delay and exploit all the information available from a feature that is not yet fully triangulated, [25] suggested an inverse depth parametrization for newly observed features, with an associated variance that allows for 2D features to contribute to the camera pose estimate, as soon as they are observed.

b. Topological map

As the camera explores large environments, metric maps suffer from the unbounded growth of their size, thereby leading to system failure. Topological maps were introduced to alleviate this shortcoming, by forfeiting geometric information in favor for connectivity information. In its most simplified form, a topological map consists of nodes corresponding to locations, and edges corresponding to connections between the locations. In the context of monocular SLAM, a topological map is an undirected graph of nodes that typically represents Keyframes linked together by edges, when shared data associations between the Keyframes exists. For a survey on topological maps, the reader is referred to [10]. In spite of the appeal of topological maps in scaling well with large scenes, metric information is still required in order to maintain camera pose estimates. The conversion from a topological to a metric map is not always trivial, and for this reason, recent monocular SLAM systems [124, 44, 100, 99] employ hybrid maps,which are simultaneously metric and topological. The implementation of a hybrid (metric-topological) map representation allows for efficient solutions to loop closures and failure recovery using topological information and increases efficiency of the metric pose estimate, by limiting the scope of the map to a local region surrounding the camera [48]. A hybrid map allows for local optimization of the metric map, while maintaining scalability of the optimization over the global topological map [86].

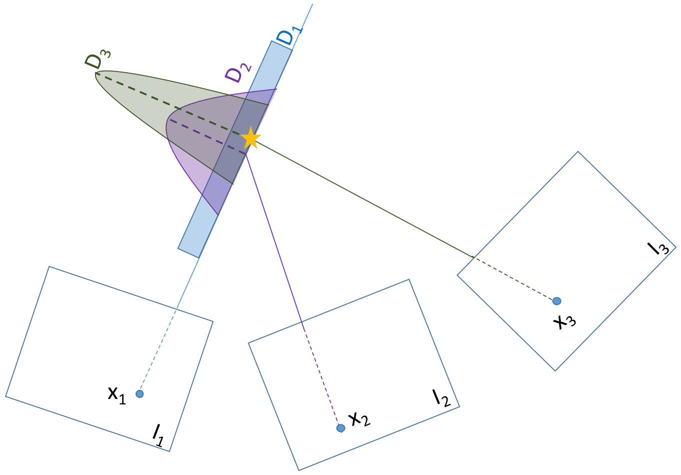


Figure 3.6: Landmark estimation using filter based methods.

3.6 BA - PGO - Map Maintenance

Map maintenance takes care of optimizing the map through either *Bundle Adjustment (BA)* or *Pose Graph Optimization (PGO)* [89]. Figure 3.7 presents the steps required for map maintenance of a generic monocular SLAM. During map exploration, 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 that were used to expand the map; therefore, a maintenance module is required to cater for such mode of failure. Table 3.6 summarizes the map maintenance procedure adopted by different KSLAM systems.

Table 3.6: Map maintenance. Abbreviations used: Local Bundle Adjustment (LBA), Global Bundle Adjustment (GBA), Pose Graph Optimization (PGO),

	PTAM	SVO	DT SLAM	LSD SLAM	ORB SLAM	DPP-TAM	DSO
Optimization	LBA & GBA	LBA	LBA & GBA	PGO	PGO & LBA	Dense mapping	LBA
Scene type	static & small	uniform depth	static & small	static & small or large	static & small or large	static & indoor planar	static & small or large

Figure 3.8 describes the map maintenance effect, where the scene's map is refined through

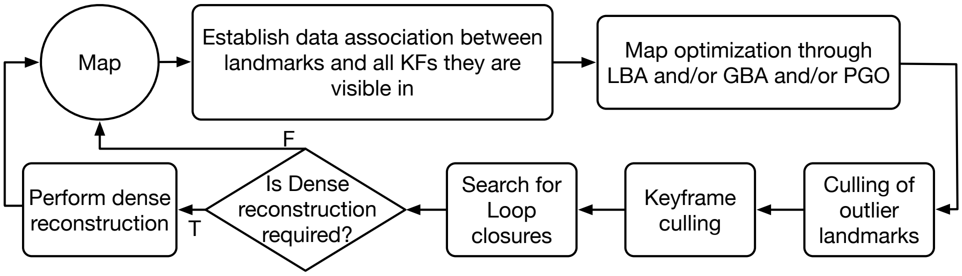

Map
Establish data association between landmarks and all KFs they are visible in
Map optimization through LBA and/or GBA and/or PGO
Culling of outlier landmarks
Keyframe culling
Search for Loop closures
Is Dense reconstruction required?
F
T
Perform dense reconstruction

Figure 3.7: Generic map maintenance pipeline.

outlier removal and error minimization, in order to yield a more accurate scene representation. The map maintenance module is also responsible for updating the connections between nodes in the topological map. When a new Keyframe is added into systems that employ hybrid maps, their topological map is updated by incorporating the new Keyframe as a node, and searching for data associations between the newly added node and surrounding ones; edges are then established to other nodes (Keyframes) according to the found data associations. Bundle adjustment is the problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter (camera pose and/or calibration) estimates. What we mean by this is that the parameter estimates are found by minimizing some cost function that quantifies the model fitting error, and that the solution is simultaneously optimal with respect to both structure and camera variations.

Bundle adjustment (BA) is the problem of refining a visual reconstruction to jointly produce an optimal structure and coherent camera pose estimates [156]. BA is an optimization that minimizes the cost function defined by:

$$\underset{T,X}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j \in S_i} Obj(e(T_i, X_j)), \tag{3.5}$$

where T_i is a Keyframe pose estimate and N is the number of Keyframes in the map. X_j corresponds to the 3D pose of a landmark and S_i represent the set of 3D landmarks observed in Keyframe i. Finally, $e(T_i, X_j)$ is the re-projection error of a landmark X_j on a Keyframe T_i, in which it is observed.

Bundle adjustment is computationally involved and intractable if performed on all frames and all poses. The breakthrough that enabled its application in realtime is the notion of Keyframes, where only select *special frames*, known as Keyframes, are used to perform map expansion and optimization. Different algorithms apply different criteria for Keyframe selection, as well as different strategies for BA; some strategies include jointly, a local (over a local number of Keyframes) *Local Bundle Adjustment (LBA)*, and global (over the entire map) *Global Bundle*

Adjustment (GBA) optimizations, other strategies argue that an *LBA* only is sufficient to maintain a good quality map. To reduce the computational expenses of bundle adjustment, [150] proposed to represent the monocular SLAM map by both a Euclidean map for *LBA*, and a topological map for pose graph optimization that explicitly distributes the accumulated drift along the entire map. *PGO* can be represented as the process that optimizes over only the Keyframe pose estimates:

$$\underset{T}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j \in S_i} Obj(e(T_i, X_j)). \tag{3.6}$$

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 monocular SLAM algorithms is that the environment is static, some algorithms such as RD SLAM[154] exploit map maintenance methods to accommodate slowly varying scenes (lighting and structural changes).

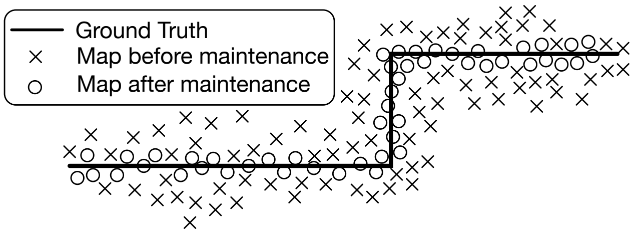


Figure 3.8: Map maintenance effects on the map.

3.7 Global Localization

Global localization is required when the camera loses track of its position and is required to situate itself in a global map. Failure recovery and loop closure are both considered a form of global localization. It is noteworthy to mention that loop closure and failure recovery evolve around the same problem, and solutions put forward to any of them could be used for the other. The interested reader is referred to [57] for a detailed survey on the topic.

3.7.1 Failure Recovery

Whether due to wrong user movement, such as abrupt changes in the camera pose, motion blur, or due to observing a featureless region, or for any other reason, monocular SLAM methods will eventually fail. Accordingly, a key module essential for the usability of any monocular SLAM system is its ability to correctly recover from such failures. Table 3.7 summarizes the failure recovery mechanisms used by different monocular *KSLAM* systems. The failure problem can be described as a lost camera pose that needs to re-localize itself in the map it had previously created of its environment before failure. Current state of the art systems in *KSLAM* perform failure recovery by generating a *Bag of Visual Words (BoVW)* representation of the image. The intermediate representation maps the images space onto a feature space made of visual words. The space of visual words is generated offline by clustering descriptors into a KD-Tree, where tree leafs are considered visual words. Failure recovery then proceeds by searching for *"closest looking"* previously observed Keyframe through either a probabilistic appearance based model [30] or through geometric consistency check in the topological map [123].

Table 3.7: Failure recovery. Abbreviations used: photometric error minimization of SBIs (p.e.m.), image alignment with last correctly tracked Keyframe (i.a.l.), image alignment with random Keyframe (i.a.r.), bag of words place recognition (b.w.), image alignment with arbitrary small rotations around the last tracked pose(i.a.a.r.)

	PTAM	SVO	DT SLAM	LSD SLAM	ORB SLAM	DPPTAM	DSO
Failure recovery	p.e.m.	i.a.l.	none	i.a.r.	b.w.	i.a.l	i.a.a.r.

3.7.2 Loop Closure

Since *KSLAM* is an optimization problem, it is prone to drifts in camera pose estimates. 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, which will eventually lead the system to erroneous measurements, and fatal failure. In an effort to correct for drift and camera pose errors, [124] and [44] detect loop closures in an online SLAM session, and optimize the loop's track and all its associated map data, using either *PGO* or *BA*. Upon the insertion of a new Keyframe, a loop is sought by looking for a connection to previously observed nodes and establishing a similarity transform $\mathbf{sim}(\mathbf{3})$ that relates both ends of the loop. Table 3.8 summarizes how loop closure is addressed by each KSLAM system.

Table 3.8: Loop closure. Abbreviations used: Bag of Words place recognition (B.W.p.r), sim(3) optimization (s.o.)

	PTAM	SVO	DT SLAM	LSD SLAM	ORB SLAM	DPPTAM	DSO
Loop closure	none	none	none	FabMap +s.o.	B.W.p.r. +s.o.	none	none

3.8 Traits of different VSLAM components

3.8.1 Traits of the matching paradigms

a. Direct methods

Direct methods can make use of virtually all image pixels with an intensity gradient in the image and are therefore more robust than feature-based methods in regions with poor texture and blur. It naturally handles points at infinity and points observed with small parallax using the inverse depth parametrization [25], and since no explicit data association is required, it can have semi-dense or sparse scene representations. More importantly, the photometric error can be interpolated over the image domain resulting in an image alignment with sub-pixel accuracy and relatively less drift than feature-based methods as shown in DSO [40](current state of the art in direct systems).
On the other hand, Direct methods are susceptible to scene illumination changes, due to the violation of the underlying brightness consistency assumption (3.1) . In an effort to gain resilience against this mode of failure, the recently released DSO models the image formation process, and attempts to incorporate the scene irradiance in the energy functional, at the expense of adding a calibrated image formation model which is used to correct the images at a pre-processing step. The model is estimated through an additional offline calibration process described in [45].

Furthermore, during the non-linear optimization process, (3.2), is linearized through a first order Taylor expansion. While the linearization is valid when the parameters of the warping transform tends to zero, higher order terms becomes dominant and the linearization becomes invalid for large transforms. Therefore, a second disadvantage of direct methods is the assumption of small motions between the images (typically not more than 1 pixel). To relax this constraint, direct monocular SLAM systems employ a pyramidal implementation, where the image alignment process takes place sequentially from the highest pyramid level to the lowest, using the results of every level as a prior to the next level. They also suggest the usage of high fame rate cameras to alleviate this issue; some systems employ an efficient second order minimization (ESM [7]) to estimate a rotation prior that helps increase the convergence radius. Despite these efforts, the tolerated baseline for data association in direct methods is considerably smaller than the tolerated baseline in feature-based methods.

Another disadvantage of direct methods is that the calculation of the photometric error at every pixel is computationally intensive; therefore, real-time monocular SLAM applications of direct methods, until recently, were not considered feasible. However, with the recent advancements in parallelized processing, hardware acceleration and with the introduction of semi-dense inverse depth filtering, it became possible to integrate direct methods into KSLAM solutions [52, 42, 29].

Direct methods do not naturally allow for loop closures detection nor failure recovery (probabilistic appearance based methods are required [30]), they are brittle against any sources of outliers in the scene (dynamic objects, occlusions), and require a spatial regularization when a semi-dense representation is employed. Most importantly, they become intractable for very large scene exploration scenarios and hence are mostly limited to odometry systems.

Finally, Direct methods suffer from the rolling shutter effect. When a rolling shutter camera is used (most smartphones), the image is serially generated which induces some time delay between different parts of the image. When the camera is moving, this delay causes a geometric distortion known as the rolling shutter effect. Since Direct methods integrate over areas in the image domain without accounting for this effect, their accuracy drops significantly in rolling shutter cameras. Indirect methods does not suffer from such limitations as features are discrete points and not areas in the image plane.

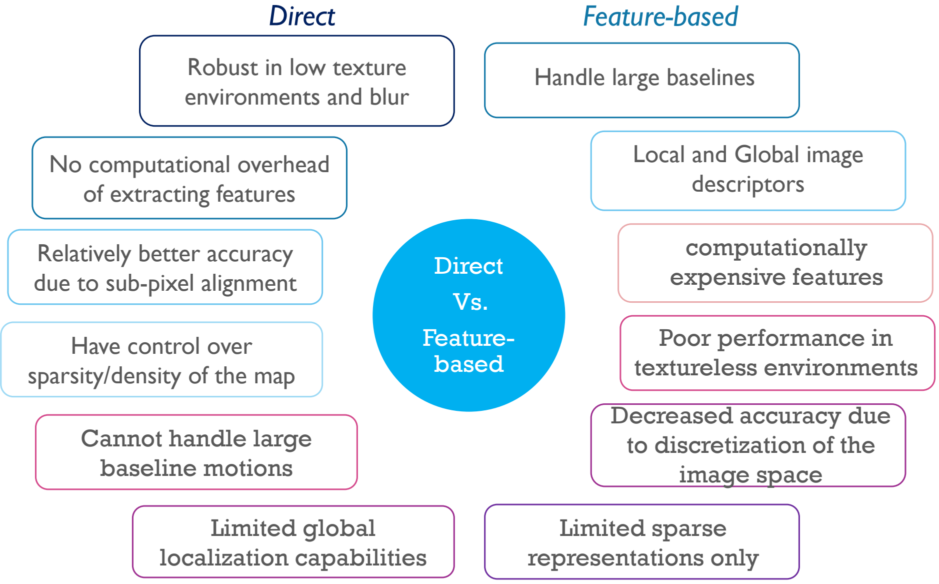
b. Feature-based methods

Feature-based methods can handle relatively large baselines between frames and tolerate, to an extent, illumination changes. When compared to direct methods, they have a relatively compact scene representation that allows for failure recovery, loop closure, and global/local optimizations. However, the extraction processes that make them resilient to these factors are generally computationally expensive. For real-time operation constraints, most systems employ a trade-of between a feature type to use in one hand, and the robustness and resilience to environment factors on the other. To mitigate this constraint, other systems, such as the work of [154], resort to parallelized GPU implementations for feature detection and extraction.

On the other hand, the density of the features is inversely correlated with the data association performance: as the density of the extracted features increases, their distinctiveness decreases, leading to ambiguous feature matches, therefore feature-based methods are limited to sparse representations. Their performance is brittle in low textured or self-repeating environments and are unstable for far-away features, or when using features that have been observed with a small parallax. More importantly, data association in feature based methods is established between features extracted at discretized locations in the images, resulting in an inferior accuracy and larger drift than the direct framework.

Another disadvantage of feature-based methods is that even the top performing feature descriptors are limited in the amount of scene change (lighting and viewpoint) they can handle before failure. Feature matching is also prone to failure in similar-self-repeating texture environments, where a feature in I_1 can be ambiguously matched to multiple other features in I_2. Outliers in the data association module can heavily degrade the system performance by inducing errors in both the camera poses and the generated map until the point of failure.

A summary of the comparison between direct and feature-based methods is shown in Fig. 3.9.



Direct
Feature-based
Robust in low texture environments and blur
Handle large baselines
No computational overhead of extracting features
Local and Global image descriptors
Relatively better accuracy due to sub-pixel alignment
computationally expensive features
Direct Vs. Feature-based
Have control over sparsity/density of the map
Poor performance in textureless environments
Cannot handle large baseline motions
Decreased accuracy due to discretization of the image space
Limited global localization capabilities
Limited sparse representations only

Figure 3.9: Comparison between the different traits of direct vs. feature-based KSLAM systems.

3.8.2 Traits of the different data association types

In general, establishing data associations remains one of the biggest challenges in KSLAM. Systems that limit the search range along the Epipolar line using the observed depth information, implicitly assume a relatively smooth depth distribution. Violating this assumption (*i.e.*, when the scene includes significance variance in the observed depth) causes the 2D features corresponding to potential future 3D landmarks to fall outside the boundaries of the Epipolar segment, and the system ends up neglecting them. Other limitations for data association arise from large erratic accelerations in the camera's motion, also causing features to fall outside the scope of the search window. Such a scenario is common in real-life applications and when the camera is operated by an untrained user. Image pollution with motion blur also negatively impacts the performance of data association methods to the point of failure.

Erroneous data association is also a very common problem that can cause false positives in self repeating environments. Most current implementations of data association address this problem through a bottom-up approach, where low level information from image pixels or from features, is used to establish correspondences. To mitigate some of these issues, a number of systems have attempted to use geometric features of a higher level, such as lines [9, 184, 37, 84, 117, 159, 161, 167], super-pixels, or planar features[28, 29, 113], or priors on 3D shapes in the scene [55]. Recent advances in machine learning are promising alternatives to remedy some of the data association issues by automatically learning to extract and match features using the methods suggested and not limited to [148, 160, 65, 180, 171, 145].

3.8.3 Traits of initialization

Aside from the random depth initialization of LSD SLAM and DSO, all the suggested methods described above suffer from degeneracies under certain conditions, such as under low-parallax movements of the camera, or when the scene's structure assumption—Fundamental matrix's assumption for general non-planar scenes or the Homography assumption of planar scenes—is violated.

PTAM's initialization procedure is brittle and remains tricky to perform, especially for inexperienced users. Furthermore, it is subject to degeneracies when the planarity of the initial scene's assumption is violated, or when the user's motion is inappropriate; thereby crashing the system, without means of detecting such degeneracies.

As is the case in PTAM, the initialization of SVO requires the same type of motion and is prone to sudden movements, as well as to non-planar scenes. Furthermore, monitoring the median of the baseline distance between features is not a good approach to automate the initial

Keyframe pair selection, as it is prone to failure against degenerate cases, with no means of detecting them.

The model based initialization of ORB SLAM attempts to automatically initialize the system by monitoring the baseline and the scene across a window of images. If the observed scene is relatively far, while the camera slowly translates in the scene, the system is not capable of detecting such scenarios, and fails to initialize.

While a random depth initialization from a single image does not suffer from the degeneracies of two view geometry methods, the depth estimation requires the processing of subsequent frames to converge, resulting in an intermediate tracking phase where the generated map is not reliable. It requires slow sideway-translational motion, and the convergence of the initialization is not guaranteed.

3.8.4 Traits of pose estimation

Systems relying on constant motion models, such as PTAM and ORB SLAM are prone to tracking failure when abrupt changes in the direction of the camera's motion occurs. While they both employ a recovery from such failures, PTAM's tracking performance is exposed to false positive pose recovery; as opposed to ORB SLAM that first attempts to increase the search window before invoking its failure recovery module.

Another limitation of feature-based pose estimation is the detection and handling of occlusions. As the camera translates in the scene, some landmarks in the background are prone to occlusions from objects in the foreground. When the system projects the 3D map points onto the current frame, it fails to match the occluded features, and counts them toward the camera tracking quality assessment. In extreme cases, the tracking quality of the system might be deemed bad and tracking failure recovery procedures are invoked even though camera pose tracking did not fail. Furthermore, occluded points are flagged as outliers and passed to the map maintenance module to be removed, depriving the map from valid useful landmarks that were erroneously flagged due to occlusions in the scene.

Other systems, that use the previously tracked pose as a prior for the new frame's pose, are also prone to the same limitations of constant velocity models. Furthermore, they require small displacements between frames, limiting their operation to relatively expensive high frame-rate cameras (typically $> 70fps$) such that the displacement limitation is not exceeded. Another limitation of these methods is inherent from their use of direct data association. Their tracking module is susceptible to variations in the lighting conditions. To gain some resilience to lighting changes in direct methods, [45] suggest an off-line photometric calibration process to parametrize and incorporate lighting variations within the camera pose optimization process.

A common limitation that plagues most tracking modules is the presence of dynamic objects in the observed environment. As most KSLAM systems assume a static scene, the tracking modules of most systems suffer from tracking failures: a significantly large dynamic object in the scene could trick the system into thinking that the camera itself is moving, while it did not move relative to the environment. Small, slowly moving objects can introduce noisy outlier landmarks in the map and require subsequent processing and handling to be removed. Small, fast moving objects on the other hand, don't affect the tracking module as much. 2D features corresponding to fast moving small objects tend to violate the epipolar geometry of the pose estimation problem, and are easily flagged and removed from the camera pose optimization thread; however, they can occlude other landmarks. To address the effects of occlusions and dynamic objects in the scene, [154] suggest, for slowly varying scenes, to sample based on previous camera pose locations in the image that are not reliable, and discard them during the tracking phase.

3.8.5 Traits of map generation

A major limitation in the N-view triangulation method is the requirement of a significant baseline separating two viewpoints observing the same feature. Hence, it is prone to failure when the camera's motion is made of pure rotations. To counter such modes of failure, DT SLAM introduced 2D landmarks that can be used to expand the map during pure rotations, before they are triangulated into 3D landmarks. However, the observed scene during the rotation motion is expected to be re-observed with more baseline, for the landmarks to transition from 2D to 3D. Unfortunately, in many applications this is not the case; for example, a camera mounted on a car making a turn cannot re-observe the scene, and eventually tracking failure occurs. DT SLAM addresses such cases by generating a new sub map and attempts to establish connections to previously created sub-maps by invoking a thread to look for similar Keyframes across sub-maps, and establish data associations between them. In the meantime, it resumes tracking in the new world coordinate frame of the new sub-map. This however renders the pose estimates obsolete; at every tracking failure the tracking is reset to the new coordinate frame, yielding useless pose estimates until the sub-maps are joined together, which may never occur.

In filter based triangulation methods, outliers are easily flagged as landmarks whose distribution remain approximately uniform after a number of observations have been incorporated in the framework. This reduces the need for a subsequent processing step to detect and handle outliers. Also, landmarks at infinity suffer from parallax values that are too small for triangulation purposes; but yet, can be used to enhance the camera's rotation estimates, and kept in the map, and are transitioned from infinity to the metric map, when enough parallax between the views observing them is recorded. However, these benefits come at the expense of increased complexity in implementing a probabilistic framework, which keeps track and updates the uncertainty in the feature depth distribution.

Furthermore, while the dense and semi-dense maps can capture a much more meaningful representation of a scene than a sparse set of 3D landmarks, the added value is diminished by the challenges of handling immense amounts of data in 3D. Hence, there is a need for additional higher level semantic information to reason about the observed scene, and to enhance the system's overall performance. While monocular SLAM systems have been shown to improve the results of semantic labeling [127], the feedback from the latter to the former remains a challenging problem. Previous work on the matter include but are not limited to [90, 55, 50, 143, 170].

3.8.6 Traits of BA/PGO/map maintenance

Pose Graph Optimization (PGO) returns inferior results to those produced by GBA, while PGO optimizes only for the Keyframe poses—and accordingly adjusts the 3D structure of landmarks—GBA and LBA jointly optimize 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 the other methods. PGO is often employed during loop closure as the computational cost of running a full BA is often intractable on large-scale loops; however, pose graph optimization may not yield optimal result if the errors accumulated over the loop are distributed along the entire map, leading to locally induced inaccuracies in regions that were not originally wrong.

3.8.7 Traits of global localization

For successful re-localization or loop detection, global localization methods employed by PTAM, SVO and DT SLAM require the camera's pose to be near the previously seen Keyframe's pose, and would otherwise fail when there is a large displacement between the two. Furthermore, they are highly sensitive to any change in the lighting conditions of the scene, and may yield many false positives when the observed environment is composed of self-repeating textures.
On the other hand, methods that rely on *BoVW* representations are more reliable however, *BoVW* do not keep track of the feature's geometric distribution in the image; this is especially detrimental in self-repeating environments and require subsequent geometric checks to prevent outliers. Furthermore, the high dimensional features are susceptible to failure when the training set recorded in the *BoVW* dictionary is not representative of the working environment in which the system is operating.

3.9 VSLAM: Towards a Hybrid Scene Representation, Feature-based - direct maps interactions

When the corresponding pros and cons of both feature-based and direct frameworks are placed side by side Fig. 3.9, a pattern of complementary traits emerges. For example, direct methods require small baseline motions to ensure convergence, whereas feature-based methods are more reliable at relatively larger baselines. Furthermore, due to sub-pixel alignment accuracy, direct methods are relatively more accurate but suffer from an intractable amount of data in large environments; on the other hand feature-based methods suffer from relatively lower accuracies due to the discretization of the input space but have a suitable scene representation for a SLAM formulation, which enables feature-based methods to easily maintain a reusable global map, perform loop closures and failure recovery. Therefore an ideal system should exploit both direct and feature-based advantages to benefit from the direct formulation accuracy and robustness while making use of feature-based methods for large baseline motions, maintaining a reusable global map, and reducing drifts through loop closures. Furthermore, a hybrid feature-based-direct framework allows for the metric representation to be locally semi-dense and globally sparse, facilitating interactions with other types of representations such as topological and/or semantic, while maintaining scalability and computational tractability.

In light of these observations, a fusion between a feature-based and a direct metric representation is an essential first step towards multi-level hybrid *VSLAM* and as such is the main topic presented in this thesis.

3.10 Survey update

Since the release of **Keyframe-based monocular SLAM: design, survey, and future directions**, and the subsequent work in **Towards a Hybrid Scene Representation in VSLAM** [174] that theorized the potential and capabilities of using multiple matching paradigms in Hybrid systems, several papers were published on the matter. [168] evaluated the state of the art direct and Indirect systems, effectively quantifying the discretization disadvantage of Indirect methods, and revealing the motion bias: Indirect methods performs worse when the camera is moving forwards than backwards due to N-view Epipolar triangulation limitations. [1]

[1] N-view triangulation requires a large baseline to reliably triangulate features. When the camera is moving forward, the points that are close to the camera center (unreliable parallax) are more abundant than those around the image edge (reliable parallax); the situation is reversed when the camera is moving backward, causing the motion bias.

Direct Sparse Odometry with Loop Closure (LDSO) [56] used the Direct Framework at frame-rate and generated a *BoVW* representation of ORB features at keyframe-rate and queried it for loop closure detection.

Direct Sparse Odometry with Rolling Shutter [144] attempted to mitigate the sensitivity of direct methods to rolling shutter cameras by introducing a time component in DSO's optimization, with the assumption that the time t when a pixel is read out is linearly related to the vertical y-coordinate of the pixel in the image. However the introduction of the time factor has a large impact on the computational cost, requires a large number of new variables to be optimized, and requires more keyframes to maintain tracking. It is therefore about four times slower than DSO and can only operate in near-real-time conditions.

Loosely-Coupled Semi-Direct Monocular SLAM (LCSD) [92] implemented a cascaded approach where marginalized keyframes from DSO are fed into ORB SLAM 2 to maintain a global map representation. Since both types of residuals are loosely coupled they did not make use of the complementary nature between the different types of features and they ended up falling short of DSO's default performance.

Direct Sparse Mapping (DSM) [191] extended the capabilities of Direct maps to reuse previously built parts of the map. Keyframes along with their corresponding images and depth values are kept even after marginalization and are queried through a covisibility graph to contribute residuals to the currently active keyframe, thereby allowing previously mapped points to influence the current state of the system. However keeping track of the keyframes data is memory expensive and re-projecting previously marginalized keyframes to the currently active keyframes is computationally expensive, causing a large slow down in the back-end side. Alongside the global map reuse, DSM proposed the use of a t-distribution to weight the residuals in the optimization, thereby improving the overall system's performance.

Vitamin-E [172] is very close in spirit to SVO [52], where indirect features are extracted but matched using dominant flow rather than their descriptors. Vitamin-E also employs a sub-optimal optimization that iterates between pose only and structure only optimizations similar to that proposed in SVO albeit under a different naming. A mesh is then fitted to the 2D projection of the reconstructed map on each keyframe using the method proposed in [64], which are in turn fused across multiple keyframes using a TSDF formulation [85] to return a dense monocular reconstruction of the scene in real-time.

Alternatives to the traditional point-wise geometric entities used in all of the systems mentioned so far, were also suggested. Some papers tackled the use of lines and hybrid point-line residuals in solving *VSLAM*. Such methods include [188, 61, 134, 169, 163, 93, 46] which extract lines from images and define a geometric line re-projection error. The line re-projection error is in turn used in a multi-objective optimization with the traditional point re-projection error.

The end result is systems that are robust to texture-deprived man-made environments (such as hallways); however, extracting and maintaining lines at frame-rate is computationally expensive. To mitigate the high computational cost, [23] perform global canny edge alignment across two frames, circumventing the need to explicitly extract and maintain the lines. While the authors report on promising results performing global alignment is brittle to any outliers and dynamic objects in the scene.

Alongside the development of the aforementioned monocular systems, the research community has put forward in the past two years a large number of papers on sensor fusion systems; such systems include a combination of visual, inertial, Lidar and event based cameras. A survey on such systems is beyond the scope of this work, but the interested reader is referred to [77] for more information or to [18] and [139] for recent open source implementations of such sensor fusion systems.

Finally, *BoVW* has been the go-to method for global loop closure detection in most previously mentioned *VSLAM* systems. However recent deep learning powered supervised and unsupervised image retrieval methods have shown promising performance, especially under varying conditions (severe illumination change, seasonal change, *etc*.) which are typically challenging for *BoVW* methods. While they are still not mature enough to replace *BoVW* in practice, image retrieval is a highly active area of research and the reader is referred to [185, 22, 116, 138] for more up-to-date information on the topic.

Chapter 4

Feature Assisted Direct Monocular Odometry

When a *VO* system is calibrated photometrically, and images are captured at high rates, direct methods have been shown to outperform feature-based ones in terms of accuracy and processing time; they are also more robust to failure in feature-deprived environments. On the downside, direct methods rely on heuristic motion models to seed an estimate of camera motion between frames; in the event that these models are violated (*e.g.*, erratic motion), direct methods easily fail. In real-life applications, the motion of a hand-held or head-mounted camera is predominantly erratic thereby violating the motion models used, causing large baselines between the initializing pose and the actual pose, which in turn negatively impacts the VO performance.

As the camera transitions from a leisure device to a viable sensor, robustifying Direct *VO* to real-life scenarios becomes of utmost importance. In that pursuit, *FDMO* is a hybrid VO that makes use of Indirect residuals to seed the Direct pose estimation process. Two variations of *FDMO* are presented: one that only intervenes when failure in the Direct optimization is detected, and another that performs both Indirect and Direct optimizations on every frame. Various efficiencies are also introduced to both the feature detector and the Indirect mapping process, resulting in a computationally efficient approach. Finally, an experimental procedure designed to test the resilience of VO to large baseline motions is used to validate the success of the proposed approach. The content presented in this chapter was published in [177, 176, 59].

The *VO* problem formulates camera pose estimation as an iterative optimization of an objective function. Central to each optimization step is data association, where cues (features) from a new image are corresponded to those found in previous measurements. The type of cues used split VO systems along three different paradigms: Direct, Indirect, or a hybrid of both, with each using its own objective function and exhibiting dissimilar but often complementary traits (Chap.

3) summarized in 4.1.

An underlying assumption to all paradigms is the convexity of their objective functions, allowing for iterative Newton-like optimization methods to converge. However, none of the objective functions are convex; and to relax this limitation, VO systems assume local convexity and employ motion models to perform data association, as well as to seed the optimization. Typical motion models include a constant velocity model (CVMM) [83] and [124], or a zero motion model [52], or, in the case the CVMM fails, a combination of random motions [41]. In real-life applications (*e.g.*, Augmented Reality), the motion of a hand-held or head-mounted camera is predominantly erratic, easily violating the assumed motion models, and effectively reducing the *VO* performance from what is typically reported in most benchmark experiments. In fact, erratic motions can be viewed as a special case of large motions that induces discrepancies between the assumed motion model and the actual camera motion. The error induced is also further amplified when a camera is arbitrarily accelerating or decelerating with low frame-rates, causing errors in the VO data association and corresponding optimization. Similar effects are observed when the computational resources are slow, and VO cannot add keyframes in time to accommodate fast motions; VO is then forced to perform pose estimation across keyframes separated by relatively large distances.

Table 4.1: Comparison between Direct and Indirect methods. The more of the symbol +, the higher the attribute.

Trait	**Feature-based**	**Direct**
Large baseline	+++	+
Computational efficiency	++	+++
Robustness to Feature Deprivation	+	+++
Recovered scene point density	+	+++
Accuracy	+	+++
Optimization Non-Convexity	+	++
Global Localization	+++	None
Map reuse	+++	None

The impact large motions can have depends on various components of a *VO*; namely, on the resilience of the data association step, on the radius of convergence of the objective function, and on the ability of the VO system to detect and account for motion model violations.

In an effort to handle large baseline motions, *FDMO* [177] performs photometric image alignment for pose estimation at frame-rate, and invokes an Indirect pose estimation only when tracking failure is detected. The reason for not using Indirect measurements on every frame in FDMO is to avoid the large computational costs associated with extracting features; as a consequence, *FDMO* maintains the computational efficiency of Direct methods but requires a heuristic approach to detect local minima in the Direct objective function optimization.

To alleviate the need for a heuristic failure detection approach, a variant FDMO-f [176] (Feature Assisted Direct Monocular Odometry at Frame-rate) is also proposed. In FDMO-f the expensive computational cost associated with feature extraction is overcome by an efficiently parallelizable feature detector, allowing the use of both types of measurements sequentially on every frame, and requiring various modifications to the overall architecture of the *VO* system.

The contributions of the Feature assisted Direct Odometry includes:

- The ability to use both Direct and Indirect residuals when needed, or on every frame via a computationally cheap feature detector.
- Resilience to large baseline motions.
- Achieves the sub-pixel accuracy of Direct methods.
- Maintains the robustness of Direct methods to feature-deprived environments.
- A computationally efficient Indirect mapping approach.
- An experimental procedure designed to evaluate the robustness of VO to large baseline motions.

While various hybrid (Direct and Indirect) systems were previously proposed in the literature, few manage to successfully integrate the advantages of both paradigms. For example, [52] did not extract feature descriptors, it relied on the direct image alignment to perform data association between the features. While this led to significant speed-ups in the processing required for data association, it could not handle large baseline motions; as a result, their work was limited to high frame-rate cameras, which ensured frame-to-frame motion is small. On the other hand, both [88] and [2] adopted a feature-based approach as a front-end to their system, and subsequently optimized the measurements with a direct image alignment; as such, both systems suffer from the limitations of the feature-based framework, *i.e.* they are subject to failure in feature-deprived environments and therefore not able to simultaneously meet all of the desired traits of Table 4.1. To address this issue, both systems resorted to stereo cameras.

FDMO is a system designed to complement the advantages of both direct and featured based techniques to achieve sub-pixel accuracy, robustness in feature deprived environments, resilience to erratic and large inter-frame motions, all while maintaining a low computational cost at frame-rate. Efficiencies are also introduced to decrease the computational complexity of the feature-based mapping part. FDMO shows an average of 10% reduction in alignment drift, and 12% reduction in rotation drift when compared to the best of both ORB-SLAM and DSO, while achieving significant drift (alignment, rotation & scale) reductions (51%, 61%, 7% respectively) going over the same sequences for a second loop. FDMO is further evaluated on the EuroC dataset and was found to inherit the resilience of feature-based methods to erratic motions, while maintaining the accuracy of direct methods.

4.1 FDMO

To capitalize on the advantages of both feature-based and direct frameworks, FDMO consists of a local direct visual odometry, assisted with a feature-based map, such that it may resort to feature-based odometry only when necessary. Therefore, FDMO does not need to perform a computationally expensive feature extraction and matching step at every frame. During its feature-based map expansion, FDMO exploits the localized keyframes with sub-pixel accuracy from the direct framework, to efficiently establish feature matches in feature-deprived environments using restricted epipolar search lines.

Similar to DSO, FDMO's local temporary map is defined by a set of seven direct-based keyframes and 2000 active direct points. Increasing these parameters was found by [41] to significantly increase the computational cost without much improvement in accuracy. Direct keyframe insertion and marginalization occurs frequently according to conditions described in [41]. In contrast, the feature-based map is made of an undetermined number of keyframes, each with an associated set of features and their corresponding ORB descriptors $\Phi(x, Q(x))$.

4.1.1 Notation

To address any ambiguities, the superscript d will be assigned to all direct-based measurements and f for all feature-based measurements; not to be confused with underscript f assigned to the word frame. Therefore, M^d refers to the temporary direct map, and M^f to the feature-based map, which is made of an unrestricted number of keyframes κ^f and a set of 3D points X^f. I_{f_i} refers to the image of frame i and T_{f_i,KF^d} is the $se(3)$ transformation relating frame i to the latest active keyframe KF in the direct map. We also make the distinction between z referring to depth measurements associated with a 2D point x and Z refering to the Z coordinate of a 3D point. Finally, the symbol π is used to denote the pinhole projection function mapping a point from the camera coordinate frame to the image coordinate frame.

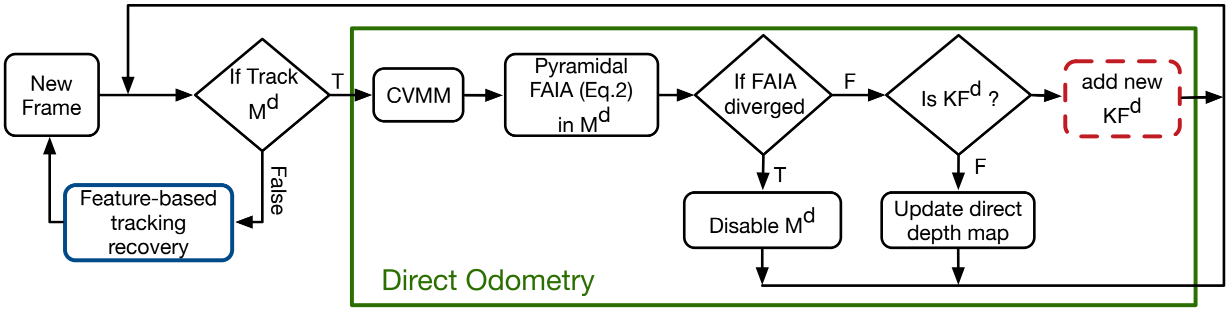


Figure 4.1: Front-end flowchart of FDMO. It runs on a frame by frame basis and uses a constant velocity motion model (CVMM) to seed a Forward Additive Image Alignment (FAIA) to estimate the new frame's pose and update the direct map depth values. It also decides whether to invoke the feature-based tracking or add a new keyframe into the system. The blue (solid line) and red (dashed line) boxes are further expanded in figures 4.2 and 4.3 respectively.

4.1.2 Odometry

Direct image alignment

Frame by frame operations are handled by the flowchart described in Fig. 4.1. Similar to [41], newly acquired frames are tracked by minimizing

$$\underset{T_{f_i,KF^d}}{\operatorname{argmin}} \sum_{x^d} \sum_{x \in N(x^d)} Obj(I_{f_i}(\omega(x,z,T_{f_i,KF^d}) - I_{KF^d}(x,z))), \qquad (4.1)$$

where f_i is the current frame, KF^d is the latest added keyframe in M^d, $x^d \in \Omega I_f$ is the set of image locations with sufficient intensity gradient and an associated depth value d. $N(x^d)$ is the set of pixels neighbouring x^d and $w(\cdot)$ is the projection function that maps a 2D point from f_i to KF^d.

The minimization is seeded from a constant velocity motion model (CVMM). However, erratic motion or large motion baselines can easily violate the CVMM, erroneously initializing the highly-non convex optimization, and yielding unrecoverable tracking failure. We detect tracking failure by monitoring the RMSE of (4.1) before and after the optimization; if the ratio $\frac{RMSE_{after}}{RMSE_{before}} > 1+\varepsilon$ we consider that the optimization has diverged and we invoke the feature-based tracking recovery, summarized in the flowchart of Fig. 4.2. The ε is used to restrict feature-based intervention when the original motion model used is accurate, a value of $\varepsilon = 0.1$ was found as a good trade-off between continuously invoking the feature-based tracking and not detecting failure in the optimization. To avoid extra computational cost, feature extraction and matching is not performed on a frame by frame basis, it is only invoked during feature-based tracking recovery and feature-based KF insertion.

Feature-based tracking recovery

The proposed feature-based tracking operates in M^f. When direct tracking diverges, FDMO considers the CVMM estimate to be invalid and seeks to estimate a new motion model using the feature-based map. The proposed feature-based tracking recovery is a variant of the global re-localization method proposed in [124]; we first start by detecting $\Phi f_i = \Phi(x^f, Q(x^f))$ in the current image, which are then parsed into a vocabulary tree. Since we consider the CVMM to be invalid, we fall back on the last piece of information the system was sure of before failure: the pose of the last successfully added keyframe. We define a set κ^f of feature-based keyframes KF^f connected to the last added keyframe KF_d through a covisibility graph [150], and their associated 3D map points X^f.

Blind feature matching is then performed between Φf_i and all keyframes in κ^f, by restricting feature matching to take place between features that exist in the same node in a vocabulary tree [54]; this is done to reduce the computational cost of blindly matching all features. Once data association is established between f_i and the map points, we set up an EPnP (Efficient Perspective-n-Point Camera Pose Estimation) [96] to solve for an initial pose T_{f_i} using 3D-2D correspondences in an non-iterative manner. The new pose is then used to define a 5×5 search window in f_i surrounding the projected locations of all 3D map points $X^f \in \kappa^f$. Finally, the pose T_{f_i} is refined through the traditional feature-based optimization:

$$\underset{T_{f_i}}{\operatorname{argmin}} \sum_{X^f \in M^f} Obj(\pi(X^f, T_{f_i}) - obs), \tag{4.2}$$

where $obs \in \mathbb{R}^2$ is the feature's matched location in f_i, found through descriptor matching. To achieve sub-pixel accuracy, the recovered pose T_{f_i} is then converted into a local increment over the pose of the last active direct keyframe, and then further refined in a direct image alignment optimization (4.1).

Note that the EPnP step could have been skipped in favour of using the last correctly tracked keyframe's position as a starting point; however, data association would then require a relatively larger search window, which in turn increases its computational burden in the subsequent step. Data association using a search window was also found to fail when the baseline motion was relatively large.

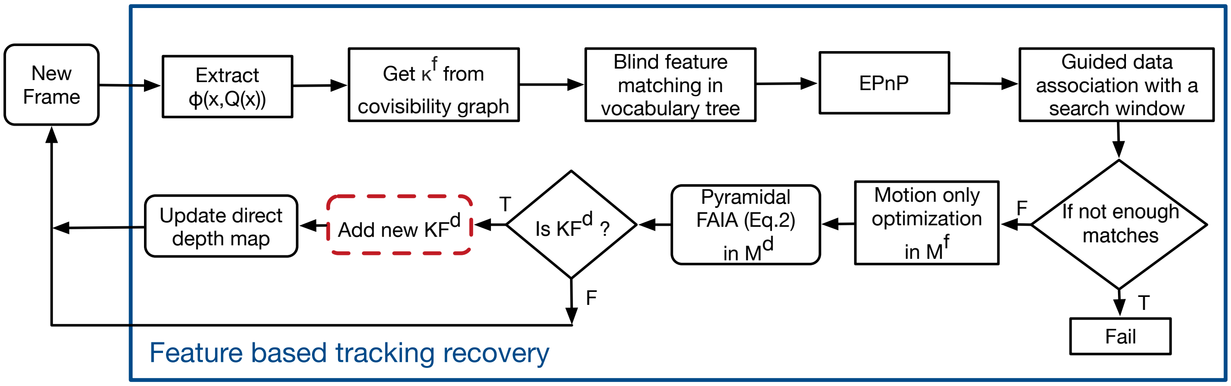

New Frame
Extract ϕ(x,Q(x))
Get κ^f from covisibility graph
Blind feature matching in vocabulary tree
EPnP
Guided data association with a search window
If not enough matches
T
Fail
F
Motion only optimization in M^f
Pyramidal FAIA (Eq.2) in M^d
Is KF^d ?
T
Add new KF^d
Update direct depth map
F
Feature based tracking recovery

Figure 4.2: FDMO Tracking Recovery flowchart. Only invoked when direct image alignment fails, it takes over the front end operations of the system until the direct map is re-initialized. FDMO's tracking recovery is a variant of ORB-SLAM's global failure recovery that exploits the information available from the direct framework to constrain the recovery procedure locally. We start by extracting features from the new frame and matching them to 3D features observed in a set of keyframes κ^f connected to the last correctly added keyframe from KF^d. Efficient Perspective-n-Point (EPnP) camera pose estimation is used to estimate an initial guess which is then refined by a guided data association between the local map and the frame. The refined pose is then used to seed a Forward additive image alignment step to achieve sub-pixel accuracy.

4.1.3 Mapping

FDMO's mapping process is composed of two components: direct, and feature-based as described in Fig. 4.3. The direct map propagation used here is the same as suggested in [41]; however we expand its capabilities to propagate the feature-based map. When a new keyframe is added to M^d, we create a new feature-based keyframe KF^f that inherits its pose from KF^d. $\Phi KF^f(x^f, Q(x^f))$ is then extracted and data association takes place between the new keyframe and a set of local keyframes κ^f surrounding it via epipolar search lines. The data association is used to keep track of all map points X^f visible in the new keyframe and to triangulate new map points.

To ensure an accurate and reliable feature-based map, typical feature-based methods employ local bundle adjustment (LBA)[121] to optimize for both the keyframes poses and their associated map points; however, employing an LBA may generate inconsistencies between both map representations, and is computationally expensive; instead, we make use of the fact that the new keyframe's pose is already locally optimal, to replace the typical local bundle adjustment with a

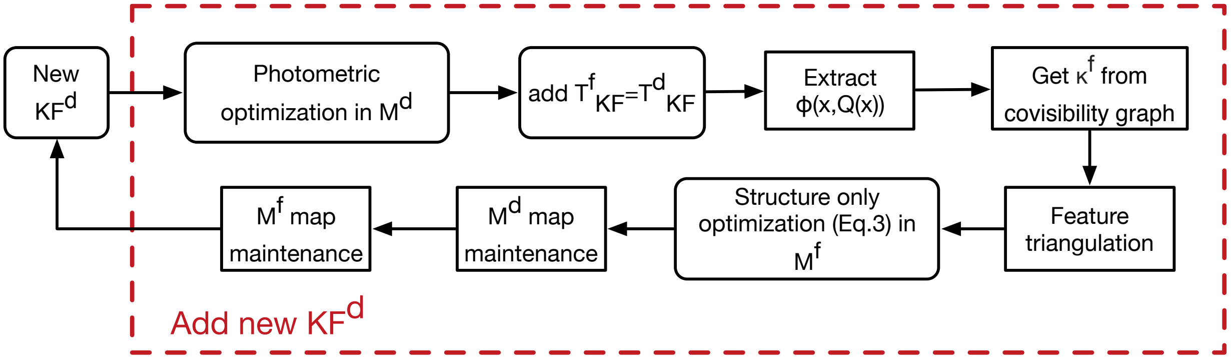


Figure 4.3: The mapping flowchart, is a variant of DSO's mapping backend; we augment its capabilities to expand the feature-based map with new KF^f. It operates after or parallel to the direct photometric optimization of DSO, by first establishing feature matches using restricted epipolar search lines; the 3D feature-based map is then optimized using a computationally efficient *structure-only* bundle adjustment, before map maintenance ensures the map remain outliers free.

computationally less demanding structure only optimization defined for each 3D point X_j^f:

$$\underset{X_j^f}{\operatorname{argmin}} \sum_{i \in \kappa^f} Obj(x_{i,j}^f - \pi(T_{KF_i^f} X_j^f)), \tag{4.3}$$

where X_j spans all 3D map points observed in all keyframes $\in \kappa^f$. We use ten iterations of Gauss-Newton to minimize the normal equations associated with (4.3) which yield the following update rule per 3D point X_j per iteration:

$$X_j^{t+1} = X_j^t - (J^T W J)^{-1} J^T W e, \tag{4.4}$$

where e is the stacked reprojection residuals e_i associated with a point X_j and its found match x_i in keyframe i. J is the stacked Jacobians of the reprojection error which is found by stacking:

$$J_i = \begin{bmatrix} \frac{f_x}{Z} & 0 & -\frac{f_x X}{Z^2} \\ 0 & \frac{f_y}{Z} & -\frac{f_y Y}{Z^2} \end{bmatrix} R_{KF_i}, \tag{4.5}$$

and R_{KF_i} is the 3×3 orientation matrix of the keyframe observing the 3D point X_j. Similar to ORB-SLAM, W is a block diagonal weight matrix that down-weighs the effect of residuals computed from feature matches found at high pyramidal levels[1] and is computed as

$$W_{ii} = \begin{bmatrix} Sf^{-2n} & 0 \\ 0 & Sf^{-2n} \end{bmatrix} \tag{4.6}$$

where Sf is the scale factor used to generate the pyramidal representation of the keyframe (we use $Sf = 1.2$) and n is the pyramid level from which the feature was extracted ($0 \leq n < 8$). The Huber norm is also used to detect and remove outliers. We have limited the number of iterations in the optimization of (4.3) to ten, since no significant reduction in the feature-based re-projection error was recorded beyond that.

4.1.4 Feature-based map maintenance

To ensure a reliable feature-based map, the following practices are employed. For proper operation, direct methods require frequent addition of keyframes, resulting in small baselines between the keyframes, which in turn can cause degeneracies if used to triangulate feature-based points. To avoid numerical instabilities, following the suggestion of [83], we prevent feature triangulation between keyframes with a $\frac{baseline}{depth}$ ratio less than 0.02 which is a trade-off between numerically unstable triangulated features and feature deprivation problems. We exploit the frequent addition of keyframes as a feature quality check. In other words, a feature has to be correctly found in at least 4 of the 7 keyframes subsequent to the keyframe it was first observed in, otherwise it is considered spurious and is subsequently removed. To ensure no feature deprivation occurs, a feature cannot be removed until at least 7 keyframes have been added since it was first observed. Finally, similar to [124] a keyframe with ninety percent of its points shared with other keyframes is removed from M^f only once marginalized[2] from M^d

The aforementioned practices ensure that sufficient reliable map points and features are available in the immediate surrounding of the current frame, and that only necessary map points and keyframes are kept once the camera moves on.

[1] Features matched at higher pyramidal levels are less reliable.

[2] Marginalization is the process of splitting a multivariate Gaussian distribution $\mathbf{X}$ into $\left[\mathbf{x}_a, \mathbf{x}_b\right]$, where $\mathbf{x}_b$ represent state variables that are no longer useful for the system (no longer visible in the camera's field of view) and should therefore be removed from the state vector. Marginalization then proceeds by computing the conditional distribution $p(\mathbf{x}_a|\mathbf{x}_b)$ using the Schur complement, after which the new state vector becomes $\mathbf{X} = [\mathbf{x}_a]$. For more details on marginalization the reader is referred to [41].

4.2 FDMO-F

FDMO-f addresses the dependence of FDMO on a heuristic failure detection test, by using both Direct and Indirect residuals on every frame. To overcome the computational expenses of extracting features, an efficient and parallelizable alternative to the feature detector employed in typical Indirect methods is proposed. Finally, an Indirect map quality feedback from the frame-to-frame feature matches is used to introduce various efficiencies in the mapping process, resulting in a 50 % faster Indirect mapping process while maintaining the same performance.

4.2.1 Feature Extraction

Several design considerations are taken into account when designing a feature detector for a SLAM algorithm. In particular, the detected keypoints should be repeatable, discriminable, and homogeneously distributed across the image. ORB SLAM [124] takes into account these considerations by extracting features using an octomap, which adapts the FAST [140] corner detector thresholds to different image regions. However, this process is computationally involved; for example, it takes 12 ms on our current hardware to extract 1500 features along with their ORB descriptors from 8 pyramid levels. Unfortunately, this means that the feature extraction alone requires more time than the entire Direct pose estimation process. Several attempts at parallelizing ORB SLAM's feature extraction process were made in the literature; Zhang *et al*. [183] attempted parallelizing the extraction process on a CPU resulting in no speedups, and ended up adopting a CPU - GPU acceleration to reduce the computational cost by 40 %. Similar results were also reported in [13].

In contrast to the aforementioned methods, we forego the adaptive octomap approach in favor of an efficiently parallelizable feature detector implementation on a CPU. Our proposed feature detector first computes the image pyramid levels, which are then distributed across parallel CPU threads. Each thread operates on its own pyramid level independent of the others. The following describes the operations performed by each thread: FAST corners [140] are first extracted with a low cutoff threshold, resulting in a large number of noisy corners with an associated *corner-ness* score (the higher the score the more discriminant). The corners are then sorted in descending order of their scores and accordingly added as features, with each added corner preventing other features from being added in a region of 11×11 pixels around its location. This ensures that the most likely repeatable corners are selected, while promoting a homogeneous distribution across the image. The area 11×11 is chosen to ensure small overlap between the feature descriptors, thereby improving their discriminability. The features orientation angles are then computed and a Gaussian kernel is applied before extracting their ORB descriptors.

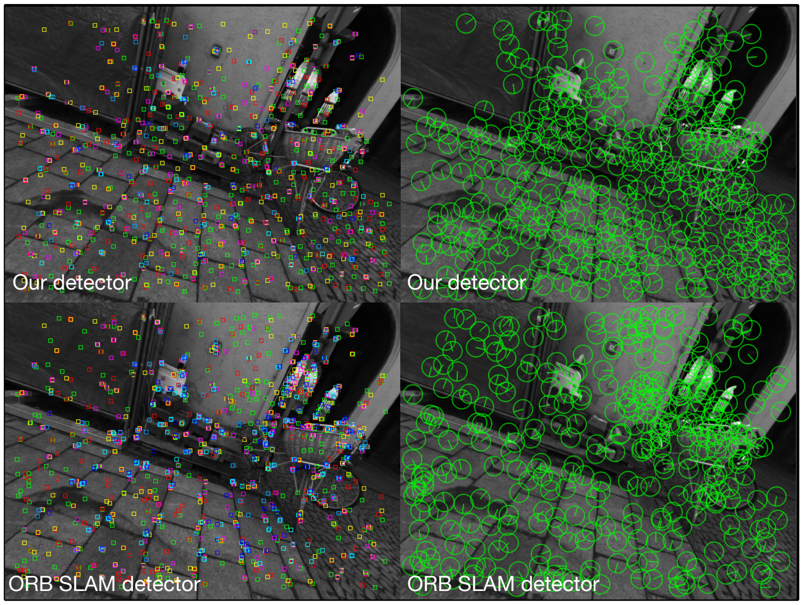


Figure 4.4: The top left image shows the features detected by our approach whereas the bottom left shows the features detected by ORB SLAM. The different colors correspond to the different pyramid levels. The right images show the image areas that were used to compute the descriptors of the features from the lowest pyramid level. It can be seen how our feature detector results in a better feature distribution; in contrast, ORB SLAM's detector extracts a large number of features from the same area (right part of the image) resulting in a large overlap between the feature descriptors.

When compared to the 12 ms required by ORB SLAM's detector, our proposed feature detector extracts the same number of features in 4.4 ms using the same CPU, making feature extraction on every frame feasible. A side by side comparison between the extracted features from ORB SLAM and our proposed detector are shown in Fig. 4.4.

4.2.2 Pose Estimation

Unlike FDMO, FDMO-f extracts and uses Indirect features on every frame. The CVMM from frame-to-frame pose is usually accurate enough to establish feature correspondences with the local map using a search window. However, if few matches are found, the motion-model-independent pose recovery described in section 4.1.2 is used to obtain a more accurate pose for feature matching to take place. The frame pose is then optimized using the Indirect features as described in Eq. 4.2 before being used to seed the direct image alignment process which ensures a sub-pixel level accuracy of the pose estimation process. FDMO-f pose estimation process is summarized in Fig. 4.5.

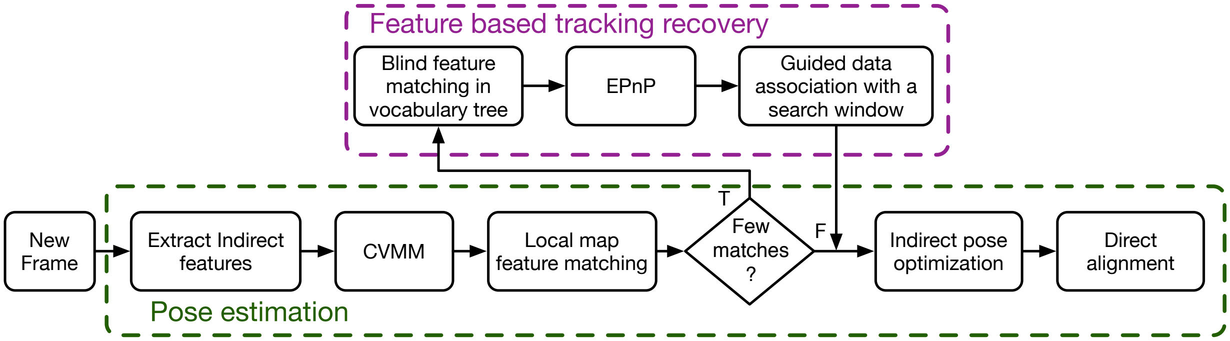

Feature based tracking recovery
Blind feature matching in vocabulary tree
EPnP
Guided data association with a search window
New Frame
Extract Indirect features
CVMM
Local map feature matching
Few matches ?
T
F
Indirect pose optimization
Direct alignment
Pose estimation

Figure 4.5: FDMO-f pose estimation flow chart.

4.2.3 Local Mapping

Similar to FDMO, FDMO-f uses hybrid keyframes that contains both Direct and Indirect measurements. However, unlike FDMO, keyframe insertion is triggered from two sources, either from (1) the Direct optimization using the criteria described in [41], or from (2) the Indirect optimization by monitoring the ratio of the inlier feature matches in the latest frame to that of the latest keyframe $r = \frac{inliers\,in\,inf_i}{inliers\,in\,KF}$; if r drops below a threshold (0.8), a keyframe is added, thus ensuring an ample amount of reliable Indirect features present in the local Indirect map M^f.

While all added keyframes are used to expand the set of direct map points x^d, they contribute differently to the Indirect mapping process depending on which criteria was used to create the keyframe. In particular, only keyframes that are triggered from the Indirect inlier ratio are used to triangulate new Indirect map points X^f. Keyframes that were not selected for Indirect triangulation are used to provide constraints on the previously added Indirect map points in the structure-only optimization. As a result, the modified mapping process is significantly more efficient than that of FDMO which did not have frame-to-frame feedback on the quality of the Indirect map, forcing it to triangulate new Indirect map points on every added keyframe. The final mapping process of FDMO-f is shown in Fig. 4.6.

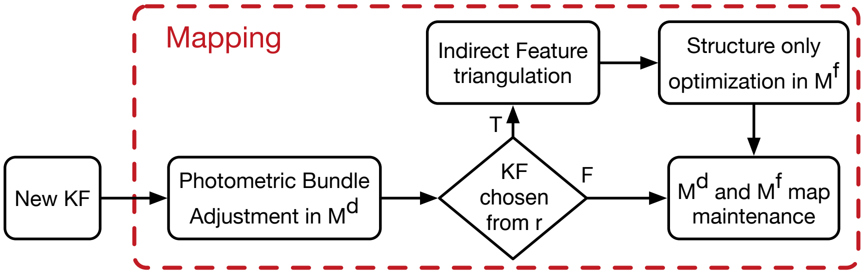

Mapping
New KF
Photometric Bundle Adjustment in M^d
KF chosen from r
T
F
Indirect Feature triangulation
Structure only optimization in M^f
M^d and M^f map maintenance

Figure 4.6: FDMO-f mapping flow chart.

4.3 Experimental Setup and Procedure

To evaluate FDMO's tracking robustness, experiments were performed on several well-known datasets [15] and [43], and both qualitative and quantitative appraisal was conducted. To further validate FDMO's effectiveness, the experiments were also repeated on state of the art open-source systems in both direct (DSO) and feature-based (ORB-SLAM2). For fairness of comparison, we evaluate ORB-SLAM2 as an odometry system (not as a SLAM system); therefore, similar to [41] we disable its loop closure thread but we keep its global failure recovery, local, and global bundle adjustments intact. Note that we've also attempted to include results from SVO [52] but it continuously failed on most datasets, so we excluded it.

4.3.1 Computational Cost

The experiments were conducted on an Intel Core i7-8700 3.4GHZ CPU, 32 GB memory; no GPU acceleration was used. Both DSO and ORB-SLAM2 consist of two parallel components, a tracking process (at frame-rate[3]) and a mapping process (keyframe-rate[4]). On the other hand, FDMO has three main processes: a direct tracking process (frame-rate), a direct mapping process (keyframe-rate), and a feature-based mapping process (keyframe-rate). Both of FDMO's mapping processes can run either sequentially for a total computational cost of 200 ms on a single thread, or in parallel on two threads. As Table 4.2 shows, the mean tracking time for FDMO remains almost the same that of DSO: we don't extract features at frame-rate; feature-based tracking in FDMO is only performed when the direct tracking diverges. The highest computational cost during FDMO tracking occurs when the recovery method is invoked, with a highest recorded processing time during our experiments of 35 ms. As for FDMO's mapping processes,

[3] occur at every frame.

[4] occur at new keyframes only.

its direct part remains the same as DSO, whereas the feature-based part takes 116 ms which is significantly less than ORB-SLAM2's feature-based mapping process that requires 198 ms.

The mean computational costs for DSO, FDMO, FDMO-f and ORB SLAM 2 are shown in table 4.2.

Table 4.2: Computational time (ms) for processes in DSO, FDMO, FDMO-f and ORB-SLAM2. Note that FDMO has two parallel threads for the mapping, the Direct mapping thread (averaging at 51 ms) and the Indirect mapping averaging at 116 ms.

Process	**DSO**	**FDMO**	**FDMO-f**	**ORB-SLAM2**
Tracking (averaged over number of frames)	9.05	10.26	16.13	20.05
Mapping (averaged over number of keyframes)	50.64	51.05 + 115.85	73.58	198.52

FDMO-f requires an average of 16.13 ms to track a frame using both Direct and Indirect residuals and an average of 73.58 ms to add a keyframe and expands both Direct and Indirect map representations. Note that the Indirect map triangulation and Structure BA require an average of 90 ms to compute, however it is only called once every twenty keyframes on average, thereby significantly reducing its average computational cost. The time required by each of FDMO-f processes was recorded and summarized in Table 4.3.

Table 4.3: Computational time (ms) breakdown for every processes in FDMO-f.

	Process	Mean Time
Tracking	Direct data preparation	0.8
	Features and Descriptors Extraction	4.24
	Feature Matching and Indirect Pose optimization	7.11
	Direct pose refinement	3.98
Mapping	Photometric Bundle Adjustment	30.76
	Indirect map maintenance	10.11
	Indirect map triangulation and Structure BA	12.45
	Direct map maintenance and marginalization	20.26

4.3.2 Datasets

The quantitative evaluation was performed using the following publicly available datasets.

TUM MONO dataset [43]: contains 50 sequences of a camera moving along a path that begins and ends at the same location. The dataset is photometrically calibrated: camera response function, exposure times and vignetting are all available; however, ground truth pose information is only available for two small segments at the beginning and end of each sequence; fortunately, such information is enough to compute translation, rotation, and scale drifts accumulated over the path.

EuRoC MAV dataset [15]: contains 11 sequences of stereo images recorded by a drone mounted camera. Ground truth pose for each frame is available from a Vicon motion capture system.

4.3.3 Quantitative evaluation

To investigate the resilience of various monocular odometry systems to erratic motions, two experiments are proposed namely, the Frame drop experiment and the Two loop experiment.

Frame drop experiment

The proposed experiment, as shown in Fig. 4.7, consists of inducing artificial motions by dropping a number of frames, and measuring the accuracy of the relative pose found across the gap. However, in monocular systems the measured poses are up to a random scale; to compute the error between the measured values and the ground truth, we compute the scale as $S = \frac{\|T_i^{gt}\|_2}{\|T_i^{vo}\|_2}$, where $\|T_i^{gt}\|_2$ is the L2-norm of the ground-truth translation vector of the i^{th} frame (last tracked frame) and $\|T_i^{vo}\|_2$ is its VO measured counterpart. The perecentage translation error can then be computed as:

$$E_t = 100 \times \frac{\mid (S\|T_i^{vo} - T_{i+n+1}^{vo}\|_2 - \|T_i^{gt} - T_{i+n+1}^{gt}\|_2 \mid}{\|T_i^{gt} - T_{i+n+1}^{gt}\|_2}, \tag{4.7}$$

where T_{i+n+1} is the translation vector of the $(i{+}n{+}1)^{th}$ frame corresponding to the first frame after skipping n frames. As for rotations, we compute the angle between the i^{th} and $(i+n+1)^{th}$ frames as the geodesic angle separating them on a unit sphere (*i.e.*, the shortest angle that will align the orientations) which is computed as:

$$\theta = arccos(2\langle q_i, q_{i+n+1}\rangle^2 - 1), \tag{4.8}$$

where q is the normalized quaternion representation of the frame rotation and $\langle\cdot,\cdot\rangle$ is the inner product. Equation 4.8 is derived from [79, (23)] using simple trigonometric identities. The rotation error percentage is then computed as $E_r = 100 \times \frac{|\theta^{vo} - \theta^{gt}|}{\theta^{gt}}$.

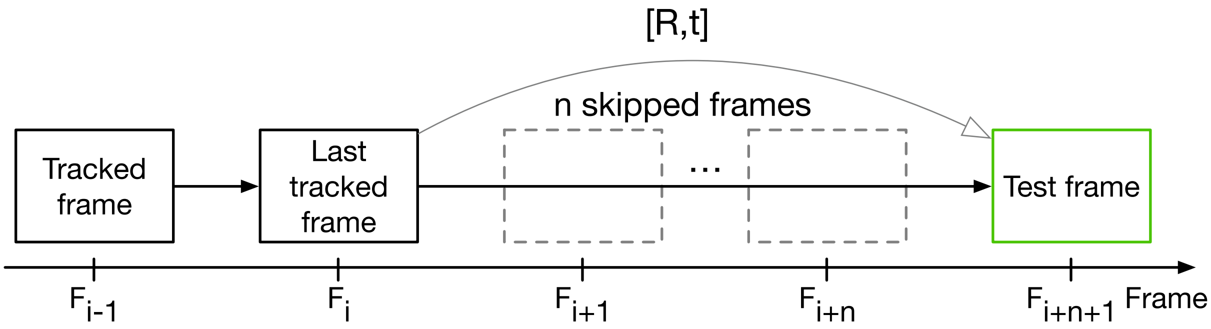


Figure 4.7: Depiction of the forward frame drop experiment where frames are skipped towards the future and the measured [R,T] between the last tracked frame and a test frame are compared against the ground truth.

The experiment is repeated until *VO* failure, with each time starting from the same frame i but increasing the number of dropped frames n (*e.g.*, $n = 0, 5, 10$, etc.). We record and plot for every n dropped frames E_t against the ground truth translation $\|T_i^{gt} - T_{i+n+1}^{gt}\|_2$ and E_r against the ground truth rotation angle θ^{gt}. Note that it is essential to measure the relative pose rather than the absolute pose for both E_r and E_t so that any previously accumulated drift does not have an effect on the obtained results. The same can be said about estimating the scale using the latest tracked frame so that accumulated scale drift doesn't distort the reported measurements.
Since the relative poses are used, aligning the two path which might yield over-optimistic results is not required.

Experiment Variation. While the suggested experiment skips frames towards previously unseen parts of the scene (future frames), one could argue that the camera can randomly move towards previously observed parts as well; therefore, to account for such motions, we propose a variation of the first experiment shown in Fig. 4.8, where the dropped frames are taken in the opposite direction, after reaching F_{i+n+1} without any interruption. To distinguish between both experiments, we will refer to the first experiment as the forward experiment (as in skipping frames towards forward in time) and the second as the backward experiment. Note that the same equations as before hold for both experiments.

Experiment analysis. While the effect of violating the motion model could be simulated by manually changing the motion model values instead of dropping frames, we chose to drop frames for two reasons. First, varying the motion model is equivalent to measuring the convergence basin of the objective function; however, it is not indicative of the system's ability to handle changes in the observed scene, which is usually the case in large motions. Second, dropping frames models another real-life problem namely, when an image feed is broadcast from a device to be processed

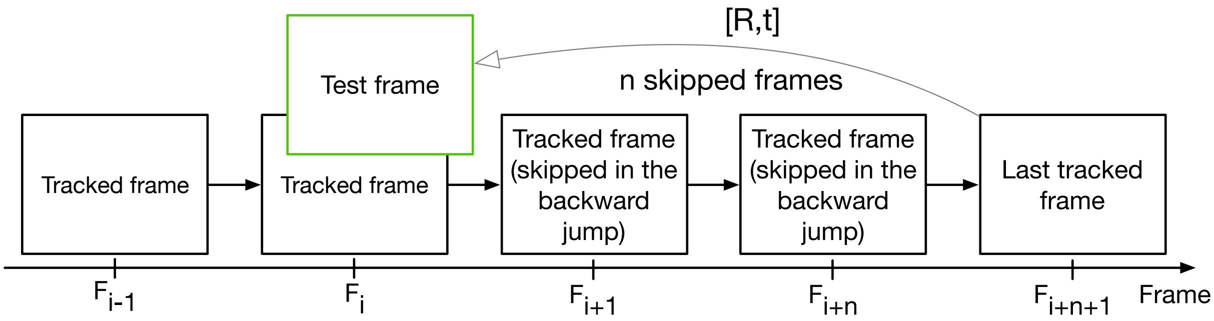


Figure 4.8: The backward frame drop experiment where frames are skipped towards the past and the measured [R,T] between the last tracked frame and a test frame are compared against the ground truth.

on a ground station, connectivity issues may arise, causing frame drops which a VO system must handle.

Other aspects of the experiment must also be considered. First, the observed scene plays an important role; as one can intuitively reason, a far-away scene requires larger motions to yield small changes in the image plane, whereas a close scene may easily go out of the field of view causing instant failure. Therefore, it is meaningless to report, for example, a camera motion of one meter causes a 30% error, without taking into account the scene depth. However, for the sake of comparing various VO systems against each other, it is sufficient to compare their results when observing the same scene and undergoing the same frame drops, as long as it is established that these results will be different and cannot be compared to other scenes.

Second, the camera motion also plays an important role; in particular, a camera moving along the same direction as its optical axis may exhibit different performance than a camera undergoing motion perpendicular to its optical axis, or than a camera that is purely-rotating. While our experiment can be applied to random motions, it is more insightful to separate the three cases and run the experiment on each, as such we have identified three locations in different sequences from the EuroC dataset [15], where the camera motion is one of the three cases and accordingly report on the found results.

Results. As mentioned earlier, the experiment was repeated three times for different camera motions as follows:

- Experiment A: camera motion parallel to its optical axis, performed in sequence MH01 with the forward experiment starting at frame 200 and the backward at frame 250.

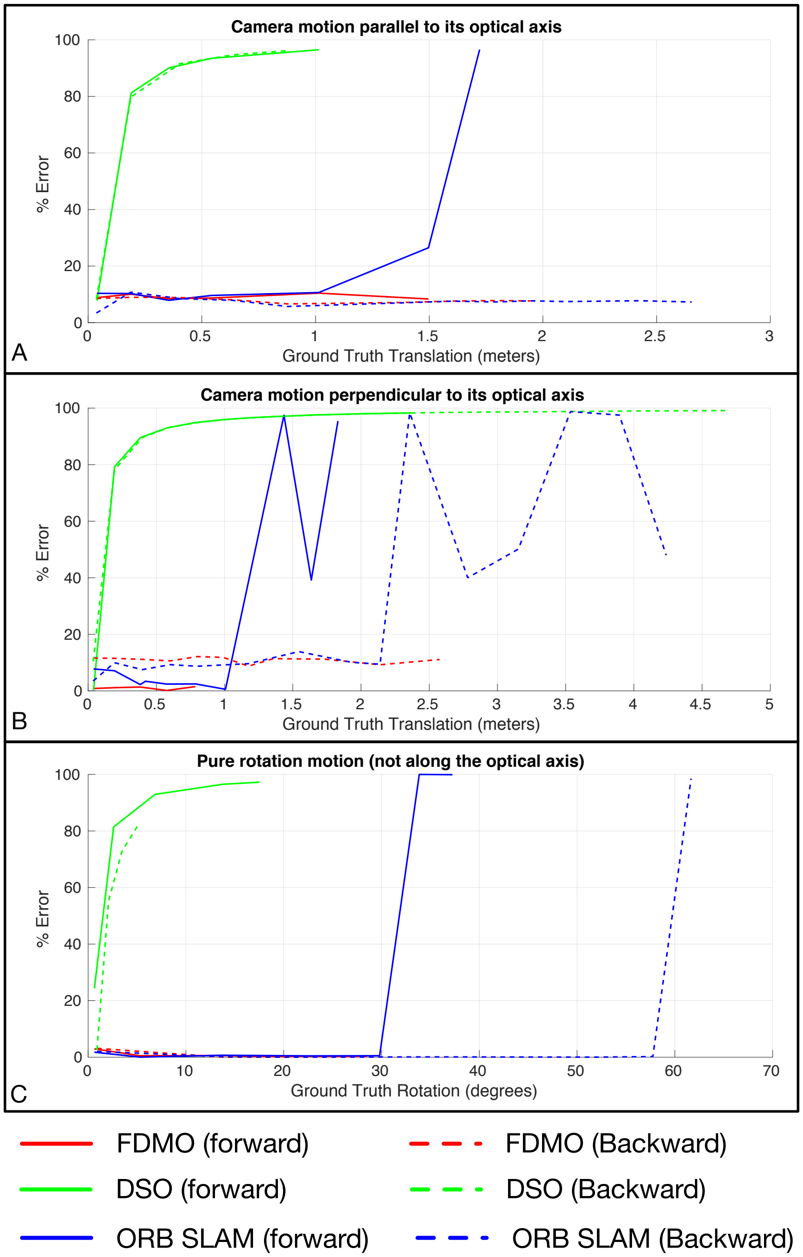


Figure 4.9: The results reported by the experiment in three different locations of the EuroC dataset [15] with each location corresponding to a different camera motion.

- Experiment B: camera motion perpendicular to its optical axis, performed in sequence MH03 with the forward starting at frame 950 and the backward at frame 1135.
- Experiment C: pure rotation motion (not along the optical axis), performed in sequence MH02 with the forward starting at frame 510 and the backward at frame 560.

The experiment was repeated in the three locations for ORB-SLAM 2 as a representative of Indirect systems, DSO [41] for Direct systems and FDMO [177] for Hybrid systems. The reported results are shown in Figure 4.9.

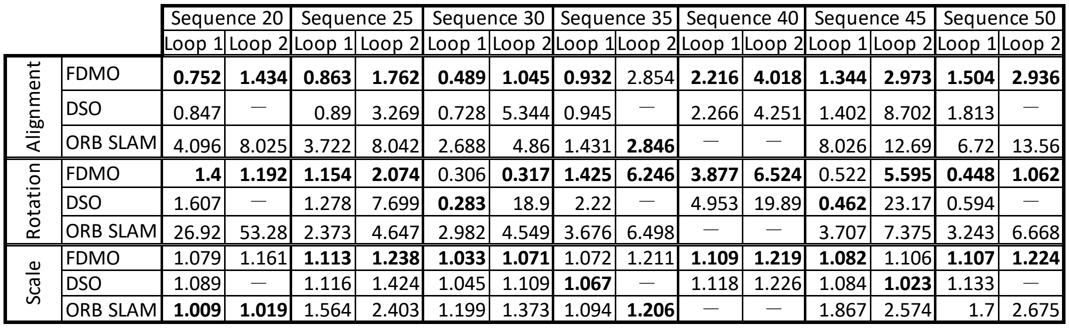
Discussion. DSO performed consistently the worst across the three experiments (Fig. 4.9-A,B,C). It is interesting to note that DSO's behavior was very similar in both forward and backward variations, *i.e.*, it could not handle any violations to its motion model irrespective of the observed scene. The reported result is expected due to its highly non-convex objective function. Meanwhile, ORB SLAM 2 and FDMO performed consistently better in all experiments on the backward variation than on the forward. This is due to the resilience of their data association step to large motions *i.e.*, where they were capable of matching image features to old map points. On the other hand, their reduced performance in the forward variation is mainly attributed to feature deprivation, as the new part of the scene had not been mapped yet.
On another note, a notable performance decrease in ORB SLAM 2 is observed before complete failure (oscillatory behavior shown in Fig. 4.9-B), which is indicative of its failure recovery procedure localizing the frames at erroneous poses.
Finally, FDMO was able to track in both variations of the experiment an angle of 30 degrees before failure as shown in Fig .4.9; whereas ORB SLAM 2 was able to track 58 degrees before failure in the backward variation of the experiment. The reduced performance in FDMO is due to its need to track the latest frame with respect to the **latest keyframe** in its map, as opposed to ORB SLAM 2 which tracks the latest frame with respect to the local map (a set of keyframes) without the need for tracking with respect to a particular keyframe.

As a concluding thought the proposed experiment revealed various insights into the behavior of different VO systems that are usually not tested in traditional experiments. Furthermore, the same experiment can be used to define the operational range of different VO systems; for example, one could repeat the experiment at varying scene depths to establish the operational speed range a specific VO system can handle. Finally, the experiment is also capable of evaluating the effects of fast erratic motions on VO; however, it does not model the motion blur induced in an actual erratic motion. Therefore modelling motion blur according to the dropped frames remains part of our future work.

Two loop experiment

In the two loop expriment, all three systems (DSO, FDMO, and ORB SLAM 2) are run on various sequences of the Tum_Mono dataset [43] across various conditions, both indoors and outdoors. Each system is allowed to run through every sequence for two continuous loops where each sequence begins and ends at the same location.

Table 4.4: Measured drifts after finishing one and two loops over various sequences from the TumMono dataset. The alignment drift (meters), rotation drift (degrees) and scale($\frac{m}{m}$) drifts are computed similar to [43].



		Sequence 20		Sequence 25		Sequence 30		Sequence 35		Sequence 40		Sequence 45		Sequence 50	
		Loop 1	Loop 2	Loop 1	Loop 2	Loop 1	Loop 2	Loop 1	Loop 2	Loop 1	Loop 2	Loop 1	Loop 2	Loop 1	Loop 2
Alignment	FDMO	**0.752**	**1.434**	**0.863**	**1.762**	**0.489**	**1.045**	**0.932**	2.854	**2.216**	**4.018**	**1.344**	**2.973**	**1.504**	**2.936**
	DSO	0.847	—	0.89	3.269	0.728	5.344	0.945	—	2.266	4.251	1.402	8.702	1.813	—
	ORB SLAM	4.096	8.025	3.722	8.042	2.688	4.86	1.431	**2.846**	—	—	8.026	12.69	6.72	13.56
Rotation	FDMO	**1.4**	**1.192**	**1.154**	**2.074**	0.306	**0.317**	**1.425**	**6.246**	**3.877**	**6.524**	0.522	**5.595**	**0.448**	**1.062**
	DSO	1.607	—	1.278	7.699	**0.283**	18.9	2.22	—	4.953	19.89	**0.462**	23.17	0.594	—
	ORB SLAM	26.92	53.28	2.373	4.647	2.982	4.549	3.676	6.498	—	—	3.707	7.375	3.243	6.668
Scale	FDMO	1.079	1.161	**1.113**	**1.238**	**1.033**	**1.071**	1.072	1.211	**1.109**	**1.219**	**1.082**	1.106	**1.107**	**1.224**
	DSO	1.089	—	1.116	1.424	1.045	1.109	**1.067**	—	1.118	1.226	1.084	**1.023**	1.133	—
	ORB SLAM	**1.009**	**1.019**	1.564	2.403	1.199	1.373	1.094	**1.206**	—	—	1.867	2.574	1.7	2.675

The translational, rotational, and scale drifts are recorded at the end of each loop, as described in [43]. The drifts recorded at the end of the first loop are indicative of the system's performance across the unmodified generic datasets, whereas the drifts recorded at the end of the second loop consist of three components: (1) the drift accumulated from the first loop, (2) an added drift accumulated over the second run, and (3) an error caused by a large baseline motion induced at the transition between the loops. The reported results are shown in Table 4.4 and some of the recovered trajectories are shown in Fig. 4.10.

4.3.4 Qualitative assessment

Fig. 4.11 compares the resilience of FDMO and ORB-SLAM2 to feature-deprived environments. FDMO exploits the sub-pixel accurate localized direct keyframes to propagate its feature-based map, therefore its capable of generating accurate and robust 3D landmarks that have a higher matching rate even in low textured environments. In contrast, ORB-SLAM2 fails to propagate its map causing tracking failure.

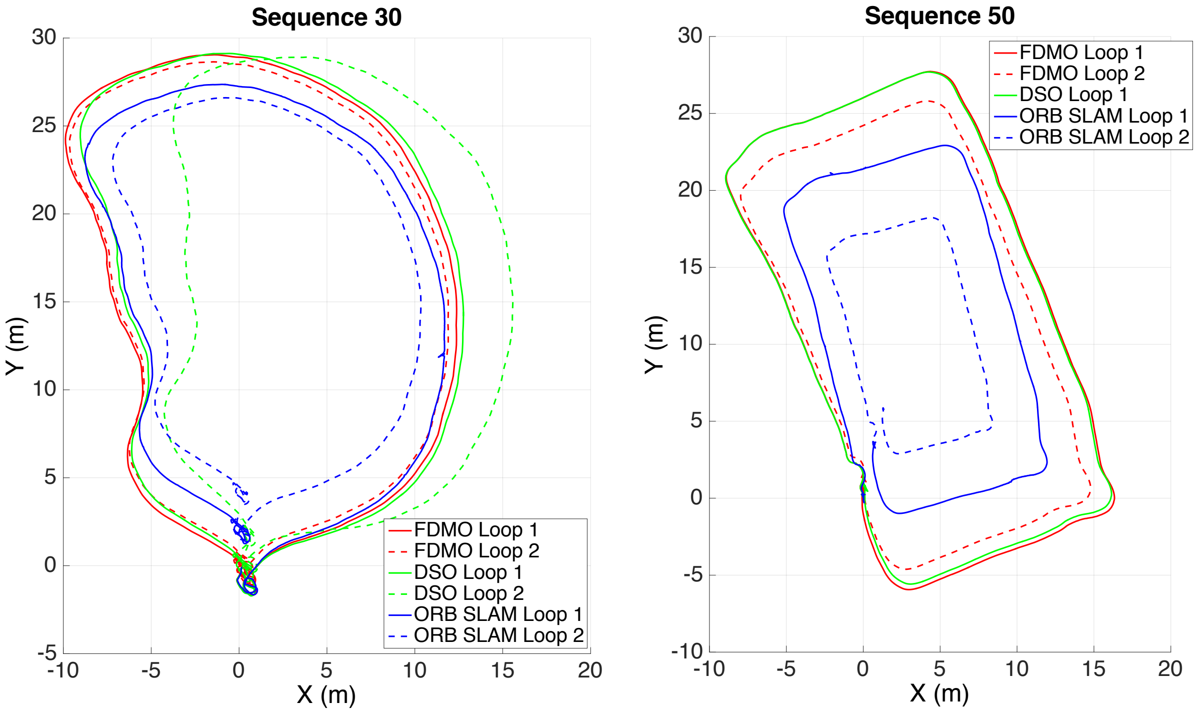


Figure 4.10: Sample paths estimated by the various systems on Sequences 30 and 50 of the Tum_Mono dataset. The paths are all aligned using ground truths available at the beginning and end of each loop. Each solid line corresponds to the first loop of a system while the dashed line correspond to the second loop. Ideally, all systems would start and end at the same location, while reporting the same trajectories across the two loops. Note that in Sequence 50, there is no second loop for DSO as it was not capable of dealing with the large baseline between the loops and failed.

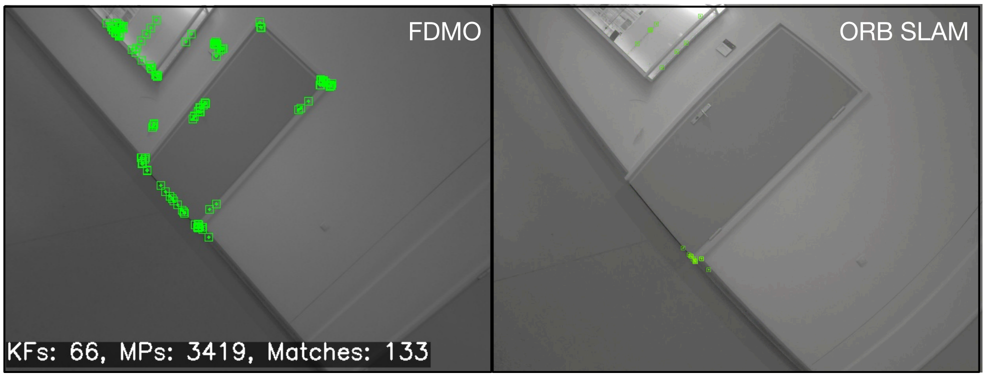


Figure 4.11: Features matched of FDMO (left) and ORB-SLAM2 (right) in a feature deprived environment (sequence 40 of the Tum_mono dataset).

Chapter 5

UFVO: A Unified Formulation for Visual Odometry

While the previously suggested feature-assisted methods were successful in leveraging the advantages of both Direct and Indirect methods, and despite the various efficiencies introduced, at their core the Direct and Indirect formulations are separate processes where special treatment is required to perform the pose estimation using either of them, and requiring separate threads to propagate both scene representations.
To address this issue, *UFVO* is proposed with the following key contributions: (1) a tight coupling of photometric (Direct) and geometric (Indirect) measurements using a joint multi-objective optimization, (2) the use of a utility function as a decision maker that incorporates prior knowledge on both paradigms, (3) descriptor sharing, where a feature can have more than one type of descriptor and its different descriptors are used for tracking and mapping, (4) the depth estimation of both corner features and pixel features within the same map using an inverse depth parametrization, and (5) a corner and pixel selection strategy that extracts both types of information, while promoting a uniform distribution over the image domain. Experiments show that *UFVO* can handle large inter-frame motions, inherits the sub-pixel accuracy of direct methods, can run efficiently in real-time, can generate an Indirect map representation at a marginal computational cost when compared to traditional Indirect systems, all while outperforming state of the art in Direct, Indirect and hybrid systems. Fig. 5.1 shows an overview of the different *UFVO* components.

Before getting into further details, a nomenclature must be established so that misconceptions are avoided. In particular we refer to the image locations from which measurements are taken as features. This means that we consider corners and pixel locations as features that can be used interchangeably as Direct or Indirect features. We also employ the word *descriptor* for both

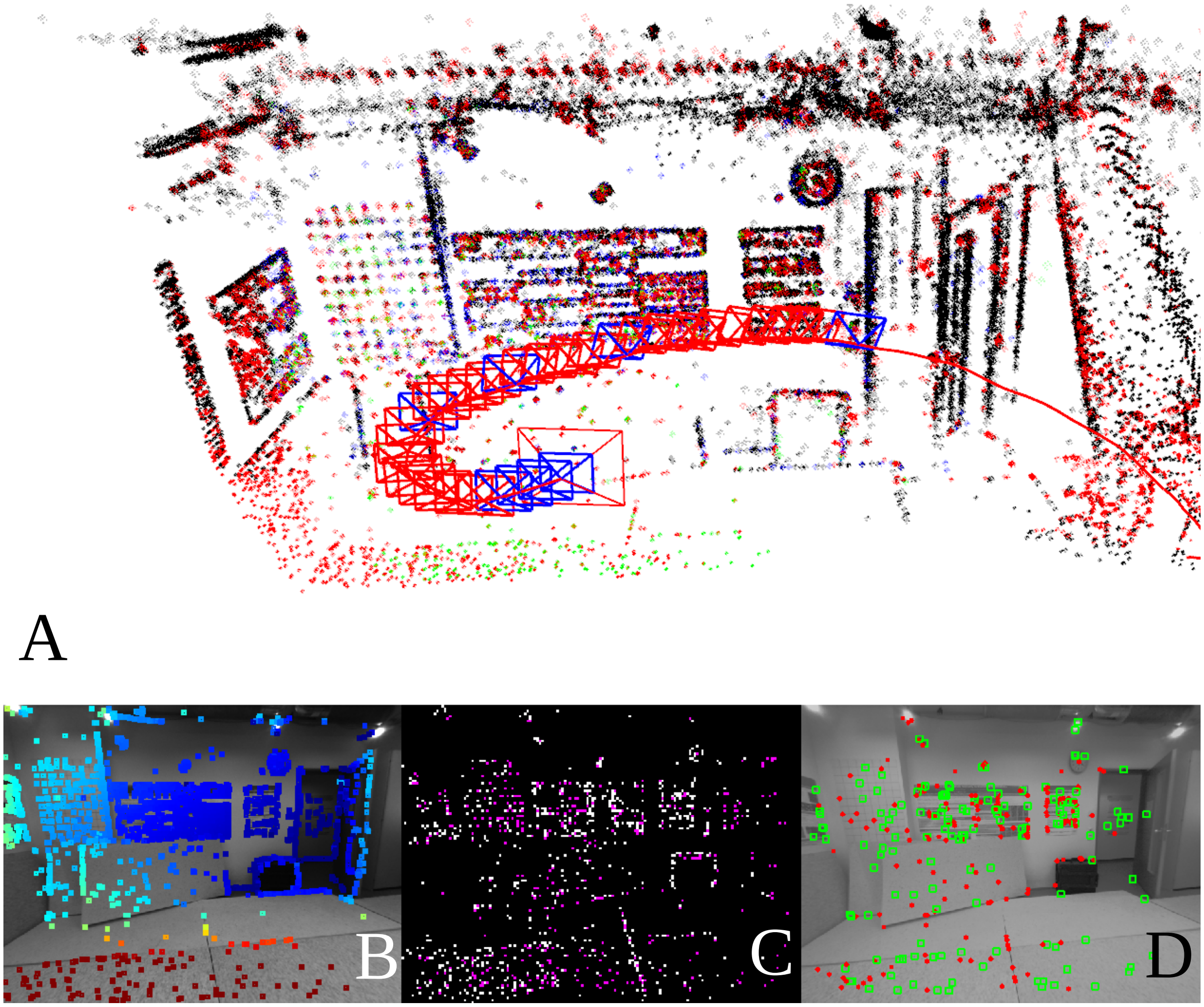


Figure 5.1: **UFVO**: A− shows the 3D recovered map and the different types of points used: green points contribute to both geometric and photometric residuals, red points geometric residuals only, blue points photometric residuals only and black points are marginalized (do not contribute any residuals). The blue squares are hybrid keyframes (contains both geometric and photometric information), whereas red squares are Indirect Keyframes whose photometric data was marginalized. B− shows the projected depth map of all active points. C− shows the occupancy grid which we use to ensure a homogeneously distributed map point sampling process. The white squares correspond to the projected map points while the magenta squares represent newly extracted features from the newest keyframe. Finally D− shows the inlier geometric features (active in green and marginalized in red) used during tracking (Figure better seen in color).

paradigms, where an Indirect feature descriptor is a high dimensional vector computed from the area surrounding the feature (*e.g.* ORB [141]), and a Direct feature descriptor is a local patch of pixel intensities surrounding the feature. Finally, we refer to geometric residual as the 2-D geometric distance between an indirect feature location and its associated map point projection on the image plane. In contrast, we refer to photometric residual as the intensity difference between a direct feature descriptor and a patch of pixel intensities at the location where the feature projects to in the image plane.

5.1 RelatedWork

Aside the previously proposed feature-assisted methods, several other hybrid approaches were proposed. Gao *et al.* [56] suggested using both corners and pixels as Direct features, where both contribute photometric residuals; however, geometric residuals are not used. The ORB descriptors associated with the corners are only used to detect loop closure. Nevertheless, the introduction of corners as Direct features affects the odometry performance. The reason is that corners have high intensity gradients along two directions, and as such are not susceptible to drift along the edge of an intensity gradient; in contrast, their pixel feature counterparts can drift along edges. While this is expected to improve performance, its results where different across various sequences.

In Lee and Civera [92], the state of the art in both Direct (DSO [41]) and Indirect (ORB SLAM 2[124]) are cascaded one on top of the other, where DSO is used for odometry and its marginalized keyframes are fed to ORB SLAM 2 for further optimization. Aside the inefficiency of running both systems sequentially and independently, the Indirect residuals do not contribute towards real-time odometry unless loop closure is detected, limiting the odometry performance to that of [41].

Finally, the closest in spirit to *UFVO* is that of Yu *et al.* [179], where geometric and photometric residuals are combined in a joint optimization. However, the residuals are treated as completely independent entities, requiring separate processes to extract and maintain each type of residual, irrespective of the other, and the optimization is used as a black box without incorporating crucial priors on the residuals' behavior.

UFVO addresses all of the aforementioned limitations by using both types of residuals at frame-rate. The contribution is a novel monocular odometry system that:

1. can operate in texture-deprived environments,
2. can handle large inter-frame motions (as long as a sufficient number of geometric residuals are established),

3. does not require a separate process to generate two different types of maps,
4. incorporates prior information on the various feature types via a utility function in the optimization.

5.2 Nomenclature

Throughout the remainder of this chapter, matrices are denoted by bold upper case letters $\mathbf{M}$, vectors by bold lower case letters $\mathbf{v}$, camera poses as $\mathbf{T} \in SE(3)$ with their associated Lie element in the groups' tangent space as $\hat{\xi} \in se(3)$. A 3D point in the world coordinate frame is denoted as $\mathbf{x} \in \Re^3$, its coordinates in the camera frame as $\tilde{\mathbf{x}} = \mathbf{T}\mathbf{x}$, and its projection on the image plane as $\mathbf{p} = (u,v)^T$. The projection from 3D to 2D is denoted as $\pi(\mathbf{c},\mathbf{x}) : \Re^3 \rightarrow \Re^2$ and its inverse $\pi^{-1}(\mathbf{c},\mathbf{p},d) : \Re^2 \rightarrow \Re^3$ where $\mathbf{c}$ and d represent the camera intrinsics and the points' depth respectively. $\mathscr{L}(a,b) : I(\mathbf{p}) \mapsto e^{-a}(I(\mathbf{p}) - b)$ is an affine brightness transfer function that models the exposure change in the entire image and $I(\mathbf{p})$ is the pixel intensity value at $\mathbf{p}$. To simplify the representation we define $\xi := (\hat{\xi}, \mathscr{L})$ as the set of variables over which we perform the camera motion optimization. We define the operator $\boxplus : se(3) \times SE(3) \rightarrow SE(3)$ as $\hat{\xi} \boxplus \mathbf{T} = e^{\hat{\xi}}\mathbf{T}$. The incremental updates over ξ are then defined as $\delta\xi \oplus \xi = (log(\delta\hat{\xi} \boxplus e^{\hat{\xi}}), a + \delta a, b + \delta b)$. Finally, a subscript p is assigned for photometric measurements and g for geometric measurements. Also note that in this work the terms Direct and Indirect are associated with the words photometric and geometric respectively, as such both names are used interchangeably.

5.3 Proposed System

5.3.1 Feature types

UFVO concurrently uses both photometric and geometric residuals at frame-rate. For this purpose, its important to make the distinction between two types of features: salient corner features, and pixel features.

Corner features are FAST [140] corners extracted at $\mathbf{p}$, associated with a Shi-Tomasi score [146] that reflects their saliency as a corner, an ORB descriptor [141], and a patch of pixels surrounding the point $\mathbf{p}$. Corner features contribute two types of residuals during motion estimation: a geometric residual using its associated descriptor, and a photometric residual using its pixels patch.

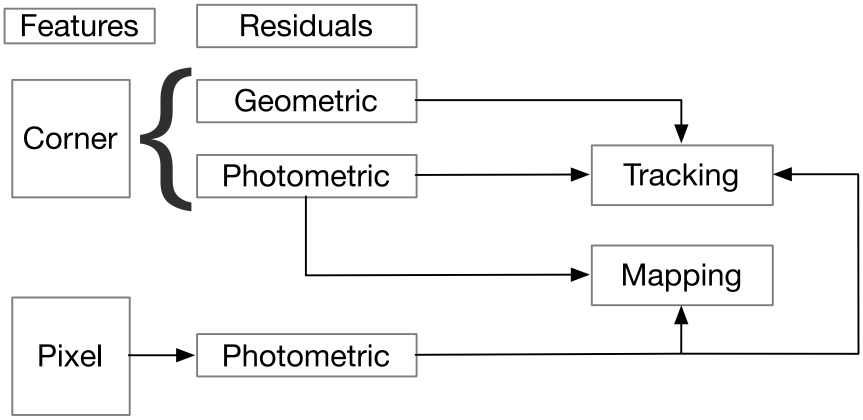


Figure 5.2: Summary of the feature types, their associated residuals, and their usage in UFVO.

Pixel features are sampled from the images at any location **p** that is not a corner and has sufficient intensity gradient; they are only associated with a patch of pixels; therefore, pixel features only contribute photometric residuals during motion estimation.

The types of residuals each feature type contributes in tracking and mapping are summarized in Fig. 5.2. Features are classified as either:

1. *Candidate*: new features whose depth estimates have not converged; they contribute to neither tracking nor mapping.
2. Active: features with depth estimates that contribute to both tracking and mapping.
3. Marginalized: features that went outside the current field of view or features that belong to marginalized keyframes.
4. Outliers: features with high residuals or corners that frequently failed to match other features.

5.3.2 System Overview

UFVO is an odometry system that operates on two threads: tracking and local mapping. Fig. 5.3 depicts an overview of the system's operation, which starts by processing new frames to create a pyramidal image representation, from which both corners features are first sampled. A constant velocity motion model is then used to define a search window for corner feature matching, which are used to compute the geometric residuals. A joint multi-objective pyramidal image alignment then minimizes both geometric and photometric residuals associated with the two types of features over the new frame's pose. The frame is then passed to the mapping thread where a keyframe selection criterion (as described in [41]) is employed. If the frame was not flagged as a keyframe, all the candidate points' depth estimates are updated using their photometric residuals in an inverse depth formulation and the system awaits a new frame.

On the other hand, if the frame was deemed a keyframe, a 2D occupancy grid is first generated over the image by projecting the local map points to the new keyframe; each map point occupies a 3x3 pixels area in the grid. A subset of the candidate features from the previous keyframes are then activated such that the new map points project at empty grid locations. New *Candidate* corner features are then sampled at the remaining empty grids before a local photometric bundle adjustment takes place which minimizes the photometric residuals of all features in the active window. Note that we do not include the geometric residuals in this optimization; their use during tracking ensures that the state estimates are as close as possible to their global minima; hence, there is no added value of including them. Furthermore, since the geometric observation models' precision is limited, including them would actually cause jitter around the minimum.

Outlier Direct and Indirect features are then removed from the system. The local map is then updated with the new measurements and a computationally cheap structure only optimization is applied to the marginalized Indirect features that are part of the local map. Finally, old keyframes are marginalized from the local map.

5.3.3 Corner and pixel features sampling and activation

Feature sampling: when a new frame is acquired, a pyramidal representation is created over which we apply a pyramidal image alignment; however, unlike [124] or most Indirect methods in the literature, we only extract Indirect features at the highest image resolution. Since Indirect features in UFVO are only tracked in a local set of keyframes, which are relatively close to each other (*i.e.*, don't exhibit significant variations in their scale), extracting features from one pyramid level allows us to save on the computational cost typically associated with extracting Indirect features without significantly compromising performance. Since corners contribute to both types of residuals, we avoid sampling pixel features at corner locations; therefore, we sample pixel features at *non-corner* locations with sufficient intensity gradient.

Feature activation: when a keyframe is marginalized, a subset of the candidate features from the previous keyframes are activated to take over in its place. Our feature activation policy is designed to enforce the following priorities in order:

1. To minimize redundant information, features should not overlap with other types of features.

2. Ensure maximum saliency for the new Indirect features.

3. To maintain constant computational cost, add a fixed number of new Direct and Indirect Candidates.

To enforce the activation policy and ensure a homogeneous feature distribution, we employ a coarse 2D occupancy grid over the latest keyframe, such that the grid is populated with the projection of the current map on the keyframe with each point occupying a 3×3 area in the occupancy grid. Since corners are scarce, we prioritize their activation, that is, we employ a two stage activation process: the first sorts corner features in a descending order according to their Shi-Tomasi score, and activates a fixed number of the strongest corners from unoccupied grid cells. The second stage activates pixel features at locations different than the newly activated corners, and that maximizes the distance to any other feature in the keyframe.

Fig. 5.4 demonstrates the effectiveness of our sampling and activation strategy by showing an example of the occupancy grid (left) alongside its keyframe's image (right). The occupancy grid (left) shows the current map points in white and the newly added map points in magenta. It can be seen that there is no overlap between old and new points, the points are homogeneously distributed throughout the frame.

5.3.4 Joint Multi-Objective Image alignment

The core component of UFVO is a joint optimization that minimizes an energy functional, which combines both types of residuals, over the relative transformation ξ relating the current frame to last added keyframe. The joint optimization is best described as:

$$\underset{\xi}{\operatorname{argmin}}(\mathbf{e}_p(\xi), \mathbf{e}_g(\xi)). \tag{5.1}$$

The optimization process itself must be computationally efficient, delivers a single pareto optimal solution, and capable of achieving superior performance than both individual frameworks. While the research community has provided an ample amount of methods for Multi-Objective

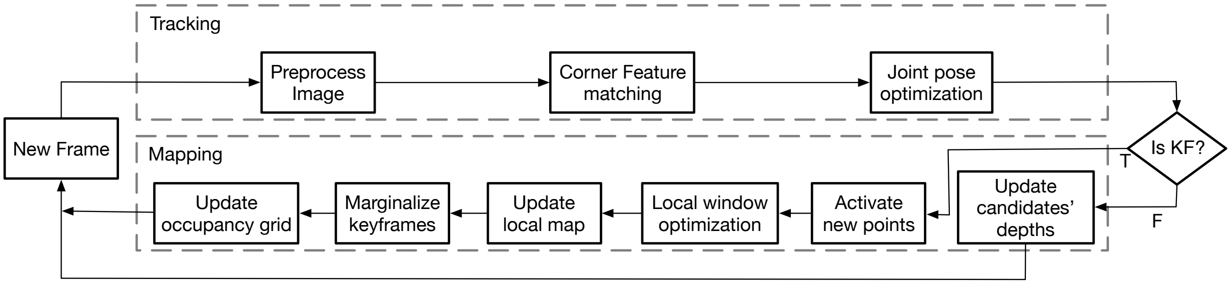


Figure 5.3: Overview of UFVO's tracking and mapping threads. A new frame is first preprocessed to compute pyramid levels and extract corner features. Corner features matching then takes place before a joint pose optimization. UFVO then updates an occupancy grid over the last added keyframe which records the locations of active corners and pixel features. The frame is then passed to the mapping thread and a decision is made whether it should be a keyframe or not. If it is not selected as a keyframe, it is used to update the depth estimates of the local map's candidate points. Otherwise, candidate points from the local map are activated and a local photometric optimization takes place; the local map is then updated with the optimized variables and old keyframes with their associated points are marginalized.

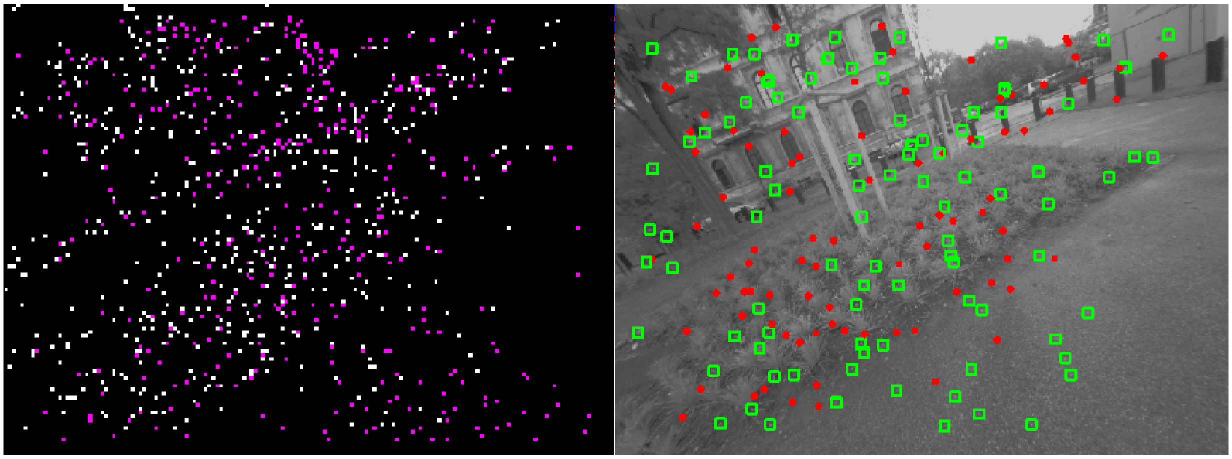


Figure 5.4: Occupancy grid (left), and its associated keyframe's image (right). The white squares in the occupancy grid represent current map points and the magenta squares represent newly added ones. As for the right image, the green squares represent indirect active map point matches, and the red dots represent marginalized indirect feature matches.

optimizations, few meet the harsh constraints of real-time performance, and allow for explicit a priori articulation of preferences. One optimization method that meets the aforementioned requirements is the *Weighted Sum Scalarization* method [112], that transforms the optimization of (5.1) to:

$$\underset{\xi}{\operatorname{argmin}}\,(\alpha_1 \mathbf{e}_p(\xi) + \alpha_2 \mathbf{e}_g(\xi)), \tag{5.2}$$

where α_1 and α_2 represent the contribution of each residual type to the final solution. For simplicity, the problem is reformulated using $K = \frac{\alpha_2}{\alpha_1}$, which represents the weight of the geometric residuals relative to the photometric residuals; *e.g.*, $K = 2$ assigns twice as much importance to the geometric residuals than to the photometric residuals.

For this weighing scheme to have any sense, both energies must be dimensionless, and normalized such that imbalances in the numbers of the two residuals does not inherently bias the solution. The Huber norm is also used to account for outliers. The joint energy functional becomes:

$$\underset{\xi}{\operatorname{argmin}}\,\mathbf{e}(\xi) = \underset{\xi}{\operatorname{argmin}}\left[\frac{\|\mathbf{e}_p(\xi)\|_\gamma}{n_p\sigma_p} + K\frac{\|\mathbf{e}_g(\xi)\|_\gamma}{n_g\sigma_g}\right], \tag{5.3}$$

where n is the count of each feature type, σ^2 is the residual's variance, $\|\cdot\|_\gamma$ is the Huber norm, and the energy per feature type is defined as:

$$\mathbf{e}_p(\xi) = (\mathbf{r}^T W \mathbf{r})_p,\ \textit{and}\ \mathbf{e}_g(\xi) = (\mathbf{r}^T W \mathbf{r})_g, \tag{5.4}$$

$\mathbf{r}$ is the vector of stacked residuals per feature type and W is a weight matrix that will be further discussed by the end of this section.

We seek an optimal solution $\bar{\xi}$ that minimizes (5.3); however, $\mathbf{r}$ is non-linear, therefore we linearize it with a first order Taylor expansion around the initial estimate ξ, with a perturbation $\delta\xi$, that is:

$$\mathbf{r}(\xi \oplus \delta\xi) \simeq \mathbf{r}(\xi) + \mathbf{J}\delta\xi, \tag{5.5}$$

where $\mathbf{J} = \frac{\partial \mathbf{r}}{\partial \xi}$. If we replace $\mathbf{r}$ in (5.4) by its linearized value from (5.5), substitute the result in (5.3), differentiate the result with respect to ξ, and set it equal to zero, we arrive at the step increment equation $\delta\xi \in \Re^8$ of the joint optimization:

$$\delta\xi = -\left[\left(\frac{\mathbf{J}^t\mathbf{W}\mathbf{J}}{n\sigma}\right)_p + \left(\frac{K\mathbf{J}^t\mathbf{W}\mathbf{J}}{n\sigma}\right)_g\right]^{-1}\left[\left(\frac{\mathbf{J}^t\mathbf{W}\mathbf{r}}{n\sigma}\right)_p + \left(\frac{K\mathbf{J}^t\mathbf{W}\mathbf{r}}{n\sigma}\right)_g\right], \tag{5.6}$$

which is iteratively applied using $\xi = \delta\xi \oplus \xi$ in a Levenberg-Marquardt formulation, until convergence. The optimization is repeated from the coarsest to the finest pyramid levels, where the result of each level is used as an initialization to the subsequent one. At the end of each pyramid level, outliers are removed and the variable K is updated according to 5.3.5.

The photometric residual $r_p \in \Re$ per feature can be found by evaluating:

$$r_p = \sum_{p \in \mathcal{N}_p}\left[(I_j[\mathbf{p}'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}}(I_i[\mathbf{p}] - b_i)\right], \tag{5.7}$$

where $\mathcal{N}_p$ is the neighboring pixels of the feature at $\mathbf{p}$, the subscript i denote the reference keyframe, and j the current frame; t is the image exposure time which is set to 1 if not-available, and $\mathbf{p}'$ is the projection of a feature from the reference keyframe to the current frame, which is found using:

$$\mathbf{p}' = \pi(\mathbf{c}, e^{\hat{\xi}}\pi^{-1}(\mathbf{c}, \mathbf{p}, d)) \tag{5.8}$$

where $\hat{\xi}$ is the relative transformation from the reference keyframe to the new frame. Note that we compute the photometric residual for both types of features (corners and pixels). The geometric residual $r_g \in \Re^2$ per corner feature is defined as:

$$\mathbf{r}_g = \mathbf{p}' - \mathbf{obs}, \tag{5.9}$$

where $\mathbf{obs} \in \Re^2$ is the corners' matched location in the new image, found through descriptor matching. Regarding the weight W matrices, the photometric weight is defined as $w_p = h_w(\gamma_p)$ where:

$$h_w = \begin{cases} 1 & if\, e < \gamma^2 \\ \frac{\gamma}{\sqrt{e}} & if\, e \geq \gamma^2 \end{cases}, \tag{5.10}$$

is the Huber weight function. As for the geometric weight, we combine two weighing factors: a Huber weight as defined in (5.10), and a variance weight associated with the variance of the corners' depth estimate:

$$w_d = \frac{\frac{1}{\sigma_d}}{max\left(\frac{1}{\sigma_d}\right)}, \tag{5.11}$$

with $max(\frac{1}{\sigma_d})$ the maximum $\frac{1}{\sigma_d}$ in the current frame, which down-weighs features with high depth variance. The final geometric weight is then found as $W_g = w_d h_w(\gamma_g)$.

Finally, the photometric Jacobian $J_p|_{1\times 8}$ required to solve (5.6) per pixel is found using:

$$\begin{aligned} J_p = \Bigg[& \frac{f_u \nabla I_u}{d}, \frac{f_v \nabla I_v}{d}, -\frac{1}{d}(\nabla I_u u f_u + \nabla I_v v f_v), \\ & -\nabla I_v f_v (1+v^2) - \nabla I_u f_u uv, \nabla I_u f_u (1+u^2) + \nabla I_v f_v uv, \\ & \nabla I_v f_v u - \nabla I_u f_u v, -\frac{t_j e^{a_j}}{t_i e^{a_i}}(I_i[\mathbf{p}] - b_i), -1 \Bigg] \end{aligned} \tag{5.12}$$

where u and v are the coordinates of the point $\mathbf{p}'$ in the new frame's image plane, f_u and f_v are the focal length parameters, and ∇I is the directional image gradient. The geometric Jacobian $\mathbf{J}_g||_{2\times 8}$ per corner is computed as:

$$\mathbf{J}_g = \begin{bmatrix} \frac{f_u}{d}, & 0, & -\frac{f_u u}{d}, & -f_u uv, & f_u(1+u^2), & -f_u v, & 0, & 0 \\ 0, & \frac{f_v}{d}, & -\frac{f_v v}{d}, & -f_v(1+v^2), & f_v uv, & f_v u, & 0, & 0 \end{bmatrix} \tag{5.13}$$

5.3.5 Utility function

Due to the high non-convexity of the Direct formulation, erroneous or initialization points far from the optimum, cause the Direct optimization to converge to a local minimum, far from the actual solution. While Indirect methods are robust to such initializing points, they tend to flatten around the actual solution due to their discretization of the image space. The interested reader is referred to [177] for an experiment on the matter. Ideally, an optimization process would follow the descent direction of the Indirect formulation until it reaches a pose estimate within the local concavity of the actual solution, after which it would follow the descent direction along the Direct formulation.

The introduction of K in (5.3) allows expressing such a-priori preference within the optimization. As $K \to 0$ the optimization discards geometric residuals, whereas as $K \gg$, geometric residuals dominate. Therefore a function that controls K is sought such that the descent direction behaves as described earlier. Furthermore, the geometric residuals tend to be unreliable in texture-deprived environments, therefore K should be $\propto$ number of matches. We heuristically design the following logistic utility function:

$$K = \frac{5e^{-2l}}{1+e^{\frac{30-N_g}{4}}}, \tag{5.14}$$

where l is the pyramid level at which the optimization is taking place, and N_g is the number of current inlier geometric matches. While the number of iterations does not explicitly appear in 5.14, it is embedded within l; as the optimization progresses sequentially from a pyramid level to another, the optimization follows the descent direction of the geometric residuals,with a decay induced by (5.14) that down-weighs the contribution of the geometric residuals as the solution approaches its final state. K also penalizes the Indirect energy functional at low number of matches, allowing UFVO to naturally handle texture-deprived environments.

5.3.6 Mapping

Map representation

Unlike typical Indirect formulations found in the literature [124],[177],[83], etc., we adopt for our Indirect features an inverse depth parametrization, which allows us to circumvent the need for a separate multi-view geometric triangulation process that is notorious for its numerical instability for observations separated by small baselines. Instead, we exploit the Direct pixel descriptors associated with our Indirect corner features. Aside its numerical stability, this is also advantageous in terms of computational resources, as it allows us to compute the depth of Indirect features at virtually no extra computational cost.

Local Map

Our local map is made of a moving window of keyframes in which we distinguish between two types of keyframes:

- hybrid keyframes: a fixed-size set of keyframes that contains Direct and Indirect residuals, over which a photometric bundle adjustment optimizes both types of features using their photometric measurements.
- Indirect keyframes: previously hybrid keyframes whose photometric data was marginalized, but still share Indirect features with the latest frame.

As new keyframes are added, hybrid keyframes are removed by marginalization using the Schur complement. On the other hand, Indirect keyframes are dropped from the local map once they no longer share Indirect features with the current camera frame.

To maintain the integrity of marginalized Indirect points we resort to a structure-only optimization as described in [177], that refines their depth estimates with new observations; however, one should note that the use of marginalized indirect features is restricted to features that are still in the local map. Furthermore, their use, and the structure only optimization for that matter, is optional; however, we found that using them increases the system's performance as it allows previously marginalized reliable data to influence the current state of the system, thereby reducing drift.

5.4 Evaluation

Our evaluation includes a computational cost analysis and a thorough evaluation of our system on various sequences from the TUM Mono dataset [45], covering a wide range of full camera exploration scenarios with fast motions, texture-deprived environments, short and long paths in indoor and outdoor scenes.

Since our system is a monocular system, we also report on the state-of-the-art in monocular Direct (DSO [41]), Indirect (ORB SLAM2 [124]) and Hybrid (LDSO [56] and LCSD [92]). For fairness of comparison and similar to [41] and [177], we evaluate ORB SLAM2, LDSO and LCSD as odometry systems by disabling their global loop closure modules. We don't compare against SVO 2 [53] since it failed on most sequences, and when it didn't fail it returned errors of an order of magnitude larger than the other systems; for reported results on [53] the reader is referred to [168]. Finally we don't compare against our own previous hybrid system (FDMO [177]) since UFVO is a much more efficient extension of FDMO that runs on every frame, and it naturally outperforms FDMO.

5.4.1 Computational Cost

We analyze the computational cost associated with the major components of UFVO on an Intel Core i7-8700K CPU @ 3.70GHz CPU; no GPU acceleration was used. The time required by each process is summarized in Tab. 5.1.

In total, UFVO requires on average 14.2 ms per frame for tracking and 72.9 ms on average per keyframe for mapping both Direct and Indirect map representations. In contrast, DSO requires 4.46 ms for tracking and 46.77 for mapping a Direct representation while ORB SLAM 2 requires 19.08 for tracking and 104.44 ms for mapping an Indirect map only.

Table 5.1: Computational cost of different components of UFVO in ms.

Process	mean time (ms)
Direct data preparation and Image Pyramids	3.81
Features and Descriptors Extraction	3.38
Feature Matching	2.52
Joint Optimization	4.51
Occupancy map Update	2.62
Candidate Points Depth Update	3.5
New map point initialization	4.27
Photometric BA	50.6
Local Map Update	4.62
Structure only optimization (optional)	7.32

5.4.2 Quantitative Evaluation

To evaluate the performance of our system we report on the alignment error e_{align}, as described in [45], which encompasses the effects of translation, rotation and scale drifts accumulated over an entire path. Each sequence is repeated ten times, with the alignment error measured for each sequence. We report on the alignment error of UFVO (ours), LCSD [92], LDSO [56], DSO [41] and ORB SLAM 2 [124]. Note that I do not include FDMO and FDMO-f in this comparison as they were not designed to be full systems, but rather proof of concepts where FDMO established the robustness gain of a hybrid system while FDMO-f showed that such robustness can be achieved efficiently (relatively small extra computational cost).

Fig. 5.5 shows the median of the alignment error over the ten runs for each sequence (the lower the better). whereas Fig. 5.6 shows the number of runs for which the alignment error (y-axis) was lower than a specific value (x-axis), the steeper the better.

Figures 5.5 and 5.6 summarize the alignment error graphically. The detailed errors (displayed numerically) for the translation, rotation and scale drifts are shown in Appendix .2.

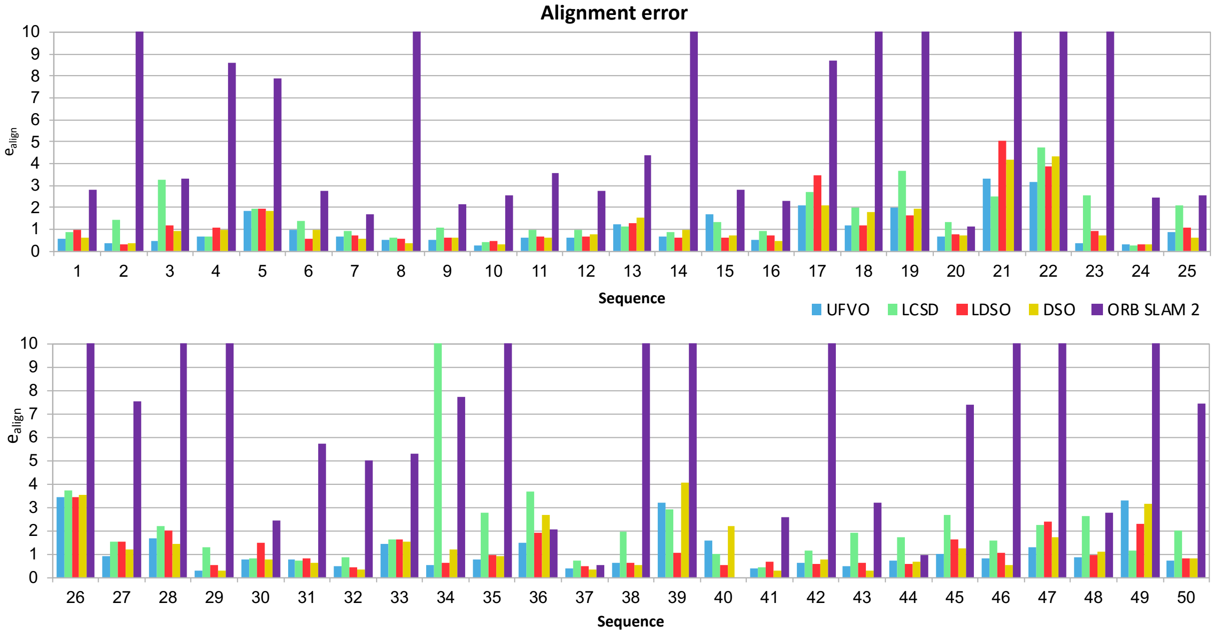


Figure 5.5: The alignment error e_{align}, as defined in [45], for UFVO (ours), LCSD, LDSO, DSO, and ORB SLAM 2 on all sequences from the TUM Mono dataset[45]. The results for LDSO are taken from the extra material provided in [92], DSO and ORB SLAM 2 results are taken from the extra material provided in [41]. Finally the results of UFVO and LDSO are obtained the same way the other systems results were generated, by repeating each sequence ten times and reporting on the median. For the sake of visualization, error values beyond 10 were truncated and are not representative of the actual error.

5.5 Discussion

The computational cost reported in Tab. 5.1 demonstrates UFVO's real-time capabilities; while tracking is relatively slower than DSO, we argue that 14 ms per frame is well below real-time requirements, and is completely justified by the improved robustness. Furthermore, the joint optimization itself can be turned off at frame-rate, and only invoked when tracking failure is detected, thereby reducing the tracking cost to roughly the same as DSO while operating in a state similar to [177]. As for the mapping, at a mere increase of 18 ms, we are able to generate and maintain two types of data representations which can be further exploited in a SLAM formulation. In contrast, maintaining an Indirect map alone requires at least 100 ms per keyframe in ORB SLAM 2.

The results reported on the alignment error (Fig. 5.5 and Fig. 5.6) reflect the complementary

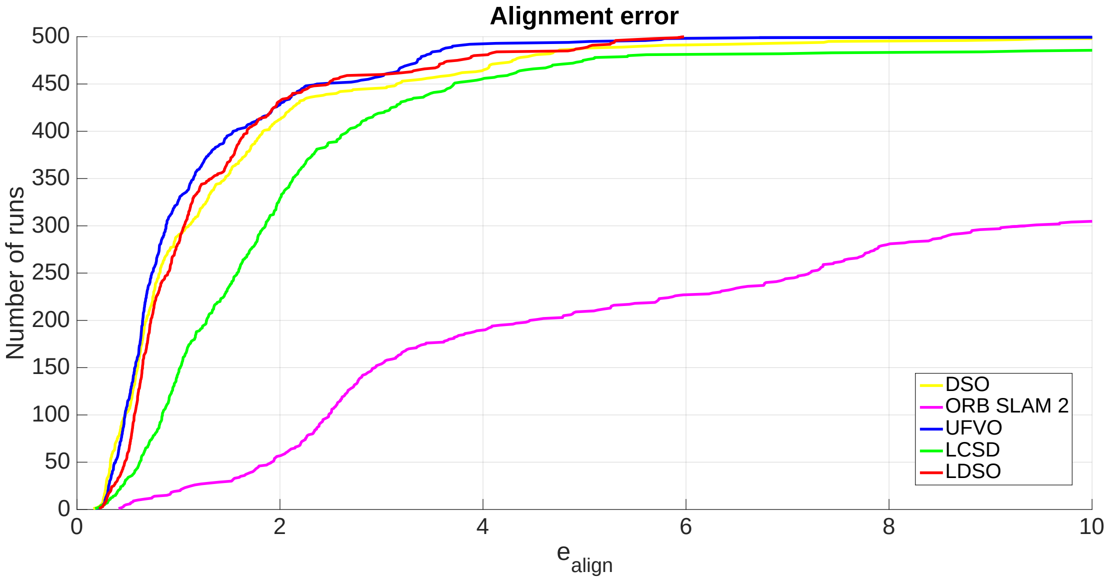


Figure 5.6: The number of runs for which the alignment error was lower than the alignment error shown along the x-axis. The steeper and closer to the point (0,500) the better.

nature of the Direct and Indirect formulations; while the state of the art in Indirect methods (ORB SLAM2) performs consistently worse than DSO, our unified formulation was able to leverage the advantages of both paradigms to achieve a consistent improvement over both DSO and ORB SLAM 2 throughout the various sequences.

Meanwhile, the performance of LCSD was limited in most cases to that of DSO, performing similarly or worse on most sequences. The reason for the similar results is that LCSD performs odometry using a direct formulation [41] without any use of geometric residuals at frame-rate. However, the reduced performance compared to DSO can be attributed to three reasons: 1) first, LCSD uses DSO-reduced [41] instead of regular DSO, which compromises accuracy for computational cost; 2) second, indirect corners suffer from reduced precision due to the discretization of the image domain to corner locations, therefore performing a geometric based optimization using Indirect corners only causes an offset around the previously found DSO optimal state; 3) third, a large number of outlier Indirect feature matches in as little as one keyframe may result in significant discrepancy between the two separate Direct and Indirect maps. All of these reasons highlight the importance of the joint formulation UFVO employs.

As for LDSO, since its ability to perform loop closure is disabled, it theoretically behaves the same as DSO, with the difference that it uses a mixture of both corners and pixels as Direct features. The immunity of corners to drift along edges, introduces performance improvement in

LDSO compared to DSO; however, it does also cause worse results on some sequences, which can be attributed to the lack of a sampling mechanism that enforces homogeneous feature extraction of both types across the image like the 2D occupany grid UFVO employs. Furthermore, the contrast between the results of LDSO and UFVO on most sequences shows that the increased performance UFVO achieves is not due to the mere use of corner features and that UFVO is capable of using the geometric residuals to mitigate the shortcomings of its photometric counterpart.

Despite the improved performance, UFVO under-performs on a few sequences; upon closer inspection of said sequences, they all shared a significant portion of footage on highly repetitive textures (*e.g.*, flooring carpets, grass, etc.), resulting in a large number of outlier indirect feature matches, in turn causing the reduced accuracy. While this is an inherent problem of all Indirect formulations, it was introduced to UFVO by our adaptive corner extraction mechanism that allows weak indirect corners to be extracted at such locations. In fact, increasing the lower bound on the minimum shi-tomasi score for a corner feature, caused an increase in performance by upwards of 50% in the reported error on most of these sequences. However, doing so causes corner deprivation in weakly textured locations across other sequences. Therefore, to maintain the results coherency we evaluate all sequences using the same configuration parameters.

5.6 Conclusion

By allowing corner features to have both photometric and geometric residuals, and adopting an inverse depth parametrization, *UFVO* is capable of generating a single unified map in a single thread that is naturally resilient to texture-deprived environments, at a relatively small computational cost. Furthermore, the suggested joint pyramidal image alignment is capable of exploiting the best traits of both Direct and Indirect paradigms, allowing *UFVO* to cope with initializations far from the optimum pose, while gaining the sub-pixel accuracy of Direct methods. Finally, the proposed point sampling and activation process ensures a homogeneously distributed, rich scene representation. While *UFVO* already outperforms state of the art systems, it does not make use of a global map representation, which is obtainable by maintaining a co-visibility graph over the local Indirect keyframes. Furthermore, the presence of an Indirect observation model allows for the integration of an online photometric calibration which would improve performance on non-photometrically calibrated datasets.

Chapter 6

UFVS: A Unified Formulation for Visual SLAM

In the previous chapter I've shown that UFVO efficiently addresses some of the limitations of Direct, Indirect and Hybrid methods by homogeneously distributing both types of features across an image, and by concurrently associating two types of descriptors with the same features. This chapter, extends UFVO's capabilities to a SLAM system, titled **A Unified Formulation for Visual SLAM**, **UFVS** further exploits the complementary nature of Direct and Indirect representations to perform loop closure, resulting in an efficient global and reusable map.

But before I get into the details of the proposed loop closure process, it is important to highlight the essential components needed to perform loop closure.

6.1 Loop Closure Essentials

Visual Odometry is a numerical optimization process that integrates new measurements given old ones. As the camera explores its environment accumulated numerical errors can grow unbounded, resulting in a *drift* between the true camera trajectory and scene map, and their current system estimates. A typical monocular SLAM system can drift along all rotation, translation and scale axis, resulting in an ill-representation of the camera motion and scene reconstruction, which in turn can lead to catastrophic failure.

Loop closure is a mechanism used to identify and correct drift. The process can be split into three main tasks, (1) loop closure detection, (2) error estimation, and (3) map correction.

Loop closure detection is the process of flagging a candidate Keyframe from the set of Keyframes that were previously observed, and share *visual cues* with the most recent Keyframe (*i.e*. the camera has returned to a previously observed scene). However, significant drift might have occurred between past and current measurements, therefore the Keyframe poses cannot be used to detect loop closure, instead SLAM systems must rely on visual cues only to flag candidate keyframes, hence the naming appearance-based methods.

Error estimation.: once a candidate Keyframe is detected, its corresponding map point measurements are matched to the most recently added 3D map points, and a similarity transform $T \in$**Sim(3) :=** $[s\mathbf{R}|\mathbf{t}]$) that minimizes the error between the two 3D sets is computed. The found Similarity transform is a measure of how much one end of the loop closure keyframes must change to account for the accumulated drift, and is therefore applied on the keyframes such that the matched old and new 3D points are fused together.

Path Correction. The Error estimation process corrects the loop-end keyframes, however it does not correct the accumulated drift throughout the entire path; for that, a SLAM system must keep track of the connectivity information between the path keyframes in the form of a graph, and subsequently correct the entire trajectory using a Pose Graph Optimization (**PGO** —a sub-optimal optimization that distributes the accumulated error along the path by considering pose-pose constraints only while ignoring the 3D observations).

6.2 Literature Review

Loop Candidate Detection. While all SLAM systems perform loop closure following the above three tasks, their implementations differ according to the type of information at their disposal. For example low level features (*e.g*., patches of pixels) are susceptible to viewpoint and illumination changes, and are inefficient at encoding and querying an image, therefore most Direct odometry systems discard them once they go out of view (*e.g*. DSO [41]). To perform loop closure, Direct systems require auxiliary features; LSD SLAM [44] (a Direct system) extracted STAR features [1] from its Keyframes and associated them with SURF descriptors [4], which were in turn parsed through OpenFABMAP [60] (an independent appearance-only SLAM system) just to detect loop closure candidates. In contrast ORB SLAM [124] did not require such process; the same features used for odometry (FAST features [140] and ORB descriptors [141]) were parsed in a Bag of Visual Words model (BoVW) [54] to compute a global Keyframe descriptor. The global Keyframe descriptor was then used to query a database of previously added Keyframes for potential loop closure candidates.
In an effort to extend DSO to a SLAM system, LDSO [56] extracted corner features [146] with their associated ORB descriptors [141]; however, it did not use them in an Indirect formulation,

instead it used the corner features as a source of direct residuals (patches of pixels), and used the ORB descriptors to encode and query the keyframes for loop closure candidates using a BoVW model similar to that of ORB SLAM.

Error Estimation Once a loop candidate is found, all systems proceed by computing a corrective Similarity transformation. LDSO first matches features between the latest Keyframe and the loop candidate, resulting in a matched set of 3D map points in the loop closure candidate's coordinate frame and their corresponding 2D matches in the latest keyframe. OpenCV's RANSAC implementation of EPnP [96] is then used to recover an SE(3) relating the two Keyframes, which initializes a subsequent optimization over a similarity transformation (Sim(3)) that minimizes the error between the set of 3D points in the candidate Keyframe and their corresponding 3D point matches in the latest Keyframe as computed from their most recent measurements in the local active window.

ORB SLAM first establishes 3D-3D map points matches between the loop-ends, which are subsequently used in a RANSAC implementation of [76] to recover a similarity transform directly. The new similarity is used to search for more map point matches across other Keyframes connected to the loop ends. The result is a relatively larger set of 3D map point matches that originates from several Keyframes and not just two, which are then used in a subsequent optimization to further refine the Similarity estimate. The result is a corrective similarity that can be applied to all Keyframes that contributed map points to the similarity estimation process and not just the two loop-ends.

Path Correction. The corrective similarity aligns both ends of the loop so that drift between them is minimized; however, this does not correct the remaining Keyframes along the path, causing a discontinuity between the corrected Keyframes and their non-corrected counterparts. To fix this, the accumulated drift before the correction must be spread throughout the entire path using Pose Graph Optimization. The idea is to modify the entire path to account for the drift measured by the corrective similarity, while preserving as much as possible the relative transformations between connected keyframes. The relative transformations between the connected Keyframes are referred to as pose-pose constraints.

Therefore, to be able to perform PGO, some notion of connectivity between the Keyframes must be established.

ORB SLAM employs several representations of such connections; in particular the **Covisibility** graph is a byproduct of ORB SLAM's feature matching on every Keyframe, where a connectivity edge is inserted between Keyframes that observe the same 3D map points. ORB SLAM also uses a **Spanning tree** graph, made from a minimal number of connection, where each Keyframe is connected to its reference Keyframe. Finally, to perform PGO, ORB SLAM uses an **Essential graph**, which contains the spanning tree and all edges between Keyframes that share more than 100 feature matches. Note that the Spanning tree $\subseteq$ Essential graph $\subseteq$ Covisibility graph; and while the Covisibility graph can be used for PGO, ORB SLAM uses the Essential graph and cites the computational efficiency as a reason since the Covisibility graph might introduce a large number of constraints.

ORB SLAM's connectivity graphs is based on finding enough feature matches between Keyframes; however, LDSO does not have access to such information as it does not perform nor keep track of feature matches between Keyframes. Instead, it considers all Keyframes that are currently active in the local window to be connected and accordingly adds an edge between them in its connectivity graph. While this works well in feature-deprived environments (where not enough feature matches can be established), it has several drawbacks when compared to its ORB SLAM's counterpart. For example when a loop closure takes place, new connections between keyframes that were previously disconnected due to drift can be updated based on their mutually observed map points. This however is not possible in LDSO's graph.
Furthermore, the only criteria for accepting loop closure candidates is that the candidate is not in between the most recent and the oldest currently active Keyframes in the local window. This unfortunately results in a loop closure detection every time the camera re-observes a scene after a mere few seconds of leaving it, thereby triggering unnecessary pose graph optimizations and in some cases corrupting the map when the estimated similarity is noisy or based on a very small number of matches.

Global Map Memory management For proper operation, Direct systems typically add a relatively large number of Keyframes per second; while this is generally not an issue for pure odometry methods (constant memory consumption), the unbounded memory growth becomes an issue for SLAM systems that maintain and re-use a global map representation.

To maintain a reasonable memory consumption and keep compute-time low, ORB SLAM invokes a Keyframe culling strategy that removes Keyframes whose 90% of map points are visible in other Keyframes. This, however has a negative impact on the final results accuracy as culled Keyframes are completely removed from their system, whereas their poses could have been used to better constrain the estimated trajectory.

On the other hand, even though LDSO does not make use of its global map for pose estimation and mapping, it has to store in memory all Keyframes, along with their associated Indirect features, and their depth estimates to be able to perform loop closure. Since LDSO does not perform feature matching to detect redundant Keyframes, it cannot invoke a Keyframe culling strategy, resulting in a relatively large memory consumption.

Finally, despite the fact that DSM [191] does not have a loop closure module, it blurs the line between what can be considered an odometry system *vs.* a SLAM system: it is a Direct method that keeps track of a global map, and attempts to re-use it for both pose estimation and mapping. This however means storing a large number of information per Keyframe and querying them whenever a new Keyframe is added. The result is a global map that requires a relatively large memory consumption, while suffering from a large computational cost whenever a new Keyframe is added. DSM mitigates these issues by adding less Keyframes than other Direct methods.

6.3 Proposed System

UFVS extends the capabilities of UFVO to maintain and reuse a global map with some modifications. The entire SLAM system is shown in Fig. 6.1, and is made of 3 threads: (1) pose estimation at frame-rate, (2) Mapping estimation at Keyframe-rate, and (3) loop closure at Keyframe-rate.

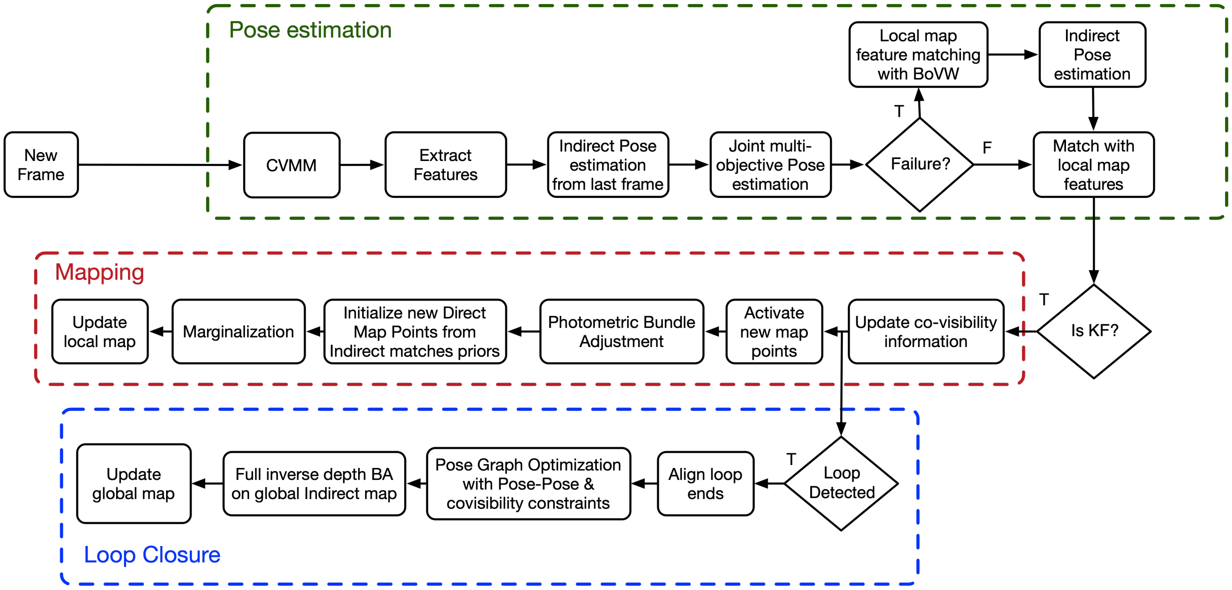

Pose estimation
New Frame
CVMM
Extract Features
Indirect Pose estimation from last frame
Joint multi-objective Pose estimation
Failure?
T
F
Local map feature matching with BoVW
Indirect Pose estimation
Match with local map features
Is KF?
T
Mapping
Update co-visibility information
Activate new map points
Photometric Bundle Adjustment
Initialize new Direct Map Points from Indirect matches priors
Marginalization
Update local map
Loop Detected
T
Align loop ends
Pose Graph Optimization with Pose-Pose & covisibility constraints
Full inverse depth BA on global Indirect map
Update global map
Loop Closure

Figure 6.1: UFVS flowchart showing the various processes in the three main threads.

6.3.1 Pose Estimation

UFVS initializes the new frame pose with a constant velocity motion model and extracts from the image FAST corners with ORB descriptors; the extracted features are matched against the features of the last processed frame using a small search window, and an indirect pose optimization that minimizes the geometric re-projection error (2.6) over the incremental camera pose takes place. The new pose estimate acts as a seed to the multi-objective optimization as described in Sec. 5.3.4 where both Direct and Indirect feature matches from a local map (managed by the mapping thread) are used to compute the camera pose using:

$$\underset{T}{argmin}\, e_{pho}(T, X_d, X_i) + \lambda e_{geo}(T, X_i), \tag{6.1}$$

where T $\in SE(3)$ is the camera pose relating the current frame to the last added Keyframe, X_d and X_i are the set of Direct and Indirect map points respectively, and λ is the relative weight between the photometric and geometric residuals. λ is controlled by a logistic utility function defined in [178], which starts by assigning a large weight λ to the geometric residuals and subsequently reduces it as the optimization progress such that the Direct residuals eventually dominate. This allows UFVS to handle large inter-frame motions while benefiting from the sub-pixel accuracy of Direct methods. Furthermore, the logistic utility function keeps track of the number of inlier geometric residuals within the optimization and accordingly reduces λ in feature deprived environments, allowing UFVS to handle texture-less regions and motion blur using the photometric residuals only. The ability of UFVS pose estimation module to handle large inter-frame motions is tested in [177] where the feature assisted Direct pose estimation was able to handle motions as large as that of state of the art in Indirect methods. On the other hand, UFVS's accuracy is tested in [178] where it outperformed state of the art in Direct, Indirect and other Hybrid systems, over an exhaustive set of sequences covering challenging indoor and outdoor environments.

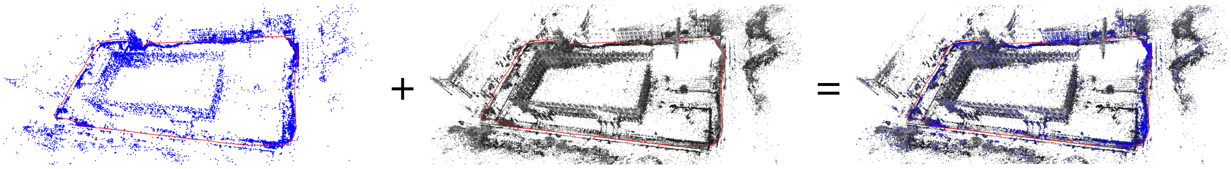


Figure 6.2: Density control and global/local reconstructed maps; the Indirect features are shown in the left image (in blue), the Direct features are shown in the middle image (in black) and the Joint map representation from UFVS is shown in the right image.

6.3.2 Map representation

Local Map

Similar to DSO, UFVS explores the scene as a moving window of locally active Keyframes which are managed as described in ([41], Sec 3.1). The local active window includes both photometric and geometric residuals from the set of currently active 7 Keyframes. However, Indirect map points from the global map can be re-observed across large view-point differences, therefore the local window in UFVS is expanded to include all Keyframes that are connected to the current active Keyframes from the covisibility graph. The set of active and non-active keyframes along with their associated direct and Indirect map points are hereafter referred to as the local map.

Point parametrization

Unlike typical Indirect VSLAM in the literature such as [124] and [83], etc., UFVS adopts an inverse depth parametrization for both Direct and Indirect features and uses the photometric descriptors to triangulate them using small baselines as described in [44]. Aside from its ability to triangulate both representations at virtually no extra computational cost than that of DSO, the short baseline triangulation avoids the numerical instability typically associated with n-view triangulation methods when the views are separated by small baselines. The end result is a single map with both representations as shown in figure 6.2.

Connectivity graph

Since UFVS performs feature matching at frame-rate, it maintains a connectivity graph similar to ORB SLAM's connectivity graph, which includes a spanning tree, an essential graph and a covisibility graph. Furthermore, since UFVS explores the scene in a moving window approach, a pose-pose connectivity graph is also maintained by adding edges between keyframes that are concurrently active in the local active window.

6.3.3 Mapping

While the main photometric Bundle adjustment and marginalization processes are similar to those of [41], some modifications are introduced to make use of the global map. When a

Keyframe is added to the mapping thread, its associated map points are used to update the covisibility graph and the Keyframe is passed to a parallel loop closure thread. Meanwhile in the mapping thread, a set of new 3D map points initialized from previous Keyframes are activated, and points that originate from Indirect features are added to both the local and global maps, whereas points that originate from DSO's point sampling startegy are added to the local map only. To ensure global map points distinctiveness, if an indirect map point to be activated, matches with any map points from the global map, it is converted to a local map point (*i.e.*, no longer considered an Indirect global map point, and only contributes photometric residuals). Once new map points are added, a photometric Bundle adjustment takes place as described in [41] and new 3D map point seeds are initialized from the latest keyframe. If a seed source happens to match a global map point, its inverse depth estimate is initialized to that of its corresponding global match projected to the new Keyframe $\pm 5\sigma_d$, where σ_d is the standard deviation associated with the inverse depth of the global map point. This allows UFVS to make use of global map points to initialize new data, without violating the priors already integrated within the local window (the local window optimization refines the depth estimates in subsequent steps). Finally, the global 3D map point is considered active again, allowing the local window to modify its global pose without the need of a separate structure only optimization process similar to that employed in UFVO [178]. Finally, The local map used for pose estimation is updated.

6.3.4 Loop Closure

UFVS maintains a covisibility graph over the global set of Keyframes allowing it to perform loop closure detection and error estimation using techniques similar to those of ORB SLAM's loop closure module (described in sec. 6.2). It starts parsing the new keyframe into a BoVW model [54] to generate a global Keyframe descriptor, which is in turn used to query the global database of Keyframes for potential matches. To prevent spurious detections (which are common in LDSO [56]), candidates connected to the latest Keyframe in the covisibility graph are discarded. If no loop candidates are found (which is the common case for most Keyframes), the loop closure thread ends here while requiring an average of 8 ms to parse the new Keyframe into DBoW3 and to query the global map. However if a candidate is found, UFVS performs a 3D-3D map point matches between the candidate keyframe and the most recently added Keyframe (at the time of loop closure detection) and uses the matches to compute the corrective sim(3) in a RANSAC implementation of [76]. The corrective Sim(3) is used to establish more 3D-3D map point matches from keyframes connected to both sides of the loop, which are in turn used to refine the Sim(3) estimate. If not enough matches are found, the loop closure thread rejects the candidate Keyframe, otherwise it uses the corrective Sim(3) to correct the poses of all Keyframes that contributed feature matches from one side of the loop. The corrected Keyframes are then considered fixed, and a Pose Graph Optimization (PGO) as described in [124] eq. (9) is used to correct the remainder of the path using constraints from both the essential graph and the pose-pose connectivity graphs.

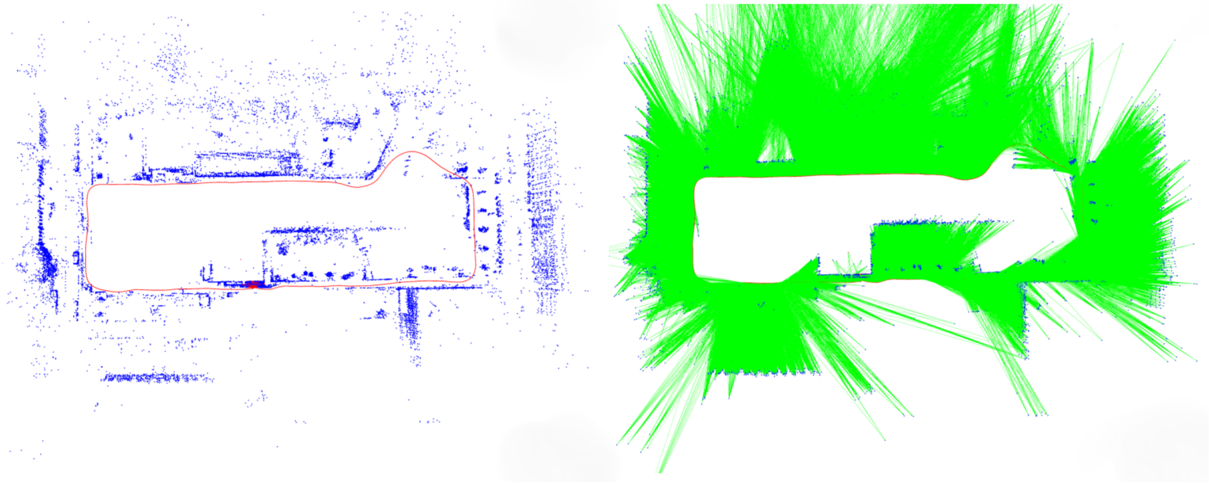


Figure 6.3: Global Bundle Adjustment graph where green lines represent constraints between the 3D map points and their corresponding Keyrames in which they are observed.

Since UFVS relies on a moving active window to explore the scene, modifying the active window keyframe introduces unwanted artifacts as it breaks the pre-integrated priors in the active window. Therefore, similar to [56], UFVS loop closure corrects the side of the loop corresponding to the old previously observed Keyframes, while keeping the most recent active Keyframes fixed. Aside from not breaking the priors in the local window, this has the advantage of running the loop closure correction without the need to lock the mapping thread, that is regular pose estimation and mapping processes can continue normally in their respective threads while the Loop Closure correction takes place on the third parallel thread.
PGO is relatively fast to compute, requiring about 800 ms to correct a path made of 700 Keyframes. It achieves this speed by discarding 3D observations during its path correction, therefore its results are sub-optimal. To achieve optimal results, UFVS further refine the loop closure result by performing a full inverse depth Bundle Adjustment using all connectivity and 3D map point observations. The connected graph of the full BA is shown in Fig. 6.3, and while it is relatively slow to compute requiring about 5 to 8 seconds on average, it can be computed without interfering with the other thread operations as it considers all keyframes in the active window at the time of loop closure detection fixed, it discards all Keyframes that were added after the loop closure detection, and only modifies marginalized Keyframes and map points.

6.4 Results and Discussion

6.4.1 Qualitative Results

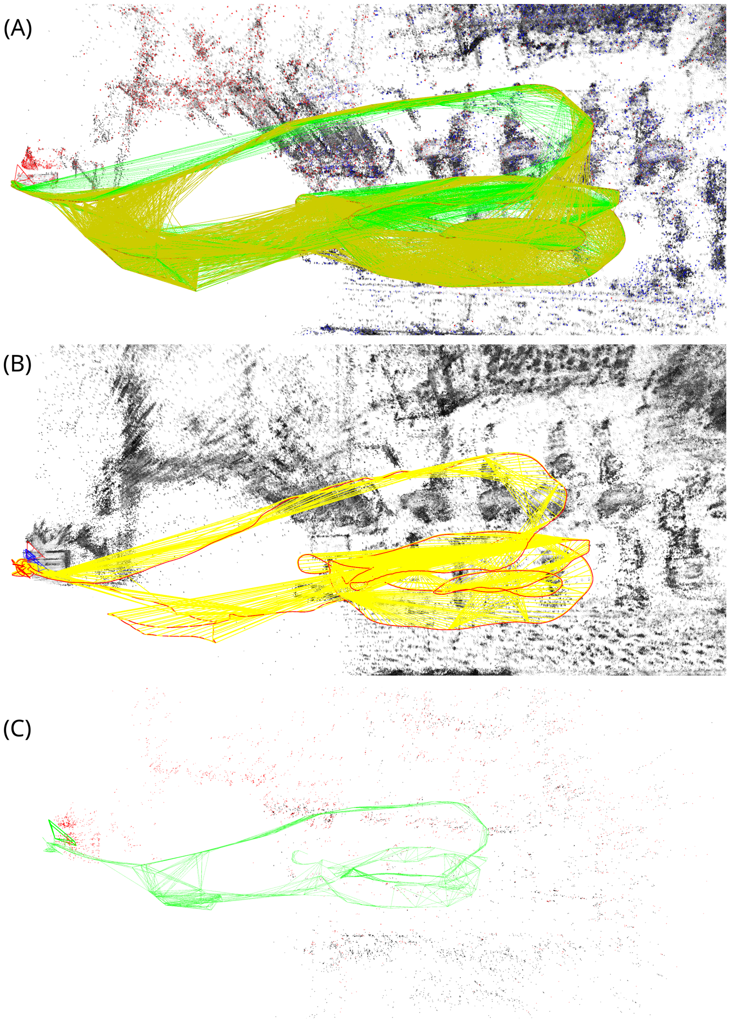


Figure 6.4: Constraints available for Pose Graph Optimization in the Euroc [15] dataset, MH_01_easy left images sequence. (A) Top figure shows both the pose-pose and covisibility constraints of UFVS. (B) Middle figure shows the pose-pose constraints from LDSO [56]. (C) Bottom figure shows the covisibility constraints from ORB SLAM 2 [124].

The use of both covisibility and pose-pose graphs allow UFVS to generate a relatively dense network of constraints when compared to that of ORB SLAM 2's or LDSO's networks as shown in Fig. 6.4. Note that ORB SLAM employs a keyframe culling strategy, thereby removing redundant keyframes from its map. This in turn results in a pruned set of covisibility constraints. On the other hand, LDSO's pose-pose constraints cannot be updated to account for new constraints when loop closure takes place. In contrast, UFVS can recognize and add new connections between keyframes that were once unconnected due to large drift.

6.4.2 Quantitative Results

Computational cost

The computational cost analys of the various SLAM systems was performed on the same CPU Intel core i7-8700 CPU @ 3.70GHz CPU; no GPU acceleration was used. The results are shown in table 6.1. To ensure fairness, all systems are ran across the same sequence and the results are averaged across the entire MH_01_easy sequence of the Euroc dataset [15]. Note that this sequence is mostly a camera moving around a closed room. The obtained times may differ in different scenarios (*e.g.*, the computational cost might be different in a pure exploratory sequences).

Table 6.1: Average computational cost (ms) associated with tracking and mapping threads for various VSLAM systems and DSO as odometry system.

Average Time (ms)	UFVS	LDSO	ORB SLAM 2	DSM	DSO
Tracking (frame-rate)	14.12	6.11	27.38	8.41	6.08
Mapping (Keyframe-rate)	65.43	93.64	312.08	620.17	51.6

The average tracking time per frame for UFVS is 14 ms, during which an indirect optimization first takes place, followed by a joint multi-objective optimization. In contrast, LDSO and DSM are Direct systems, thus only require around 6 ms to perform Direct image alignment. Finally ORB SLAM 2 requires an average of 27 ms to extract features from several pyramid levels and computes the frame's pose.

The entire mapping process used in UFVS requires on average 65 ms per keyframe to generate and maintain both the direct and Indirect global maps. In contrast LDSO mapping thread requires 93 ms to process a keyframe, and ORB SLAM 2 requires 312 ms. Note that ORB SLAM 2's mapping process performs a local Bundle Adjustment every time a new Keyframe is added, and since the tested sequence is a closed room, the local map is relatively large when compared to pure exploratory motion; and hence the relatively slow time. Finally DSM requires an average of 620 ms per Keyframe. The very large increase in computational cost is attributed to DSM's pyramidal photometric Bundle Adjustment, which it repeats on three levels. In contrast UFVS, DSO and LDSO performs their local photometric BA on a single pyramid level. Finally, note that DSO is an odometry system and not a SLAM system but it is just added to the table to give some perspective into the computational cost differences between odometry and SLAM.

As for loop closure detection and PGO, they are typically infrequent and can happen at different locations in the scene for various SLAM systems, and as such it is difficult to report and compare their average computational cost. Therefore, the results for a sample case in UFVS are reported, where for a sequence of 597 Keyframes, it took 8 ms to query the global map for a loop candidate, 16 ms to estimate the similarity transformation and correct the loop ends, 850 ms to perform Pose Graph Optimization using both covisibility and pose-pose constraints on the entire sequence, and 5.4 seconds to perform the full photometric Bundle Adjustment.

Memory cost

The memory cost associated with the various VSLAM systems run on the same Euroc MH_01_easy sequence are shown in Table 6.2. Note that UFVS, LDSO, and ORB SLAM 2 use a Bags of Visual words dictionary to detect loop closure candidates. The dictionary must be loaded in memory and has a constant size of 439 MBs (not shown in the table). In contrast DSM does not perform loop closures, and therefore does not require the bags of words dictionary; however, its memory cost per keyframe is significantly higher than the other systems.

Since UFVS has a covisbility graph, it is possible to prune its global map similar to what is proposed in ORB SLAM [124] by removing redundant Keyframes that share more than 90% of their map points with other Keyframes. Therefore UFVS has the optional setting of performing Keyframe culling. for example on the MH_01_sequence, it keeps 249 Keyframes, and 13 062 map points for a total memory cost of 317 MB. Note that unlike ORB SLAM 2, culled Keyframes still contribute pose-pose constraints, which can be helpful to correcting the estimated trajectory around them. Keyframe culling is turned off for all experiments performed in this chapter.

Table 6.2: Memory cost associated with the various VSLAM systems. UFVS, LDSO and ORB SLAM 2 require a constant extra 439 MBs to load the Bag of Words dictionary into memory.

	UFVS	LDSO	ORB SLAM 2	DSM
Number of Keyframes	484	705	204	122
Number of Map Points	23023	-	8335	31974
Memory Usage (MB)	409	722	336	805

Path accuracy

While in previous chapters the TUM MONO dataset [45] was used to evaluate the performance of various proposed systems, it cannot be used to evaluate UFVS. The dataset does not contain ground truth information for the entire path, it only provides localization information for the beginning and end of each sequence. However, UFVS is a SLAM system and performs loop closure, thus its computed trajectory beginning and end are likely be aligned through the loop closure process, resulting in overconfident results that might not reflect the true trajectory errors. For this reason the experimental evaluation for UFVS is performed using the EuroC dataset [15] which provides ground truth data throughout the entire sequence. Each sequence was repeated ten times and the resulting median is shown in Table 6.3.

Table 6.3: Localization error (meters) on the Euroc dataset (left images) [15]. Each experiment is repeated 10 times and the resulting medians are reported below. Note that LDSO's, ORB SLAM 2's and DSM's results were lifted from [191]. The reported results for UFVS were obtained without the global Bundle Adjustment process.

Sequence	UFVS	LDSO	ORB SLAM 2	DSM
MH_01_easy	**0.035**	0.053	0.07	0.039
MH_02_easy	**0.034**	0.062	0.066	0.036
MH_03_medium	0.111	0.114	0.071	**0.055**
MH_04_difficult	0.111	0.152	0.081	**0.057**
MH_05_difficult	0.064	0.085	**0.06**	0.067
V1_01_easy	0.039	0.099	**0.015**	0.095
V1_02_medium	0.085	0.087	**0.02**	0.059
V1_03_difficult	0.266	0.536	-	**0.076**
V2_01_easy	0.037	0.066	**0.015**	0.056
V2_02_medium	0.047	0.078	**0.017**	0.057
V2_03_difficult	1.218	-	-	0.784

UFVS outperforms LDSO on all sequences despite that both systems use a local moving window to explore the scene. The performance improvement is attributed to three reasons: (1) the improved accuracy of the hybrid pose estimation process, (2) the improved connectivity graph that contains both covisibility and pose-pose constraints, and (3) the global Bundle Adjustment that optimizes the global map. On the other hand, UFVS scores between ORB SLAM and DSM, outperforming each on several sequences while under-performing on others. The mixed results can be attributed to several factors. One of the common reasons for UFVS's under-performance is not invoking loop closures: UFVS rejects loop closure candidates if recent map points are observed in them, since that is an indication that not drift has occurred. Since loop closures are not performed, neither Pose Graph Optimization nor Bundle Adjustments are invoked over the entire sequence, resulting in reduced accuracy.

In contrast, ORB SLAM 2 rejects loop closure candidates for similar reasons, but invokes a local Bundle Adjustment after every Keyframe. Since most sequences in Euroc are of a relatively small room, the consistently re-occurring Local Bundle Adjustment covers most Keyframes several times, resulting in the improved accuracy.

Finally, DSM achieves superiority over UFVS and ORB SLAM on several sequences by limiting the number of new Keyframes it adds, and reusing old Keyfarmes with their associated data. While the other SLAM systems query a relatively sparse global map of Indirect features, DSM keeps track and queries all of its photometric residuals in a global Direct map. This unfortunately comes at a very high computational and memory costs, resulting in below real-time performance. In contrast, UFVS's back-end is about nine and five times faster than DSM's and ORB SLAM's back-ends respectively, while achieving competitive results on most sequences.

6.5 Conclusion

I have presented in this chapter UFVS, a SLAM extension to UFVO capable of maintaining and reusing a global Indirect map and a local hybrid one. By further exploring the complementary nature of Direct and Indirect residuals, UFVS is capable of generating an Indirect global map at a fraction of the computational cost required by other Indirect SLAM systems.

UFVS is also capable of performing loop closure using Pose Graph Optimization over both a covisibility and pose-pose graphs concurrently; the sub-optimal results of the PGO are further refined in a global Indirect Inverse depth Bundle Adjustment process. The end result is a competitive SLAM system that is computationally and memory efficient, and achieves state of the art performance on some sequences of the Euroc dataset while being competitive with other state of the art SLAM systems despite being almost an order of magnitude faster than them.

Chapter 7

Deep Camera Intrinsic Calibration from Natural Scene Images

An underlying assumption to everything presented in this thesis thus far is the availability of a camera calibration matrix **K**, that is used to project 3D points from the camera's homogeneous space to the image frame. However, obtaining the camera calibration requires the presence of a special calibrating pattern, which may not always be viable.

In this regard, Recent deep learning approaches have shown some promise in recovering camera intrinsic variables from natural scene images without the presence of a calibrating pattern; nevertheless, their relatively low accuracy, coupled with their large architectures have prohibited their use in real life applications. Inspired by traditional auto-calibration methods, this chapter proposes a new data-driven approach, which exploits the underlying structure encoded in multiple views of the same scene to recover the camera intrinsic parameters. This is the first work that leverages the traditional multiple-view solutions in the design and training of a model for the sole purpose of estimating the focal length and principal point position of a camera. The accuracy of the proposed model outperforms traditional methods by 2 to 30% and outperforms recent deep learning approaches by a factor of 2 to 4 times. Achieving these results would not have been possible without a new dataset, which covers both synthetic and real scenes and includes a varying range of focal lengths, aspect ratios and principal points, tailored to the camera calibration task.

7.1 Camera Calibration: An Introduction

Camera calibration is the process of recovering intrinsic parameters *e.g.* the focal length *f*, principal point *c*, skew angle γ, *etc*., which are necessary for mapping from the 3D coordinate frame of the camera to its corresponding 2D image frame. The accurate knowledge of this projection is therefore vital for various vision-based (Structure from Motion) and robotics (Simultaneous Localization and Mapping) systems, where image observations are continuously projected from 3D to 2D and vice versa. It then comes to no surprise that the camera calibration process has been extensively studied for decades, resulting in an ample amount of solutions put forward in the literature like Kruppa's equations from Epipolar geometry and vanishing points methods to name a few.

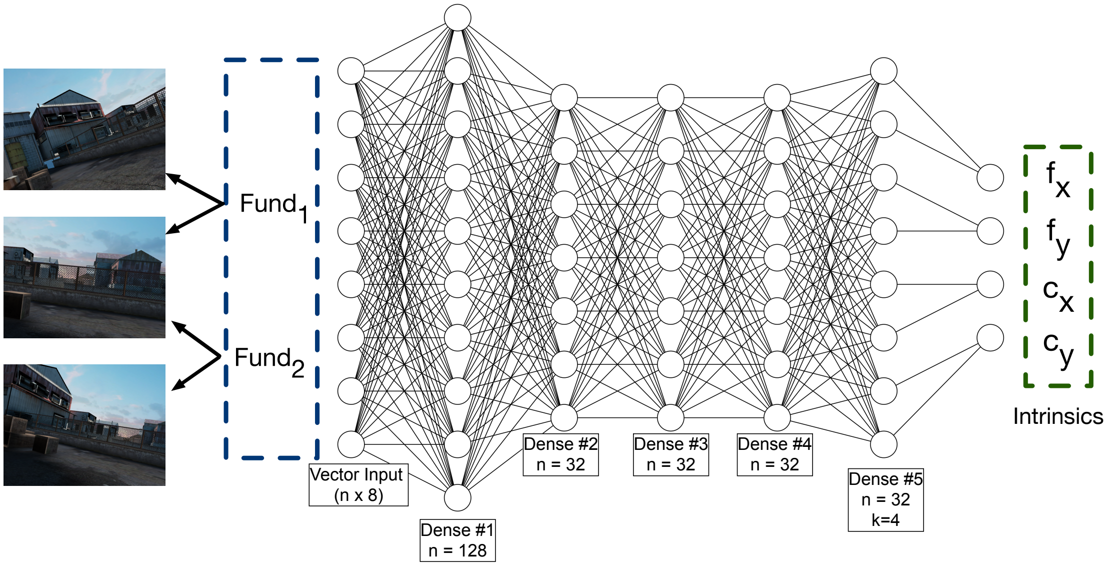


Figure 7.1: Summary of the proposed camera calibration: a minimal set of two Fundamental matrices relating three sequential images is fed into a compact deep model to recover both the focal length (f_x, f_y) and principle point coordinates (c_x, c_y).

Most calibration solutions suffer from various limitations, for example, recovering the camera calibration using Epipolar geometry [47] suffers from computational challenges in solving the coupled non-linear Kruppa equations. On the other hand, recovering the camera calibration from a single image is inherently degenerate (infinite possible camera calibrations), and is only reduced to a unique solution if one can reliably extract 3 perpendicular planes (under the Manhattan world assumption), from which the vanishing points can be recovered.

Recent successes in data driven approaches have enticed the research community to leverage the advantages of modern machine learning algorithms with automatic feature selection to address these limitations. Several attempts such as [165], [74] and [104] were made to retrieve the camera parameters from a single image using an end-to-end approach, thereby automating the entire camera calibration process. To achieve this, the networks were implicitly biased to learn the vanishing points method by introducing hyper-parameters involving the horizon line, roll and tilt into their loss function. However, most methods relied on a generic deep convolutional architecture designed for image classification: the learned models had millions of parameters and yet did not yield accurate results for real-life deployment. Furthermore, due to the lack of datasets tailored for the camera calibration problem, the required training data was generated by cropping images from large panoramas, and therefore lacked any information on the principal point, an essential component in the calibration of real cameras.

While the noise prone nature of vanishing points did not deter researchers from attempting camera calibrations using single images (Sec. 7.2), recovering the camera calibration using the minimum case of three images instead of one, is inherently more stable as it returns a unique solution under random Euclidean motion and does not require a Manhattan world assumption. Given that most applications requiring camera calibrations already have a moving camera and produce image sequences, this work proposes, contrary to the recent data-driven approaches that recover the camera parameters from a single image, an architecture that exploits the inherent structure encoded in multiple views (as shown in Fig. 7.1) to perform the camera calibration.

The **first contribution** is an architecture which consists of Fundamental matrices as input (computed from temporally adjacent image pairs) to a FCN (Fully Connected Network) trained using the ground truth camera parameters as a supervisory signal. Three images is the minimum number required to fully constrain the camera intrinsic parameters to a unique solution using the underlying Epipolar geometry[119]. To train the proposed network, a custom dataset tailored for the camera calibration problem had to be created since readily available datasets lacked the necessary structure and information. The **second contribution** is the development and release of this dataset made of 80000 synthetic images and 84,000 real images, sorted into 6,000 sequences of 14 images each, where each sequence has a unique set of ground truth camera calibration parameters including the focal length and the principal point. Finally, the **third contribution** is to show that a simple compact trained Fully Connected model (only 4 hidden layers) with only 17K parameters can outperform current state of the art self-calibration models which have more than 26M parameters.

7.2 Related Work

Camera calibration solutions were proposed as early as 1986 [157], where a single image of a planar scene was used along with the camera sensor's manufacturer data to recover the camera intrinsics, extrinsic camera pose $[\mathbf{R}, \mathbf{T}]$ and radial distortion. Zhang *et al*. circumvented the need for the sensor manufacturer's data in [186] by using a 3D calibration grid of known 3D coordinates. In turn, the requirement of a 3D calibration grid was relaxed in [187] where multiple images of a two dimensional grid pattern were used to perform the calibration.

While the aforementioned approaches required the physical presence of a calibrating pattern in the observed images, others such as [19],[5], [24], and more recently [70, 142], and [21] investigated the properties of projective transformations in special scenes to recover the camera intrinsics. In particular, under a Manhattan world assumption, one can recover three vanishing points from three sets of mutually orthogonal parallel line directions (two in-plane and one corresponding to the vertical direction) which can then be used to infer the camera roll, tilt and its intrinsic parameters.

However, detecting vanishing points came with its own challenges, and in an attempt to circumvent their Manhattan world assumption and noise-prone nature, researchers investigated the behavior of virtual objects placed at infinity and their relationship to the Epipolar geometry. By construction, a virtual object known as the absolute conic located on a plane at infinity is invariant under Euclidean transformations. As such, its image in the camera frame would be the same from various point of views and is only affected by the camera intrinsic parameters. Various researchers [47, 181, 67, 105, 162], and [119] explored the constraints Epipolar geometry imposes on the image of the absolute conic, giving rise to Kruppa's equations, where each pair of images imposes two linearly independent constraints. Since the absolute conic has five degrees of freedom, at least three images are required to establish six linearly independent constraints in order to solve for a unique camera calibration. Unfortunately, the solution of such systems is cumbersome and prone to degeneracy and measurement noise. More recently, [49] revisited Kruppa's equations for the varying intrinsics case proposed in [106] and formulated a better behaved optimization. However, their modifications only applies for the varying intrinsics case and for large focal length magnitudes (tailored for camera-projector systems), which are beyond most values that are typical for consumer or computer vision cameras.

Given the successes of deep learning in various computer vision tasks, recent methods were developed to retrieve the camera parameters from images using a CNN. [165] first proposed to recover the focal length by finding the horizontal field of view (H_θ) using transfer learning with AlexNet. While the proposed approach reported a success rate of less than 50% in finding H_θ within some error threshold, it demonstrated the feasibility of the problem. In [11], both the focal length f and the image distortion parameter were estimated using the Inception V3 architecture from single images. [74] tackled the same problem, but used a DenseNet architecture [78] instead. Proxy parameters were also introduced in their loss function to implicitly bias the network towards behaving similar to the traditional camera calibration process of vanishing points. The network architecture was revised in [104] to additionally predict the distortion coefficient. However, as is the case with traditional methods, vanishing points are very sensitive to noise and require the presence of man-made structures (Manhattan world assumption). Furthermore, not embedding the domain knowledge in the design of their architectures resulted in significantly large networks, further prohibiting their applicability to practical problems. Also, [74, 104, 11] assume the principal point to be located at the center of the image and the image pixels have zero skew. While real modern cameras have minimal skew, they almost never have their principal point located at the image center. [190] followed a similar approach to [165] by a adopting a fully convolutional network based on ResNet to estimate both the camera calibration and radial distortion from single images; however, without the explicit biasing of the model towards vanishing points, the sought after parameters are ambiguous from single images, as an infinite set of solutions exists.

Finally there is a large body of research on calibrating sports cameras, however such approaches are tailored to specific scenarios as they mostly exploit PTZ (rotating and zooming only) cameras or the known markings of the observed field and its planarity to recover the camera intrinsics, and as such are out of scope for this work.

In contrast to the above, the proposed approach embeds the theory and constraints of the traditional self-calibration methods [67] in the design and structure of the proposed network, resulting in a relatively compact model. Furthermore, the assumptions made regarding the location of the principal point and aspect ratio are relaxed by allowing the model to estimate their values, thereby outperforming state of the art approaches and achieving an error of about 10%. This was made possible by a new training dataset that provides reliable and accurate supervisory signals (i.e., degenerate cases are not included).

7.3 Background

Before I proceed, it is noteworthy to mention that the input to all of the developed methods in this work is a set of radial distortion free Fundamental matrices. The algorithms presented are agnostic to the method used to find the Fundamental matrix, whether from radial distortion free images or from images with radial distortion using the 9 point algorithm [51]. For the remainder of the paper, matrices are denoted with bold upper case characters, vectors with lower case bold characters and scalars with regular font.

7.3.1 Kruppa's equations

In a distortion-free pinhole camera model, a 3D point $[X,\ Y,\ Z,\ 1]^\intercal$ in the world frame, projects to a 2D point $[u,\ v,\ 1]^\intercal$ in the image plane using:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}\mathbf{P}^c_w \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{7.1}$$

where $\mathbf{K}$ is the camera's intrinsic matrix and $\mathbf{P}^c_w \in \mathrm{SE}(3)$ is a 3×4 camera pose matrix in the world coordinate frame. However, in most computer vision applications neither $\mathbf{K}$, $\mathbf{P}$ nor the 3D points are known in advance; instead, 2D point correspondences established through feature matching between two images are typically used to estimate a 3×3 Fundamental matrix $\mathbf{F}$, that encodes both $\mathbf{K}$ and the pose $\mathbf{P}$ between the pair of images.
Under this projection model, several objects typically found at infinity, are invariant to Euclidean transformations $\mathbf{P}$, and only depends on the camera intrinsics $\mathbf{K}$. This allowed [47, 181, 67, 106] and several others to develop self-calibrating systems capable of recovering the camera intrinsics from moving cameras.

Most self-calibration methods rely on the absolute conic Ω—a virtual object located on a plane at infinity and invariant under Euclidean transformations—to estimate the camera intrinsics. Since the absolute conic is invariant to translation and rotation, its projection known as the image of the absolute conic ω is also independent of the position and orientation of the camera. Therefore, ω is only dependent on the camera matrix $\mathbf{K}$ that embeds the intrinsic parameters. Consequently, finding ω is equivalent to estimating the camera intrinsic parameters. In fact, it can be shown that:

$$\omega = (\mathbf{K}\mathbf{K}^\intercal)^{-1} = \mathbf{K}^{-\intercal}\mathbf{K}^{-1} \tag{7.2}$$

In practice, it is the "Dual Image of the Absolute Conic" (DIAC) $\omega^* = \omega^{-1}$ that is used, where $\omega^* = \mathbf{K}\mathbf{K}^\intercal$; the DIAC allows for the recovery of a unique upper triangular matrix $\mathbf{K}$ using Cholesky decomposition.

In order to properly constrain ω^* (equivalently ω), at least 8 point correspondences between 3 different images are required. Two Fundamental matrices describing two sets of motion can then be computed by solving $\mathbf{M}'^\intercal\mathbf{F}\mathbf{M} = 0$, where $\mathbf{M}$ denotes a set of 2D homogeneous points and $\mathbf{M}'$ the set of their respective point correspondences in the other images.

Each Fundamental matrix is then decomposed through SVD into $\mathbf{F} = \mathbf{U}\mathbf{D}\mathbf{V}^\intercal$. The Fundamental matrices are rank deficient (rank 2) and as such each provides three Epipolar constraints of which only two are independent. The resulting constraints known as Kruppa's equations can then be defined as:

$$\frac{r^2\mathbf{v}_1^\top\mathbf{K}\mathbf{v}_1}{\mathbf{u}_2^\top\mathbf{K}'\mathbf{u}_2} = \frac{rs\mathbf{v}_1^\top\mathbf{K}\mathbf{v}_2}{-\mathbf{u}_2^\top\mathbf{K}'\mathbf{u}_1} = \frac{s^2\mathbf{v}_2^\top\mathbf{K}v_2}{\mathbf{u}_1^\top\mathbf{K}'\mathbf{u}_1}, \tag{7.3}$$

where $\mathbf{u}_i, \mathbf{v}_i$ are the respective i^{th} column vectors of the matrices U and V. For further details on the derivation of Kruppa's equations from the Fundamental matrix, the reader is referred to [105].

7.4 Proposed System

7.4.1 Solving Kruppa's equations

Equation (7.3) assumes different camera calibrations $\mathbf{K}$ and $\mathbf{K}'$ for each of the images used to compute $\mathbf{F}$, and can therefore handle cameras with changing intrinsics as long as a sufficient number of observations is available. However, to avoid various unwanted artifacts like recurrent defocus, most practical computer vision applications restrict the camera intrinsics to be fixed. Therefore, for practical purposes the camera intrinsics are assumed fixed ($\mathbf{K} = \mathbf{K}'$). This has considerable ramifications on the performance and algorithms developed as I adopt the nonlinear optimization of [105] with various modifications. Let ρ_i be the numerator of the i^{th} equality term in (7.3) and ϕ_i the denominator. Equation (7.3) can then be rewritten as:

$$\begin{cases} \pi_1 = \frac{\rho_1}{\phi_1} - \frac{\rho_2}{\phi_2} = 0 \\ \pi_2 = \frac{\rho_1}{\phi_1} - \frac{\rho_3}{\phi_3} = 0 \\ \pi_3 = \frac{\rho_2}{\phi_2} - \frac{\rho_3}{\phi_3} = 0 \end{cases}, \tag{7.4}$$

where two of which are linearly independent.

In the fixed intrinsics case, the number of independent equations needed to solve for a unique camera's intrinsics is equal to the number of unknowns in $\mathbf{K}$, which is parametrized with the assumption of zero skew as:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}.$$

With the four unknowns (f_x, f_y, c_x, c_y), two Fundamental matrices each providing two independent equations are sufficient to uniquely determine $\mathbf{K}$. However, I found that the stability and accuracy of the objective function is positively correlated with the number of Fundamental matrices used; as such, the objective function is designed to handle n Fundamental matrices.

Since the goal is to minimize all three equations for n Fundamental matrices, the nonlinear optimization is then formulated as:

$$\operatorname*{argmin}_{\mathbf{K}\in\Re_{+}^{3\times 3}} \sum_{i=0}^{n} w_i(\pi_{i,1}^2 + \pi_{i,2}^2 + \pi_{i,3}^2), \tag{7.5}$$

where i refers to the i^{th} Fundamental matrix, and w_i is a weighting factor that down-weights the contribution of unreliable Fundamental matrices to the overall objective function. While there are many ways of quantifying the reliability of a Fundamental matrix, I chose to use a normalized version of the fundamental matrix constraint:

$$w_i = \frac{\frac{1}{(\mathbf{M'FM})_i}}{\sum_{j=1}^{n} \frac{1}{(\mathbf{M'FM})_j}}, \tag{7.6}$$

since it is readily available from the Fundamental matrix estimation process. The non-linear constrained optimization defined in (7.5) is then solved using the interior point method, which requires an initialization point.

To find this initialization point (7.4) is solved by making two assumptions, first the principal point is centered: $(c_x, c_y) = (width/2, height/2)$, and second, that the aspect ratio is equal to one $(f_x = f_y = f)$. Note that since each Fundamental matrix provides two independent equations, one could solve (7.4) for two parameters at a time; that is, I could have gotten away with the second assumption and simultaneously solved for both f_x and f_y. However, because of noise in the Fundamental matrices, solving a system of equation in both unknowns often resulted in physically meaningless imaginary pairs of solutions, not to mention that it is not trivial to identify which of the two equations are independent. For these reasons, and since the values found will only serve as initializing points for the subsequent optimization, I opted to go with the unit aspect ratio assumption.

Each equation in (7.4) is quartic in f and yields four possible values for a total of 12 possible answers per Fundamental matrix; most of these values are either negative or imaginary and are as such discarded for being physically meaningless. The process is repeated for all Fundamental matrices and the median of all the valid (real positive) values is used as the initializing point to both f_x and f_y. I also experimented with refining the principal point initialization using the newly found focal length but found no significant improvement in the final optimization accuracy, as a result, kept it as is.

7.4.2 Network Architecture

While solving Kruppa's equations provides a systematic solution to the calibration, it requires some ad-hoc procedures and is relatively slow to compute. Instead, a computationally efficient and compact deep network that leverages the lessons learned in solving Kruppa's equations is sought. The proposed network (Fig. 7.1) consists of 4 fully connected hidden layers whose input is a set of Fundamental matrices; since Fundamental matrices are rank deficient, each one is flattened to a vector of 8 elements (the fundamental matrices were normalized so that their 9th element = 1). The architecture design was guided by minimizing the number of parameters and the hyperparameters were determined empirically. The number of input Fundamental matrices is a also hyperparameter to tune but cannot be less than two as anything less is not enough to uniquely constrain the intrinsics matrix. The output is fed to four independent regressors, each dedicated to a specific calibration parameter. In order to ensure better generalization, a dropout of 10% is applied to the first four layers. Early stopping yields an optimal result at around 280-300 epochs. The network is trained with a Huber loss and optimized using Adam solver with a learning rate of 0.0001. Finally, the batch size of 64 generalized the best and was therefore adopted.

7.4.3 Dataset

A major challenge to the proposed data-driven camera calibration approach is the lack of suitable datasets tailored to the problem's nature. This forced [74], [104], [11] and [182] to adapt the SUN360 [166] dataset, originally designed for scene recognition, to the task at hand, by warping the panoramas from an equi-rectangular projection to a pinhole one. However, the warping approach can only generate sequences of purely rotating cameras and is therefore not suitable to generate continuous image sequences undergoing general euclidean transformations, typically found in real life applications. Furthermore, the aforementioned papers only consider the case of centered principal point with unit aspect ratio, which is hardly the case for most cameras.

In light of the lack of datasets for the self intrinsic calibration problem, a large dataset of video sequences had to be generated and made from both real and synthetic images, along with their associated ground truth intrinsic calibrations.

Synthetic Video Sequences

The synthetic dataset is generated using the Unity engine; over $80,000$ images are produced in batches of 14, resulting in about $6,000$ different video sequences. Images within the same sequence have the same camera intrinsic parameters as they undergo general and incremental

Euclidean transformations. Camera rotation and translation is performed by smoothly transitioning from an initial pose $[\mathbf{R_0}|\mathbf{t_0}]$ to another $[\mathbf{R_1}|\mathbf{t_1}]$ across a number of frames in a linear fashion as described in (7.7).

$$\begin{aligned}
x_{t+1} &= x_t + d_x(t) + \mathcal{N}_x(0,1)\\
y_{t+1} &= y_t + d_y(t) + \mathcal{N}_y(0,1)\\
z_{t+1} &= z_t + d_z(t) + \mathcal{N}_z(0,1)\\
\phi_{t+1} &= \tfrac{\phi_f-\phi_i}{t_f-t_i}(t+1)\\
\gamma_{t+1} &= \tfrac{\gamma_f-\gamma_i}{t_f-t_i}(t+1)\\
\psi_{t+1} &= \tfrac{\psi_f-\psi_i}{t_f-t_i}(t+1)
\end{aligned} \tag{7.7}$$

Additional white noise is added to the incremental displacement vectors (d_x, d_y, d_z) to introduce randomness in the camera's motion.

Since the camera is free to move in 3D space, invalid sequences caused by camera clipping into objects are accordingly detected and removed. The 640x480 pixels images originate from a dozen of different indoor and outdoor scenes, covering rural, urban, natural, and man-made environments (Fig. 7.2).

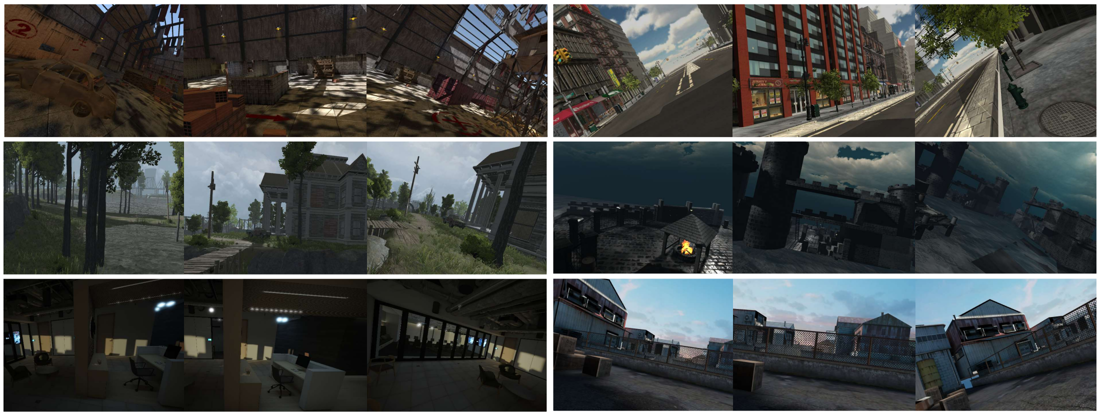


Figure 7.2: Sample image sets from the synthetically generated image dataset (showing randomly selected 3 of 14 images per set). The samples show the wide variety of scenes, covering both indoor and outdoor conditions, and the motion randomness between the frames.

The distribution of the variables used in the synthetic dataset generation along with the camera intrinsic parameters range are presented in Table 7.1. The generated dataset includes the ground truth focal length (f_x, f_y), principal point (c_x, c_y), relative camera position and orientation to the first frame in each sequence as well as the absolute camera pose in the world coordinate frame. The presence of these labels allows the use of this dataset in a variety of computer vi-

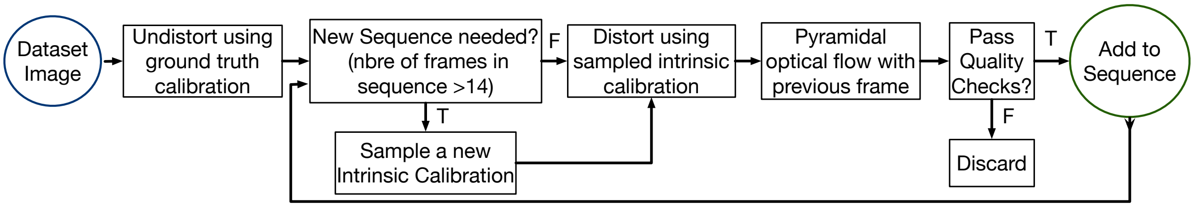

Dataset Image
Undistort using ground truth calibration
New Sequence needed? (nbre of frames in sequence >14)
F
T
Sample a new Intrinsic Calibration
Distort using sampled intrinsic calibration
Pyramidal optical flow with previous frame
Pass Quality Checks?
T
F
Discard
Add to Sequence

Figure 7.3: Real Dataset sequence generation process. The quality checks are detailed in Sec. 7.4.3

sion applications that require the knowledge of the camera's pose and calibration. The generated images are then divided such that 64,000 images are used to train the network, 13,000 for validation, and 3,000 for testing. Note that the test sequences were generated from a set of Unity scenes that were not used to generate the training and validation sets.

The Fundamental matrices needed to train the proposed model are computed from the pose estimates and known intrinsic matrices **K**. Let **P** and **P**′ denote the camera poses of two different images of the same sequence. The Fundamental matrix relating them is then computed as

$$\mathbf{F} = \mathbf{K}^{-\mathsf{T}}[\mathbf{t}]_{\mathbf{x}}\mathbf{R}\mathbf{K}^{-\mathbf{1}}, \tag{7.8}$$

where **R** and **t** are the relative rotation and translation matrices relating **P**′ and **P**, and $[\cdot]_x$ is the skew-symmetric operator. In order to ensure scale independent Fundamental matrices, the

Table 7.1: Parameter distribution used in generating the synthetic dataset. The principal point is computed as image center * (1 + lens shift). Displacement units are in meters.

Extrinsic Parameters	Value
Displacement in x	$\mathscr{U}(-0.9, 0.9)$
Displacement in z	$\mathscr{U}(-0.9, 0.9)$
Displacement in y	$\mathscr{U}(-0.36, 0.36)$
Tilt	$\mathscr{U}(-40, 40)$
Roll	$\mathscr{U}(-20, 40)$
Pan	$\mathscr{U}(-180, 180)$
Intrinsic Parameters	**Value**
Focal Length in x	$\mathscr{U}(200, 775)$
Focal Length in y	$\mathscr{U}(150, 580)$
Lens Shift in x	$\mathscr{U}(-0.2, 0.2)$
Lens Shift in y	$\mathscr{U}(-0.2, 0.2)$

translation vector is normalized using its L2 norm.

Real Video Sequences

Since collecting a large body of video sequences and manually calibrating them is a very daunting task, alternatives from the visual SLAM community were considered, which has put forward over the past couple of decades a large number of indoor and outdoor datasets with their corresponding calibrations. In particu[41] and Euroc [16] datasets were used with a combined total of three distinct camera calibrations and covering a wide range of indoor and outdoor scenarios.

The sequence generation process, shown in Fig. 7.3, starts by randomly sampling a virtual pinhole model that will serve as the new camera intrinsics to each sequence. The dataset images are then processed sequentially and transformed into the new pinhole model. Since each dataset uses its own calibration model, I first un-distort each using [35] for TumMono and [14] for Euroc. The image is then transformed to the new virtual pinhole projection $\mathbf{K}_{new}$ using:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_{old} = \mathbf{K}_{old}\mathbf{K}_{new}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_{new}. \tag{7.9}$$

To avoid unnatural looking images through this transformation, the virtual projection parameters $(f_x, f_y, c_x, c_y)_{new}$ are sampled as a function of the ground truth calibration as described in Table 7.2. Pyramidal Optical flow is then performed between the current image and the previous one,

Table 7.2: New camera intrinsics as a function of the ground truth ones. The subscripts n and o refers to new and old respectively. $\mathscr{U}$ is a uniform distribution, s_x is the scale applied to the ground truth focal length and a is the aspect ratio of the new image.

New Intrinsic Parameters	Value
$(W,H)_n$	$(W,H)_o$ / $\mathscr{U}(1.0, 2.0)$
s_x	$\mathscr{U}(0.7, 2.0)$
a	$\mathscr{U}(0.8, 1.2)$
f_x	$s_x f_{xo} \frac{W_n}{W_o}$
f_y	$a s_x f_{yo} \frac{H_n}{H_o}$
c_x	$c_{xo} \frac{W_n}{W_o} + \mathscr{U}(-50, 50)$
c_y	$c_{yo} \frac{H_n}{H_o} + \mathscr{U}(-50, 50)$

allowing for the recovery of the Fundamental matrix relating them with a RANSAC scheme. Various procedures are also set in place to ensure the quality of the Fundamental matrices. First, FAST features [140] with low cut off threshold ensures a large number of extracted features, even in texture deprived environments, at the expense of added noise. The large number of noisy features is then countered by non maximum suppression with a radius of 10 pixels. This ensures the survival of a large number of the best homogeneously distributed features in the frame. A frame then passes through a series of Quality checks and is discarded if:

1. the number of extracted features $n_f < 100$,
2. the number of feature matches $n_m < 0.65n_f$,
3. the percentage of inliers supporting the Fundamental matrix through RANSAC $n_i < 0.5n_m$,
4. the fundamental matrix constraint $(\sum_i \mathbf{x_i'Fx_i}) > 15$ pixels.

To ensure sufficient camera motion while maintaining randomness between the frames, a frame is only accepted into a sequence if it meets the aforementioned quality checks, as well as the following condition on its optical flow:

$$\|meanOptflow\|_2 > a(W+H)+b+\mathscr{U}(-10,10),$$

where W,H are the image width and height respectively, and (a,b) tuned on a per dataset basis, taking into account the camera speed and typical depths observed[1]. The end result is a dataset of 6,000 sequences (4,300 from TumMono and 1,700 from Euroc) with 14 images per sequence and their corresponding 7 relative Fundamental matrices and unique ground truth camera calibration.

7.5 Experiments and Results

The main goal of the proposed system is deployability and generalizability, therefore to quantitatively evaluate the validity of the model, it was trained using various schemes: SynthToReal (trained on synthetic, evaluated on real data), RealToReal (trained on real, evaluated on real data), and Synth-RealToReal (trained on synthetic, refined on real, evaluated on real data). Note that the Euroc dataset was not used for training whatsoever, and the testing data from the other datasets were taken from entire sequences that were not seen during training.

[1] (a,b) = (0.08, 22) and (0.06, 65) for TumMono and Euroc datasets respectively.

7.5.1 Error Distribution

The cumulative error distributions of the predicted intrinsic parameters for each model is shown in Fig. 7.4. As expected, the *SynthToReal* did not generalize as well as the other models in f_x and

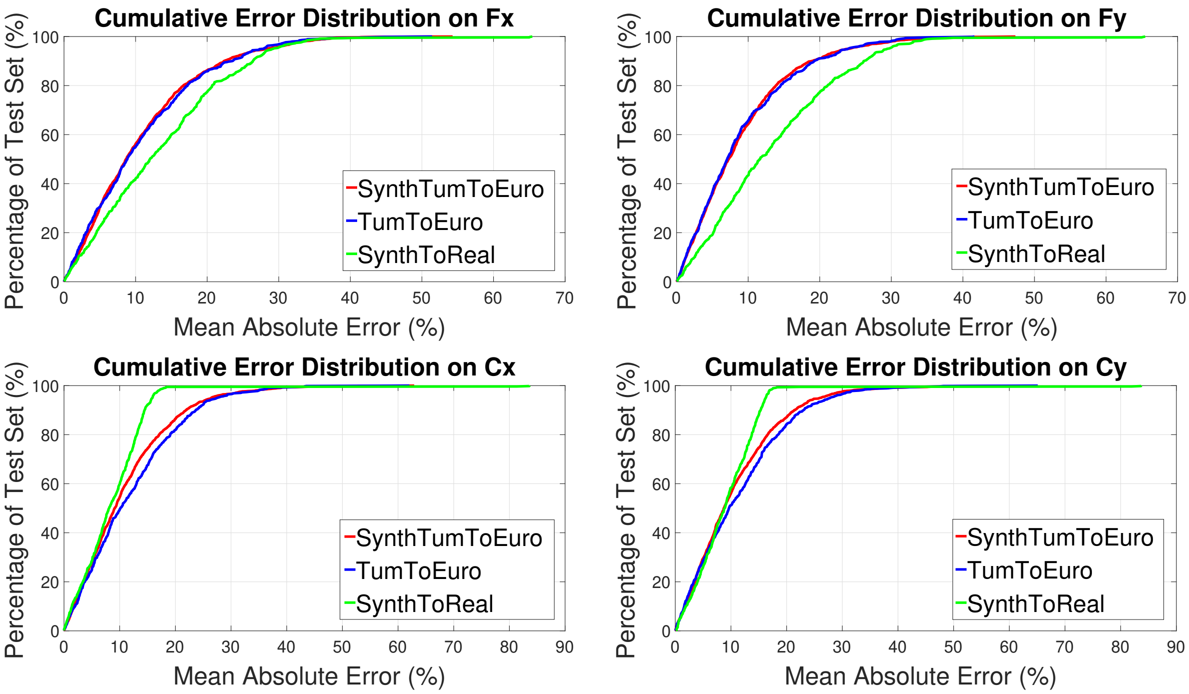


Figure 7.4: Cumulative error distributions of the estimated intrinsics using the different variations of the proposed model (with 7 Fundamental matrices as input). SynthToReal (trained on synthetic, evaluated on real data), RealToReal (trained on real, evaluated on real data), and Synth-RealToReal (trained on synthetic, refined on real, evaluated on real data)

f_y, however the surprise was that it performed better on estimating the principal point. The result shows the high sensitivity of the principal point to any noise. Since the SynthToReal model was trained on synthetic noise-free data, the network was able to learn a proper correlation between the input Fundamentals and the principal point. However, once that same model is refined with real (noisy) data, the estimated correlation is corrupted, thereby decreasing the performance. It is this high sensitivity to any noise in the inputs that makes the principal point difficult to predict and as such was widely ignored in recent works on the topic. Despite the slightly reduced performance in estimating the principal point, the reported mean absolute error of the Synth-RealToReal model remains around 10% across all intrinsic parameters, and as such was adopted as the final proposed model.

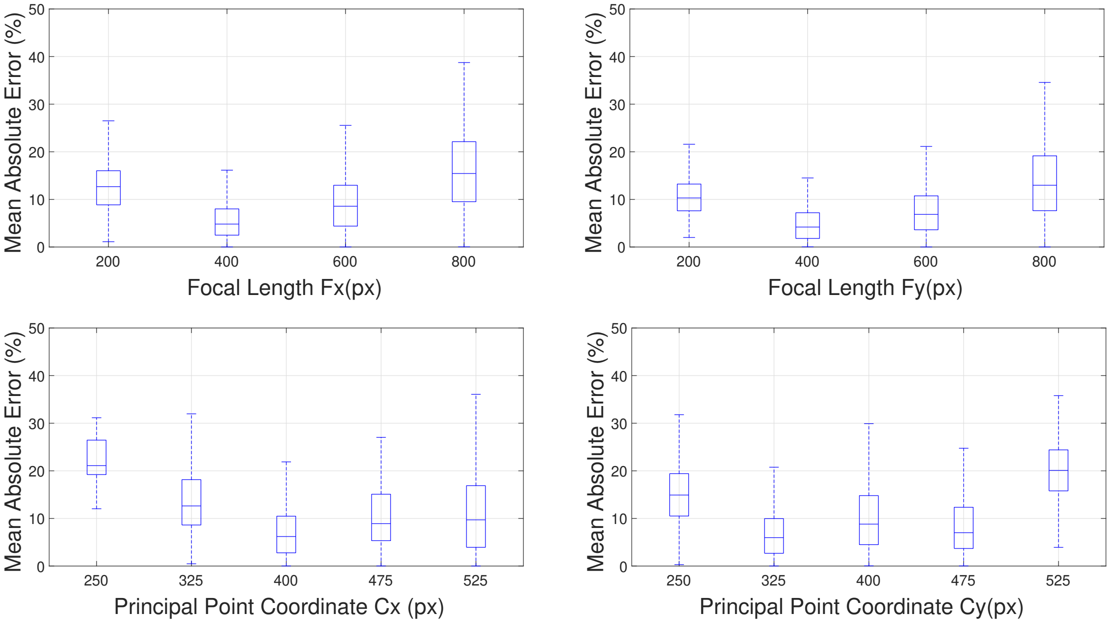


Figure 7.5: Box-plot of the error distributions of the proposed model against the ground truth f_x, f_y, c_x, c_y

The error achieved with the proposed model against the ground intrinsic parameters is reported in Fig.7.5, showing its validity against a wide range of values.

7.5.2 Comparison to Literature

Unfortunately, state of the art approaches do not predict the full set of intrinsic parameters; they all report on the focal length f with unit aspect ratio, making it difficult to perform a direct comparison to the results obtained on f_x, f_y, c_x, c_y.

The closest in spirit work presented in [104] assumes unit aspect ratio, which is not the case for most cameras. However, for the sake of comparison, their model was replicated exactly as described in their paper and its performance was assessed; it achieved an error of 13.4% on the focal length f (unit aspect ratio), 11.7% and 12.9% on the camera's tilt and roll angles respectively, and 14.7% on the distortion parameter. The obtained results agree with the accuracy results presented in their paper, thereby validating the fidelity of the replicated implementation. Their architecture was then modified to output the full camera intrinsic calibration (f_x, f_y, c_x, c_y), which is hereafter referred to as the *Single Net model* (it takes single images as input and outputs its camera calibration).

State of art in traditional methods were also evaluated, that is solving Kruppa's equations as described in [105], with the modifications proposed in Section 7.4.1. This solution is referred to as the *traditional* self-calibration approach. The results of all three experiments are shown in Table 7.3. The proposed model performs the best, achieving errors as low as 9%; it is also noteworthy to mention the contrast between the proposed model and the other two when estimating the principal point, where the former achieves errors three times lower on the principal point. These results can be attributed to two factors: (1) the quality of the supervisory signal on each parameter from the dataset generation step, and (2) an architecture dedicated to exploiting the set of Fundamental matrices instead of implicitly inferring the required knowledge from images. The second reason also has important ramifications on the size of the model, which requires a fraction of the parameters employed in [104], and as such is faster to compute than the traditional self-calibration approach. In fact, the proposed model consists of 17,000 parameters, in contrast to the DenseNet model adopted in [104] which includes 26 million parameters.

Table 7.3: Percentage Mean absolute error in intrinsics. Traditional refers to kruppa's solution as described in sec. 7.4.1 and Single Net refers to the architecture described in sec. 7.5.2 that predicts the intrinsics from a single image.

	Ours (%)	Traditional (%)	Single Net (%)
Fx	**10.704**	12.93	46.61
Fy	**8.992**	9.30	41.62
Cx	**10.856**	31.44	21.94
Cy	**10.568**	30.86	30.16

7.5.3 Effect of input size

As mentioned earlier, a minimum of three images, or equivalently two Fundamental matrices, are required to find a unique camera calibration using either the proposed approach or the traditional self-calibration method (solving Kruppa's equations). It is therefore interesting to examine the impact of the number of inputs on the resulting accuracy. To that end 6 different models of the proposed architecture were trained, starting at two Fundamental matrices for the first model, and increasing one Fundamental matrix for each subsequent model to reach seven in the final model. The experiment is also repeated for the traditional self-calibration approach While the traditional approach registered an improvement of about 4% going from 2 to 7 Fundamental matrices, the proposed architecture's accuracy improved by only 1%, suggesting it sufficient to use two Fundamental matrices in the proposed method.

Table 7.4: Mean absolute error of the various developed models. The blue color correspond to the results of the final adopted model.

	Mean Absolute Error in %			
	Fx	Fy	Cx	Cy
Ours trained on TumMono and Tested on TumMono	10.80	8.99	11.88	11.45
Ours trained on TumMono and Tested on EuroC	14.84	14.27	16.16	15.93
Ours trained on TumMono and Tested on Kitti	31.79	28.30	26.76	28.78
Ours trained on Synthetic and Tested on Synthetic	11.10	11.93	7.80	8.66
Ours trained on Synthetic and Tested on TumMono	13.40	13.36	8.62	9.14
Ours trained on Synthetic and Tested on EuroC	14.84	14.27	16.16	15.93
Ours trained on Synthetic and Tested on Kitti	27.10	25.97	36.86	17.76
Ours trained on Synthetic Refined on TumMono and Tested on TumMono	10.70	8.99	10.85	10.56
Ours trained on Synthetic Refined on TumMono and Tested on EuroC	15.20	14.64	15.67	15.62
Ours trained on Synthetic Refined on TumMono and Tested on Kitti	32.57	29.05	26.58	30.63
Single Net Trained on TumMono and Tested on TumMono	46.61	41.62	21.94	30.16
Traditional Self-Calibration Tested on EuroC	14.91	11.77	26.96	33.12
Traditional Self-Calibration Tested on TumMono	12.93	9.30	31.44	30.86

7.5.4 Limitations

It is a well known fact that Kruppa's equation can lead to ambiguous results under certain critical motion sequences [153]. These degenerate cases are intrinsic to the self-calibration problem and apply uniformly to all absolute-conic-based algorithms [155]. Examples of degenerate motion sequences include purely translating cameras, in-plane motion, *etc*.

Unfortunately, some of these degenerate cases are common in application: for example, a car-mounted camera is almost always undergoing a pure translation motion or an in-plane motion, thereby making self-calibration in such cases degenerate. One dataset plagued with degenerate motions is the Kitti dataset [58], and was therefore used to evaluate the performance of the proposed model to degenerate sequences. The results are reported in Table 7.4, revealing a significant decrease in performance over every system/model that was tested and as such remains a limitation to this work and to all absolute-conic-based self-calibration algorithms. Note that absolute-quadric-based solutions do not suffer from such degeneracies [155]; however, they require the solution to the entire Structure from Motion problem before recovering the camera calibration [80].

7.6 Conclusion

We have presented a deep intrinsic camera calibration model from fundamental matrices; to the best of our knowledge this is the first work that exploits the traditional multi-view constraints in a deep learning approach to predict the focal lengths (f_x, f_y) and the principal point (c_x, c_y), while achieving state of the art performance on each parameter, with a mean absolute error of 10% across; all of this with a small fraction of the number of parameters involved in the state of the art systems. This was possible due to the generated synthetic and real datasets, tailored to the intrinsic calibration task. Finally we perform a thorough experimental procedure which validated the model's accuracy and generalizability. We believe this system has merit to the research community as it can be deployed for the camera self-calibration either from fundamental matrices or from images by coupling it with systems that recover the fundamental matrices from image sequences like the work presented in [132] and [136].

Chapter 8

Conclusion and future directions

VSLAM is an essential technology at the core of various industries; from autonomous robotics, to self driving cars, to multimedia use such as augmented reality, the reliability and robustness of *VSLAM* is vital to its expanding applications across the various domains. While viable solutions to *VSLAM* has been around for a couple of decades now, there is still ample room for improvements across its different components.

Throughout this thesis, I have tackled the potential and the various ways Direct and Indirect features can interact with each other to concurrently leverage their advantages while diminishing their shortcomings in a unified approach. I have developed various systems that, at a small computational cost, drastically improves the accuracy of *VSLAM* while maintaining the robustness of direct methods to textureless regions and the resilience of indirect methods to large baseline motions. A byproduct of this unified formulation is a mean to control the density of the reconstructed maps, allowing the use of indirect features for global map localization and reuse while locally maintaining the reconstruction density from the direct methods.

This however is only one step forward towards a unified hybrid formulation for *VSLAM*, where various sources of information, not just direct and indirect features, interact with each other across multiple layers of representations to expand current *VSLAM* capabilities. An example of such extra source of information is semantic labels that can, thanks to recent development in deep learning, provide pixel-wise semantic labels in real-time. Therefore, semantic segmentation can be considered an independent source of information that can interact with semi-dense direct features, and sparse indirect features in various ways and are part of the future directions of my work.

8.1 Sparse metric maps and semantic labels interactions

In a feature-based framework, the 2D features are formed by sparsely sampling the 3D world surface into 2D geometric primitives, significantly abstracting the properties of the surface structure itself; after which the geometric primitives are treated as conditionally independent entities. Integrating semantic labels in such formulation is particularly challenging as the geometric primitives are usually defined over small areas in the image domain (*e.g.* keypoints are defined at a single pixel), whereas semantic labels are associated with objects typically spanning a large number of pixels. Nevertheless, semantic labels can be integrated in the sparse map in many ways: Semantic labels can be used to identify potentially dynamic objects in the scene, allowing VSLAM to properly handle keypints from those image regions that violate the static scene assumption; furthermore the set of features associated with such regions can be grouped together to form the notion of dynamic objects, which can be tracked separately from the traditional static map, allowing VSLAM to perform path collision detection and dynamic object tracking within the same framework. Also, descriptors of 2D features, associated with semantic labels of rigid objects in the static map, can be grouped together to form a global based descriptor for observed objects in the scene. Such formulation allows for an object-based VSLAM framework that preserves the rigidity of the reconstructed objects, as opposed to the current conditionally independent 3D features reconstructions.

8.2 Semi-dense metric map and semantic labels interactions

Semi-dense metric maps are highly susceptible to outliers from dynamic objects and occlusions; semantic labels can significantly reduce the outlier's spurious effects by flagging out potential dynamic objects and by providing consistency checks over occluded pixels, in different images, that belong to different semantic labels. Furthermore, semantic labels can benefit the semi-dense regularization process by tightly integrating the labels into the regularization, such that local points belonging to the same labels with discontinuous 3D reconstructions are penalized [98]. Also, semantic labels can help increase the density of semi-dense maps through planar detection (*e.g.* a 3D surface of a road is a continuously smooth surface, and support information *e.g.* a building has to be supported by a ground at its bottom.

On the other hand, semantic labeling can benefit from integrating structural and temporal measurements, provided from the semi-dense map, to provide segmentation priors on subsequent images. Furthermore, the semi-dense map can provide geometric consistency checks on the 2D segmentation procedure, reducing outliers and increasing prediction accuracy.

8.3 Camera Calibration

While most cameras are designed for human consumption, their applicability to *VSLAM* requires special calibration procedures to recover both the geometric and photometric calibrations.

8.3.1 Photometric Calibration

While photometric calibration is not essential for *VSLAM*, its knowledge can significantly improve the accuracy of the components that rely on direct features.
For example, during regular operation, a camera is allowed to adjust its exposure to accommodate varying lighting conditions. However, for the sake of brightness constancy, auto exposure is typically turned off in a *VSLAM* session so that the brightness constancy assumption is not violated. In an effort to increase the robustness of Direct VSLAM to varying lighting conditions, [43] proposed an offline photometric calibration process to estimate the camera response function and vignetting map. If a photometric calibration for a camera is then available along with the exposure times per frame, they can be used to map the image intensities to the scene radiance, which is independent of the camera response, thereby exploiting auto-exposure to maintain the brightness constancy assumption. However, obtaining a photometric calibration and exposure times is a daunting offline task. Nevertheless, [8] showed that if features can be reliably matched between photometrically distorted images in a video sequence, they can be used to recover both the photometric calibration and the exposure time per frame in real-time.
Since indirect features are mostly invariant (to some degree) to illumination variations, they can be used to recover the exposure time, gamma response and camera vignetting. The required feature matches are inherently available in *UFVO* through its Indirect features that use ORB descriptors to establish correspondences across frames. Therefore, a joint formulation can be exploited to perform the photometric calibration online.

8.3.2 Geometric Calibration

An accurate geometric calibration is essential for proper *VSLAM* operation. In chapter 7 I have presented a deep learning approach that exploits the sequential nature of images taken from a video to recover the camera intrinsics. The approach is inherently biased to solve kruppa's equations since the input to the model is a set of pairwise independent fundamental matrices. While the obtained results outperformed state-of-the-art self-calibration methods, they are still not enough for deployability in a *VSLAM* session. The main reason for this shortcoming is due to the underlying reliance of kruppa's equations on the absolute conic, which under certain motions can be degenerate (sec. 7.5.4); unfortunately these types of motion are often encountered in *VSLAM* applications and addressing these limitations remain part of my future work on the topic.

References

[1] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008*, pages 102–115, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[2] R. Ait-Jellal and A. Zell. Outdoor obstacle avoidance based on hybrid stereo visual SLAM for an autonomous quadrotor MAV. In *IEEE 8th European Conference on Mobile Robots (ECMR)*, 2017.

[3] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517, June 2012.

[4] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[5] Paul Beardsley and David Murray. Camera calibration using vanishing points. In David Hogg and Roger Boyle, editors, *BMVC92*, pages 416–425, London, 1992. Springer London.

[6] P. R. Beaudet. Rotationally invariant image operators. In *International Conference on Pattern Recognition*, 1978.

[7] S Benhimane and E Malis. Homography-based 2D Visual Tracking and Servoing. *International Journal of Robotics Research*, 26(7):661–676, July 2007.

[8] P. Bergmann, R. Wang, and D. Cremers. Online photometric calibration of auto exposure video for realtime visual odometry and slam. *IEEE Robotics and Automation Letters (RA-L)*, 3:627–634, April 2018.

[9] S. R. Bista, P. R. Giordano, and F. Chaumette. Appearance-based indoor navigation by ibvs using line segments. *IEEE Robotics and Automation Letters*, 1(1):423–430, Jan 2016.

[10] Jaime Boal, Álvaro Sánchez-Miralles, and Álvaro Arranz. Topological simultaneous localization and mapping: a survey. *Robotica*, 32(5):803–821, Aug 2014.

[11] Oleksandr Bogdan, Viktor Eckstein, Francois Rameau, and Jean-Charles Bazin. Deepcalib: a deep learning approach for automatic intrinsic calibration of wide field-of-view cameras. In *Proceedings of the 15th ACM SIGGRAPH European Conference on Visual Media Production*, pages 1–10, 2018.

[12] G. Bourmaud and R. Megret. Robust large scale monocular visual slam. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1638–1647, June 2015.

[13] Donald Bourque. *CUDA-Accelerated ORB-SLAM for UAVs*. Worcester Polytechnic Institute, 2017.

[14] Duane C. Brown. Close-range camera calibration. *PHOTOGRAMMETRIC ENGINEERING*, 37(8):855–866, 1971.

[15] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 2016.

[16] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 2016.

[17] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. Brief: Computing a local binary descriptor very fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298, July 2012.

[18] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual-inertial and multimap slam. *arXiv preprint arXiv:2007.11898*, 2020.

[19] Bruno Caprile and Vincent Torre. Using vanishing points for camera calibration. *International journal of computer vision*, 4(2):127–139, 1990.

[20] Koray Celik and Arun K. Somani. Monocular vision slam for indoor aerial vehicles. *Journal of Electrical and Computer Engineering*, 2013:15, 2013.

[21] Huan Chang and Fuan Tsai. Vanishing point extraction and refinement for robust camera calibration. *Sensors*, 18(1), 2018.

[22] Baifan Chen, Dian Yuan, Chunfa Liu, and Qian Wu. Loop closure detection based on multi-scale deep feature fusion. *Applied Sciences*, 9(6):1120, 2019.

[23] Kevin Christensen and Martial Hebert. Edge-direct visual odometry. *CoRR*, abs/1906.04838, 2019.

[24] Roberto Cipolla, Tom Drummond, and Duncan P Robertson. Camera calibration from vanishing points in image of architectural scenes. In *BMVC*, volume 99, pages 382–391, 1999.

[25] J. Civera, A.J. Davison, and J. Montiel. Inverse Depth Parametrization for Monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945, October 2008.

[26] Javier Civera, Andrew 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, pages 412–419, Berlin, Heidelberg, 2007. Springer-Verlag.

[27] Laura A. Clemente, Andrew J. Davison, Ian D. Reid, José Neira, and Juan D. Tardós. Mapping large loops with a single hand-held camera. In *Robotics: Science and Systems*, 2007.

[28] A. Concha and J. Civera. Using superpixels in monocular slam. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 365–372, May 2014.

[29] A. Concha and J. Civera. Dpptam: Dense piecewise planar tracking and mapping from a monocular sequence. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 5686–5693, Sept 2015.

[30] Mark Cummins and Paul Newman. Appearance-only slam at large scale with fab-map 2.0. *Int. J. Rob. Res.*, 30(9):1100–1123, August 2011.

[31] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893 vol. 1, June 2005.

[32] A. J. Davison, Y. Cid, and N. N. Kita. Real-time 3D SLAM with wide-angle vision. In *Proc. IFAC Symposium on Intelligent Autonomous Vehicles, Lisbon*, July 2004.

[33] A.J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Ninth IEEE International Conference on Computer Vision*, pages 1403–1410 vol.2, 2003.

[34] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: real-time single camera SLAM. *Pattern Analysis and Machine Intelligence (PAMI), IEEE Transactions on*, 29(6):1052–67, June 2007.

[35] Frédéric Devernay and Olivier Faugeras. Straight lines have to be straight: automatic calibration and removal of distortion from scenes of structured enviroments. *Machine Vision and Applications*, 13(1):14–24, 2001. Erratum available at http://devernay.free.fr/publis/distcalib-erratum.html.

[36] Zilong Dong, Guofeng Zhang, Jiaya Jia, and Hujun Bao. Efficient keyframe-based real-time camera tracking. *Computer Vision and Image Understanding*, 118:97 – 110, 2014.

[37] Renaud Dubé, Daniel Dugas, Elena Stumm, Juan Nieto, Roland Siegwart, and Cesar Cadena. Segmatch: Segment based loop-closure for 3d point clouds. *CoRR*, abs/1609.07720, 2016.

[38] E. Eade and Tom Drummond. Scalable monocular slam. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 469–476, June 2006.

[39] E. Eade and Tom Drummond. Monocular slam as a graph of coalesced observations. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, Oct 2007.

[40] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2017.

[41] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2017.

[42] J. Engel, J. Stuckler, and D. Cremers. Large-scale direct slam with stereo cameras. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1935–1942, Sept 2015.

[43] J. Engel, V. Usenko, and D. Cremers. A photometrically calibrated benchmark for monocular visual odometry. In *arXiv:1607.02555*, July 2016.

[44] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014 SE - 54*, volume 8690 of *Lecture Notes in Computer Science*, pages 834–849. Springer International Publishing, 2014.

[45] Jakob Engel, Vladyslav C. Usenko, and Daniel Cremers. A photometrically calibrated benchmark for monocular visual odometry. *CoRR*, abs/1607.02555, 2016.

[46] Baofu Fang and Zhiqiang Zhan. A visual slam method based on point-line fusion in weak-matching scene. *International Journal of Advanced Robotic Systems*, 17(2):1729881420904193, 2020.

[47] Olivier D Faugeras, Q-T Luong, and Stephen J Maybank. Camera self-calibration: Theory and experiments. In *European conference on computer vision*, pages 321–334. Springer, 1992.

[48] Eduardo Fernández-Moral, Vicente Arévalo, and Javier González-Jiménez. Hybrid Metric-topological Mapping for Large Scale Monocular SLAM . In Jean-Louis Ferrier, Oleg Gusikhin, Kurosh Madani, and Jurek Sasiadek, editors, *Informatics in Control, Automation and Robotics*, pages 217–232. Springer International Publishing, 2015.

[49] T. Fetzer, G. Reis, and D. Strieker. Stable intrinsic auto-calibration from fundamental matrices of devices with uncorrelated camera parameters. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 221–230, 2020.

[50] N. Fioraio and L. Di Stefano. Joint detection, tracking and mapping by semantic bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1538–1545, June 2013.

[51] Andrew W Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001.

[52] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo : Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), IEEE International Conference on*, 2014.

[53] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, April 2017.

[54] D Galvez-López and J D Tardos. Bags of Binary Words for Fast Place Recognition in Image Sequences. *Robotics, IEEE Transactions on*, 28(5):1188–1197, oct 2012.

[55] Dorian Gálvez-López, Marta Salas, Juan D. Tardós, and J. M. M. Montiel. Real-time monocular object SLAM. *CoRR*, abs/1504.02398, 2015.

[56] X. Gao, R. Wang, N. Demmel, and D. Cremers. Ldso: Direct sparse odometry with loop closure. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2198–2204, Oct 2018.

[57] Emilio Garcia-Fidalgo and Alberto Ortiz. Vision-based topological mapping and localization methods: A survey. *Robotics and Autonomous Systems*, 64:1 – 20, 2015.

[58] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[59] Georges Younes, Daniel Asmar, and John Zelek. Robustness of vo systems to subsampled motions. In International Conference on Robotics and Automation (ICRA), editors, *Workshop on Dataset Generation and Benchmarking of SLAM Algorithms for Robotics and VR/AR*, 2019.

[60] Arren Glover, William Maddern, Michael Warren, Stephanie Reid, Michael Milford, and Gordon Wyeth. OpenFABMAP: An open source toolbox for appearance-based loop closure detection. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4730–4735. IEEE, May 2012.

[61] R. Gomez-Ojeda, J. Briales, and J. Gonzalez-Jimenez. Pl-svo: Semi-direct monocular visual odometry by combining points and line segments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4211–4216, 2016.

[62] O.G. Grasa, E. Bernal, S. Casado, I. Gil, and J.M.M. Montiel. Visual slam for handheld monocular endoscope. *Medical Imaging, IEEE Transactions on*, 33(1):135–146, Jan 2014.

[63] W. N. Greene, K. Ok, P. Lommel, and N. Roy. Multi-level mapping: Real-time dense monocular slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 833–840, May 2016.

[64] W. N. Greene and N. Roy. Flame: Fast lightweight mesh estimation using variational smoothing on delaunay graphs. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4696–4704, 2017.

[65] Xufeng Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3279–3286, June 2015.

[66] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[67] Richard I Hartley. In Defense of the Eight-Point Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(6):580–593, June 1997.

[68] Richard I. Hartley and Peter Sturm. Triangulation. *Comput. Vis. Image Underst.*, 68(2):146–157, November 1997.

[69] J Hartmann, J H Klussendorff, and E Maehle. A comparison of feature descriptors for visual SLAM. In *Mobile Robots (ECMR), 2013 European Conference on*, pages 56–61, sep 2013.

[70] B.W. He and Y.F. Li. Camera calibration from vanishing points in a vision system. *Optics & Laser Technology*, 40(3):555 – 561, 2008.

[71] Daniel Herrera, Juho Kannala, Kari Pulli, and Janne Heikkila. DT-SLAM: Deferred Triangulation for Robust SLAM. In *3D Vision, 2nd International Conference on*, volume 1, pages 609–616. IEEE, December 2014.

[72] Antti Hietanen, Jukka Lankinen, Joni-Kristian Kämäräinen, Anders Glent Buch, and Norbert Krüger. A comparison of feature detectors and descriptors for object class matching. *Neurocomputing*, 2016.

[73] 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*, pages 1–6, June 2009.

[74] Yannick Hold-Geoffroy, Kalyan Sunkavalli, Jonathan Eisenmann, Matthew Fisher, Emiliano Gambaretto, Sunil Hadap, and Jean-François Lalonde. A perceptual measure for deep single image camera calibration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2354–2363, 2018.

[75] Steven A. Holmes, Georg Klein, and David W. Murray. A square root unscented kalman filter for visual monoslam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(7):1251–1263, 2008.

[76] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[77] Baichuan Huang, Jun Zhao, and Jingbin Liu. A survey of simultaneous localization and mapping. *CoRR*, abs/1909.05214, 2019.

[78] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[79] Du Q. Huynh. Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, Oct 2009.

[80] E. Ito and T. Okatani. Self-calibration-based approach to critical motion sequences of rolling-shutter structure from motion. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4512–4520, 2017.

[81] WooYeon Jeong and Kyoung Mu Lee. Cv-slam: a new ceiling vision-based slam technique. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3195–3200, Aug 2005.

[82] James L. Crowley Jérôme Martin. Experimental Comparison of Correlation Techniques. In *IAS-4, International Conference on Intelligent Autonomous Systems*, 1995.

[83] Georg Klein and David Murray. Parallel Tracking and Mapping for Small AR Workspaces. *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10, November 2007.

[84] Georg Klein and David Murray. Improving the Agility of Keyframe-Based {SLAM}. In *Proc. 10th European Conference on Computer Vision (ECCV)*, pages 802–815, Marseille, October 2008.

[85] Matthew Klingensmith, Ivan Dryanovski, Siddhartha S Srinivasa, and Jizhong Xiao. Chisel: Real time large scale 3d reconstruction onboard a mobile device using spatially hashed signed distance fields. In *Robotics: science and systems*, volume 4. Citeseer, 2015.

[86] Kurt Konolige. Sparse sparse bundle adjustment. In *Proceedings of the British Machine Vision Conference*, pages 102.1–102.11. BMVA Press, 2010. doi:10.5244/C.24.102.

[87] Scott Krig. *Interest Point Detector and Feature Descriptor Survey*, pages 217–282. Apress, Berkeley, CA, 2014.

[88] N. Krombach, D. Droeschel, and S. Behnke. Combining feature-based and direct methods for semi-dense real-time stereo visual odometry. In *International Conference on Intelligent Autonomous Systems*, pages 855–868. Springer, 2016.

[89] Rainer Kummerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 3607–3613. IEEE, May 2011.

[90] Abhijit Kundu, Yin Li, Frank Dellaert, Fuxin Li, and JamesM. Rehg. Joint semantic segmentation and 3d reconstruction from monocular video. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, volume 8694 of *Lecture Notes in Computer Science*, pages 703–718. Springer International Publishing, 2014.

[91] Junghyun Kwon and Kyoung Mu Lee. Monocular slam with locally planar landmarks via geometric rao-blackwellized particle filtering on lie groups. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1522–1529, June 2010.

[92] S. H. Lee and J. Civera. Loosely-coupled semi-direct monocular slam. *IEEE Robotics and Automation Letters*, 4(2):399–406, April 2019.

[93] Sang Jun Lee and Sung Soo Hwang. Elaborate monocular point and line slam with robust initialization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[94] Seok-Han Lee. Real-time camera tracking using a particle filter combined with unscented kalman filters. *Journal of Electronic Imaging*, 23(1):013029, 2014.

[95] T. Lemaire and S. Lacroix. Monocular-vision based slam using line segments. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2791–2796, April 2007.

[96] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal of Computer Vision*, 81(2):155–166, 2009.

[97] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555, Nov 2011.

[98] Xuanpeng Li and Rachid Belaroussi. Semi-dense 3d semantic mapping from monocular SLAM. *Arxiv*, abs/1611.04144, 2016.

[99] Hyon Lim, Jongwoo 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*, pages 1532–1539, May 2014.

[100] J. Lim, J. M. Frahm, and M. Pollefeys. Online environment mapping. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3489–3496, June 2011.

[101] Tony Lindeberg. Feature detection with automatic scale selection. *Int. J. Comput. Vision*, 30(2):79–116, November 1998.

[102] Haomin Liu, Guofeng Zhang, and Hujun Bao. Robust keyframe-based monocular slam for augmented reality. *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2016.

[103] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Letters to Nature*, 293(5828):133–135, 09 1981.

[104] Manuel Lopez, Roger Mari, Pau Gargallo, Yubin Kuang, Javier Gonzalez-Jimenez, and Gloria Haro. Deep single image camera calibration with radial distortion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11817–11825, 2019.

[105] Manolis I.A. Lourakis and Rachid Deriche. Camera Self-Calibration Using the Singular Value Decomposition of the Fundamental Matrix: From Point Correspondences to 3D Measurements. Technical Report RR-3748, INRIA, August 1999.

[106] Manolis I.A. Lourakis and Rachid Deriche. Camera Self-Calibration Using the Kruppa Equations and the SVD of the Fundamental Matrix: The Case of Varying Intrinsic Parameters. Research Report RR-3911, INRIA, 2000.

[107] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[108] D.G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision (ICCV), the seventh IEEE*, volume 2, pages 1150–1157 vol.2. IEEE, 1999.

[109] Bruce D Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[110] I. Mahon, S. B. Williams, O. Pizarro, and M. Johnson-Roberson. Efficient view-based slam using visual loop closures. *IEEE Transactions on Robotics*, 24(5):1002–1014, Oct 2008.

[111] Elmar Mair, Gregory D Hager, Darius Burschka, Michael Suppa, and Gerhard 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.

[112] R. Timothy Marler and Jasbir S. Arora. The weighted sum method for multi-objective optimization: new insights. *Structural and Multidisciplinary Optimization*, 41(6):853–862, 2010.

[113] Jose Martinez-Carranza and Andrew Calway. Unifying planar and point mapping in monocular slam. In *Proceedings of the British Machine Vision Conference*, pages 43.1–43.11. BMVA Press, 2010. doi:10.5244/C.24.43.

[114] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. BMVC*, pages 36.1–36.10, 2002. doi:10.5244/C.16.36.

[115] J. Meltzer, Rakesh Gupta, Ming-Hsuan 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*, volume 2, pages 1550–1555 vol.2, Sept 2004.

[116] Azam Rafique Memon, Hesheng Wang, and Abid Hussain. Loop closure detection using supervised and unsupervised deep neural networks for monocular slam systems. *Robotics and Autonomous Systems*, 126:103470, 2020.

[117] B. Micusik and H. Wildenauer. Descriptor free visual indoor localization with line segments. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3165–3173, June 2015.

[118] Davide Migliore, Roberto Rigamonti, Daniele Marzorati, Matteo Matteucci, and Domenico G. Sorrenti. Use a single camera for simultaneous localization and mapping with mobile object tracking in dynamic environments, 2009.

[119] Theo Moons, Luc Van Gool, and Maarten Vergauwen. 3d reconstruction from multiple images: Part 1 - principles. *Foundations and Trends in Computer Graphics and Vision*, 4:287–404, 01 2009.

[120] Pierre Moreels and Pietro Perona. Evaluation of features detectors and descriptors based on 3d objects. *Int. J. Comput. Vision*, 73(3):263–284, July 2007.

[121] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real time localization and 3d reconstruction. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 363–370, June 2006.

[122] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *In VISAPP International Conference on Computer Vision Theory and Applications*, pages 331–340, 2009.

[123] R. Mur-Artal and J. D. Tardós. Fast relocalisation and loop closing in keyframe-based slam. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 846–853, May 2014.

[124] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, PP(99):1–17, 2015.

[125] Richard A Newcombe and Andrew J Davison. Live dense reconstruction with a single moving camera. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1498–1505. IEEE, 2010.

[126] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. Dtam: Dense tracking and mapping in real-time. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2320–2327, Washington, DC, USA, 2011. IEEE Computer Society.

[127] Sudeep Pillai and John J. Leonard. Monocular SLAM supported object recognition. *CoRR*, abs/1506.01732, 2015.

[128] P. Pinies and J.D. Tardos. Large-scale slam building conditionally independent local maps: Application to monocular vision. *Robotics, IEEE Transactions on*, 24(5):1094–1106, Oct 2008.

[129] C. Pirchheim and G. Reitmayr. Homography-based planar mapping and tracking for mobile phones. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 27–36, Oct 2011.

[130] C. Pirchheim, D. Schmalstieg, and G. Reitmayr. Handling pure camera rotation in keyframe-based slam. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, pages 229–238, Oct 2013.

[131] Katrin Pirker, Matthias Rüther, and Horst Bischof. CD SLAM - Continuous localization and mapping in a dynamic world. In *IEEE International Conference on Intelligent Robots Systems (IROS)*, pages 3990–3997. IEEE, 2011.

[132] Omid Poursaeed, Guandao Yang, Aditya Prakash, Qiuren Fang, Hanqing Jiang, Bharath Hariharan, and Serge Belongie. Deep fundamental matrix estimation without correspondences. In *European Conference on Computer Vision*, pages 485–497. Springer, 2018.

[133] 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*, pages 3289–3296, May 2011.

[134] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer. Pl-slam: Real-time monocular visual slam with points and lines. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4503–4508, 2017.

[135] Mark Pupilli and Andrew Calway. Real-time camera tracking using a particle filter. In *In Proc. British Machine Vision Conference*, pages 519–528, 2005.

[136] René Ranftl and Vladlen Koltun. Deep fundamental matrix estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 284–299, 2018.

[137] Ives Rey-Otero, Mauricio Delbracio, and Jean-Michel Morel. Comparing feature detectors: A bias in the repeatability criteria, and how to correct it. *CoRR*, abs/1409.2465, 2014.

[138] Josiane Rodrigues, Marco Cristo, and Juan G Colonna. Deep hashing for multi-label image retrieval: a survey. *Artificial Intelligence Review*, pages 1–47, 2020.

[139] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020.

[140] Edward Rosten and Tom Drummond. Machine Learning for High-speed Corner Detection. In *9th European Conference on Computer Vision - Volume Part I, Proceedings of the*, ECCV'06, pages 430–443, Berlin, Heidelberg, 2006. Springer-Verlag.

[141] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *International Conference on Computer Vision (ICCV)*, pages 2564–2571, November 2011.

[142] Daniel Santana-Cedrés, Luis Gomez, Miguel Alemán-Flores, Agustín Salgado, Julio Esclarín, Luis Mazorra, and Luis Alvarez. Automatic correction of perspective and optical distortions. *Computer Vision and Image Understanding*, 161:1 – 10, 2017.

[143] S. Savarese, Y-W. Chao, M. Bagra, and S. Y. Bao. Semantic structure from motion with points, regions, and objects. *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 00(undefined):2703–2710, 2012.

[144] David Schubert, Nikolaus Demmel, Vladyslav Usenko, Jorg Stuckler, and Daniel Cremers. Direct sparse odometry with rolling shutter. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[145] Xuelun Shen, Cheng Wang, Xin Li, Zenglei Yu, Jonathan Li, Chenglu Wen, Ming Cheng, and Zijian He. Rf-net: An end-to-end image matching network based on receptive field. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8132–8140, 2019.

[146] 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*, pages 593–600, June 1994.

[147] G. Silveira, E. Malis, and Patrick Rives. An efficient direct approach to visual slam. *Robotics, IEEE Transactions on*, 24(5):969–979, Oct 2008.

[148] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative Learning of Deep Convolutional Feature Point Descriptors. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[149] Paul Smith, Ian Reid, and Andrew Davison. Real-time monocular SLAM with straight lines. In *Proc. British Machine Vision Conference*, pages 17–26, Edinburgh, 2006.

[150] Hauke Strasdat, Andrew J Davison, J M M Montiel, and Kurt Konolige. Double Window Optimisation for Constant Time Visual SLAM. In *International Conference on Computer Vision, Proceedings of the*, ICCV '11, pages 2352–2359, Washington, DC, USA, 2011. IEEE Computer Society.

[151] Hauke Strasdat, J M M Montiel, and Andrew J Davison. Real-time monocular SLAM: Why filter? In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 2657–2664, May 2010.

[152] Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison. Scale drift-aware large scale monocular slam. In *In Proceedings of Robotics: Science and Systems*, 2010.

[153] P. Sturm. Critical motion sequences for monocular self-calibration and uncalibrated euclidean reconstruction. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1100–1105, 1997.

[154] Wei Tan, Haomin Liu, Zilong Dong, Guofeng Zhang, and Hujun Bao. Robust monocular SLAM in dynamic environments. *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 209–218, 2013.

[155] B. Triggs. Autocalibration and the absolute quadric. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 609–614, 1997.

[156] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment — a modern synthesis. In Bill Triggs, Andrew Zisserman, and Richard Szeliski, editors, *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings*, pages 298–372. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[157] R. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344, August 1987.

[158] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(4):376–380, April 1991.

[159] Alexander Vakhitov, Jan Funke, and Francesc Moreno-Noguer. Accurate and linear time pose estimation from points and lines. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII*, pages 583–599. Springer International Publishing, Cham, 2016.

[160] Y. Verdie, Kwang Moo Yi, P. Fua, and V. Lepetit. Tilde: A temporally invariant learned detector. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5279–5288, June 2015.

[161] Bart Verhagen, Radu Timofte, and Luc Van Gool. Scale-invariant line descriptors for wide baseline matching. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 493–500. IEEE, March 2014.

[162] Guanghui Wang, Q.M. Jonathan Wu, and Wei Zhang. Kruppa equation based camera calibration from homography induced by remote plane. *Pattern Recognition Letters*, 29(16):2137 – 2144, 2008.

[163] Runzhi Wang, Kaichang Di, Wenhui Wan, and Yongkang Wang. Improved point-line feature based visual slam method for indoor scenes. *Sensors*, 18(10):3559, Oct 2018.

[164] B. Williams and I. Reid. On combining visual slam and visual odometry. In *Proc. International Conference on Robotics and Automation*, 2010.

[165] S. Workman, C. Greenwell, M. Zhai, R. Baltenberger, and N. Jacobs. Deepfocal: A method for direct focal length estimation. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 1369–1373, 2015.

[166] Jianxiong Xiao, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Recognizing scene viewpoint using panoramic place representation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2695–2702. IEEE, 2012.

[167] G. Yammine, E. Wige, F. Simmet, D. Niederkorn, and A. Kaup. Novel similarity-invariant line descriptor and matching algorithm for global motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(8):1323–1335, Aug 2014.

[168] N. Yang, R. Wang, X. Gao, and D. Cremers. Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect. *IEEE Robotics and Automation Letters*, 3(4):2878–2885, Oct 2018.

[169] S. Yang and S. Scherer. Direct monocular odometry using points and lines. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3871–3877, 2017.

[170] Shichao Yang, Yu Song, Michael Kaess, and Sebastian Scherer. Pop-up slam: Semantic monocular plane slam for low-texture environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, October 2016.

[171] Kwang Moo Yi, Yannick Verdie, Pascal Fua, and Vincent Lepetit. Learning to Assign Orientations to Feature Points. In *Proceedings of the Computer Vision and Pattern Recognition*, 2016.

[172] M. Yokozuka, S. Oishi, S. Thompson, and A. Banno. Vitamin-e: Visual tracking and mapping with extremely dense feature points. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9633–9642, 2019.

[173] Georges Younes, Daniel Asmar, Elie Shammas, and John Zelek. Keyframe-based monocular SLAM: design, survey, and future directions. *Robotics and Autonomous Systems*, 98:67 – 88, 2017.

[174] Georges Younes, Daniel Asmar, and John Zelek. Towards a hybrid scene representation in vslam. *Robotics: Science and Systems (RSS): Workshop on Spatial-Semantic Representations in Robotics*, 2017.

[175] Georges Younes, Daniel Asmar, and John Zelek. The advantages of a joint direct and indirect vslam in ar. *CVPR Workshop on Computer Vision for Augmented and Virtual Reality, Long Beach, CA*, 2019.

[176] Georges Younes, Daniel Asmar, and John Zelek. Robustifying direct vo to large baseline motions. In Ana Paula Claudio, Kadi Bouatouch, Manuela Chessa, Alexis Paljic, Andreas

Kerren, Christophe Hurter, Alain Tremeau, and Giovanni Maria Farinella, editors, *Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 477–496, Cham, 2020. Springer International Publishing.

[177] Georges Younes, Daniel C. Asmar, and John Zelek. FDMO: feature assisted direct monocular odometry. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. INSTICC, 2019.

[178] Georges Younes, Daniel C. Asmar, and John S. Zelek. A unified formulation for visual odometry. *CoRR*, abs/1903.04253, 2019.

[179] Q. Yu, J. Xiao, H. Lu, and Z. Zheng. Hybrid-residual-based RGBD visual odometry. *IEEE Access*, 6:28540–28551, 2018.

[180] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4353–4361, June 2015.

[181] Cyril Zeller and Olivier Faugeras. Camera Self-Calibration from Video Sequences: the Kruppa Equations Revisited. Technical Report RR-2793, INRIA, February 1996.

[182] C. Zhang, F. Rameau, J. Kim, D. M. Argaw, J. Bazin, and I. S. Kweon. Deepptz: Deep self-calibration for ptz cameras. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1030–1038, 2020.

[183] Chaofan Zhang, Yong Liu, Fan Wang, Yingwei Xia, and Wen Zhang. Vins-mkf: A tightly-coupled multi-keyframe visual-inertial odometry for accurate and robust state estimation. *Sensors*, 18(11), 2018.

[184] Lilian Zhang and Reinhard Koch. Hand-held monocular slam based on line segments. In *Proceedings of the 2011 Irish Machine Vision and Image Processing Conference*, IMVIP '11, pages 7–14, Washington, DC, USA, 2011. IEEE Computer Society.

[185] X. Zhang, Y. Su, and X. Zhu. Loop closure detection for visual slam systems using convolutional neural network. In *2017 23rd International Conference on Automation and Computing (ICAC)*, pages 1–6, 2017.

[186] X. F. Zhang, A. Luo, W. Tao, and H. Burkhardt. Camera calibration based on 3d-point-grid. In Alberto Del Bimbo, editor, *Image Analysis and Processing*, pages 636–643, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[187] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov 2000.

[188] Huizhong Zhou, Danping Zou, Ling Pei, Rendong Ying, Peilin Liu, and Wenxian Yu. Structslam: Visual slam with building structure lines. *IEEE Transactions on Vehicular Technology*, 64(4):1364–1375, 2015.

[189] Huizhong Zhou, Danping Zou, Ling Pei, Rendong Ying, Peilin Liu, and Wenxian Yu. Structslam: Visual slam with building structure lines. *Vehicular Technology, IEEE Transactions on*, 64(4):1364–1375, April 2015.

[190] Bingbing Zhuang, Quoc-Huy Tran, Gim Hee Lee, Loong Fah Cheong, and Manmohan Chandraker. Degeneracy in self-calibration revisited and a deep learning solution for uncalibrated slam. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3766–3773. IEEE, 2019.

[191] Jon ”Zubizarreta, Iker Aguinaga, and J. M. M.” Montiel. Direct sparse mapping. *IEEE Transactions on Robotics*, 2020.

APPENDICES

.1 Monocular VSLAM Survey

Survey of all monocular *VSLAM* systems. Keyframe bases systems are highlighted with a gray color.

Table 1: VSLAM survey summary from 2003 to 2016.

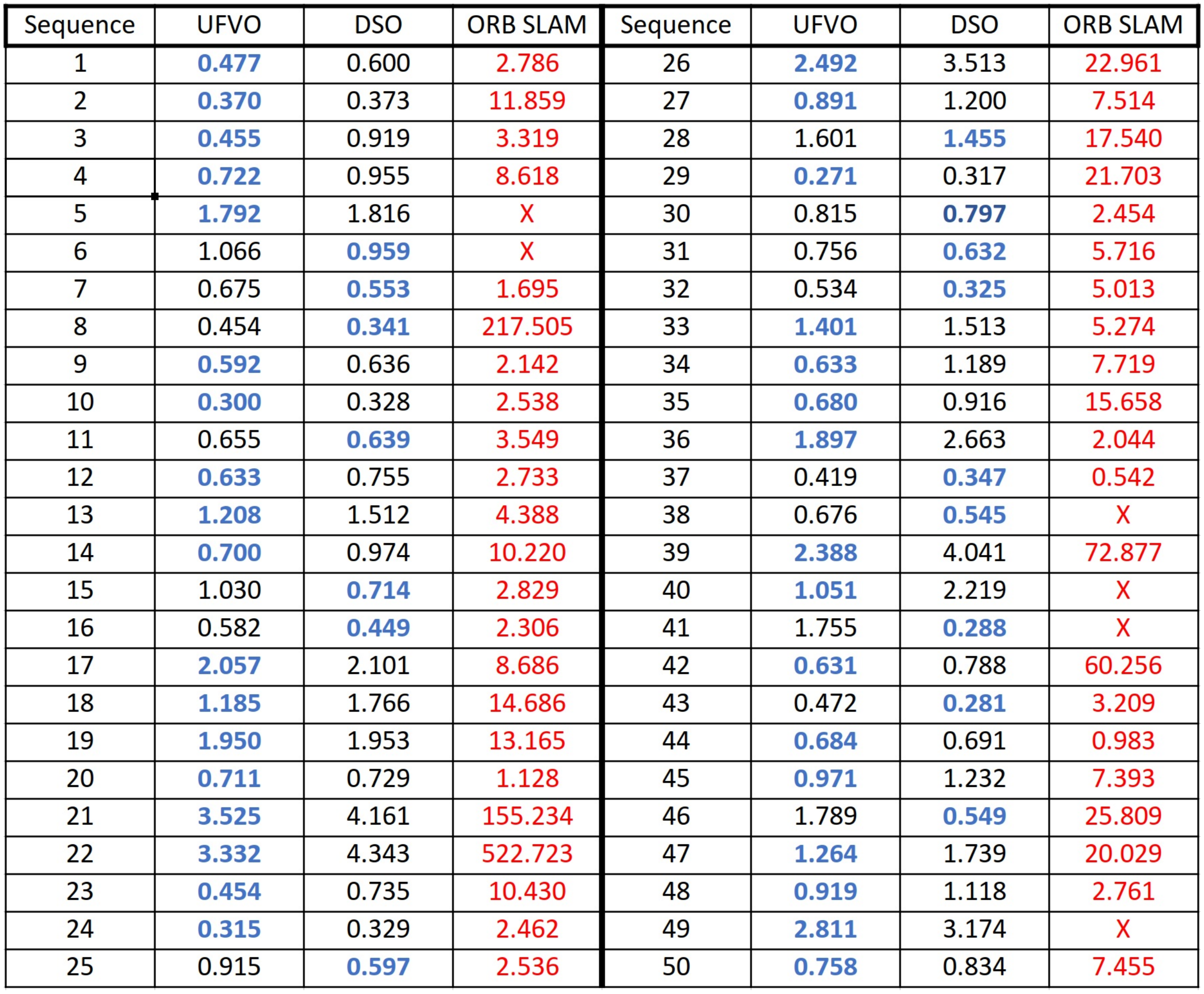
Year	Name	Method	Type	Reference
2003	Real-time simultaneous localization and mapping with a single camera	filter	indirect	[33]
2004	Simultaneous localization and mapping using multiple view feature descriptors	filter	indirect	[115]
2004	Real-Time 3D SLAM with Wide-Angle Vision	filter	indirect	[32]
2005	Real-Time Camera Tracking Using a Particle Filter	filter	indirect	[135]
2005	CV-SLAM	filter	indirect	[81]
2006	Real-time Localization and 3D Reconstruction	keyframe	indirect	[121]
2006	Scalable Monocular SLAM	filter	indirect	[38]
2006	Real-Time Monocular SLAM with Straight Lines	filter	indirect	[149]
2007	Monocular-vision based SLAM using Line Segments	filter	indirect	[95]
2007	MonoSLAM	filter	indirect	[34]
2007	Parallel Tracking and Mapping (PTAM)	keyframe	indirect	[83]
2007	Monocular SLAM as a Graph of Coalesced Observations	filter	indirect	[39]
2007	Mapping Large Loops with a Single Hand-Held Camera	filter	indirect	[27]
2007	Dimensionless Monocular SLAM	filter	indirect	[26]

2008	A Square Root UKF for visual monoSLAM	filter	indirect	[75]
2008	An Efficient Direct Approach to Visual SLAM	keyframe	direct	[147]
2008	Efficient View-Based SLAM Using Visual Loop Closures	filter	indirect	[110]
2008	Large-Scale SLAM Building Conditionally Independent Local Maps: Application to Monocular Vision	filter	indirect	[128]
2009	Towards a robust visual SLAM approach: Addressing the challenge of life-long operation	filter	indirect	[73]
2009	Use a Single Camera for Simultaneous Localization And Mapping with Mobile Object Tracking in dynamic environments	filter	indirect	[118]
2010	On Combining Visual SLAM and Visual Odometry	filter	indirect	[164]
2010	Scale Drift-Aware Large Scale Monocular SLAM	keyframe	indirect	[152]
2010	Live dense reconstruction with a single moving camera	keyframe	hybrid	[125]
2010	Monocular SLAM with locally planar landmarks via geometric rao-blackwellized particle filtering on Lie groups	filter	indirect	[91]
2011	Dense Tracking and Mapping (DTAM)	keyframe	direct	[126]
2011	Omnidirectional dense large-scale mapping and navigation based on meaningful triangulation	keyframe	direct	[133]
2011	Continuous localization and mapping in a dynamic world (CD SLAM)	keyframe	indirect	[131]
2011	Online environment mapping	keyframe	indirect	[100]
2011	Homography-based planar mapping and tracking for mobile phones	keyframe	indirect	[129]
2013	Robust monocular SLAM in Dynamic environments (RD SLAM)	keyframe	indirect	[154]
2013	Handling pure camera rotation in keyframe-based SLAM (Hybrid SLAM)	keyframe	indirect	[130]
2013	Monocular Vision SLAM for Indoor Aerial Vehicles	filter	indirect	[20]
2014	Efficient keyframe-based real-time camera tracking	keyframe	indirect	[36]
2014	Visual SLAM for Handheld Monocular Endoscope	filter	indirect	[62]
2014	Real-time camera tracking using a particle filter combined with unscented Kalman filters	filter	indirect	[94]
2014	Semi-direct Visual Odometry (SVO)	keyframe	hybrid	[52]

2014	Large Scale Direct monocular SLAM (LSD SLAM)	keyframe	direct	[44]
2014	Deferred Triangulation SLAM (DT SLAM)	keyframe	indirect	[71]
2014	Real-Time 6-DOF Monocular Visual SLAM in a Large Scale Environment	keyframe	indirect	[99]
2015	StructSLAM: Visual SLAM With Building Structure Lines	filter	indirect	[189]
2015	Robust large scale monocular Visual SLAM	keyframe	indirect	[12]
2015	ORB SLAM	keyframe	indirect	[124]
2015	Dense Piecewise Parallel Tracking and Mapping (DPPTAM)	keyframe	direct	[29]
2016	Multi-level mapping: Real-time dense monocular SLAM	keyframe	direct	[63]
2016	Robust Keyframe-based Monocular SLAM for Augmented Reality	keyframe	indirect	[102]
2016	Direct Sparse Odometry (DSO)	keyframe	direct	[41]

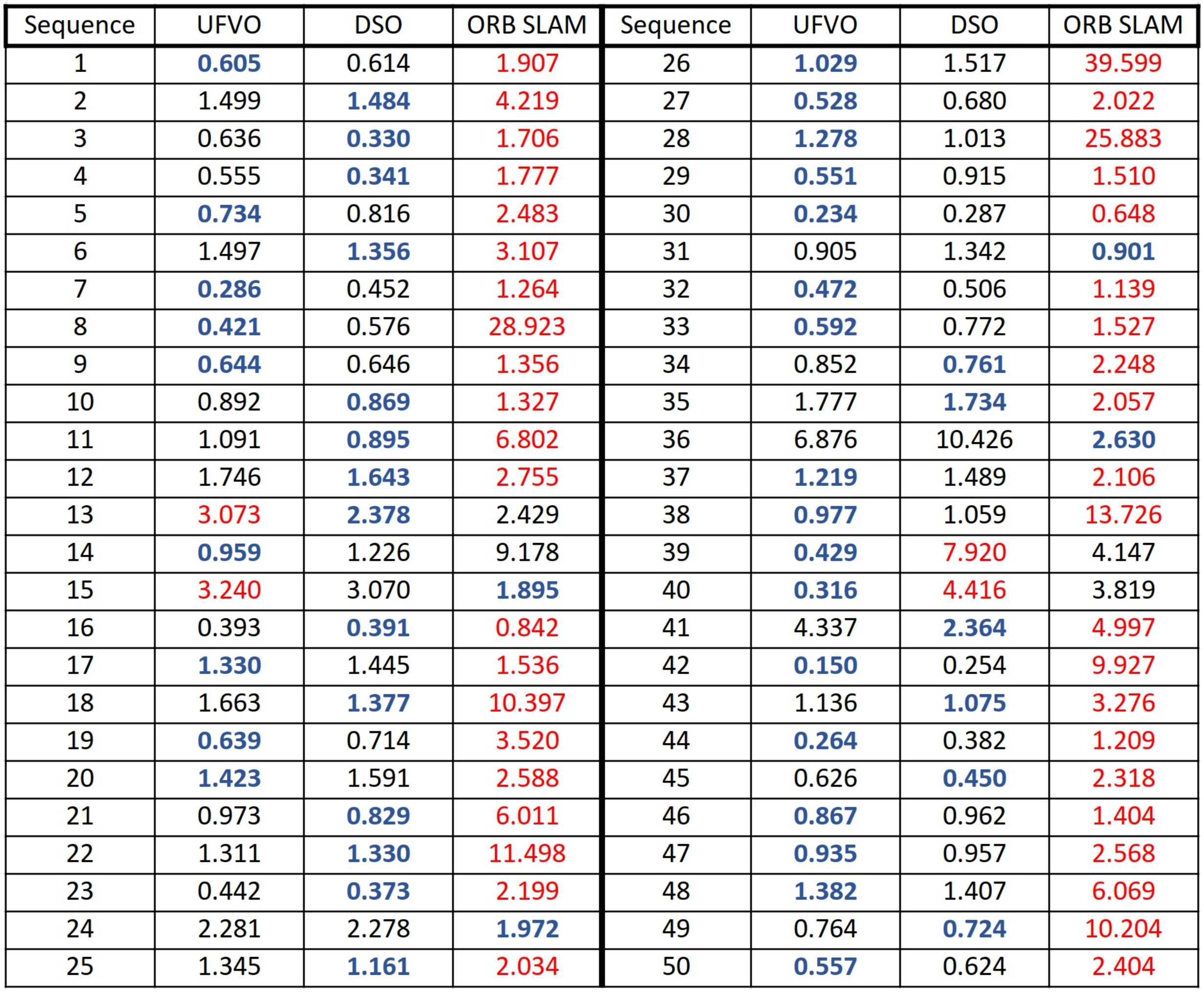
.2 Alignment error on the TumMono dataset.

Table 2: Translational drift.



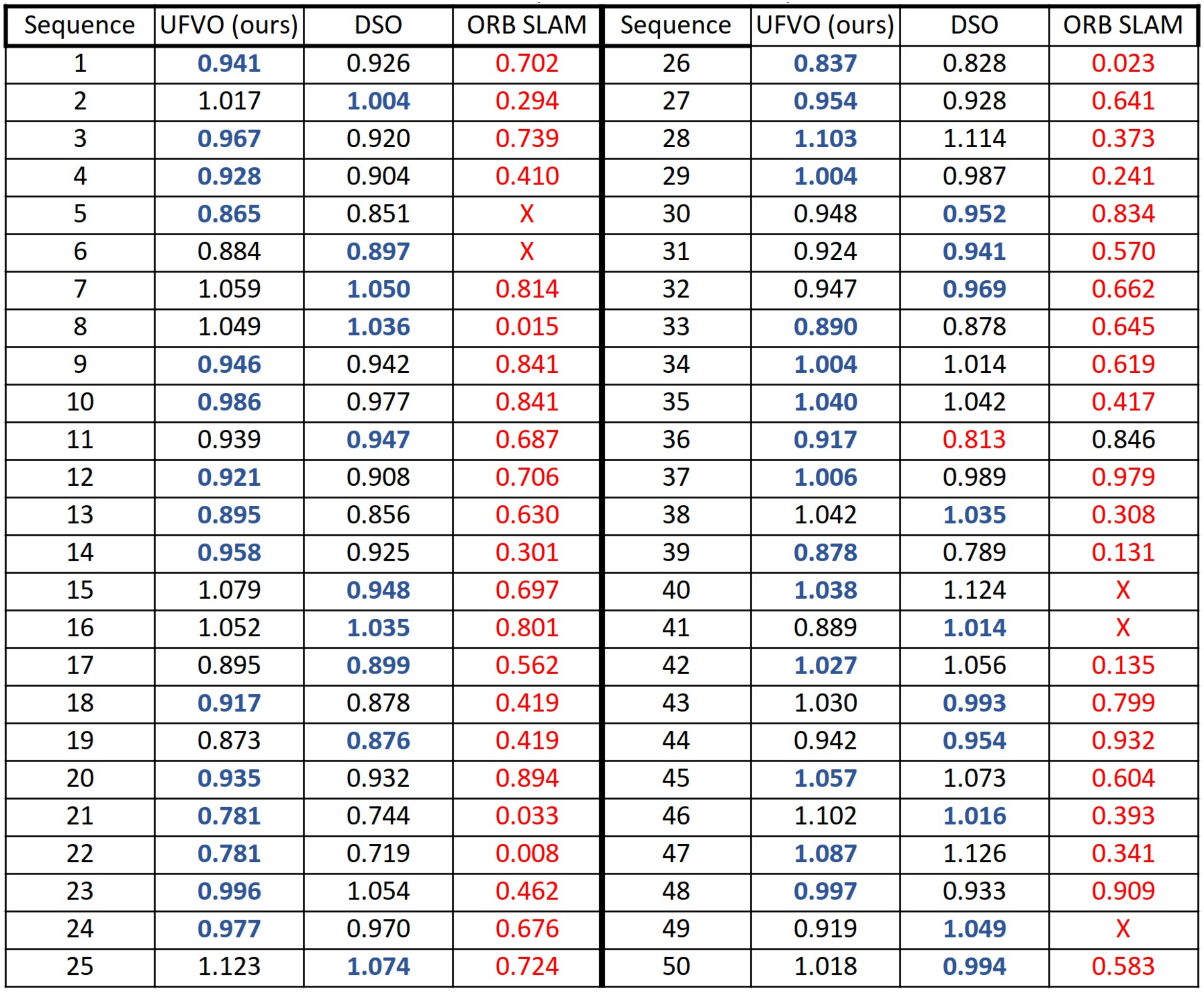
Sequence	UFVO	DSO	ORB SLAM	Sequence	UFVO	DSO	ORB SLAM
1	**0.477**	0.600	2.786	26	**2.492**	3.513	22.961
2	**0.370**	0.373	11.859	27	**0.891**	1.200	7.514
3	**0.455**	0.919	3.319	28	1.601	**1.455**	17.540
4	**0.722**	0.955	8.618	29	**0.271**	0.317	21.703
5	**1.792**	1.816	X	30	0.815	**0.797**	2.454
6	1.066	**0.959**	X	31	0.756	**0.632**	5.716
7	0.675	**0.553**	1.695	32	0.534	**0.325**	5.013
8	0.454	**0.341**	217.505	33	**1.401**	1.513	5.274
9	**0.592**	0.636	2.142	34	**0.633**	1.189	7.719
10	**0.300**	0.328	2.538	35	**0.680**	0.916	15.658
11	0.655	**0.639**	3.549	36	**1.897**	2.663	2.044
12	**0.633**	0.755	2.733	37	0.419	**0.347**	0.542
13	**1.208**	1.512	4.388	38	0.676	**0.545**	X
14	**0.700**	0.974	10.220	39	**2.388**	4.041	72.877
15	1.030	**0.714**	2.829	40	**1.051**	2.219	X
16	0.582	**0.449**	2.306	41	1.755	**0.288**	X
17	**2.057**	2.101	8.686	42	**0.631**	0.788	60.256
18	**1.185**	1.766	14.686	43	0.472	**0.281**	3.209
19	**1.950**	1.953	13.165	44	**0.684**	0.691	0.983
20	**0.711**	0.729	1.128	45	**0.971**	1.232	7.393
21	**3.525**	4.161	155.234	46	1.789	**0.549**	25.809
22	**3.332**	4.343	522.723	47	**1.264**	1.739	20.029
23	**0.454**	0.735	10.430	48	**0.919**	1.118	2.761
24	**0.315**	0.329	2.462	49	**2.811**	3.174	X
25	0.915	**0.597**	2.536	50	**0.758**	0.834	7.455

Table 3: Rotational drift.



Sequence	UFVO	DSO	ORB SLAM	Sequence	UFVO	DSO	ORB SLAM
1	**0.605**	0.614	1.907	26	**1.029**	1.517	39.599
2	1.499	**1.484**	4.219	27	**0.528**	0.680	2.022
3	0.636	**0.330**	1.706	28	**1.278**	1.013	25.883
4	0.555	**0.341**	1.777	29	**0.551**	0.915	1.510
5	**0.734**	0.816	2.483	30	**0.234**	0.287	0.648
6	1.497	**1.356**	3.107	31	0.905	1.342	**0.901**
7	**0.286**	0.452	1.264	32	**0.472**	0.506	1.139
8	**0.421**	0.576	28.923	33	**0.592**	0.772	1.527
9	**0.644**	0.646	1.356	34	0.852	**0.761**	2.248
10	0.892	**0.869**	1.327	35	1.777	**1.734**	2.057
11	1.091	**0.895**	6.802	36	6.876	10.426	**2.630**
12	1.746	**1.643**	2.755	37	**1.219**	1.489	2.106
13	3.073	**2.378**	2.429	38	**0.977**	1.059	13.726
14	**0.959**	1.226	9.178	39	**0.429**	7.920	4.147
15	3.240	3.070	**1.895**	40	**0.316**	4.416	3.819
16	0.393	**0.391**	0.842	41	4.337	**2.364**	4.997
17	**1.330**	1.445	1.536	42	**0.150**	0.254	9.927
18	1.663	**1.377**	10.397	43	1.136	**1.075**	3.276
19	**0.639**	0.714	3.520	44	**0.264**	0.382	1.209
20	**1.423**	1.591	2.588	45	0.626	**0.450**	2.318
21	0.973	**0.829**	6.011	46	**0.867**	0.962	1.404
22	1.311	**1.330**	11.498	47	**0.935**	0.957	2.568
23	0.442	**0.373**	2.199	48	**1.382**	1.407	6.069
24	2.281	2.278	**1.972**	49	0.764	**0.724**	10.204
25	1.345	**1.161**	2.034	50	**0.557**	0.624	2.404

Table 4: Scale drift. The closer to 1 the better.



Sequence	UFVO (ours)	DSO	ORB SLAM	Sequence	UFVO (ours)	DSO	ORB SLAM
1	**0.941**	0.926	0.702	26	**0.837**	0.828	0.023
2	1.017	**1.004**	0.294	27	**0.954**	0.928	0.641
3	**0.967**	0.920	0.739	28	**1.103**	1.114	0.373
4	**0.928**	0.904	0.410	29	**1.004**	0.987	0.241
5	**0.865**	0.851	X	30	0.948	**0.952**	0.834
6	0.884	**0.897**	X	31	0.924	**0.941**	0.570
7	1.059	**1.050**	0.814	32	0.947	**0.969**	0.662
8	1.049	**1.036**	0.015	33	**0.890**	0.878	0.645
9	**0.946**	0.942	0.841	34	**1.004**	1.014	0.619
10	**0.986**	0.977	0.841	35	**1.040**	1.042	0.417
11	0.939	**0.947**	0.687	36	**0.917**	0.813	0.846
12	**0.921**	0.908	0.706	37	**1.006**	0.989	0.979
13	**0.895**	0.856	0.630	38	1.042	**1.035**	0.308
14	**0.958**	0.925	0.301	39	**0.878**	0.789	0.131
15	1.079	**0.948**	0.697	40	**1.038**	1.124	X
16	1.052	**1.035**	0.801	41	0.889	**1.014**	X
17	0.895	**0.899**	0.562	42	**1.027**	1.056	0.135
18	**0.917**	0.878	0.419	43	1.030	**0.993**	0.799
19	0.873	**0.876**	0.419	44	0.942	**0.954**	0.932
20	**0.935**	0.932	0.894	45	**1.057**	1.073	0.604
21	**0.781**	0.744	0.033	46	1.102	**1.016**	0.393
22	**0.781**	0.719	0.008	47	**1.087**	1.126	0.341
23	**0.996**	1.054	0.462	48	**0.997**	0.933	0.909
24	**0.977**	0.970	0.676	49	0.919	**1.049**	X
25	1.123	**1.074**	0.724	50	1.018	**0.994**	0.583