Recursive Estimation of Structure and Motion from Monocular Images

by

Adel H. Fakih

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2010

Abstract

The determination of the 3D motion of a camera and the 3D structure of the scene in which the camera is moving, known as the Structure from Motion (SFM) problem, is a central problem in computer vision. Specifically, the recursive (online) estimation is of major interest for robotics applications such as navigation and mapping. Many problems still hinder the deployment of SFM in real-life applications namely, (1) the robustness to noise, outliers and ambiguous motions, (2) the numerical tractability with a large number of features and (3) the cases of rapidly varying camera velocities. Towards solving those problems, this research presents the following four contributions that can be used individually, together, or combined with other approaches.

A motion-only filter is devised by capitalizing on algebraic threading constraints. This filter efficiently integrates information over multiple frames achieving a performance comparable to the best state of the art filters. However, unlike other filter based approaches, it is not affected by large baselines (displacement between camera centers).

An approach is introduced to incorporate, with only a small computational overhead, a large number of frame-to-frame features (i.e., features that are matched only in pairs of consecutive frames) in any analytic filter. The computational overhead grows linearly with the number of added frame-to-frame features and the experimental results show an increased accuracy and consistency.

A novel filtering approach scalable to accommodate a large number of features is proposed. This approach achieves both the scalability of the state of the art filter in scalability and the accuracy of the state of the art filter in accuracy.

A solution to the problem of prediction over large baselines in monocular Bayesian filters is presented. This problem is due to the fact that a simple prediction, using constant velocity models for example, is not suitable for large baselines, and the projections of the 3D points that are in the state vector can not be used in the prediction due to the need of preserving the statistical independence of the prediction and update steps.

Acknowledgements

I am indebted to numerous people without whom my PhD studies would have been a much harder and less fruitful experience, and to only few of them I can give a particular mention here.

I had the privilege to be supervised by Dr. John Zelek who offered me his assistance and unequivocal support in all possible fashions. I would like to express my sincere gratitude to him for that and for his valuable guidance, critical advice, patience and for being a great academic role model.

Many thanks are extended to the members of my committee, professors, John Barron, Richard Mann, David Clausi, and Hamid Tizhoosh for reading my thesis and for providing helpful and insightful comments. Also, the feedback and comments of Dr. Paul Fieguth on my PhD proposal are very much appreciated.

I owe thanks to the Department of Systems Design Engineering for offering all the necessary amenities and facilities. Particularly, I am very grateful for the graduate coordinator Vicky Lawrence for all her indispensable efforts and her readiness to help at all times.

Words fail me to express my heartfelt gratitude for my parents who, through my childhood and study career, provided me with all their love and encouragement and worked hard to secure me an excellent education. I regret that I would never be able to pay them back.

Daniel Asmar is especially thanked for being a mentor both during my Masters and PhD studies. My interest in Computer Vision is greatly inspired by him and by the mind lifting discussions I had, and still have, with him.

I convey special acknowledgement to my friends, Firas Mansour and Kamal Mroueh and their spouses Melissa and Nesrine, who made my stay in Waterloo enjoyable and made me feel at home. I will always remember the great times we spent together and the delicious meals they generously prepared. I wholeheartedly thank Mohamed Awad for being a true brother and for being there for me whenever I needed help. I also highly value the friendship of Mobarak El Mutairi and his support during the early years of my PhD studies and for being the perfect office mate.

Last but not least, I owe my loving thanks to my sweetheart Noor whose presence, love and never-ending support were a constant source of motivation and inspiration that kept me going through my PhD journey.

Dedication

To my parents.

Contents

List of Tables

List of Figures

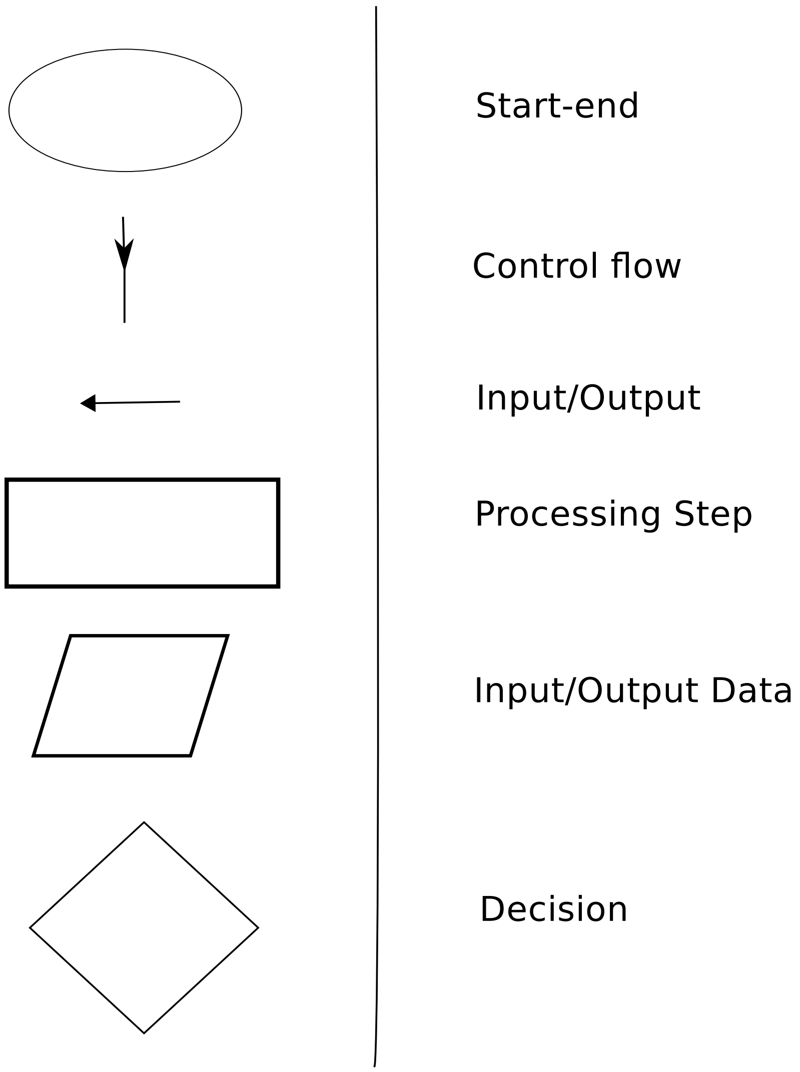
Nomenclature

$[\overrightarrow{V}]_\times$	Skew symmetric matrix corresponding to the vector $\overrightarrow{V}$
$\underline{R}(t)$	Rotation matrix at time t
$\overrightarrow{\Omega}(t)$	Rotation vector at time t
$\overrightarrow{\omega}(t)$	Rotational velocity vector at time t
$\overrightarrow{T}(t)$	Translation vector at time t
$\overrightarrow{V}(t)$	Translational velocity vector at time t
$\underline{M}(t)$	Motion parameters ($\underline{R}$ and $\overrightarrow{T}$ concatenated in a 4×4 matrix)
$\overrightarrow{M}(t)$	Motion parameters in vector form at time t
$\overrightarrow{M}(t_1, t_2)$	Relative motion between t_1 and t_2
$\overrightarrow{P}$	3D point expressed in the inertial frame
$\overrightarrow{Q}(t)$	3D point in the camera reference frame at time t
$\overrightarrow{S}(t)$	State vector (containing the parameters to be estimated)
$\underline{\Sigma}_{\overrightarrow{V}\overrightarrow{U}}$	Cross-covariance between vectors $\overrightarrow{V}$ and $\overrightarrow{U}$
$\underline{\Sigma}_{\overrightarrow{V}}$	Covariance matrix of the vector $\overrightarrow{V}$
I_n	Identity matrix of order n
$\overrightarrow{Z}$	Observation (Measurement) vector
$h(\overrightarrow{S}, \overrightarrow{Z})$	Observation function of the vector $\overrightarrow{S}$
$\underline{H}_{\overrightarrow{S}}$	Jacobian matrix of h with respect to $\overrightarrow{S}$
$\underline{H}_{\overrightarrow{Z}}$	Jacobian matrix of h with respect to $\overrightarrow{Z}$
$\mathbb{R}^3 to SO_3$	Map from 3-vectors(angle-axis or quaternion) to rotation matrices
$SO_3 to \mathbb{R}^3$	Map from rotation matrices to 3-vectors (angle-axis or quaternion)

List of Abbreviations

BA	Bundle Adjustment
BLUE	Best Linear Unbiased Estimate
CenSurE	Center Surround Extrema
EIF	Extended Information Filter
DoG	Difference of Gaussians
EKF	Extended Kalman Filter
FAST	Features from Accelerated Segment Test
FEJ-EKF	First Estimate Jacobian-EKF
GLF	Graph of Local Filters
ISAM	Incremental Smoothing And Mapping
IMU	Inertial Measurement Unit
LoG	Laplacian of Gaussian
LM	Levenberg-Marquardt
NEES	Normalized Estimation Error Squared
PTAM	Parallel Tracking And Mapping
RANSAC	RANdom SAmple Consensus
RBPF	Rao-Blackwellized particle filter
RMS	Root Mean Square
SFM	Structure From Motion
SIFT	Shift Invariant Feature Transform
SIS	Sequential Importance Sampling
SLAM	Simultaneous Localization And Mapping
SRSAM	Square Root Smoothing and Mapping
SRUKF	Square Root Unscented Kalman Filter
SURF	Speeded Up Robust Features
SSD	Sum of Squared Distances
UKF	Unscented Kalman Filter

Flowchart Conventions



Chapter 1

Introduction

The estimation of structure and motion from monocular images known as Structure From Motion (SFM) is a central problem in computer vision. This problem has numerous applications especially in the robotics field for navigation and mapping purposes. The most important applications are visual odometry when only the motion of the camera is required, and Simultaneous Localization And Mapping (SLAM) when both camera motion and scene structure are required. Actually, the only difference between SFM and visual SLAM is that SLAM maintains a map of the available features and involves techniques to recognize the same features when seen again.

The recursive estimation of structure and motion allows the online integration of measurements over frames on the basis that motion and structure change incrementally. Many problems are not yet properly addressed by the state-of-the-art recursive estimation techniques, namely the robustness to noise and ambiguous motions, the numerical tractability with a large number of features and the cases of rapidly changing camera motions. This thesis presents four contributions towards addressing those problems. Specifically, (1) capitalization on three-frames algebraic constraints to provide fast motion-only filtering regardless of the size of the baseline between consecutive frames. (2) A computationally efficient approach to include frame-to-frame features in SFM filters for enhanced accuracy. (3) A multi-hypothesis approach that scales lineraly to large number of features and at the same time having the accuracy of the best state of the art filters. (4) Providing a solution to deal with wide-baseline cases due to rapidly varying camera motions.

The first part of this chapter presents a high level overview of the SFM problems and the related difficulties. The second part presents the contributions of this thesis towards solving the mentioned problems.

1.1 The Recursive SFM Estimation Problem

1.1.1 Structure from Motion

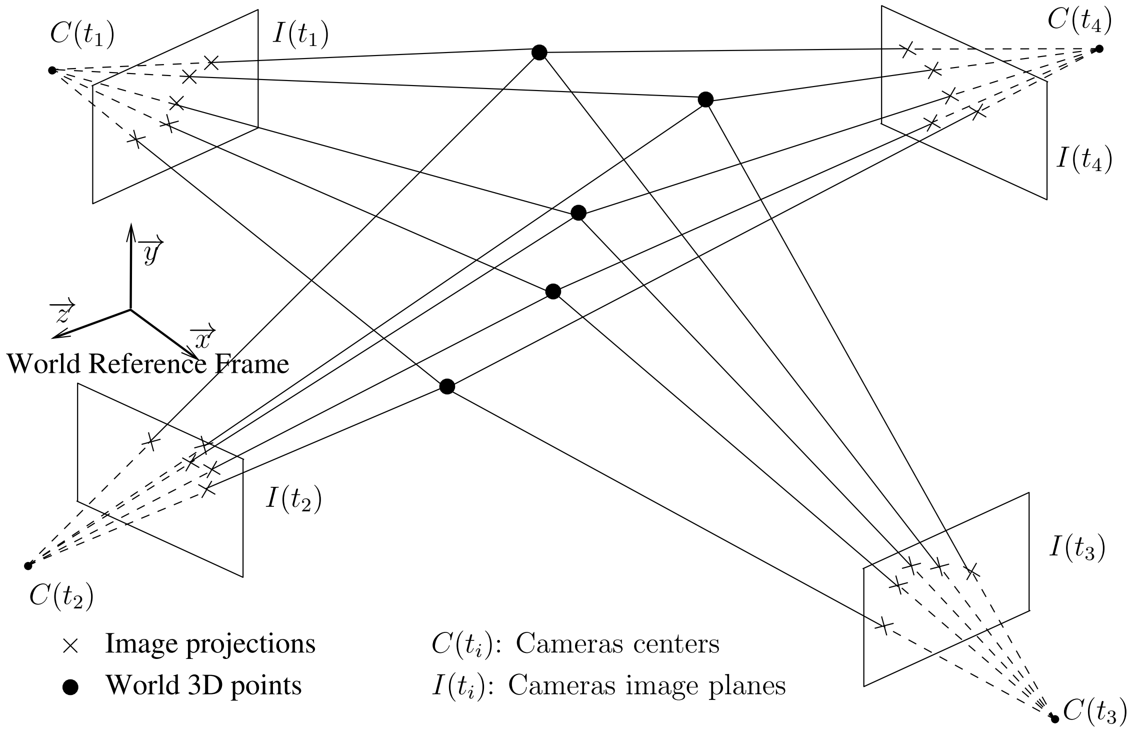


Figure 1.1: The SFM problem: Given a set of $3D$ points imaged from different vantage points (different cameras or same camera at different times), determine from the observed images: the $3D$ locations of the points and the camera pose at every image.

The SFM problem, as depicted in Figure 1.1, aims to determine from a sequence of temporally sequential images of the same scene, the poses of the camera(s) while taking those images and the structure of the scene. The pose of a camera is the position and orientation of the camera with respect to a specified reference frame. The structure of the scene, in the context of this thesis, represents the $3D$ positions of physical world points in the scene. Those 3D points can be sparse key points in the scene or dense providing a cloud representation of almost the whole scene.

In the remainder of this thesis we will use the term “$3D$ parameters” to refer to the structure and/or motion. The SFM problem generally comprises three stages (Figure 1.2). Those three stages are:

1. “feature detection”
2. “feature matching”

3. "$3D$" parameter estimation.

Although, this thesis is only concerned with the third step, the first two steps are important to us since the errors in the outputs of those steps are carried on to and should be dealt with in the third step. Those three steps are briefly described as follows:

- **Feature Detection** Theoretically, the input to SFM should be the whole intensity image treated as one discretized two-dimensional signal. However, although there has been some successful approaches working directly with the image intensities called direct approaches [16, 50, 59], methods based on sparse salient image features are generally more robust. Selected image features with rich image structure can be tracked more successfully than other non textured image regions. That is why most current state of the art approaches are feature based. Nevertheless, many people still advocate the direct approaches as Jitendra Malik once stated in a discussion at ICCV1999 Workshop on Vision Algorithms Applications [106]:

  > From a scientific or aesthetic point-of-view, I don't think there's any comparison. The world consists of surfaces, and the visual world consists of surfaces, with occlusion, non-rigid motion, etc. For engineering purposes, if it so happens that there's just a single moving camera and you can reduce your world to a collection of 20 or 30 points, that's fine. But it's engineering, not fundamental vision.

  In the context of this thesis, we adopt a feature-based point of view, in which the first step in an SFM system is the identification and detection of a set of image features. The image features can be points, lines, or any image structure. The quality of the obtained features, in terms of localization and repetitiveness directly affects the subsequent steps of feature matching and $3D$ parameter estimation; a feature that is not well localized represents a noisy measurement and features that are repetitively detected in a large number of frames carry more information about the $3D$ motion than features detected in a small number of frames. Also, if the feature detector can not detect when a feature gets occluded, it will be transferred to the SFM stage as an outlier in the input data.

- **Feature Matching** The detected features are matched across frames so that features corresponding to the same world point are associated together. Two categories of feature matching exist:

1. Optical flow based where a feature is detected in a single frame only and then is tracked in subsequent frames using the variation of the spatio-temporal image derivatives.
2. Matching based which perform a feature detection in every frame, then attempt to match the new features, to previously detected ones in earlier frames based on some similarity measure.

- **3D parameters estimation** Once image features are detected and matched, they can be used to recover the 3D parameters. This is an inverse problem based on the perspective projection equation. This inverse problem is difficult for five reasons:

  1. The large number of degrees of freedom: Even in the calibrated case (as we consider in this thesis), the 3D parameters are determined up to seven degrees of freedom representing a similarity transformation (Section 2.3.8). Therefore, when performing the estimation using several frames, some parameters should be fixed as a common gauge across all the frames. This causes problems in sequential estimation and induces a drift when the gauge parameters are changed.
  2. The large dimensionality of the problem when many frames and many $3D$ features are considered (6 parameters per frame and 3 parameters per feature) which makes real-time [1] performance very challenging.
  3. The inherent ambiguities (Section 2.3.8) which combined with noise and outliers, make an erroneous motion fit the noisy and contaminated features data better than the correct motion.
  4. The high non-linearity of the perspective projection equation (Equation (2.12) involving multiplication of the $3D$ parameters and a division by the depth).
  5. The high susceptibility of the feature detection and matching to noise and incorrect matching, and the necessity to deal with the resulting errors in the estimation step.

1.1.2 Recursive Estimation of Structure and Motion

In the case of perfect point matches, SFM is a solved problem and the 3D parameters can be recovered linearly in real-time. In the presence of noise and outliers, linear approaches

[1]Real-time performance is application dependent and is often 30Hz.

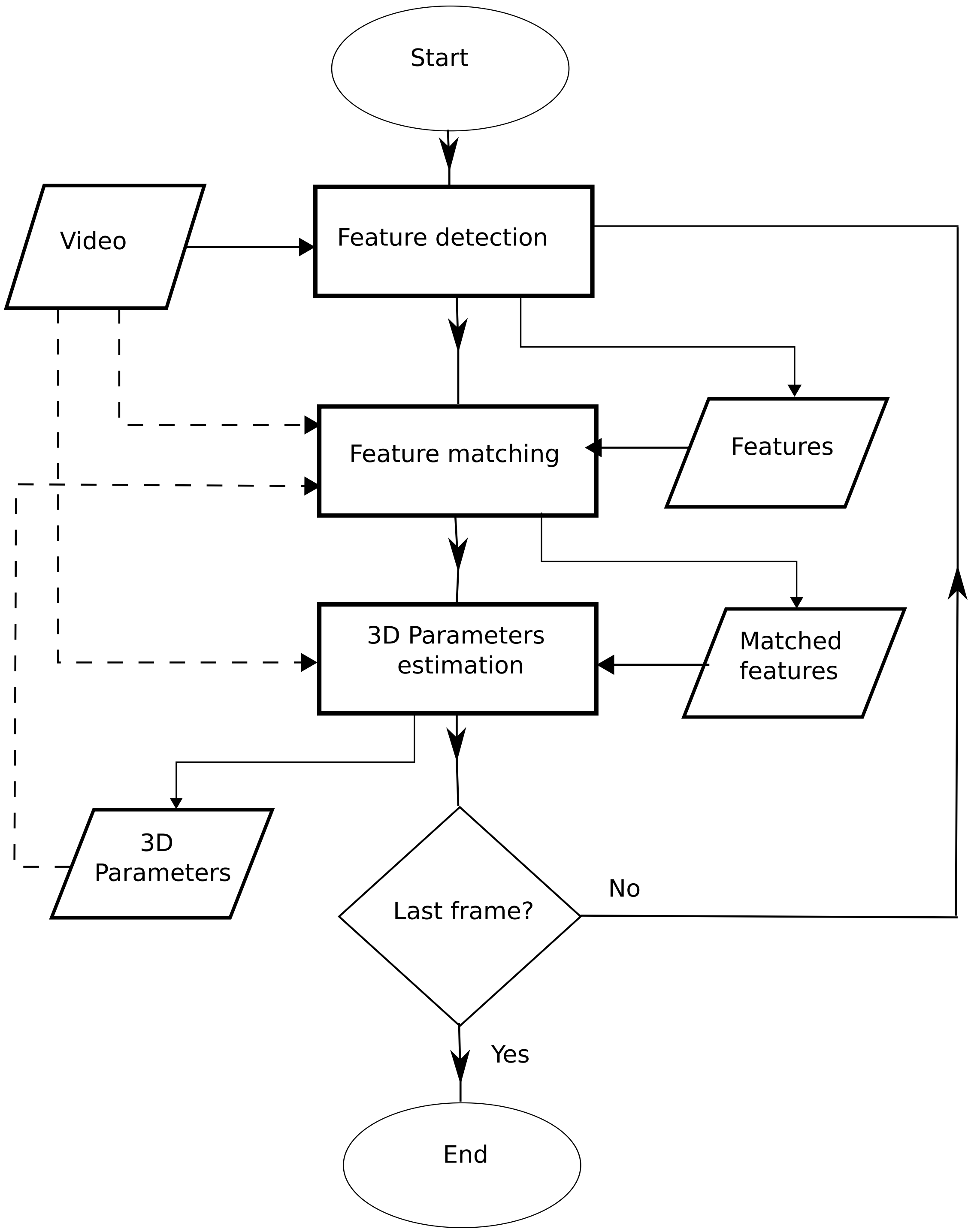


Figure 1.2: The SFM estimation flowchart depicting the three main steps: feature detection, matching and estimation. The dashed arrows represent conditional inputs to reflect the fact that many approaches do not use the images after the feature detection step, and that some approaches use the estimated motion and structure in the matching process.

become very fragile and a non-linear estimation enforcing all the geometric constraints becomes mandatory if decent estimates are expected. The estimation is carried on with the aim of determining the best motion that fits the features data. If the data for the whole sequence is available and there are no hard time limits, the best approach is to find simultaneously the $3D$ parameters that minimize the re-projected errors in all the frames using a combination of gradient descent and Gauss-Newton iterations which is known in photogrammetry and computer vision as “bundle adjustment” [129]. However, when new estimates are required as soon as new images are captured, such approach is no longer applicable. Recursive estimation aims to address this problem by using the previous estimates and the current measurement in an effort to efficiently determine a solution equivalent to the best 3D parameters that can be obtained given all the history of previous measurements up to the current time.

1.2 Thesis Contributions

This thesis tackles many aspects of the mentioned SFM problems and difficulties. The use of multi-hypothesis and probabilistic paradigms are advocated, which are consistent with Marr’s principle of least commitment [79] stating that the maximum possible amount of information should be carried between processing stages before making the final decision. This is helpful in dealing with the outliers and with the SFM ambiguities problems. The noise problem is dealt with by incorporating more measurements in a time-efficient and statistically consistent manner. Within those lines of thought, this thesis presents four contributions to improve the performance of SFM. Those contributions can be regarded as separate strategies targetting different problems in SFM and they can be combined together in one system. Furthermore, they can be incoporated within State of the art approaches framework such as the Parallel Tracking and Mapping (PTAM) approach [68]. Those contributions are the following:

- **A fast and robust motion-only filter capitalizing on threading constraints:** Threading constraints are algebraic constraints that link the motion of the camera to the features matched across three frames. For cases when only the camera motion is required, a new filtering technique capitalizing on the threading constraints is proposed to design a recursive filter which provides motion estimates with accuracy at least matching the state-of-the-art, and with better computational efficiency and robustness to outliers. Also, unlike Bayesian filters, this filter is not affected if the inter-frame dispacement between consecutive frames is large (Chapter 3).

- **A method to efficiently incorporate frame-to-frame features in analytic SFM filters:** Frame-to-frame features are features matched between every two consecutive frames only. These features are abundant and carry useful information about the motion. However, there is no easy way to incorporate them in analytic SFM filters (in addition to the features tracked over time) principally because of the hight cost and because different filters might need to treat them differently. We present a new filtering approach that allows to incorporate the information from a large number of frame-frame features within any analytic filter. The approach is very computationally efficient and is able to process hundreds of frame-to-frame features in less than 10 ms (Chapter 4).

- **A scalable accurate multi-hypothesis filter:** Having filters that scale well to accommodate a large number of features has been always an important problem in SFM. The best solution so far is based on the Rao-Blackwellized Particle Filter (RBPF) and is highly scalable, however it is accuracy is lower than other filters. We present a filter addressing the scalabilty issue that matches the scalability of the RBPF and yet achieves an accuracy similar to the best state of the art filters (Chapter 5).

- **Extending monocular filter-based estimation to large baselines:** Filtering based methods following the Bayesian predict-update style are not suitable to operate with large baselines because of the absence of a mechanism to predict the motion between consecutive frames. This problem is specifically hard in the case of monocular images. We provide a solution to this problem based on features matched only between every three consecutive frames. This solution allows the prediction of a new motion without using the projections of the features that are in the state vector and hence jeopardizing the statistical independence of the predict and update filtering stages (Chapter 6).

In Chapter 7 we discuss the impact of those four contributions on the SFM problem, how they can be integrated together and how they can be extended to include GPS and inertial measurements.

Chapter 2

Background

Chapter 1 presented a flowchart depicting the three main stages of SFM: feature matching, feature detection and $3D$ parameters estimation. This chapter reviews how these stages have been addressed previously in the literature. As the focus in this thesis is on the estimation of the $3D$ parameters, the first two stages (feature detection and feature matching and tracking) are described only briefly for the sake of completeness. Most of the chapter is devoted to the $3D$ parameter estimation and is organized in three parts: the first part (Section 2.3) introduces the notations and a mathematical statement of the SFM problem. The second part (Section 2.4) revisits the classical SFM estimation techniques which constitute the fundamentals and foundations upon which modern estimation approaches are based. As we are concerned only with the online sequential SFM, the review of the state of the art will only address online approaches (Section 2.5) .

2.1 Feature Detection

The detection of significant image features is the most fundamental problem in computer vision. Although some computer vision techniques use directly the image intensities as their input, most of them require some preprocessing of the image intensities to generate features that represent useful entities. Features can be anything ranging from point features, to patches, lines and segments. In the scope of this thesis, we limit the term feature to point features.

The anticipated use of the features affects the properties that are desired for these features. Features sought for SFM are different than those detected to be used in object recognition. For SFM the most desired properties for the features are localization, repetitivity and computational cost. Localization means that the same feature, detected in many frames, should be well localized on the projection of the same physical points. Repetitivity (or stability) means that the feature is most likely to be detected in all the frames where it appears. Repetitivity requires the features to be persistent across viewpoint change. The computational cost of the detection process is very important for online structure from motion, since for a system operating at $30Hz$ the detection step should not be allocated much more than $5ms$.

Feature detection techniques operate by computing a response function across the image, and regions for which the response function exceeds a threshold are assumed to be salient features. A non-maximal suppression is often performed to remove features whose response is not maximal locally. Harris [44] for example, computes the following matrix of second image derivatives:

$$\underline{H} = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, \tag{2.1}$$

where I_x^2, I_y^2 and I_{xy} are the spatial second derivatives of the image at every pixel. Harris then defines the saliency response to be:

$$C = ||\underline{H}|| - k(trace(\underline{H}))^2 \tag{2.2}$$

where the operator $||.||$ represents the Frobenius norm of a matrix and k is a tuning parameter. Shi and Tomasi [107] performed an analysis based on the assumption of affine image deformation and found that a better measure of saliency is the smallest eigen-value of the matrix $\underline{H}$.

Rosten and Drummond [102] introduced a new approach for corner detection based on an analysis of a circle of sixteen pixels around the corner candidate. If a number n of pixels in this circle are all brighter or darker than the center point, this point is considered as a corner. To speed up the process, they used machine learning to avoid testing all the sixteen locations in every circle and come up with a decision tree which tests on average three pixels in the circle. This method is known by the acronym FAST (Features from Accelerated Segment Test).

The approaches mentioned so far find points that are well localized and are relatively invariant to change of view. However, they are not invariant to scale and therefore not very stable across scale changes. To deal with the scale changes another set of approaches use scale-space techniques to detect features at different scales. Lowe [77], in his SIFT (Shift Invariant Feature Transform) detector, convolves the image with a difference of Gaussians (DoG) kernel at multiple scales, retaining locations which are optima in scale and space. DoG is used because it is a good approximation for the Laplacian of Gaussian (LoG) and much faster to compute. The Harris-Laplace and Hessian Laplace features [82] combine scale-space techniques with the Harris approach. They use a scale-adapted Harris measure or the determinant of the Hessian matrix to select the features and the Laplacian to select the scale. Bay *et al.* [15], in the SURF (Speeded Up Robust Feature) detector use a very basic approximation of the Hessian matrix which can be computed very efficiently using integral images. These scale-space approaches, although they achieve a much better invariance to scale than single scale detectors, they do not perform as well in localization. The reason is that features detected at higher levels of the pyramid are not well localized relative to the original image. Therefore, their use involves a tradeoff between repetitivity and localization. Agrawal and Konolige [3] addressed this in their CenSurE (Center Surround Extrema) detector by computing features at all scales at every pixel in the original image (no pyramid). They relied on a class of simple center-surround filters that can be computed in time independent of their size, which results in a fast multi-scale detector. However this detector has only an approximate rotational invariance and performs worse than SIFT when large variations in rotations and scaling exist.

There is no clear winner detector that outperforms others in all aspects. The choice of feature detector depends on many application related factors, such as the computational requirements, the type of images and expected features and the types of camera motion. In this thesis, we use Shi and Tomasi's detector [107] and FAST [102] when working with small baselines since they are faster to detect (less than $10ms$ per frame) and SURF [15] when working with large baselines (about $80ms$) per frame. One of the reasons behind choosing those specific detectors is the availability of software implementations.

2.2 Feature Matching

There are many approaches used to match features for SFM purposes. They can be classified into three categories:

- Optical flow: These approaches attempt to find the displacement of image features based on the spatio-temporal variation of image derivatives. Such approaches provide good tracking for a small number of frames and are applicable only in the case of small baseline between consecutive frames. However, due to error accumulation, they might lead to erroneous matches if used for an extended number of frames. A standard approach to track a point using optical flow is Lucas-Kanade's method [78] based on the brightness constancy equation:

$$I_x u + I_y v = -I_t, \tag{2.3}$$

where I_x and I_y are the horizontal and vertical spatial image derivatives, I_t is the temporal image derivative, and u and v are the horizontal and vertical optical flow values. This constraint is a single equation with two unknowns and is not enough to determine the optical flow. Based on the assumption of a translational local image model, Lucas and Kanade assume the optical flow to be constant within a small window, thus, they stack the brightness constancy equations for all the pixels in the window in one system which can be solved linearly for u and v. Later on, Tomasi and Kanade [124] extended this to affine image motion models. Bouguet [18] provided a pyramidal implementation of this approach to deal with large image displacements up to few pixels. Other approaches attempts to recover optical flow as the displacement that minimizes the Sum of Squared Distances (SSD) between two small rectangular windows centered on the original point and the candidate displaced point [110, 134]. Although those approaches are sometimes classified as optical flow techniques, their mode of operation is more similar to the matching based approaches.

- Matching: Instead of trying to find the displacement of a given feature, such approaches perform a new feature detection at every frame. Then every feature in one frame is compared to every feature within a fixed distance from it in the next image by computing a distance measure between some defined descriptors of the features. The feature yielding the smallest distance is considered a match. The descriptor can be a just a small patch around the feature and the distance measure can be the SSD distance. However the SSD distance is very sensitive to illumination variation and noise. The normalized cross correlation performs much better and can be implemented to be fairly fast [74, 90]. Nevertheless, the normalized cross correlation between image patches is not robust enough to rotations and scaling. More advanced approaches use more sophisticated descriptors that are invariant to scale, rotation and illuminations and are based on sets of weighted histograms of gradient orientation. The most popular and robust techniques in this category are SIFT [77] and Speeded Up Robust Features (SURF) [15]. However, the computational cost of these approaches makes them not very suitable in some cases for hard real time requirements. A performance evaluation of the most popular descriptors has been performed by Mikiolajczyk and Schmid [83].

- Direct search approaches: Those approaches such as [30, 33] track features by actively searching new frames for the projections of already estimated $3D$ points. A patch around the first observation of each feature is stored. Then in subsequent frames, a search region for every feature is determined using the current estimate of the $3D$ point corresponding to that feature, the current estimate of the camera pose and the uncertainty in these estimates which yield a Gaussian estimate (3σ ellipse) of where the feature should be in the second image. The stored feature's patch is warped by an affine homography computed from the motion between the current camera pose estimate and its estimate when the feature was seen. Then this warped patch is compared to every pixel inside the ellipse. The pixel yielding the maximal normalized cross correlation with the warped patch is taken as the new position of the feature.

In this thesis, we use a pyramidal implementation of the Kanade-Lucas-Tomasi optical flow based tracker by Bouguet [18] (provided by OpenCV) for small baseline tracking and SURF [15] (binaries provided by the authors) for large baselines.

2.3 SFM: Notations and Problem Statement

This section is intended to introduce the notations adopted in the thesis and provide a mathematical statement of the SFM problem and the main constraints relating the $3D$ parameters to the image measurements.

2.3.1 Vectors and Matrices Notations

Right pointing arrows ($\overrightarrow{V}$) are used to represent vectors and underlines ($\underline{A}$) are used to represent matrices. V_i denotes the i^{th} element of the vector $\overrightarrow{V}$ and A_i denotes the i^{th} element of the matrix $\underline{A}$ in row-major order. The vector $\overrightarrow{A}^i$ represents the i^{th} column of the matrix $\underline{A}$. Similarly, $\underline{A}^{i:j}$ represents the sub-matrix of $\underline{A}$ consisting of the columns i to j. The horizontal concatenation of vectors is represented as $[\overrightarrow{V}^1, \overrightarrow{V}^2]$ and the vertical concatentation as $[\overrightarrow{V}^1; \overrightarrow{V}^2]$. The superscript "^{T}" represents the transposition of vectors and matrices. Also, for a matrix $\underline{A}$ the notation $\overrightarrow{A}$ represents the vector obtained by stacking the elements of $\underline{A}$ in a row-major order. The superscripted index between two parentheses (ex. $\overrightarrow{V}^{(i)}$) is used to denote a particle or a hypothesis. For every vector $\overrightarrow{V}$, $\underline{\Sigma}_{\overrightarrow{V}}$ represents the covariance matrix associated with this vector, and $\underline{\Sigma}_{\overrightarrow{V}\overrightarrow{U}}$ represents the cross-covariance matrix between the vector $\overrightarrow{V}$ and the vector $\overrightarrow{U}$. The notation $||.||$ is used to denote the euclidean norm of vectors. Also we use the notation $[\overrightarrow{V}]_\times$ to represent the skew symmetric matrix corresponding to the 3-vector $\overrightarrow{V} = (V_1, V_2, V_3)^T$ and defined as:

$$[\overrightarrow{V}]_\times \overset{\text{def}}{=} \begin{bmatrix} 0 & -V_3 & V_2 \\ V_3 & 0 & -V_1 \\ -V_2 & V_1 & 0 \end{bmatrix}. \tag{2.4}$$

The skew symmetric matrix simplifies the notation of many mathematical expressions such as the cross product and the expression of the essential matrix (Section 2.4.2).

2.3.2 SFM Notations and Representation

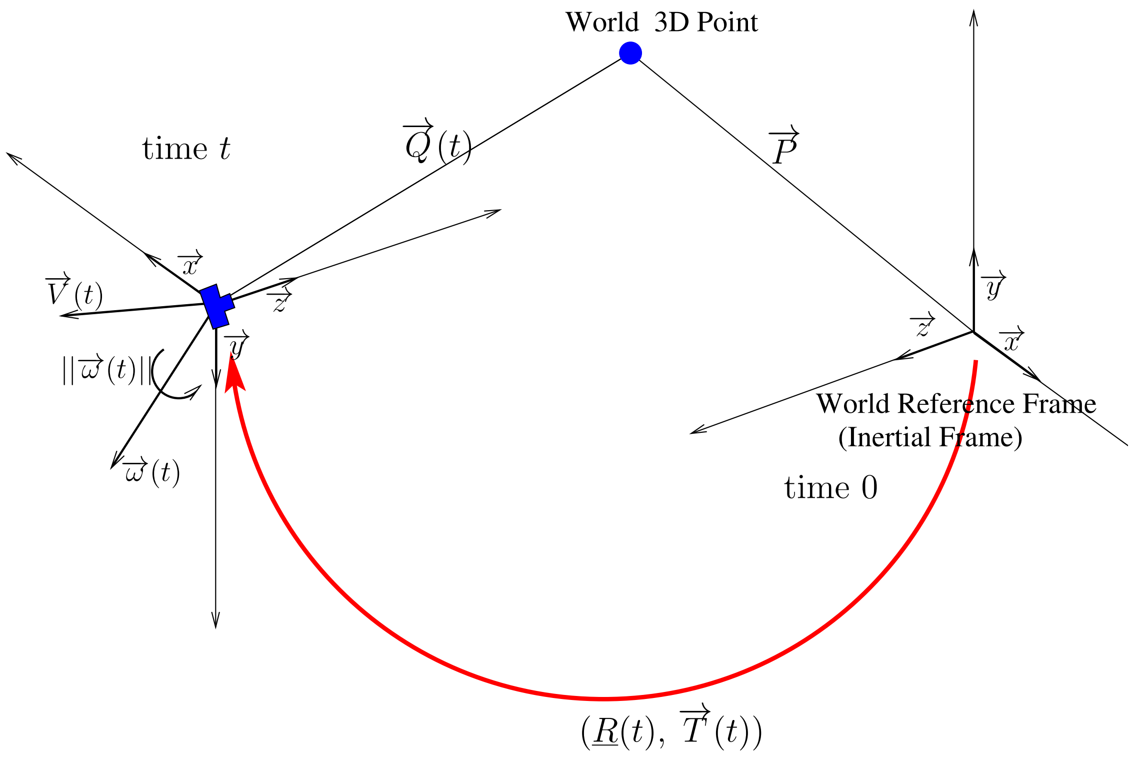


Figure 2.1: Inertial frame, camera pose represented by a rotation matrix $\underline{R}$ and a translation vector $\vec{T}$, camera velocity represented by $\vec{\omega}$ and $\vec{V}$, and $3D$ point configuration represented by $\vec{P}$ in the inertial frame and $\vec{Q}(t)$ in the camera frame.

The 3D parameters are expressed with respect to a fixed orthonormal reference frame (inertial frame Figure 2.1) with unit vectors $\overrightarrow{x}$,$\overrightarrow{y}$ and $\overrightarrow{z}$.

The structure at time t consists of a number $N(t)$ of $3D$ points represented by their Euclidean vectors $\overrightarrow{P}^i = (P_1^i, P_2^i, P_3^i)^T$, with $i \in [1, N(t)]$ and P_1, P_2 and P_3 being the projections of $\overrightarrow{P}$ on the directions $\overrightarrow{x}$, $\overrightarrow{y}$ and $\overrightarrow{z}$ respectively (i.e. $P_1 = \overrightarrow{P} \cdot \overrightarrow{x}$).

At time $t = 0$, the camera is supposed to be centered in the origin of the inertial reference frame with its optical axis coincident with the $\overrightarrow{z}$ direction and oriented in such a way that its image plane is parallel to the $(\overrightarrow{x}, \overrightarrow{y})$ plane (see Figure 2.1).

At time t, the pose of the camera with respect to the reference frame is encoded by a rotation of its optical axis about the origin of this frame and a translation of its center away from this origin. The translation is represented by a translation vector $\overrightarrow{T}(t) = (T_1, T_2, T_3)^T$. There are different ways to represent the rotation:

- **Angle-axis representation:** As per the Euler rotation theorem [36], the rotation can be represented by a minimum of three parameters: an angle θ and a direction $\overrightarrow{\Omega}$. Therefore, the 3-vector $\theta\overrightarrow{\Omega}$ provides a minimal parameterization of the rotation.

- **Rotation matrix:** The angle/axis representation, being minimal, is suitable for estimation purposes. However, the most convenient mathematical way to represent a rotation is through the Direct Cosine Matrix (DCM) $\underline{R}$ known also as the *rotation matrix* . The columns of $\underline{R}$ are the rotated unit vectors $\overrightarrow{x}$, $\overrightarrow{y}$ and $\overrightarrow{z}$ by an amount θ about the direction $\overrightarrow{\Omega}$ (3 dimensional unit vector). The matrix $\underline{R}$ belongs to the special orthogonal group of 3×3 matrices $(SO(3))$.

- **Quaternions:** The angle/axis representation can only be used for rotations between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$ because outside this range there would be two different values representing the same physical rotations. A better representation when large angles are used is the unit-quaternions. Unit-quaternions can also be represented as a 3-vector.

The details about those representations and the conversion from one to the other are provided in Appendix B. In the remainder of this thesis we will use the notation $\overrightarrow{\Omega}$ when the rotation is represented as a 3-vector whether it is in quaternion form or angle-axis form. The terminology $SO_3 to \mathbb{R}^3(\underline{R})$ and $\mathbb{R}^3 to SO_3(\overrightarrow{\Omega})$ to represent the map from $\underline{R}$ to $\overrightarrow{\Omega}$ and from $\overrightarrow{\Omega}$ to $\underline{R}$ respectively, as such maps are conversions between the three dimensional vector space of real numbers $\mathbb{R}^3$ and the special group of orthonormal 3×3 matrices $SO(3)$. Note that those maps are different depending on whether quaternions or angle-axis representation is used, but such terminology will allow us to have a unified notation for both cases. The motion parameters $\overrightarrow{T}$ and $\underline{R}$ are sometimes concatenated in the 4×4 matrix $\underline{M}$ as follows:

$$\underline{M} \overset{\text{def}}{=} \begin{bmatrix} \underline{R} & \overrightarrow{T} \\ 0 & 0 \end{bmatrix}. \tag{2.5}$$

We also use the equivalent vector form of the motion: $\overrightarrow{M} = [\overrightarrow{\Omega}; \overrightarrow{T}]$.

The velocity of the camera is represented by a translational velocity vector $\overrightarrow{V}$ and a rotational velocity vector $\overrightarrow{\omega}$ parameterized in the angle-axis form since it is the most suitable for small and infinitesimal rotations.

2.3.3 Rigid Motion Equations

As the camera moves, the relative position of the $3D$ points with respect to the camera changes. From a mathematical perspective, this is exactly equivalent to the camera being fixed and the $3D$ points moving in its field of view. As this latter case simplifies the notation, it will be assumed in the subsequent formulations. Under the effect of the motion $\underline{M}(T)$, the $3D$ point $\overrightarrow{P}$ will move to a new position $\overrightarrow{Q}(t)$ in the camera reference frame. $\overrightarrow{Q}$ is related to $\overrightarrow{P}$ by the equation of rigid bodies motion:

$$\overrightarrow{Q}(t) = \underline{R}(t)\overrightarrow{P} + \overrightarrow{T}(t) = \underline{M}(t) \begin{bmatrix} \overrightarrow{P} \\ 1 \end{bmatrix}. \tag{2.6}$$

Note that this is equivalent to having the camera undergo a rigid transformation of $(\underline{R}^T, -\underline{R}^T\overrightarrow{T})$.

If the rigid motion is infinitesimally small consisting of a rotational velocity $\overrightarrow{\omega}(t)$, and a translational velocity $\overrightarrow{V}(t)$, the differential equation of rigid bodies motion gives the differential change in the $3D$ point position as:

$$\frac{d\overrightarrow{Q}(t)}{dt} = \overrightarrow{\omega}(t) \times \overrightarrow{Q}(t) + \overrightarrow{V}(t). \tag{2.7}$$

2.3.4 Interframe Motion and Motion Evolution

If the 3D points are subjected between two time instants t_1 and t_2 to a rigid motion $\underline{M}(t_1, t_2)$ composed of the rotation $\underline{R}(t_1, t_2)$ and the translation $\overrightarrow{T}(t_1, t_2)$, the new pose

$\underline{M}(t_2)$ of the points with respect to the inertial frame is determined through composition of rigid transformation as:

$$\begin{aligned}\overrightarrow{T}(t_2) &= \underline{R}(t_1,t_2)\overrightarrow{T}(t_1) + \overrightarrow{T}(t_1,t_2)\\ \underline{R}(t_2) &= \underline{R}(t_1,t_2)R(t_1),\end{aligned} \tag{2.8}$$

Similarly, if the 3D points are subjected to a differential motion $[\overrightarrow{\omega};\overrightarrow{V}]$, their new pose with respect to the inertial frame will be:

$$\begin{aligned}\overrightarrow{T}(t+1) &= e^{[\overrightarrow{\omega}(t)]_\times}\overrightarrow{T}(t) + \overrightarrow{V}(t)\\ \underline{R}(t+1) &= e^{[\overrightarrow{\omega}(t)]_\times}\underline{R}(t),\end{aligned} \tag{2.9}$$

2.3.5 Projection Model

There are many image projection models used in the computer vision community. For a thorough discussion of these models see [40]. In this thesis, the only projection model considered is the perspective projection model shown in Figure 2.2. The perspective projection equation determines the projection $\overrightarrow{q} = (q_1, q_2, f)^T$ of the point $\overrightarrow{Q}(t)$ on the image plane:

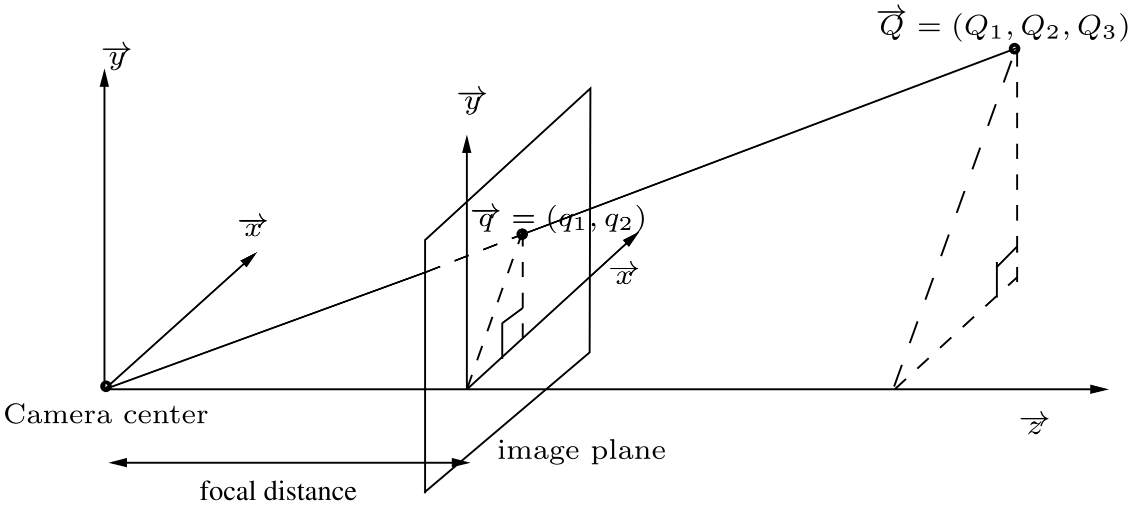


Figure 2.2: Perspective Projection: The world point $\vec{Q}$ projecting to the point $\vec{q}$ on the camera image plane.

$$\overrightarrow{q} = \begin{bmatrix} q_1 \\ q_2 \\ 1 \end{bmatrix} = proj(\overrightarrow{Q}) \overset{\text{def}}{=} \underline{K} \begin{bmatrix} \frac{Q_1}{Q_2} \\ \frac{Q_1}{Q_2} \\ 1 \end{bmatrix}, \tag{2.10}$$

where

$$\underline{K} = \begin{bmatrix} f_1 & 0 & c_1 \\ 0 & f_2 & c_2 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

is the calibration matrix: f_1 and f_2 refer to the focal length in horizontal and vertical pixels respectively; c_1 and c_2 are the coordinates of the center of the camera. In this thesis we assume a calibrated framework, i.e., $\underline{K}$ is known and the points $\overrightarrow{q}$ have been multiplied by $\underline{K}^{-1}$.

2.3.6 Structure from Motion Constraints

The SFM constraints are equations that relate the image measurements to the $3D$ parameters and are derived from the rigid motion equations, for both the discrete and continuous cases, combined with the perspective projection equation.

In the discrete case, substituting Equation (2.6) in Equation (2.10) results in the following equation:

$$\overrightarrow{q} = proj(\underline{R}\overrightarrow{P} + \overrightarrow{T}), \tag{2.12}$$

which is a constraint relating the structure and motion to be estimated to the observed image projection. We refer to this equation as the discrete SFM constraint.

In the continuous case, substituting Equation (2.7) in Equation (2.10) results in the following equation:

$$\dot{\overrightarrow{q}}(t) = -\frac{A(\overrightarrow{q})\overrightarrow{V}}{Q_z} - B(\overrightarrow{q})\overrightarrow{\omega}, \tag{2.13}$$

where

$$A(\overrightarrow{q}) = \begin{bmatrix} 1 & 0 & -q_1 \\ 0 & 1 & -q_2 \end{bmatrix} \tag{2.14a}$$

and

$$B(\overrightarrow{q}) = \begin{bmatrix} -q_1q_2 & 1+q_1^2 & -q_2 \\ -1-q_2^2 & q_1q_2 & q_1 \end{bmatrix}. \tag{2.14b}$$

$\dot{\overrightarrow{q}}$ is the image velocity (optical flow) at the pixel $\overrightarrow{q}$. We refer to this equation (derived first by Longuet-Higgins and Prazdny [76]) as the differential (or continuous) SFM constraint.

2.3.7 Structure from Motion Problem Statement

Having a set of features tracked in a sequence of images, such that at a time t, $N(t)$ features are available and possibly $K(t)$ optical flow values. Equations (2.12) and (2.13) lead to a system of equations of the form:

$$\overrightarrow{Z}^1(t) = h^1(\overrightarrow{s}(t)), \overrightarrow{Z}^2(t) = h^2(\overrightarrow{S}(t)). \tag{2.15}$$

where $\overrightarrow{S}(t)$ is a vector containing the $3D$ parameters (rotation $\overrightarrow{\Omega}$ translation $\overrightarrow{T}$, rotational velocity $\overrightarrow{\omega}$, translational velocity $\overrightarrow{V}$ and $3D$ points $\overrightarrow{P}^i$). The vector $\overrightarrow{Z}(t)$ contains the image measurements and consists of two parts: $\overrightarrow{Z}^1$ containing the discrete matches and $\overrightarrow{Z}^2$ containing the optical flow.

$$\begin{aligned} \overrightarrow{S}(t) &= [\overrightarrow{\Omega}(t); \overrightarrow{T}(t); \overrightarrow{V}(t); \overrightarrow{\omega}(t); \overrightarrow{P}^1(t); ...; \overrightarrow{P}^{N(t)}(t)] \\ \overrightarrow{Z} &= [\overrightarrow{Z}^1; \overrightarrow{Z}^2] \\ \overrightarrow{Z}^1(t) &= [\overrightarrow{q}^1; ...; \overrightarrow{q}^{N(t)}(t)] \\ \overrightarrow{Z}^2(t) &= [\dot{\overrightarrow{q}}^1; ...; \dot{\overrightarrow{q}}^{K(t)}(t)] \end{aligned} . \tag{2.16}$$

The SFM problem is the inverse problem of determining some or all of the components of $\overrightarrow{S}(t)$ given some or all of the components of $\overrightarrow{Z}(t)$. Solving this system recursively means to determine $\overrightarrow{S}(t)$ satisfying all the measurments from 0 to t ($\overrightarrow{Z}(0 : t)$) by using only the previous estimate $\overrightarrow{S}(t-1)$ and the most recent measurement $\overrightarrow{Z}(t)$. This requires casting the problem as a dynamical system whose state equation given by:

$$\overrightarrow{S}(t+1) = f(\overrightarrow{S}(t)), \tag{2.17}$$

represents the time evolution of the $3D$ parameters.

2.3.8 Structure from Motion Ambiguities

The first ambiguity in structure from motion results directly from Equation (2.12). Assume that $\hat{\underline{R}}$, $\hat{\overrightarrow{T}}$ and $\hat{\overrightarrow{P}}$ represent a solution for this equation. Let α, $\underline{R}_r$ and $\overrightarrow{T}$ be a random number, a random rotation and a random translation respectively, then we can show it can be easily seen that:

$$\hat{\underline{R}}\underline{R}_r^T(\alpha(\underline{R}_r\hat{\overrightarrow{P}} + \overrightarrow{T}_r)) + (-\alpha\hat{\underline{R}}\underline{R}_r^T\overrightarrow{T}_r + \alpha\hat{\overrightarrow{T}}) = \alpha(\hat{\underline{R}}\hat{\overrightarrow{P}} + \hat{\overrightarrow{T}}). \tag{2.18}$$

Therefore, the rotation $\underline{\hat{R}R}_r^T$, the translation $(-\alpha\underline{\hat{R}R}_r^T\overrightarrow{T}_r + \alpha\overrightarrow{\hat{T}})$ and the $3D$ point $(\alpha(\underline{R}_r\overrightarrow{\hat{P}} + \overrightarrow{T}_r))$ represent also a solution for Equation (2.12). This means that the SFM problem can only be solved up to a similarity transform (rotation, translation and scaling). This similarity transform results in a gauge freedom that is often solved, in the two frames case, by assuming the camera corresponding to the first frame to be coincident with the origin and that the translation magnitude between the two frames is one. In the case of multiple frames, this problem is a little bit trickier as the gauge freedom should be fixed to the same values across all the frames.

Another ambiguity in SFM in the case of a planar surface is the existence of a two solutions. Those two solutions are dual meaning that given one set of motion and structure parameters it is possible to derive a second set in terms of the first analytically. If this second solution is then substituted back into the equations specifying the duality, the first solution is obtained. This has been discussed by Barron *et al.* [13] and Subbarao and Waxman [121] in the case of optical flow, and by [131] in the case of discrete correspondences.

There are also other ambiguities that arise in special cases of structure from motion [2, 91, 114]. The most serious of these ambiguities is the "bas-relief" ambiguity which arises when the translation is mainly lateral, the depth difference between the $3D$ points is small and the field of view is small. In this case, a rotation about a direction perpendicular to the lateral translation direction and to the optical axis of the camera, would generate an image motion very similar to the one generated by the lateral translation. In such cases, if the noise in the image motion is statistically equal to or larger than the difference between the translation generated motion and the rotation generated motion, it would be very easy to confuse the two of them and get an inaccurate motion. Those ambiguities are inherent to the SFM problem, meaning that they are algorithm independent and they appear in both discrete and continuous SFM. Therefore, they should be accounted for by any robust algorithm. For example, Soatto and Brockett [114] introduced a method to switch between the minima of the re-projection error function, and discriminating between the true minimum and the one corresponding to the bas-relief ambiguity based on the fact that the true minimum should lead to a reconstruction with positive depth for all the 3D points.

2.4 SFM Estimation: The Standard Techniques

This section reviews the basic techniques and methods used in standard SFM estimation. While some of these techniques apply to the projective reconstruction, we cast them in this review from a Euclidean perspective. With the exception of some approaches that use factorization to determine the solution of a batch of images directly, most non-recursive standard approaches (in both the discrete and continuous settings) start with two or three frames. The first step is generally to eliminate the depth from the constraints to obtain either an algebraic or a geometric constraint from which the motion is determined either linearly or non-linearly. Then the depth can be determined by triangulation. Extension to multiple frames is done mostly by adding frames incrementally using three-points based absolute orientation algorithms. Before describing those techniques, we quickly describe the optimal SFM solution known as Bundle Adjustment (BA).

2.4.1 Bundle Adjustment: The Optimal Solution

Assuming a Gaussian additive noise in the features locations, the optimal solution, in a Maximum Likelihood sense (i.e., the solution maximizing the likelihood of the $3D$ parameters given the image projections) is the solution that minimizes the sum of the re-projection errors for all the features in all the frames. The re-projection error is the distance between the measured image feature location and the projection of the estimated corresponding $3D$ feature using the estimated pose of the camera at the considered frame. The sum of all the re-projection errors is defined as:

$$e^2 = \sum_t \sum_i W^i(t)(e^i(t))^2 = \sum_t \sum_i W^i(t)(\overrightarrow{q}^i(t) - \Pi(\underline{R}(t)\overrightarrow{P}^i + \overrightarrow{T}(t)))^2, \quad (2.19)$$

where $W^i(t)$ is a weight inversely proportional to the variance of the feature if such measure is available from the feature matching process. Unfortunately, e^2 is highly non-linear in the $3D$ parameters and therefore there is no way to determine the solution minimizing it in a closed-form computation. However, starting from an initial solution, the solution minimizing e^2 can obtained by a Levenberg-Marquardt (LM) optimization (See Appendix A). That is why such approach is known as "Bundle Adjustment" (BA) [130] is often used as the last step of reconstruction whenever no time restrictions are to be respected. The computational cost as explained by Triggs *et al.* [130] can be reduced by orders of magnitude by capitalizing on the sparse structure of the Jacobian matrix (See Figure 2.3). This sparse structure is due to the fact that the Jacobians of the image pro-

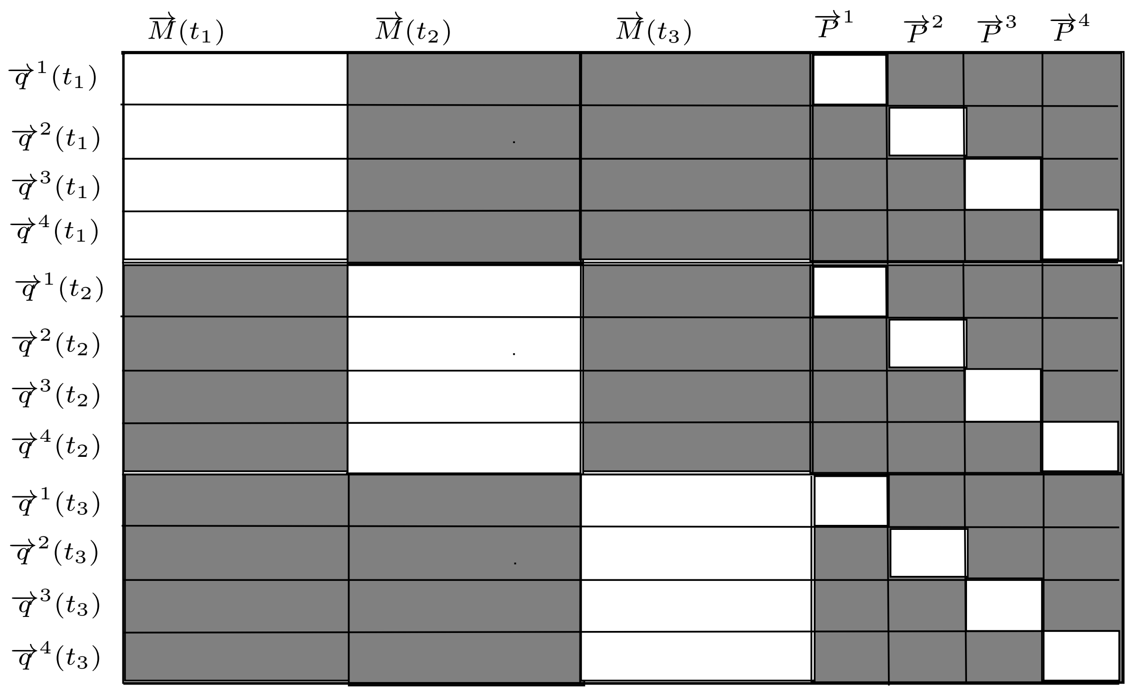


Figure 2.3: Structure of the sparse Jacobian matrix for a BA problem consisting of 3 motions and 4 $3D$ points. The gray entries are all zero.

jection of a $3D$ point with respect to other $3D$ points are zero. Also, the Jacobian of an image projection at time t with respect to the motions at other times are zero. However, even the reduced cost is too expensive to be done in real-time when a large number of frames and $3D$ points are considered given the fact that the complexity is still cubic and the process iterative.

Note that, even if the noise in the features is not Gaussian, minimizing Equation (2.19) results in the Best Linear Unbiased Estimate (BLUE). Nevertheless, since the cost function e^2 has many local minima, BA is very sensitive to the initial solution, and might not reach to the global minima.

The LM minimization of Equation (2.19) can also be used as an analysis tool for SFM as it reveals many of the involved ambiguities. Szeliski and Kang [122] examined the fundamental ambiguities and uncertainties by examining the eigenvectors associated with null or small eigenvalues of the Hessian matrix. They used it to quantify the nature of these ambiguities and predict how they affect the accuracy of the reconstructed shape.

2.4.2 Two-Frames Structure from Motion

Nearly all two frame approaches [37] start by an elimination of the depth of the $3D$ points from the constraints 2.12 or 2.13. The early approaches in both the discrete and continuous setting used an algebraic elimination resulting in a linear constraint in the discrete case and in a bilinear constraint in the continuous case. Those two constraints are as follows:

$$\vec{q}(t_2)^T[\vec{T}]_\times \underline{R}\vec{q}(t_1) = 0, \tag{2.20}$$

for the discrete case and

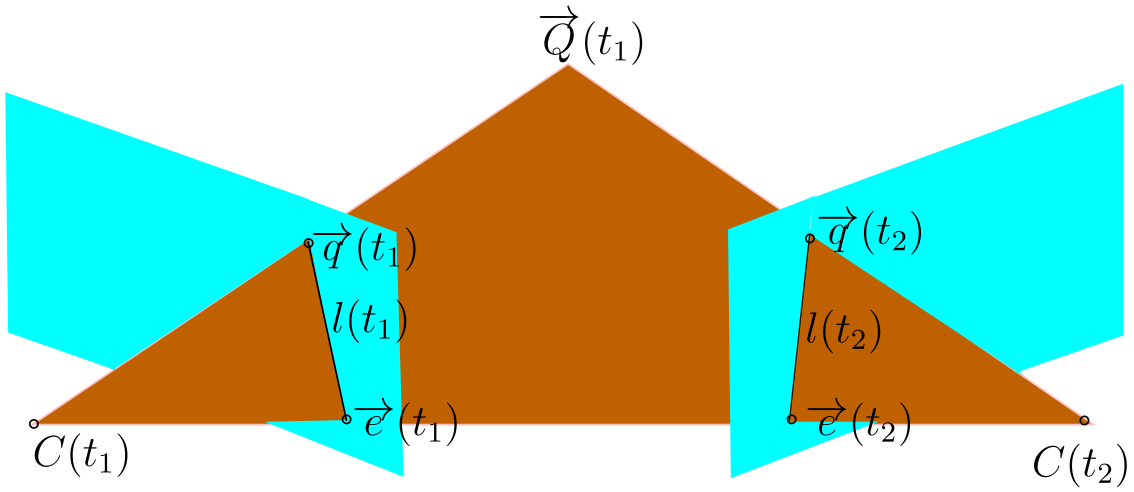


Figure 2.4: Epipolar geometry. A $3D$ point and its projections on two camera planes form what is called the epipolar plane and the intersections of this plane with the two image planes form the epipolar lines. The line connecting the two centers of projection intersects the image planes at the epipoles. Equation (2.20) constrains the projections to lie on the epipolar lines.

$$([\underline{A}(\vec{q})\vec{V}]_2, -[\underline{A}(\vec{q})\vec{V}]_y)(\dot{\vec{q}} - \underline{B}(\vec{q})\vec{\omega}), \tag{2.21}$$

for the continuous case.

The matrix $\underline{E} = [\vec{T}]_\times \underline{R}$ obtained in the discrete case is called the Essential matrix which captures the geometric relation between points matched across two views 2.4. Equation (2.20) allows the determination of the Essential matrix linearly from at least 8

points by rearranging this equation as follows:

$$(q_1(t_2), q_2(t_2), 1) \begin{pmatrix} E_1 & E_2 & E_3 \\ E_4 & E_5 & E_6 \\ E_7 & E_8 & E_9 \end{pmatrix} \begin{pmatrix} q_1(t_1) \\ q_2(t_1) \\ 1 \end{pmatrix} = 0$$

which results in a system of linear equations of the form:

$$\underline{F}(\overrightarrow{q}^i(t_1), \overrightarrow{q}^i(t_2)) \begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \\ E_5 \\ E_6 \\ E_7 \\ E_8 \\ E_9 \end{bmatrix} = 0, \tag{2.22}$$

Minimizing $\sum_i (\overrightarrow{q}^i(t_2)^T \underline{E} \overrightarrow{q}^i)^2$ subject to $||\underline{E}|| = 1$ results in an estimate of the matrix E as the eigen-vector corresponding to the smallest eigen value of $\underline{F}$. This is the 8-points algorithm [37]. However, there are two problems with this estimate. The first problem is that Equation (2.20) represents an algebraic quantity and not a physically significant geometric error. The contribution of a point to this constraint depends on the coordinates of this point, and hence, the resulting estimate is not statistically sound. The second problem is that the obtained essential matrix does not factorize exactly to the form $[\overrightarrow{T}\underline{R}]_\times$. To satisfy this condition, an essential matrix should further satisfy the following conditions:

1. $\underline{E}\overrightarrow{e}(t_1) = 0$ and $\overrightarrow{e}(t_2)^T E = 0$ where $\overrightarrow{e}(t_1)$ and $\overrightarrow{e}(t_2)$ are the epipoles or the images of the camera centers of each camera in the other.
2. $\underline{E}$ is singular
3. $\underline{E}$ has two equal non-zero singular values
4. $\underline{EE}^T\underline{E} - \frac{1}{2}trace(\underline{EE}^T)\underline{E} = 0$
5. $SVD : \underline{E} = \underline{UWV}^T$ where, $\underline{W} = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 0 \end{pmatrix}$ and $\underline{V}, \underline{U} \in SO_3$

The first problem has been dealt with firstly by Richard Hartley [46] who achieved a great improvement using a pre-conditioning of the points by renormalizing them to be centered on the origin. Later on, Phil Torr [126] used the Sampson's approximation [105] to come up with a more geomterically siginificant error. The Sampson's approximation, introduced by Sampson in fitting conics to scattered points gives the geomtric distance to the first order approximation of the conic. For the epipolar error the Sampson approximation provides the geometric distance to the first order approximation of the algebraic variety defined by Equation (2.20). This error is defined as follows:

$$e = \frac{r}{\nabla r} = \frac{r}{\sqrt{(\frac{\partial r}{\partial q_x(t-1)})^2 + (\frac{\partial r}{\partial q_y(t-1)})^2 + (\frac{\partial r}{\partial q_x(t)})^2 + (\frac{\partial r}{\partial q_y(t)})^2}}. \tag{2.23}$$

where $r = \overrightarrow{q}(t)^T E \overrightarrow{q}(t-1)$. However, $\underline{E}$ can not be solved for linearly by minimizing this error. Torr [126] suggested solving it iteratively starting from an initial solution obtained from the linear algorithm.

For the second problem, (i.e., the obtained matrix not being a true essential matrix) Hartley [46] suggested replacing the obtained essential matrix by the closest singular matrix with equal eigen-values. However, the obtained matrix might not be close enough to the original matrix to verify the data. Another approach used to deal with this problem enforces the rank two constraint by using only seven points in Equation (2.22). In this case, the matrix $\underline{F}$ is supposed to have two zero eigen values and hence, two candidates for $\underline{E}$, $\underline{E}^1$ and $\underline{E}^2$, are determined as the eigen vectors corresponding to the smallest two eigen values of $\underline{F}$. The solution is a combination of the two solutions ($\underline{E} = \underline{E}^1 + \lambda \underline{E}^2$). λ can then be determined by forcing the determinant of E to be zero which results in a third degree polynomial in λ. This is the seven points algorithm [37]. Nister [88] provided an algorithm that satisfies all the essential matrix properties and uses only the minimum number required of points (5 points). Nister combined the two equations:

$$\begin{gathered} \overrightarrow{q}(t_2)^T \underline{E} \overrightarrow{q}(t_1) = 0 \\ \underline{E}\underline{E}^T \underline{E} - \frac{1}{2} trace(\underline{E}\underline{E}^T)\underline{E} = 0 \end{gathered},$$

to get a 10^{th} order polynomial equation. Solving this system gives up to 10 essential matrices. The correct essential matrix can be found by testing the resulting matrices against a sixth point.

Usually the 5-points algorithm is used in a RANdom SAmple Consensus (RANSAC) [38] way to provide robustness against outliers. This is carried on as follows:

- Select 5 data items at random
- Estimate the correspondent essential matrix
- Find the number of inliers k. The inliers are those points whose Sampson's error is less than a given threshold.
- If K is big enough, accept and exit.
- Repeat L times
- L is determined based on the expected number of outliers and the desired probability of success

For real time performance, Nister [89] presented an alternative pre-emptive Ransac technique which fixes the number of evaluated hypotheses and compares the hypotheses against each other in a breadth-first manner.

For the continuous case, Equation (2.21) is bilinear in the motion parameters. Bruss and Horn [22] presented an approach based on determining a least-squares estimate of $\overrightarrow{\omega}$ as a function of $\overrightarrow{V}$. Then they substitute $\overrightarrow{\omega}$ back into the bilinear constraints to obtain a non-linear quadratic system which can be solved using Gauss-Newton subject to subject to $|\overrightarrow{V}| = 1$.

Jepson and Heeger [47, 60, 61] proposed a series of subspace methods for estimating egomotion. The basic idea is to construct a set of N dimensional (N is the number of considered image features) constraint vectors $\overrightarrow{\tau}^k$:

$$\overrightarrow{\tau}^k = \sum_{i=1}^{N} c_i^k \overrightarrow{q}^i. \tag{2.24}$$

They presented several methods to determine the vectors $\overrightarrow{c}^k$ from the bilinear constraint in such a way that the vectors $\overrightarrow{\tau}^k$ are orthogonal to $\overrightarrow{V}$ (i.e., $\overrightarrow{\tau}^k \overrightarrow{V} = 0$). For the N points, it can be shown there are $N - 6$ constraint vectors $\overrightarrow{c}^k$. The estimate of $\overrightarrow{V}$ in a least squares sense is the eigenvector corresponding to the smallest eigenvalue of the matrix obtained by summing $\overrightarrow{\tau}\,\overrightarrow{\tau}^T$ over all the $(N - 6)$ vectors $\overrightarrow{\tau}$. The advantage of the linear subspace method is that $\overrightarrow{V}$ is computed directly without requiring iterative numerical optimization. The disadvantage is that this method does not make use of all of the available information ($(N - 6)$ linear constraints versus N bilinear constraints). This approach gained popularity and it was extended to the discrete case by Ponce and Genc [95]. It was also used as the basis of a recursive filter in [118].

However the estimates of this approach, as all least-squares estimates of $\overrightarrow{V}$ based on the algebraic constraints (Equation (2.21)), are systematically biased. The reason is shown in Figure 2.5. This figure shows that the true constraint should be:

$$\frac{([\underline{A}(\overrightarrow{q})\overrightarrow{V}]_2, -[\underline{A}(\overrightarrow{q})\overrightarrow{V}]_1)}{||\underline{A}(\overrightarrow{q})\overrightarrow{V}||}(\dot{\overrightarrow{q}} - \underline{B}(\overrightarrow{q})\overrightarrow{\omega}), \tag{2.25}$$

Hence, the algebraic constraint (Equation (2.21)) is weighted by an extra term $||\underline{A}(\overrightarrow{q})\overrightarrow{V}||$. Kanatani [66] analyzed the statistical bias resulting from the incorrect weighing using a simple Gaussian noise model, then proposed a method (called the renormalization method) that removes the bias by automatically compensating for the unknown noise. Also Heeger and Jepson [47] tried to deal with the bias by adding extra noise to the vectors $\overrightarrow{\tau}$ in a effort to make the noise in those vectors isotropic.

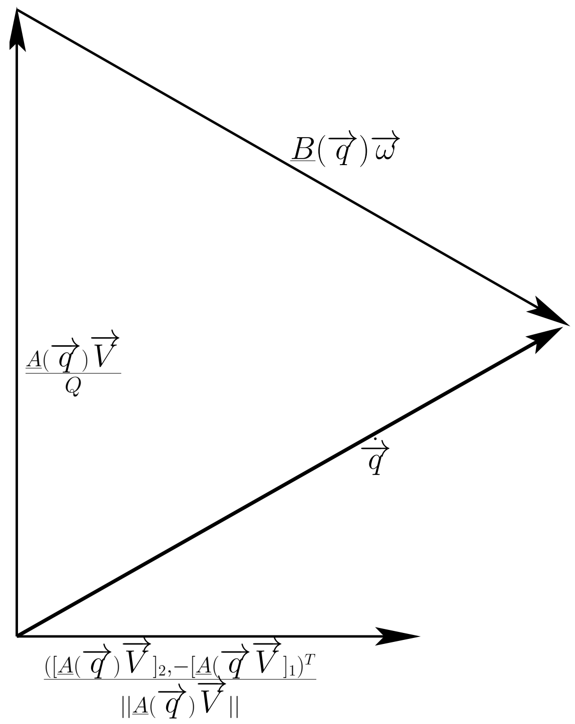

$\underline{B}(\overrightarrow{q})\overrightarrow{\omega}$
$\frac{\underline{A}(\overrightarrow{q})\overrightarrow{V}}{Q}$
$\dot{\overrightarrow{q}}$
$\frac{([\underline{A}(\overrightarrow{q})\overrightarrow{V}]_2, -[\underline{A}(\overrightarrow{q}\overrightarrow{V}]_1)^T}{||\underline{A}(\overrightarrow{q})\overrightarrow{V}||}$

Figure 2.5: Components of optical flow: The dot product of the vector $\vec{\tau}$ with the difference of the measured flow and the rotational flow should ideally be zero (Recreated from [93]).

Zhang and Tomasi [135] presented an iterative approach based on the geometric constaint (Equation (2.25)). Their approach capitalizes on the fact that this constraint is separable, in the sense that once $\overrightarrow{V}$ is known, $\overrightarrow{\omega}$ can be determined linearly using least squares. Therefore they start from an initial estimate of $\overrightarrow{V}$ to proceed with a Gauss-

Newton minimization of the differential constraint (Equation (2.13)) where each iteration is carried out as follows:

- Determine a new $\overrightarrow{V}$, $\overrightarrow{\omega}$ and Q_z of every point using a regular Gauss-Newton iteration.
- Re-determine $\overrightarrow{\omega}$ from the geometric constraint using $\overrightarrow{V}$.
- Re-determine the Q_Z from the full instantaneous constraint (Equation (2.13)).

Because of the high non-linearity of the geometric constraint, such an approach is prone to falling in one of the many local minima. To avoid this Zhang and Tomasi suggested initializing $\overrightarrow{V}$ from 15 different locations spread out on the unit sphere. As the algebraic constraint is much less non-linear, Pauwels and Van-Hulle [93] proposed a hybrid constraint parameterized with ρ:

$$(||A(\overrightarrow{q})\overrightarrow{V}||^{(1-\rho)}\overrightarrow{\tau}^T(\dot{\overrightarrow{q}} - B(\overrightarrow{q})\overrightarrow{\omega}))^2 \tag{2.26}$$

When $\rho = 0$ this constraint is equivalent to the algebraic constraint with very few local minima (only SFM inherent ones). As they proceed with iterations similar to Zhang and Tomasi, they increase ρ gradually. When in the vicinity of the optimal solution ρ would be equal to 1 and the constraint would be equal to the geometric constraint. This way they could reach the global minima by starting from only one random initial $\overrightarrow{V}$ which resulted in a significant speed-up over Zhang and Tomasi approaches.

2.4.3 Motion from Essential Matrix

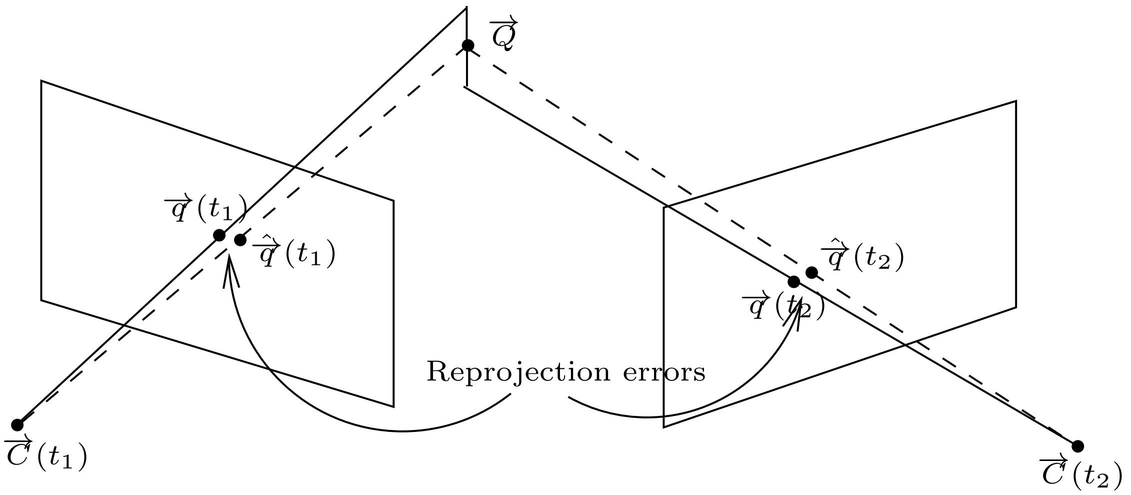
To recover the translation and rotation from the essential matrix, the most widely used approach [46] is to use an SVD decomposition of $\underline{E}$ as $\underline{UWV}^T$ which gives rise to four possible solutions:

- $[\overrightarrow{T}^1]_\times = \underline{UR}_z(\frac{\pi}{2})\underline{WU}^T$
- $[\overrightarrow{T}^2]_\times = \underline{UR}_z(-\frac{\pi}{2})\underline{WU}^T$
- $\underline{R}^1 = \underline{UR}_z(\frac{\pi}{2})\underline{V}^T$
- $\underline{R}^2 = \underline{UR}_z(-\frac{\pi}{2})\underline{V}^T$

where $\underline{R}_z(\frac{\pi}{2}) = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

2.4.4 Triangulation

After the rotation and translation are determined, the $3D$ points can be determined by triangulation (Figure 2.6). A linear approach for triangulation can be obtained by com-



$\overrightarrow{Q}$
$\overrightarrow{q}(t_1)$
$\hat{\overrightarrow{q}}(t_1)$
$\hat{\overrightarrow{q}}(t_2)$
$\overrightarrow{q}(t_2)$
Reprojection errors
$\overrightarrow{C}(t_1)$
$\overrightarrow{C}(t_2)$

Figure 2.6: Triangulation: due to noise, the rays through the image projections do not intersect in one physical point. The optimal solution is the $3D$ point which minimizes the re-projection errors in both images.

bining the projection equations for both views to get a linear system of equations. This solution can then be refined by iteratively minimizing the sum of re-projection errors $\sum(\overrightarrow{q}(t_1) - \hat{\overrightarrow{q}}(t_1))^2 + (\overrightarrow{q}(t_2) - \hat{\overrightarrow{q}}(t_2))$, where $\hat{\overrightarrow{q}}(t_1)$ and $\hat{\overrightarrow{q}}(t_2)$ are the projections of the estimated $3D$ point on the images. Hartley and Sturn [45] describe a non-iterative solution that finds the optimal triangulation through solving a sixth order polynomial. Recently, Oliensis [92] introduced an approach to reach the optimal solution in a closed form which just requires computing square roots.

2.4.5 Absolute Orientation

The absolute orientation problem is to determine the unknown pose of a camera given a set of known $3D$ points and their projection on the image frame of the camera. A linear solution to this problem results directly from the discrete constraint (Equation (2.12)) using a minimum of six points. However, this solution is far from optimal. The optimal solution, can be determined from three points using geometric reconstruction. A number of approaches provided such solution and were summarized and compared by Haralick et. al [43]. The absolute orientation is an essential task in structure from motion and is used frequently in sequential approaches in a RANSAC style to determine a new motion from a previously determined structure.

2.4.6 The Trifocal Tensor

Similar to the essential matrix, the trifocal tensor captures the multilinear relationships between the three images. In this thesis, to avoid further complication of the notation, we do not use the tensorial notation and represent the trifocal tensor $\mathcal{T}$ as a set of three matrices $\underline{\mathcal{T}}_i$, $i \in \{1, 2, 3\}$. The geometry of three views is shown in Figure 2.7. Corresponding

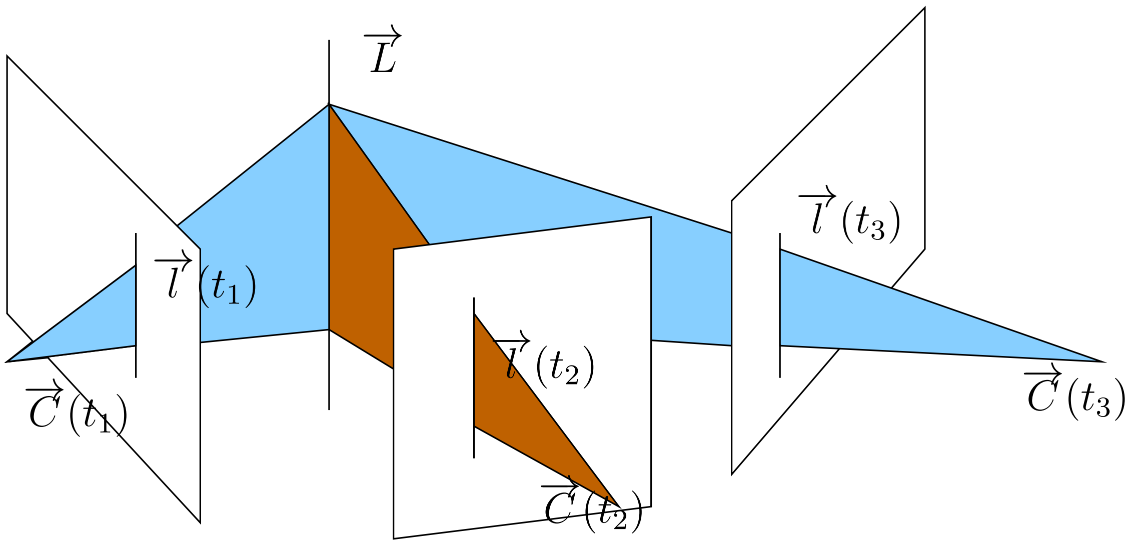


Figure 2.7: The trifocal geometry and the three lines correspondence. This correspondence results in algebraic constraints between the lines in the three frames.

lines back-projected from the first, second and third images all intersect in a single 3D line in space. This geometric constraint can be translated into an algebraic constraint on the three lines as follows:

$$l_i(t_1) = \vec{l}(t_2)^T \underline{\mathcal{T}}_i \vec{l}(t_3) \tag{2.27}$$

The matrices of the trifocal tensor are defined as follows:

$$\underline{\mathcal{T}}_i = \vec{R}^i(t_1, t_2)\vec{T}(t_2, t_3)^T - \vec{T}(t_1, t_2)\vec{R}^i(t_2, t_3)^T. \tag{2.28}$$

Equation (2.27) can be manipulated to obtain the following formula involving the projections $\overrightarrow{q}(t_1)$, $\overrightarrow{q}(t_2)$ and $\overrightarrow{q}(t_3)$ of the same world point:

$$[\overrightarrow{q}(t_1)]_\times(\sum_i q_i(t_1)\underline{\mathcal{T}}_i)[\overrightarrow{q}(t_3)]_\times = \underline{0}, \tag{2.29}$$

which is known as the point-point-point incidence equation [46]. Another constraint involving two image projections in the first and third views and a line passing by the third image projection in the second view known as the point-line-point incidence equation is expressed as follows:

$$\overrightarrow{l}(t_2)(\sum_i q_i(t_1)\underline{\mathcal{T}}_i)[\overrightarrow{q}(t_3)]_\times. \tag{2.30}$$

Having 6 point matches across the three views, the trifocal tensor can be determined linearly. As in the case of the essential matrix, more accurate results can be obtained by iteratively minimizing the Sampson's approximation of the trifocal error.

2.4.7 Factorization Approaches

Factorization methods are approches that aim to find batch solutions for the SFM problem using matrices decomposition. They were originally introduced by Tomasi and Kanade [125] for the orthographic projection case (Figure 2.8). The SFM constraint

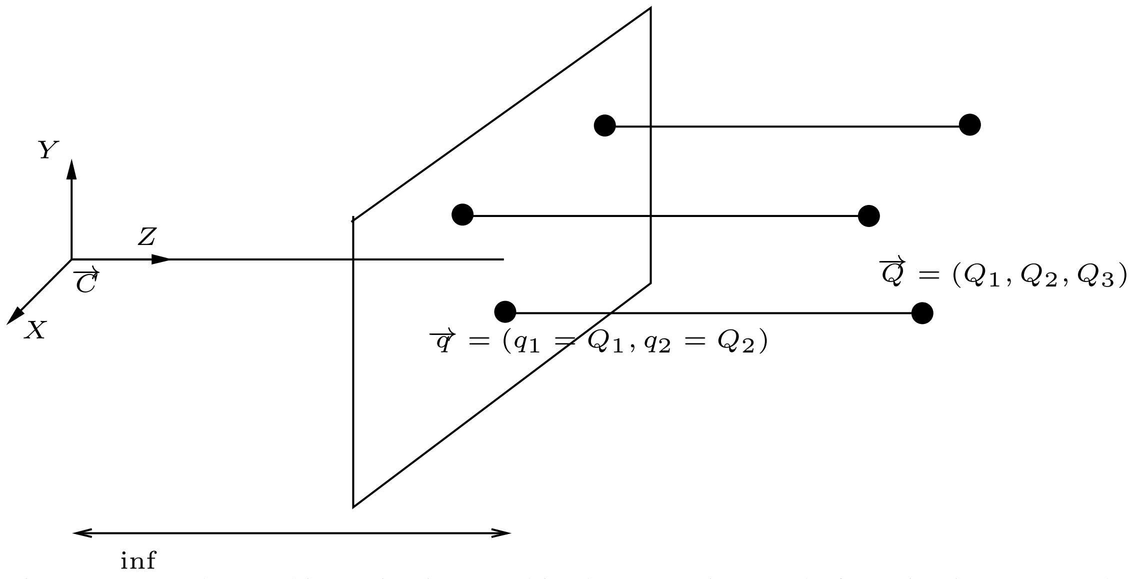

Y
Z
$\overrightarrow{C}$
X
$\overrightarrow{Q} = (Q_1, Q_2, Q_3)$
$\overrightarrow{q} = (q_1 = Q_1, q_2 = Q_2)$
inf

Figure 2.8: Orthographic projection used in the Tomasi-Kanade factorization approach: the projection of a $3D$ point has the same x and y coordinates as the $3D$ point itself.

in this case is :

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} [\underline{R}\overrightarrow{P} + \overrightarrow{T}]_1 \\ \underline{R}\overrightarrow{P} + \overrightarrow{T}]_2 \end{bmatrix} = \underline{A}\overrightarrow{P} + \overrightarrow{b} \tag{2.31}$$

where A is 2×3 and $\overrightarrow{b}$ is 2×1. Having M cameras $(A_j, \overrightarrow{b}_j)$ and N points $(\overrightarrow{P}_i)$ with projection $\overrightarrow{q}_{ij}$ on the j^{th} camera. Taking one of the points (or their center of mass) as the origin gets rid of the translation vectors:

$$\underline{A}_j \overrightarrow{P}_i + \overrightarrow{b}_j \Leftrightarrow \underline{A}_j \overrightarrow{P}_i \tag{2.32}$$

Then the 3D paramters are stacked in the matrices $\underline{M}$ and $\underline{S}$ and the $2D$ projections are stacked in the matrix $\underline{m}$:

$$\begin{gathered} \underline{m} \overset{\text{def}}{=} \begin{bmatrix} \overrightarrow{q}_{11} & \dots & \overrightarrow{q}_{N1} \\ . & . & . \\ . & . & . \\ . & . & . \\ \overrightarrow{q}_{1M} & \dots & \overrightarrow{q}_{NM} \end{bmatrix} \\ \underline{S} \overset{\text{def}}{=} \begin{bmatrix} \overrightarrow{P}^1 & \dots & \overrightarrow{P}^N \end{bmatrix} \\ \underline{M} \overset{\text{def}}{=} \begin{bmatrix} A_1 \\ . \\ . \\ A_M \end{bmatrix} \end{gathered} \tag{2.33}$$

Now the orthographic projection on all the images can be written as $\underline{m} = \underline{M} \times \underline{S}$. $\underline{m}$ at most rank 3. Therefore the recovery of structure and motion reduces to the decomposition of $\underline{m}$ into the product of $2M \times 3$ and $3 \times 3N$ matrices. An SVD decomposition allows to recover those matrices. In the prespective case, the unknown depths of the $3D$ features would figure in the matrix $\underline{m}$. A solution for this proposed by Sturn and Triggs [128] is to start from an initial value of the depths and reiterate. However, such approaches being batch and iterative are not suitable for real time online applications.

2.4.8 Filter-Based SFM

This section provides a historical perspective of the filter based SFM. The state-of-the-art approaches will be provided in Section 2.5. Filter-based approaches rely on Bayesian filtering techniques to determine the probability distributions of the $3D$ parameters given all the measurements up to the current time. The earliest attempts at casting the structure from motion problem as a dynamical system were the ones of Broida and Chellappa [21] who provided a recursive formulation for the case of one dimensional images of two dimensional objects, and of Heel [48, 49] who provided several formulations ranging from direct depth, to discrete and continuous cases. Later on, Broida and Chellapa [20] provided a recursive filter for the 3D case of sparse features. Azarbayejani [7] extended this method to estimate the focal length along with the structure and motion.

Another set of approaches uses optical flow to determine the instantaneous motion of the camera and the depth in the reference frame of the camera. A number of these approaches [14,59,81] focused on the estimation of dense depth from optical flow assuming a known camera motion, followed or simutlaneosuly accompanied by a surface regularisation. Xiong and Shafer [133] presented an approach to determine the depth of every pixel and the velocity of the camera from a dense optical flow field. The Levenberg-Marquardt method is used to perform nonlinear minimization to get an initial estimate of the motion and depth parameters at every time step. These estimates are then filtered with the previous estimates using an EKF. To alleviate the computational cost of the EKF they decompose the depth uncertainty into and independent part and a correlated part which is only of rank six because of the dependence on a single six dimensional motion. Barron and Eagleson [12] introduced a method for the determination of structure and motion from optical flow based on the assumption of constant rotational acceleration. Then by separating the translation, rotation and depth over three frames, they get linear equations which can be solved for all the estimates. Then, they use Kalman Filtering to integrate the estimates over time. Also, within this category is the system of Soatto [118] based on an Extended Kalman Filter with the subspace constraint of Jepson and Heeger [60] as measurement equation. The problem with those optical flow based methods is that dense optical flow is hard to compute reliably in general situations. Also, they only determine the velocity of the camera and not its absolute pose with respect to a fixed reference frame.

Soatto [115] presented a set of filters to determine the motion only using the epipolar constraint. The Implicit Extended Kalman Filter is used to deal with the implicit measurement equation. The problem of this approach is that it only determines the direction of translation and not its magnitude. Also, it uses the algebraic epipolar constraint and not the more accurate Sampson's approximation of the geometric error (Equation (2.23)).

2.5 Sequential SFM: State of the Art

This section describes the state of the art in online sequential SFM. The research in this field can be classified as odometry-based, filter-based, and key frames based approaches. In contrast to the optimal offline BA which uses all the eventual features in all the frames of the sequence, the online approaches employ different mechanisms to reduce the amount of data processed to be able to maintain an online performance. The differences between the mechanisms in each of the three categories, shown in the network of Figure 2.9, are as follows:

- Filter based approaches reduce the complexity by relying on the Markovian assumption which states that the information from all the previous history of measurements can be accessed through the last previous estimate.
- Odometry based approaches tackle the problem by attempting to use the information from a large amount of features matched in the last few frames only.
- Key frame based approaches try to eliminate redundant information by selecting the most representative set of key frames spread out over the whole sequence.

In the following we review the state-of-the-art approaches in each category outlining the strengths and weaknesses. We show that the best approach would have elements of the three categories and show how our approach fits within this big picture. We also provide some results of the state-of-the-art- approaches versus bundle adjustement and/or the EKF to serve as a baseline for evaluating the performance of our approaches.

2.5.1 Odometry Based Approaches

Visual odometry approaches [71, 75, 89, 94, 136] operate by incrementally updating the motion between frames by using an absolute orientation algorithm (as descibed in Section 2.4.5) followed by a non-linear optimization of the motion. An extra local bundle adjustment step over a small window of frames is often done subsequently. Those approaches work well on large sequences, typically with cameras mounted on vehicles. However, since they integrate information only from a small number of frames they are optimal only locally. The best results in this category are achieved by the approaches of Mouragnon and Lhuillier [87], Nister [90] and Konolige and Agrawal [71] which are discussed hereafter.

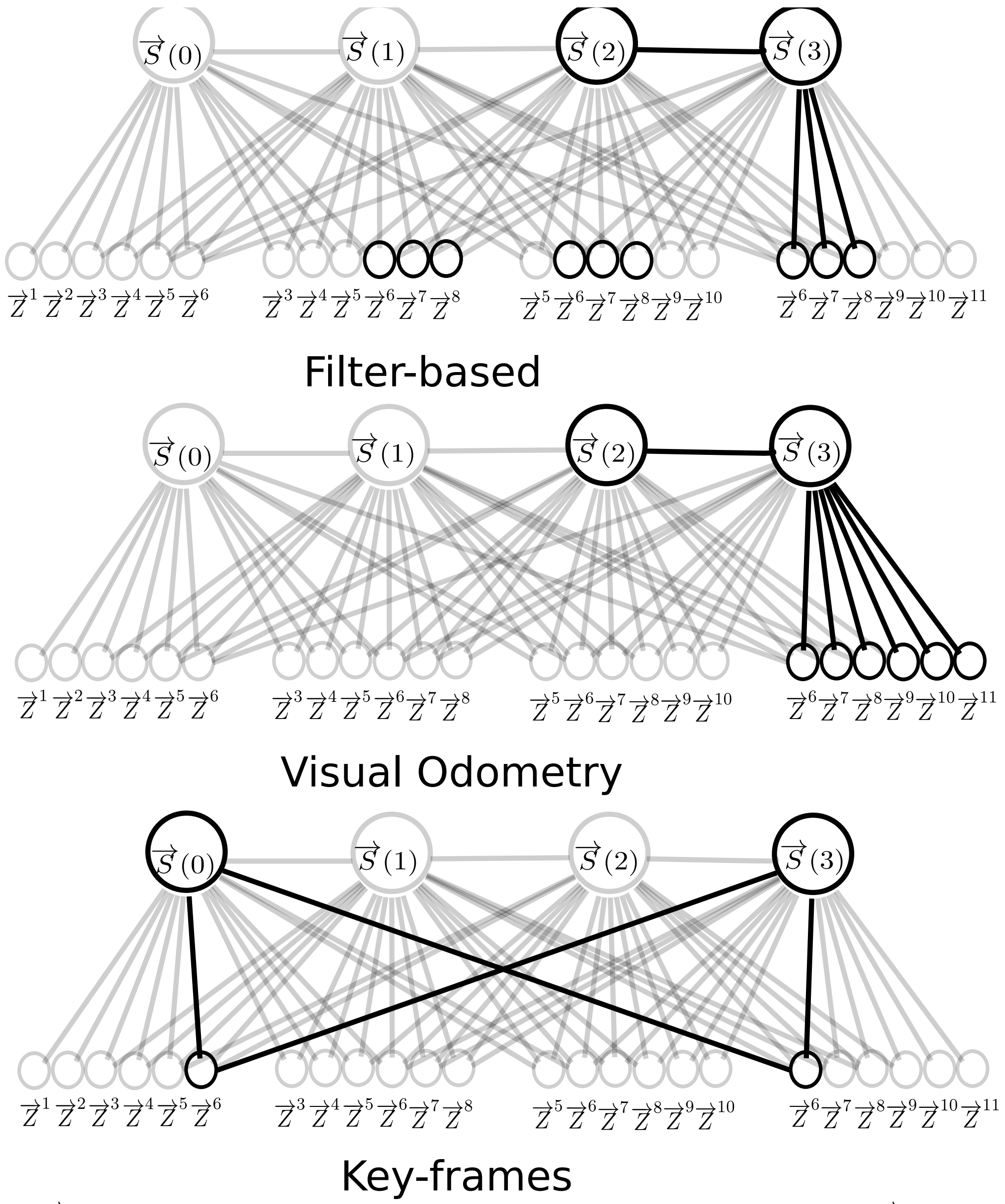

$\vec{S}(0)$ $\vec{S}(1)$ $\vec{S}(2)$ $\vec{S}(3)$
$\vec{Z}^1 \vec{Z}^2 \vec{Z}^3 \vec{Z}^4 \vec{Z}^5 \vec{Z}^6$ $\vec{Z}^3 \vec{Z}^4 \vec{Z}^5 \vec{Z}^6 \vec{Z}^7 \vec{Z}^8$ $\vec{Z}^5 \vec{Z}^6 \vec{Z}^7 \vec{Z}^8 \vec{Z}^9 \vec{Z}^{10}$ $\vec{Z}^6 \vec{Z}^7 \vec{Z}^8 \vec{Z}^9 \vec{Z}^{10} \vec{Z}^{11}$
Filter-based
Visual Odometry
Key-frames

Figure 2.9: $\vec{S}(t)$ refers to the state vector (motion+structure) at time t, and the $\vec{Z}^i$s represent the features observed at every frame. (a) Filter based approaches use the Markovian property to harness the information from previous measurements. Only a limited number of measurements can be used due to the computational cost. (b) Odometry based approaches use information from a large number of features in the last frames only. (c) Key frame based approaches use information from a few number of selected frames in the sequence.

Mouragnon and Lhuillier

Mouragnon and Lhuillier [87] presented an approach based on local bundle adjustment. This approach recovers motion and structure using only selected key frames. A key frame is chosen if it is the farthest possible from the previous key frame and yet shares a minimum numbers of features (400 and 300) with the previous two key frames. As shown in Figure 2.10, whenever a key frame C_i is selected, the new motion is determined using an absolute orientation algorithm [43] and RANSAC. Then, this motion and the features shared between the last frame and the $N-1$ ($N=20$) frames are optimized using non-linear minimization. Lastly, a local bundle adjustment is done for all the $3D$ parameters in the last N frames. Results were reported on an outdoor sequence and compared with GPS measurements. A mean error of 41 cm and a maximum error of 2 meters relative to the GPS measurements have been achieved.

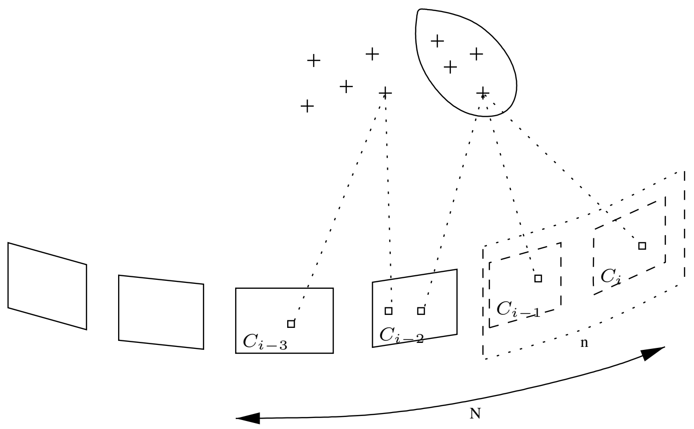


Figure 2.10: Mouragnon and Lhuillier local bundle adjustment approach (Figure from [87]). In a first step the a non-linear optimization refines only the motion across the last n frames. Then, the motion and structure parameters are refined over the last N frames.

Nister's visual odometry approach

Nister [90] presented a visual odometry system for large scale navigation. The system uses Harris features matched using normalized cross correlation and then estimates structure and motion incrementally using the following two main steps:

1. Track features over a certain number of frames. Estimate the relative poses between three of the frames using the 5-point algorithm and preemptive RANSAC followed by iterative refinement. Triangulate the observed feature tracks into 3D points using the first and last observation on each track and the optimal triangulation of Oliensis [92]. If this is not the first time through the loop, estimate the scale factor between the present reconstruction and the previous camera trajectory with another preemptive RANSAC procedure. Put the present reconstruction in the coordinate system of the previous one.

2. Track for a certain additional number of frames. Compute the pose of the camera with respect to the known 3D points using a 3-point algorithm absolute orientation algorithm and preemptive RANSAC followed by iterative refinement. Re-triangulate the 3D points using the first and last observations on their image track

The second step is repeated several times before the first step is repeated again. This system has been tested in a real life experiment and good results with comparison to GPS (less than 2% error in the trajectory) were reported. Though two schemes are proposed, a monocular one and a stereo one, results were reported only from the stereo scheme.

Konolige and Agrawal

This system [71] is similar to Mouragnon system with the following differences: Stereo images are used instead of monocular images, there is no selection of key frames instead all frames are processed, IMU and wheel encoders measurements are used whenever vision fails (while incrementing the motion using absolute orientation). Also, their local bundle adjustment interleaves the computation of structure and motion for every iteration in the main loop which results in faster convergence. The bundle adjustment is done over a window of 5 frames. Obviously, this system involving stereo, IMU and wheel encoders provided more accurate results than vision-only monocular approaches (less than 1% trajectory error).

2.5.2 Discussion

The odometry based approaches have been designed specifically for vehicle mounted camera, where the motion is somehow constrained (mostly forward with not much rotation between consecutive frames). Similar approaches applied to the case of hand held cameras where the motion might be random would face serious problems. We illustrate this using some results from knolidge approach: Figure 2.11 shows the trajectories estimated by their approach versus GPS. The trajecotry of interest is the once labeled SBA. Notice that the trajectory experiences serious drift due especially to rotations. If the motion was totally free in three dimensions the effect of rotations would have been more serious. Neverthless, the success of those approaches in tracking for large distances, shows the importance of using large number of local features. This constitutes a motivation for our approach in chapter 4 integrating frame-to-frame features with filters based on features tracked over a large number of frames.

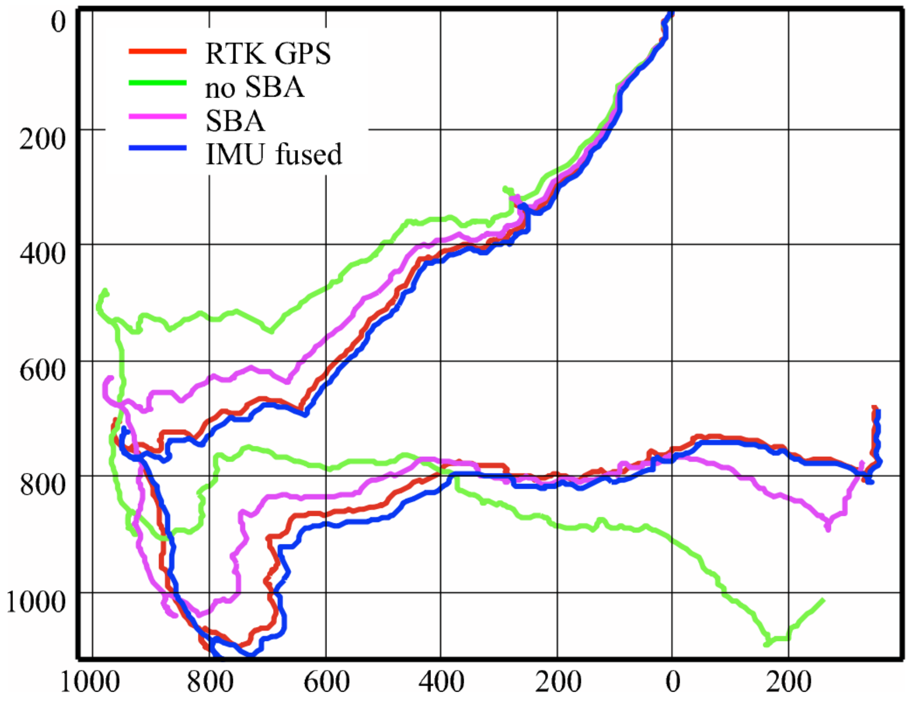


Figure 2.11: Konolige results on a large sequence. The line of interest is the one labeled sba. Notice the increase in drift due to rotations. (Figure from [71].

2.5.3 Filter-Based Recursive Structure and Motion Estimation

Extended Kalman Filter and Variations

The new generation of EKF-SFM approaches has been pioneered by the work by Chiuso *et al.* [25, 27] based on the results of a series of researches by Soatto and Perona [113, 116,117] who studied the observability, feasibility and linearization of recursive structure from motion. The approach of Chiuso *et al.* differs from earlier approaches (mentioned in the previous section) in three main aspects: (1) The earlier approaches used only one parameter per feature (depth of the feature) to represent the structure. This as explained in [27] is sub-minimal and results in an incorrect weighing of the measurements (The first set of measurement would have a weight of infinity). The correct way to have a minimal filter is to include all the coordinates of every feature in the state vector. (2) The other major improvement over earlier approaches is using structure parameters to fix the scale ambiguity across different frames (in contrast to using the translation magnitude in earlier approaches). This limits the drift due to scale ambiguity because it would only happen when the features used to fix the scale disappear and have to be replaced by other features. (3) The use of independent EKFs to initialize new features before inserting them in the main filter upon convergence. The dynamical system formulated by Chisuo *et al.* (and which is similar to most state of the art approaches) involves a state vector that contains a full representations of the $3D$ structure in addition to the absolute motion and the incremental motion from the previous frame (velocity) which drives the system. The state equations can be written as:

$$\begin{cases} \overrightarrow{P}^i(t+1) & = & \overrightarrow{P}^i(t) \\ \overrightarrow{T}(t+1) & = & \mathbb{R}^3toSO_3(\overrightarrow{\omega}(t+1))\overrightarrow{T}(t) + \overrightarrow{V}(t) \\ \underline{R}(t+1) & = & \mathbb{R}^3toSO_3(\overrightarrow{\omega}(t+1))\underline{R}(t) \\ \overrightarrow{V}(t+1) & = & \overrightarrow{V}(t) + \overrightarrow{a}_V(t) \\ \overrightarrow{\omega}(t+1) & = & \overrightarrow{\omega}(t) + \overrightarrow{a}_\omega(t) \end{cases} \tag{2.34}$$

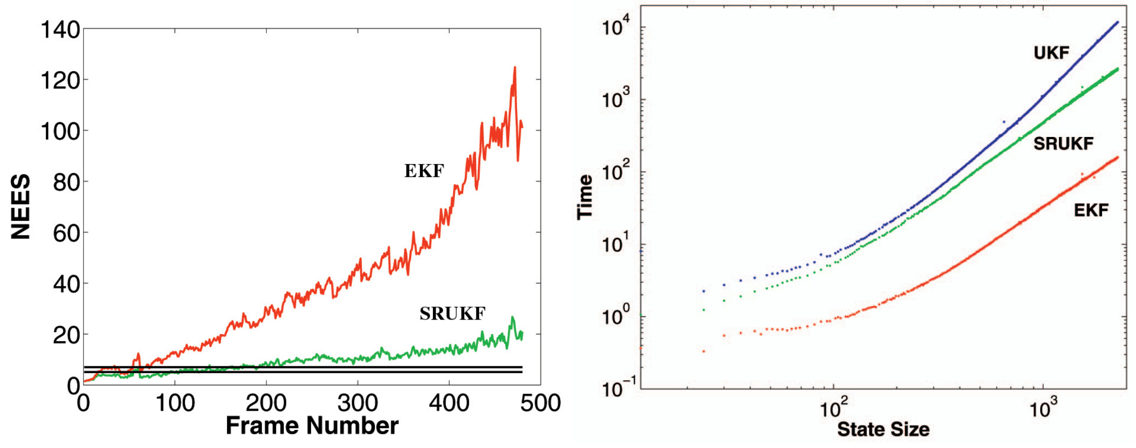
And the measurement equation:

$$\overrightarrow{q}^i(t) = \Pi(\underline{R}(t)\overrightarrow{P}^i(t) + \overrightarrow{T}(t)) \tag{2.35}$$

Davison [29] extended this EKF system to SLAM and presented one of the most convincing real-time visual systems. The main differences with the system of Chiuso is that it uses active search for the features that are already in the map, and instead of using an EKF to initialize new features, it uses a particle filter because the depth distribution at the beginning is not Gaussian. Upon convergence of the new feature it is added to the main EKF. Davison *et al.* [30] recently provided an other version of this system with more results from a camera fixed on a humanoid robot. However, these EKF filters present two major problems. First, the covariance matrices are quadratic in the number of the features, and hence updating them requires a time cubic in the number of observed features and quadratic in the total number of features in the map. This hinders their use in real time applications involving a large number of features. The second problem is related to the fact that EKFs assume that the distributions of the measurements are Gaussian which is not always true. Also, the first order linearization results in the inconsistency of the filter. An inconsistent filter is characterized by a covariance that is smaller than the true covariance of the estimates given the covariance in the noise. Inconsistency might lead to the divergence of the filter. The non-linearity in the projection equation is independent of the choice of coordinate system. Soatto and Perona [117] have proved that all choices of coordinates representation are structurally equivalent and none has an advantage based on geometric properties, instead the difference is based on computational numerical ground.

Many variations of the EKF systems have been presented, for example, using inverse depth representations by Civera *et al.* [28] which exhibit better Gaussian properties and enable the initialization of new features from within the filter. However this increases the computational cost since it represents each feature with 6 parameters. Recently, Huang *et al.* [57] provided an approach to deal with the inconsistency of the filter due to linearization and suggested to do the linearization always about the first obtained estimate. They showed this to provide more consistent and accurate results. However, the results of their system dubbed First Estimate Jacobian-EKF (FEJ-EKF) have been performed on Laser data and not on visual imagery.

To deal with the issues of non-linearity, the Unscented Kalman Filter was used for visual SLAM (Chekhlov *et al.* [23]). However the UKF is much more expensive computationally than the EKF. Holmes *et al.* [54, 55] introduced a Square Root Unscented Kalman Filter (SRUKF) for visual slam, with a reduced complexity. This filter achieved better accuracy than the EKF albeit it was still an order of magnitude more expensive than the EKF. Figure 2.12 shows the consistency and computational costs of the SRUKF with respect to the EKF.



(a) Consistency (b) Computational cost

Figure 2.12: (a) Number of normalized estimation error squared NEES (A perfectly consistent filter would have the number of NEES between the two horizontal lines) for the EKF and SRUKF. (b) Log-log plots of the total times (in seconds) taken for the core Kalman stages of the UKF, SRUKF, and EKF versus the size of the state. (Figures taken from [55]).

Within the context of general SLAM (not visual SLAM) the Extended Information Filter (EIF) has been used in some approaches. The EIF relies on the information matrix (inverse of the covariance matrix) instead of the covariance matrix directly. The advantage of the information form is that for large-scale maps the majority of the off-diagonal components of the normalised information matrix are very close to zero. Thrun *et al.* [123] exploited this fact and developed a sparsification procedure in which the elements of the normalised information matrix that were close to zero were actually set to zero. This forces the information matrix to be sparse, which saves a significant amount of memory and enables the application of very efficient sparse update procedures for information estimates [14]. While this method results in a considerable speedup over the EKF with a relatively small drop in accuracy, it was shown to be inconsistent by Bailey and Durrant-whyte [9].

The concept of information-form SLAM has been used also in general SLAM (not visual) in the constext of smoothing. Dellaert *et al.* [31] introduced the Square Root Smoothing and Mapping (SRSAM) that achieved the exact sparsification of the information matrix by augmenting the state with the new vehicle pose estimate during each update and not removing any of the past poses. This resulted in a system that is similar to bundle adjustment in SFM or visual SLAM. As a result of keeping the whole history of poses, the off-diagonal terms of the matrix are non-zero only for poses and landmarks which are directly related by observations. A similar system has been presented by Kaess *et al.* [65] called ISAM (Incremental Smoothing And Mapping) with the ability to run incrementally online. In the linear case solving the whole system reduces to solving a linear least square system, With the special structure of the system and QR factorization this can be done more efficiently. The QR factorization is done using Givens rotations, a procedure that can be performed incrementally and is the key to the online operation of the system. Good performance has been reported on Laser data (qualitative results have been provided). However those approaches require linear system and measurement equations. In the non-linear case Kaess *et al.* proposed a method based on first order linearization and on the assumption that the measurements are pretty accurate. It is not clear how this method would perform in the case of the highly non-linear and noisy visual measurements case. Also, one problem of information matrix systems is that they do not provide directly the covariance matrix. The covariance matrix is needed sometimes at every time step if one is willing to perform, for example, a search for the features in the image based on their prediction by the state vector (gated search). Kaess *et al.* provided a way to extract the covariance of a feature from the information matrix without having to invert the full covariance matrix. However, whenever the whole covariance matrix is required a direct inversion is still the only solution.

Particle Filters

Particle filters [72,96,98] have been introduced to the SFM problem in order to solve the issue of robustness to noise and ambiguities by keeping a large number of hypotheses. One of the first implementations of Particle filters is the approach of Qian and Chellapa [98] who described a method based on Sequential Importance Sampling (SIS). Their approach uses partitioning to reduce the dimensions of the state space, separating the estimation of motion from that of the depth using the epipolar constraint. However this method is too computationally expensive to be run in real time. Pupilli and Calway [96] presented an approach to track a camera using particle filtering. Their method runs in real time using features that are triangulated from all the particles. however the features are not filtered over many frames and it can only accommodate a very few number of features(less than 10). Pupilli and Calway later on [97] added auxiliary UKF filters, one per every frame in which new features are initialized. Each auxiliary UKF estimates the features initialized in the frame and the motion of the camera. The output of the tracking Particle Filter is used as prediction for the UKFs. The problem of this filter is that the correlation between thte features initialized in different frames are not taken into consideration.

Rao-BlackWellized particle filters have been introduced to SLAM by Montemerlo *et al.* [84] and then used by Sim *et al.* [109] for visual SLAM. They reduced the computational complexity by capitalizing on the important characteristic of SFM and SLAM that features are independent given the camera motion. Then, a particle filter can be used in which each particle (Figure 2.13) is composed of a motion sample and N Gaussian representations of each of the $3D$ particles conditioned on the motion sample. Each Gaussian representation consists of the mean position vector of the $3D$ point and its covariance matrix. Therefore, the required number of particles should be only large enough to represent six-dimensional entities (motion). The $3D$ features, in every particles, are estimated using individual low dimensional EKF filters conditionally on the motion. This resulted in a significant reduction of the number of particles.

However, those approaches based on FastSLAM have a serious problem in that the measurements contribute to the estimation of the motion only through weighing. Therefore, a large number of samples is needed to guarantee convergence. A drastic improvement over the simple Rao-BlackWellized particle filter was in using the optimal importance function (which is the distribution of the estimates given not only the state at the previous time-step but also the most recent observation) in the FastSLAM2.0 system by Montemerlo *et al.* [85]. This system was then extended to visual SLAM by Eade and

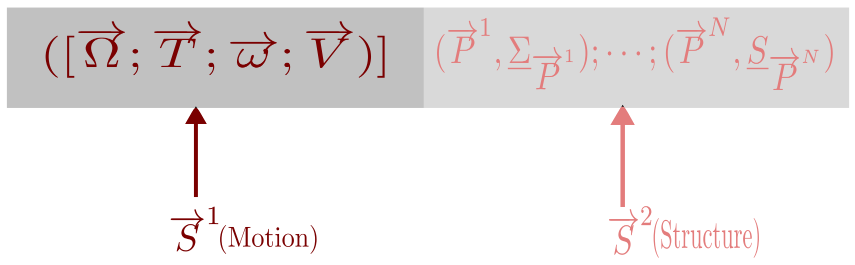

$([\overrightarrow{\Omega};\overrightarrow{T};\overrightarrow{\omega};\overrightarrow{V})]$
$(\overrightarrow{P}^1,\Sigma_{\overrightarrow{P}^1});\cdots;(\overrightarrow{P}^N,S_{\overrightarrow{P}^N})$
$\overrightarrow{S}^1$(Motion)
$\overrightarrow{S}^2$(Structure)

Figure 2.13: RBPF particle structure. It consists of a motion sample, and the Gaussian representations (mean and covariance) of every $3D$ point conditioned on the motion sample.

Drummond [33] and Sim *et al.* [108]. This ensures that samples are propagated in an efficient way which reduces the number of wasted samples. However, the way samples are drawn from this importance function in the mentioned approaches is not optimal in the sense of providing particles that lie in the regions of highest probability of the posterior. This is because they do not use the uncertainty of a motion sample during the update process and since they update the features locations based on the predicted motion which might not be very accurate if the motion is changing rapidly.

To deal with that, some research [35, 67, 108] use mixed proposal distributions that combine hypotheses from the motion model (nominal particles) and from the observation models (dual particles). One shortcoming of such approaches is the need of another modality beside vision such as wheel encoders [67], stereo [35], or the need to use the $3D$ points in the particle in generating the dual hypothesis [108] which means that the $3D$ points of a dual particles do not get updated.

Graph of Local Filters (GLF)

As mentioned earlier, the problem of linearization in the EKF leads to the inconsistency of the filter ([10]) and we have mentioned an approach [57] (has not been applied to vision yet) that proposed to deal with this based on linearizing about the first estimates only. A successful approach to address this problem in visual slam has been presented by Eade and Drummond [34]. This approach (Figure 2.14) is based on a graph of local coordinate frames (nodes) that are connected with similarity transforms (edges). At every time step, one node is active and this node is the node in which the measurement equations for the observed features are the most linear (i.e, the node with the coordinate frame in which the translation of the camera is the smallest). If no such node exists, a new node is created. The update within a node are based on a nonlinear least square procedure with a prior on the depth only. A look at the update equations in the GLF shows that the update

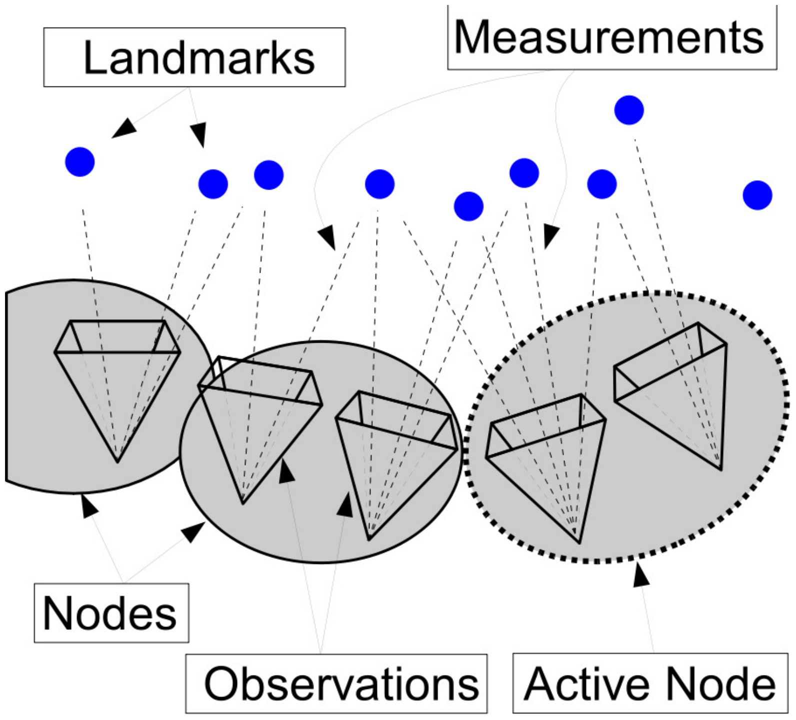


Figure 2.14: Graph of local coordinates frames: Measurements are only filtered in the closest nodes (coordinates frames). The coordinate frames are linked by similarity transforms that allow a global optimization upon loop closure. (Figure taken from [34]).

is equivalent to an iterated EKF, where each iteration is the same as a regular EKF update with the covariance of the motion set to 0 so that the prediction of the motion serves merely as an initialization. Furthermore, Eade and Drummond mention in their paper that in 95% of the cases convergence is reached in one iteration, which means that in 95% of the cases the filtering in the local nodes is equivalent to a regular EKF. After the update is performed within the active node, a global iterative optimization of the graph is done to propagate the new information to all the nodes. This system achieved the best results in the Filter-based category. Firstly, in terms of consistency, this system is more consistent that both the EKF and RBPF. Table 2.1 shows the consistency of this system relative to BA compared to the EKF and the RBPF. If a filter is consistent with respect to BA, its covariance should be greater than the covariance of BA. The good consistency of this system is due to the use of local coordinates system, so the errors due to linearization are minimized, and to the global optimization over the graph. In terms of accuracy, this system outperformed the EKF and the RBPF in as shown in table 2.2 where the errors in the map (features positions) are provided with respect to bundle adjustment (BA). Those results suggest that the accuracy of the the GLF approach is roughly three times better than the EKF. The RBPF performing slightly worse than the EKF.

Sequence	GLF	RBPF	EKF
a	2%	55%	66%
b	5%	52%	48%
c	3%	80%	68%

Table 2.1: Consistency with respect to BA: The consistency is approximated by the percentage of negative eigen values of the difference between the covariance matrix of every estimator and the covariance matrix of BA. 0% is fully consistent. (Table from [34]).

Sequence	Local GLF	RBPF	EKF
a	2.48	16.78	15.59
b	3.45	11.46	9.43
c	2.15	8.34	3.40

Table 2.2: Reconstruction errors: the root mean residual of features positions compared to BA, after registering the maps. The accuracy of the GLF is roughly three times better than the EKF which is slightly better than the RBPF (Table from [34]).

2.5.4 Key Frames Based Approaches

Although some of the visual odometry based approaches discussed earlier use key frames, the term key frame approaches is used here to refer to approaches that perform a global optimization on a select set of key frames spanning the whole (or a significant part) of the sequence, in contrast to odometry approaches which perform the optimization only locally. Some approaches within this category such as the FrameSlam system [70] optimize only the motion over the Key frames, and hence enforcing only the consistency of the motion across loops. Note that in that aspect, the nodes in the GLF approach can be considered as Key frames and the GLF approach although using filtering at the local level, can be considered as a Key frame approach globally. Other approaches perform a full BA optimization between the Key frames such as the PTAM system [68, 69] while others such as the method of Strasdat [119] use both BA on a limited number of frames, followed by an optimization of only the motion upon loop closure and then an optimization of the depth only across all the frames. This section provides a discussion of those approaches.

FrameSlam

Agarwal and Konolige [70] introduced a visual slam system based on key frames called FrameSLAM. It performs a two-frames SFM between key frames that are selected using their Visual Odometry system [71]. They derive a non-linear probabilistic relation between every two successive key frames by marginalizing out the 3D features. Then, they further marginalize out some of the key frames to obtain a skeleton of key frames related by probabilistic constraints, on which they perform non-linear optimization upon loop closure. This system was successfuly used to perform large-scale SLAM. However the features in this system are determined only using two frames each and therefore the obtained map is likely not very accurate. They reported good localization results with respect to GPS and Visual Odometery only, However they didn't report the accuracy of the obtained maps.

Parallel Tracking and Mapping (PTAM)

One of the most successful key frame approaches is the Parallel Tracking And Mapping (PTAM) of Klein and Murray [68, 69], which uses an independent back-end mapping thread that runs BA on a sequence of key frames, and another front-end thread that tracks the motion at every frame using the $3D$ estimates corresponding to the last key frame. The main innovation in this approach is due to the fact that the back-end BA thread does not have to run in real time. When the tracking thread tracks up to a translation that is greater than a minimum distance from the last key frame, and in case the re-projection error in the last frame is lower than a threshold, this last frame is added to the set of key frames. This approach achieved remarkable results. Figure 2.15 shows the localization performance of this approach versus the EKF. The BA thread could run with up to 150 key frames and keep up with the tracking thread (assuming a minimum of 20 frames between key frames). However as the authors mention in the paper, the approach is targeted for small environment in the sense that the user will spend most of his or her time in the same place (for example around a desk). This is because their tracking thread uses the features in the map to determine the camera motion and it requires a large number of features to obtain good motion estimates. Another problem with this system is that if the tracking fails at a given frame, due to an ambiguous motion for example, this would corrupt the whole map. Strasdat *et al.* [119] recently used an approach similar to the PTAM approach in their SLAM system. However they used only 10 frames for their BA back-end and since their system is not confined to a small place as in Klein and Murray, they had to use extra information in the tracking thread, which was in the form of a large number of frame-to-frame optical flow computed using GPU. Since their bundle adjustment spans only few frames, this makes their system more similar to the visual odometry approaches. However, what differentiates them from visual odometry is that upon loop

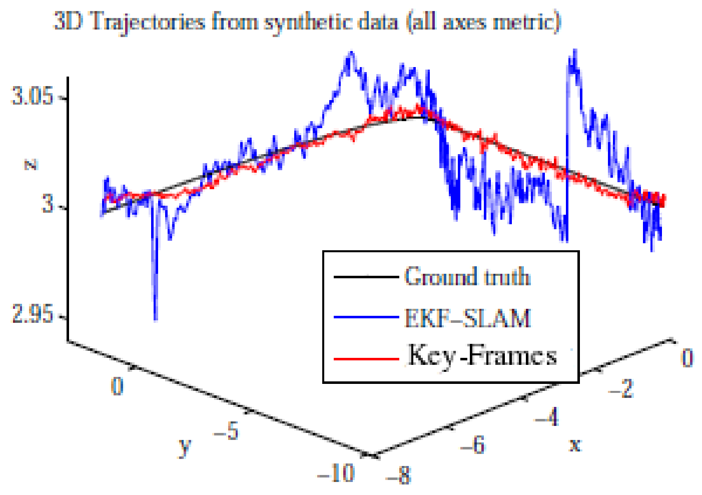
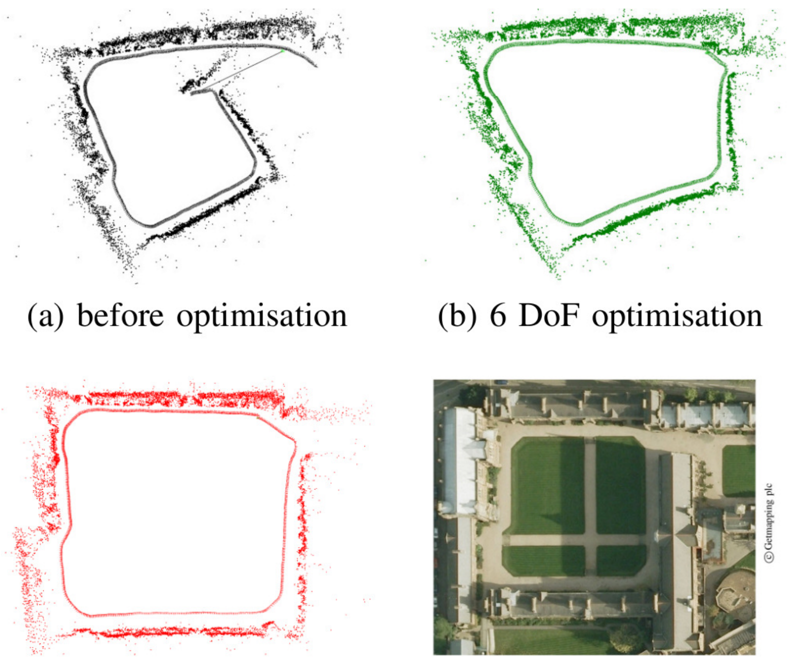


Figure 2.15: Klein and Murray PTAM's localization results versus EKF. Due to the separate BA back end the results are much better than the EKF. (Figure taken from [68]).

closure, they perform an optimisation similar to the one done by the FrameSlam system [70]. They follow this by a bundle adjustment of the structure only. The effect of this optimization is shown in Figure 2.16. Notice that before the optimization, the system exhibits a large drift typical in VO approaches, however after the optimization the drift is greatly reduced.

2.5.5 Discussion

The literature results provided above strongly suggest that the best SFM approaches are the ones based on a front end local tracker and an optimization back end that optimizes between Key Frames or nodes. The front end can be filter based as in the GLF or based on absolute orientation as in PTAM. The back end can be either a graph optimizer based on loop consistency such as in GLF or a bundle adjuster as in PTAM done in a separate thread. Furthermore, the results suggest that approaches based on a bundle adjustment back-end outperform the approaches based on graph optimization. This is reinforced by a recently published study by Strasdat *et al.* [120] that aimed to compare Bundle Adjustment key frame approaches with filter based approaches. The PTAM approach of Klein and Murray [68, 69] was selected as a representative for key frame approaches, and the GLF approach of Eade and Drummond [34] as a representative for filter based approaches. Through a series of experiments using covariance propagation and Mon-



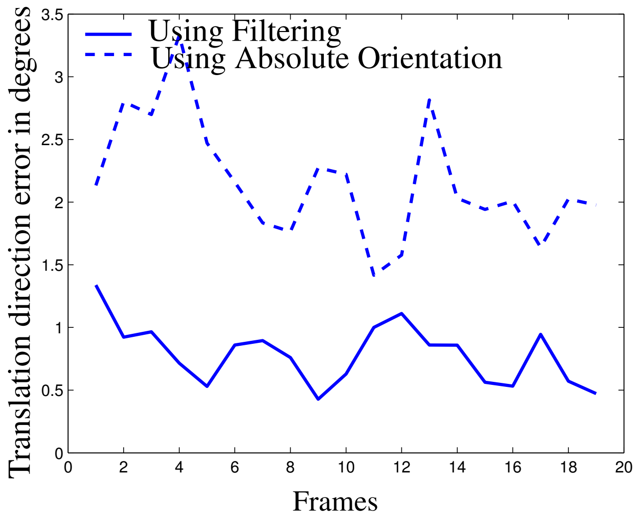
(a) before optimisation (b) 6 DoF optimisation

(c) 7 DoF optimisation (d) aerial photo

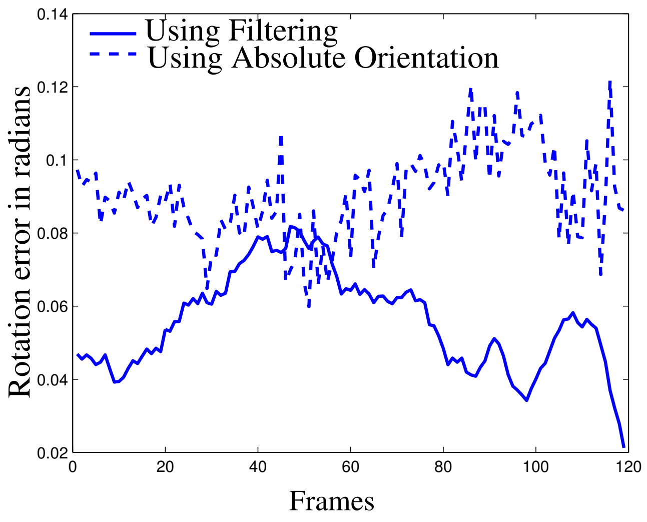
Figure 2.16: Effect of the back-end optimization in Stradast approach: before optimization the drift is very large. The back-end optimization reduces the drift significantly (Figure taken from [119]).

teCarlo simulations, they showed that the only case where filtering is preferable to key frame bundle adjustment is when the overall processing budget is small. This superiority of the Key Frame bundle adjustment approach is due to the bundle adjustment back-end. However, a front-end based on filtering can be superior to a front-end based on absolute orientation for many reasons:

- Filtering provides more accurate results than absolute orientation due to the fact that it integrates information over frames. We performed a series of simulation runs starting from the same distribution of the 3D parameters. The results are shown in Figure 2.17. It is clear that the filtering results are more accurate. This helps the optimization back-end converge faster.
- Filtering can be performed in more computationally efficient ways.
- Filtering can be made robust to erroneous motions by relying on particle filtering.
- Filtering can incorporate additional features other than the ones in the state vector of the back-end.



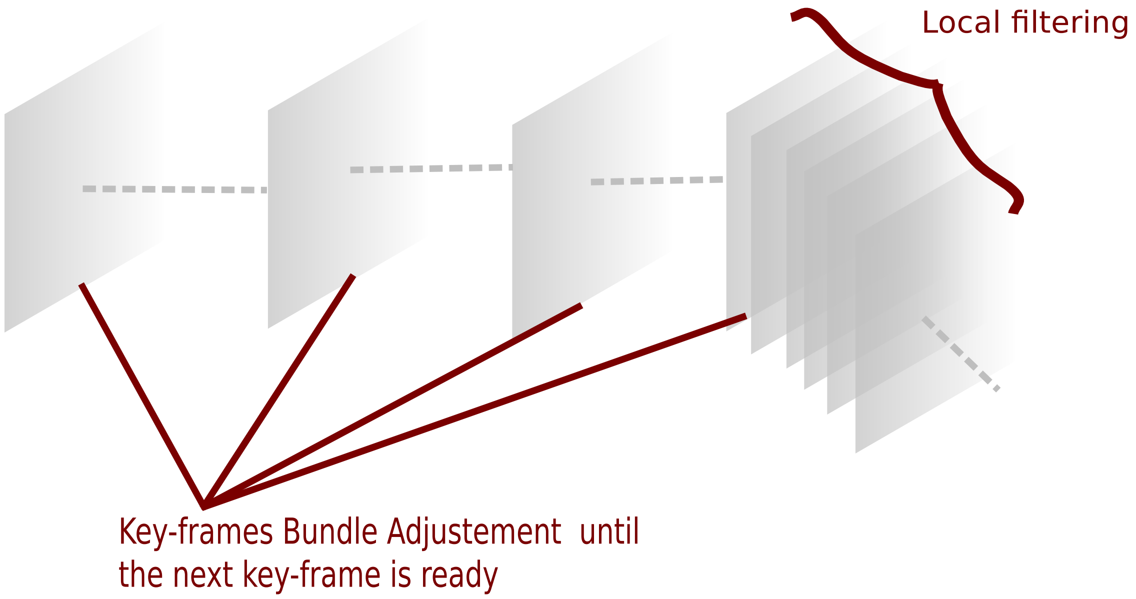
(a)



(b)

Figure 2.17: Performance of absolute orientation versus filtering starting from the same distribution of the 3D parameters. The errors are an average over 50 runs.

Therefore, the best results would be achieved by using a back-end based on key frame bundle adjustment thread and a front-end based on local filtering as shown in Figure 2.18. The research presented in this thesis focuses on developing filters to improve the front-end, by providing filtering algorithms that are faster, more accurate and more robust. In this aspect, it is natural to compare our approach with the GLF approach as it is the state of the art in filtering approaches. The comparison will be based on the fact that the GLF achieves almost a three times accuracy better than the EKF. Therefore, by comparing our approaches to the EKF, we can estimate their performances versus the GLF. Also, it is worth noting here, that the GLF approach is orthogonal to all the contributions of the


Local filtering
Key-frames Bundle Adjustement until the next key-frame is ready
Figure 2.18: Best sequential SFM setup based on the most recent literature results: A separate thread performs bundle adjustment on key frames, while a local filtering thread selects the key frames and computes the relative motion beyond the last key frame. The room for improvement is in the local filtering.

thesis, in the sense that the GLF and our contributions can be complementary and can be implemented together. The same also applies to other approaches such as the SRUKF, the inverse depth approach of Civera [28] or the FEJ-EKF of Huang [57]. This will be further discussed in each chapter.

Chapter 3

Fast Robust Motion Filtering Using Threading Constraints

3.1 Introduction

This chapter is concerned with motion-only estimation. The benefit of motion-only estimation is that it can be performed much faster than the joint estimation of structure and motion. There are many applications where only the motion is required such as vechicle odometry or robot localization. It can be used also as the tracking front end of system similar to the PTAM approach of Klein and Murray [68, 69] described in Chapter 2. Towards this aim, we explore the potential of the algebraic "threading constraints". These constraints involve three views and allow the determination, in a linear fashion, of the 3D motion between the second and third views from the motion between the first and the second and a set of points matched across the three views. In other words, $\overrightarrow{M}(t_2, t_3)$ or $\overrightarrow{M}(t_2, t_2 + \delta t)$ can be determined linearly using $\overrightarrow{M}(t_1, t_2)$ and the projections of a set of points on the views at times t_1, t_2 and t_3 or $t_2 + \Delta t$. Two types of such constraints have been presented in the literature: A discrete-discrete constraint introduced by Avidan and Shashua [6], who first coined the term "threading" for linking fundamental matrices across adjacent views, and a discrete-differential one presented by Heyden in [52].

In the first part of this chapter, we describe the threading constraint and show that, in the discrete-discrete case, a constraint stronger than the one provided by Avidan and Shashua [6] can be derived using the point-point-point incidence equation (Equation (2.29)) directly. As a byproduct of this, we develop a simple new approach to determine a unique trifocal tensor from five calibrated point matches across three views, while previous methods require at least six. The second part of the chapter focuses on exploiting those constraints for the purpose of recursive filtering.

In contrast to the previous usage of the threading constraints as merely a way to infer a new motion from a previous one, we propose to exploit the threading constraints to solve the following problem: Having a set of image matches across a sequence of images indexed from 0 to t and assuming the poses of the camera corresponding to frames 0 to $t-1$ are all known, determine the pose at t using the information from all the previous measurements.

To solve this problem, we combine the motion evolution equations with the threading constraints to derive a new constraint providing two linear equations in $\overrightarrow{M}(t)$ from every point matched in any two previous frames with known corresponding motions. By stacking those constraints together, $\vec{M}(t)$ can be determined even from only one point matched in at least four previous frames. As the use of all the possible constraints is computationally prohibitive, an approach based on Spatio-Temporal Random Sample Consensus is proposed. Via comparisons with the EKF, we show this approach to yield better performance than state of the art filters. Also, unlike filters based on the predict-update paradigm which breaks down when the motion changes rapidly between frames, the proposed approach maintains a good performance regardless of the change between successive camera positions.

3.2 Related Research

The closest work in the literature to our approach are a category of offline approaches that use all the pairwise motions between every two frames and enforce the motion evolution equations as a consistency measure to perform a global optimization. Most notably Govindu [42] presented an approach to combine all the pairwise constraints using averaging on the manifold of 3D rigid transformations. Snavely *et al.* [112] introduced the Image-Pair graph, in which every node is an image and two nodes are linked together if they share common features. The edges between the nodes are similarity transforms representing the pair-wise motion. They associate with every edge a weight representing the uncertainty of this edge computed from the covariance matrix of the pair wise motion. This allows them to select a reduced set of images forming a skeletal graph, by removing all the paths between every pair of images that do not lead to a minimum uncertainty. Sinha *et al.* [111] used this graph to perform an averaging similar to the method of Govindu but incorporating the weight of every pairwise motion instead of assuming all of them as equally accurate. All those approaches are iterative offline batch methods that are performed after all the relative motions are estimated and their scales determined with respect to a common gauge. Contrarily, the approach presented in this chapter is designed to determine in one step the motion at time t (including a consistent scale with the previous estimates) using all the previous motions and measurments.

3.3 Threading Constraints

Although all the formulations presented in this chapter belong to the discrete-discrete case, we briefly provide, for the sake of completeness, a description of the discrete-continuous constraint.

3.3.1 Discrete-Continuous Constraint

The discrete-continuous case used to infer the infinitesimal incremental motion $\overrightarrow{M}(t_2, t_2 + \delta t)$ given the motion $\overrightarrow{M}(t_1, t_2)$ is much simpler. This is due to the possibility of performing a first order Taylor approximation of the rotation across the differential baseline as follows:

$$\exp([\overrightarrow{\omega}]_\times) \approx [\overrightarrow{\omega}]_\times. \tag{3.1}$$

Heyden [52] used such approximation to derive a constraint on both the instantaneous and discrete motion by stacking the re-projection equations at t_1, t_2 and $t_2 + \delta t$ in one system:

$$\underbrace{\begin{bmatrix} \tilde{\overrightarrow{q}} & \overrightarrow{q}(t_2) & 0 & \overrightarrow{T}(t_1,t_2) \\ [\overrightarrow{\omega}(t_1)]_\times \tilde{\overrightarrow{q}} & \dot{\overrightarrow{q}}(t_2) & \overrightarrow{q}(t_2) & \overrightarrow{V}(t_2) \end{bmatrix}}_{D} \begin{bmatrix} -\lambda_{t_1} \\ \lambda_{t_2} \\ \delta\lambda \\ 0 \end{bmatrix} = 0, \tag{3.2}$$

where $\tilde{\overrightarrow{q}}(t_1) = \underline{R}(t_1,t_2)\overrightarrow{q}(t_1)$ and λ_{t_1}, λ_{t_2} and $\delta\lambda$ are projective scaling constants. The matrix D in Equation (3.2) is 6×4, and this equation implies that the rank of D is less than four since the system in this equation should have in general a unique 3-dimensional solution (λ_{t_1}, λ_{t_2} and $\delta\lambda$). Being of rank less than four means that all the minors of order four are equal to zero. The useful minors (nine) are the ones containing two rows of the first three and two of the last three because these minors give equations relating $\underline{R}(t_1,t_2)$, $\overrightarrow{T}(t_1,t_2)$, $\overrightarrow{\omega}(t_2)$ and $\overrightarrow{V}(t_2)$. Only two independent equations can be obtained. We use the following two minors:

$$det \begin{bmatrix} \tilde{q}_1 & q_1(t_2) & 0 & T_1(t_1,t_2) \\ \tilde{q}_2 & q_2(t_2) & 0 & T_2(t_1,t_2) \\ \omega_2(t_2)\tilde{q}_3 - \omega_3(t_2)\tilde{q}_2 & \dot{q}_1(t_2) & q_1(t_2) & V_1(t_2) \\ \omega_3(t_2)\tilde{q}_1 - \omega_1(t_2)\tilde{q}_3 & \dot{q}_2(t_2) & q_2(t_2) & V_2(t_2) \end{bmatrix} = 0 \qquad \text{(3.3a)}$$

and

$$det \begin{bmatrix} \tilde{q}_2 & q_2(t_2) & 0 & T_2(t_1,t_2) \\ \tilde{q}_3 & 1 & 0 & T_3(t_1,t_2) \\ \omega_2(t_2)\tilde{q}_3 - \omega_3(t_2)\tilde{q}_2(t_2) & \dot{q}_1(t_2) & q_1(t_2) & V_1(t_2) \\ \omega_1(t_2)\tilde{q}_2 - \omega_2(t_2)\tilde{q}_1(t_2) & 0 & 1 & V_3(t_2) \end{bmatrix} = 0 \qquad \text{(3.3b)}$$

Expanding the above two determinants gives two linear equations in $\overrightarrow{\omega}(t_2)$ and $\overrightarrow{V}(t_2)$ for each point. Therefore, three points are enough to determine $\overrightarrow{V}(t_2)$ and $\overrightarrow{\omega}(t_2)$.

After this step, as in the case of the Discrete-continuous constraint, an LM minimization can be performed in order to optimize the motion over the five point. However, this can be considerably faster by capitalizing on the separability of the discrete and continuous motion, in the sense that, if $\underline{M}(t_1,t_2)$ is known then $\overrightarrow{\omega}(t_2)$ and $\overrightarrow{V}(t_2)$ can be determined linearly in closed-form. This, separability is used inside the LM iterations, where the obtained estimate of $\overrightarrow{\omega}(t_2)$ and $\overrightarrow{V}(t_2)$ after each iteration is replaced by their value obtained linearly from Equations (3.3) using $\underline{M}(t_1,t_2)$. This makes the optimization 5 dimensional instead of 11 dimensional. However, as mentioned earlier, this constraint involves an approximation that does not hold unless the differential baseline is really small.

3.3.2 Discrete-Discrete Constraint

The geometry of three views has been studied extensively through the trifocal tensor. However, the first explicit formulation of a constraint to recover the motion between the second and third views from the motion between the first and second has been presented by Avidan and Shashua [6]. They relied on their earlier introduced bifocal tensor $\mathcal{F}$ [5] which is a $3 \times 3 \times 3$ embedding of the fundamental matrix of the two views, and is defined in a similar fashion as the trifocal tensor (Equation (2.28)) with the second and third views being the same. They used the point-line-point incidence (Equation (2.30)) to derive their constraint. Although, we will show that a stronger constraint (enforcing more of the true geometry of the problem) can be derived directly from the point-point-point incidence (Equation (2.29)) in terms of the trifocal tensor, we present firstly their original formulation based on the bifocal tensor. However, instead of using their covariant-contravariant tensorial notation, we provide an equivalent matrix-vector formulation for the Euclidean case.

Let $\underline{\mathcal{F}}_i, i \in \{1,2,3\}$ be the three matrices of $\mathcal{F}$ defined as in Equation (2.28) and let $\underline{\mathcal{F}}^j, j \in \{1,2,3\}$ be the three matrices defined as: $\underline{\mathcal{F}}^j = [\overrightarrow{\mathcal{F}}^j_1, \overrightarrow{\mathcal{F}}^j_2, \overrightarrow{\mathcal{F}}^j_3]$ (see Section 2.3.1 for the notations), the equation of Avidan and Shashua [6] can be written as follows:

$$\sum_j \overrightarrow{l}(t_2)\overrightarrow{q}(t_1)(\underline{\mathcal{F}}^j \underline{R}(t_2,t_3) - T_j(t_2,t_3)\underline{R}(t_1,t_2)) \cong \overrightarrow{q}(t_3), \tag{3.4}$$

where $\overrightarrow{l}(t_2)$ is any line passing by $\overrightarrow{q}(t_2)$ and can then be written as $\overrightarrow{l}(t_2) = (l_1, l_2, l_3)^T$ with $l_3 = -q_2(t_1)l_1 - q_2(t_2)l_2$. In this equation $\underline{R}(t_2,t_3)$ and $\overrightarrow{T}(t_2,t_3)$ appear only linearly. Therefore, two independent linear equations in $\overrightarrow{M}(t_2,t_3) = [\overrightarrow{\Omega}(t2,t3); \overrightarrow{T}(t2,t3)]$ can be obtainted. Before providing those equations, we simplify Equation (3.4) to a more convenient form. Let $\underline{U}(\overrightarrow{T}(t_1,t_2))$ and $\underline{W}(\overrightarrow{l}(t_2)$ be the two matrices defined as follows:

$$\underline{U} = \begin{pmatrix} 0 & -T_2(t_1,t_2) & -T_3(t_1,t_2) \\ 0 & T_1(t_1,t_2) & 0 \\ 0 & 0 & T_1(t_1,t_2) \\ T_2(t_1,t_2) & 0 & 0 \\ -T_1(t_1,t_2) & 0 & -T_3(t_1,t_2) \\ 0 & 0 & T_2(t_1,t_2) \\ T_3(t_1,t_2) & 0 & 0 \\ 0 & T_3(t_1,t_2) & 0 \\ -T_1(t_1,t_2) & -T_2(t_1,t_2) & 0 \end{pmatrix} \tag{3.5}$$

and

$$\underline{W} = \begin{pmatrix} l_1 & 0 & 0 & l_2 & 0 & 0 & l_3 & 0 & 0 \\ 0 & l_1 & 0 & 0 & l_2 & 0 & 0 & l_3 & 0 \\ 0 & 0 & l_1 & 0 & 0 & l_2 & 0 & 0 & l_3 \end{pmatrix}. \tag{3.6}$$

Then, equation (3.4) can be written in the following way:

$$\overrightarrow{q}(t_1)^T R(t_1,t_2)^T(\underline{WUR}(t_2,t_3)^T - \overrightarrow{l}(t_2)\overrightarrow{T}(t_2,t_3)^T) \cong \overrightarrow{q}(t_3)^T \tag{3.7}$$

which is equivalent to:

$$[\overrightarrow{q}(t_3)]_\times \overrightarrow{q}(t_1)^T R(t_1,t_2)^T(\underline{WUR}(t_2,t_3)^T - \overrightarrow{l}(t_2)\overrightarrow{T}(t_2,t_3)^T) = \overrightarrow{0}. \tag{3.8}$$

For each different line $\overrightarrow{l}(t_2)$ passing by $\overrightarrow{q}(t_2)$, two linear equations in term of $\underline{R}(t_2,t_3)$ and $\overrightarrow{T}(t_2,t_3)$ can be derived and hence, six points at least are required to recover those quantities. For example, assuming a slope of 1 for $\overrightarrow{l}(t_2)$ (i.e., $l_1 = l_2 = 1$), the two obtained equations are:

$$\begin{gathered}\begin{bmatrix} 1 & 0 & -q_1(t_3) \\ 0 & 1 & -q_2(t_3) \end{bmatrix} (\underline{R}(t_2,t_3)\underline{C} - \overrightarrow{T}(t_2,t_3)) \begin{bmatrix} 1 & 1 & -(q_1(t_2)+q_2(t_2)) \end{bmatrix} \\ \underline{R}(t_1,t_2)\overrightarrow{q}(t_1) = 0, \end{gathered} \tag{3.9}$$

where $\underline{C}$ is the matrix

$$\begin{bmatrix} T_2(t_1,t_2) & -T_1(t_1,t_2) & -T_3(t_1,t_2) \\ -T_2(t_1,t_2) & T_1(t_1,t_2) & -T_3(t_1,t_2) \\ (q_1(t_2)+q_2(t_2))T_1(t_1,t_2) & (q_1(t_2)+q_2(t_2))T_2(t_1,t_2) & T_1(t_1,t_2)+T_2(t_1,t_2) \end{bmatrix}.$$

Stacking Equations (3.9) for at least six points results in a system of equations of the form $\underline{A}\overrightarrow{M}(t_2,t_3) = 0$. The solution can be determined through SVD decomposition as the right singular vector corresponding to the smallest singular value.

Point-Point-Point Threading Constraint

The point-line-point incidence used to derive the above constraint is weaker than the point-point-point incidence constraint given by Equation (2.29) because it constrains two rays passing through $\overrightarrow{q}(t_1)$ and $\overrightarrow{q}(t_3)$ to intersect on a plane passing through $\overrightarrow{l}(t_2)$ instead of intersecting a line passing through$\overrightarrow{q}(t_2)$). This is also manifested by the fact that the constraint can be obtained by setting random values to l_1 and l_2. Therefore, a constraint based on the point-point-point incidence would be more well-behaved. In this thesis such constraint is derived from Equation (2.29) as follows: Let $\overrightarrow{\mathcal{T}}$ be the 27-vector containing the elements of the trifocal tensor $\mathcal{T}$ then, Equation (2.29) can be rearranged in the form:

$$\underline{K}\overrightarrow{\mathcal{T}} = \overrightarrow{0}, \tag{3.10}$$

where $\underline{K}$ is a 9×27 matrix whose entries are trilinear products of the image coordinates at times t_1, t_2 and t_3 , $\overrightarrow{q}(t_1)$, $\overrightarrow{q}(t_2)$ and $\overrightarrow{q}(t_3)$ (see Appendix C). However, the rank of

$\underline{K}$ is only 4 [46] and hence, only 4 of its 9 rows are independent. Furthermore, following from the definition of the trifocal tensor (Equation 2.28), $\overrightarrow{\mathcal{T}}$ can be expanded as follows:

$$\overrightarrow{\mathcal{T}} = \begin{bmatrix} R_1(t_1,t_2)T_1(t_1,t_3) - R_1(t_1,t_3)T_1(t_1,t_2) \\ R_1(t_1,t_2)T_2(t_1,t_3) - R_4(t_1,t_3)T_1(t_1,t_2) \\ R_1(t_1,t_2)T_3(t_1,t_3) - R_7(t_1,t_3)T_1(t_1,t_2) \\ R_4(t_1,t_2)T_1(t_1,t_3) - R_1(t_1,t_3)T_2(t_1,t_2) \\ R_4(t_1,t_2)T_2(t_1,t_3) - R_4(t_1,t_3)T_2(t_1,t_2) \\ R_4(t_1,t_2)T_3(t_1,t_3) - R_7(t_1,t_3)T_2(t_1,t_2) \\ R_7(t_1,t_2)T_1(t_1,t_3) - R_1(t_1,t_3)T_3(t_1,t_2) \\ R_7(t_1,t_2)T_2(t_1,t_3) - R_4(t_1,t_3)T_3(t_1,t_2) \\ R_7(t_1,t_2)T_3(t_1,t_3) - R_7(t_1,t_3)T_3(t_1,t_2) \\ R_2(t_1,t_2)T_1(t_1,t_3) - R_2(t_1,t_3)T_1(t_1,t_2) \\ R_2(t_1,t_2)T_2(t_1,t_3) - R_5(t_1,t_3)T_1(t_1,t_2) \\ R_2(t_1,t_2)T_3(t_1,t_3) - R_8(t_1,t_3)T_1(t_1,t_2) \\ R_5(t_1,t_2)T_1(t_1,t_3) - R_2(t_1,t_3)T_2(t_1,t_2) \\ R_5(t_1,t_2)T_2(t_1,t_3) - R_5(t_1,t_3)T_2(t_1,t_2) \\ R_5(t_1,t_2)T_3(t_1,t_3) - R_8(t_1,t_3)T_2(t_1,t_2) \\ R_8(t_1,t_2)T_1(t_1,t_3) - R_2(t_1,t_3)T_3(t_1,t_2) \\ R_8(t_1,t_2)T_2(t_1,t_3) - R_5(t_1,t_3)T_3(t_1,t_2) \\ R_8(t_1,t_2)T_3(t_1,t_3) - R_8(t_1,t_3)T_3(t_1,t_2) \\ R_3(t_1,t_2)T_1(t_1,t_3) - R_3(t_1,t_3)T_1(t_1,t_2) \\ R_3(t_1,t_2)T_2(t_1,t_3) - R_6(t_1,t_3)T_1(t_1,t_2) \\ R_3(t_1,t_2)T_3(t_1,t_3) - R_9(t_1,t_3)T_1(t_1,t_2) \\ R_5(t_1,t_2)T_1(t_1,t_3) - R_3(t_1,t_3)T_2(t_1,t_2) \\ R_5(t_1,t_2)T_2(t_1,t_3) - R_6(t_1,t_3)T_2(t_1,t_2) \\ R_5(t_1,t_2)T_3(t_1,t_3) - R_9(t_1,t_3)T_2(t_1,t_2) \\ R_9(t_1,t_2)T_1(t_1,t_3) - R_3(t_1,t_3)T_3(t_1,t_2) \\ R_9(t_1,t_2)T_2(t_1,t_3) - R_6(t_1,t_3)T_3(t_1,t_2) \\ R_9(t_1,t_2)T_3(t_1,t_3) - R_9(t_1,t_3)T_3(t_1,t_2) \end{bmatrix}. \tag{3.11}$$

which in turn can be further expanded as

$$\overrightarrow{\mathcal{T}} = \underline{L}[\overrightarrow{R}(t_1,t_3);\overrightarrow{T}(t_1,t_3)], \tag{3.12}$$

where $\underline{L}$ is a matrix whose entries are elements of $\overrightarrow{M}(t_1,t_2)$. (Recall that $\overrightarrow{R}(t_1,t_3)$ is the vector obtained by stacking the elements of the matrix $\underline{R}(t_1,t_3)$ in a row-major order). $\underline{L}$

which is a 27×12 matrix defined as (omitting the time indices (t_1, t_2)):

$$\underline{L}(\overrightarrow{M}(t_1, t_2)) =$$

$$\begin{bmatrix}
-T_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & R_1 & 0 & 0 \\
0 & 0 & 0 & -T_0 & 0 & 0 & 0 & 0 & 0 & 0 & R_1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -T_0 & 0 & 0 & 0 & 0 & R_1 \\
-T_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & R_4 & 0 & 0 \\
0 & 0 & 0 & -T_1 & 0 & 0 & 0 & 0 & 0 & 0 & R_4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -T_1 & 0 & 0 & 0 & 0 & R_4 \\
-T_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & R_7 & 0 & 0 \\
0 & 0 & 0 & -T_2 & 0 & 0 & 0 & 0 & 0 & 0 & R_7 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -T_2 & 0 & 0 & 0 & 0 & R_7 \\
0 & -T_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & R_2 & 0 & 0 \\
0 & 0 & 0 & 0 & -T_0 & 0 & 0 & 0 & 0 & 0 & R_2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -T_0 & 0 & 0 & 0 & R_2 \\
0 & -T_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & R_5 & 0 & 0 \\
0 & 0 & 0 & 0 & -T_1 & 0 & 0 & 0 & 0 & 0 & R_5 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -T_1 & 0 & 0 & 0 & R_5 \\
0 & -T_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & R_8 & 0 & 0 \\
0 & 0 & 0 & 0 & -T_2 & 0 & 0 & 0 & 0 & 0 & R_8 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -T_2 & 0 & 0 & 0 & R_8 \\
0 & 0 & -T_0 & 0 & 0 & 0 & 0 & 0 & 0 & R_3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -T_0 & 0 & 0 & 0 & 0 & R_3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -T_0 & 0 & 0 & R_3 \\
0 & 0 & -T_1 & 0 & 0 & 0 & 0 & 0 & 0 & R_6 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -T_1 & 0 & 0 & 0 & 0 & R_6 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -T_1 & 0 & 0 & R_6 \\
0 & 0 & -T_2 & 0 & 0 & 0 & 0 & 0 & 0 & R_9 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -T_2 & 0 & 0 & 0 & 0 & R_9 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -T_2 & 0 & 0 & R_9
\end{bmatrix} \tag{3.13}$$

Hence, Equation (2.29) can be written as:

$$\underline{A}\overrightarrow{M}(t_1, t_3) = 0, \tag{3.14}$$

with $\underline{A} = \underline{KL}$. The motion across the first and third frames $\overrightarrow{M}(t_1, t_3)$ can be recovered linearly from Equation (3.14) as the right singular vector corresponding to the smallest eigen value of the matrix $\underline{A}$. Note that $\underline{A}$ has a rank two and therefore, only two rows of this matrix are used. Figure 3.1 illustrates the relative performance of this point-point-point based method versus the point-line-point method of Avidan and Shashua [6] in a 1000 simulation runs involving different randomly generated motions and 3D points and with Gaussian noise added to the image projections. The figure shows the error (in log scale) in the recovered translation and rotation using both methods. The runs are arranged so that the corresponding errors are sorted in an ascending order. It is clear that the stronger point-point-point constraint exhibits a better performance. Therefore, in the filter presented later in this chapter this method will be adopted and will be referred to as "linear threading".

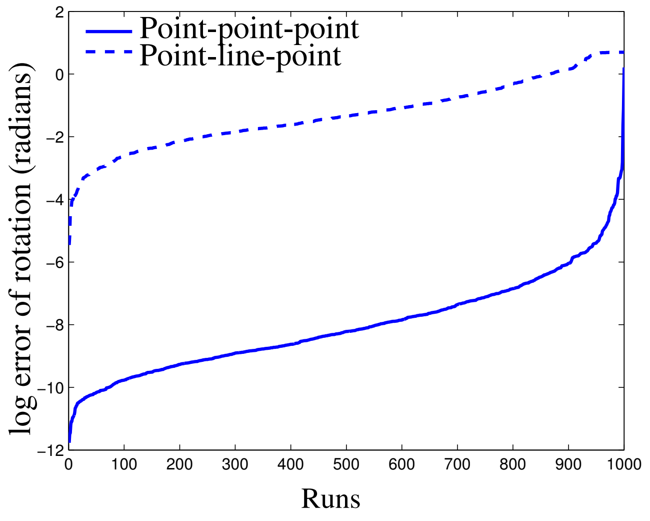
The linear threading as explained above results in a rotation matrix that does not satisfy the conditions of a true rotation matrix (i.e., unit norm with orthogonal columns). Our solution for this is to use SVD decomposition to get the closest rotation matrix to the recovered one. Let $\underline{\hat{R}}$ be this linearly recovered rotation matrix, via SVD decomposition, $\underline{\hat{R}}$ can be written as the product $\underline{UWV^T}$. The closest rotation matrix to $\underline{\hat{R}}$ is the matrix $\underline{UV^T}$. Once the corrected rotation matrix is obtained, it is substituted back in the original system of equations as follows:

$$\begin{aligned}
\underline{A}[\overrightarrow{R};\overrightarrow{T}] &= 0\\
[\underline{A}^{1:9}\underline{A}^{10:12}][\overrightarrow{R};\overrightarrow{T}] &= 0\\
\underline{A}^{1:9}\overrightarrow{R} + \underline{A}^{10:12}\overrightarrow{T} &= 0\\
\underline{A}^{10:12}\overrightarrow{T} &= -\underline{A}^{1:9}\overrightarrow{R},
\end{aligned} \tag{3.15}$$

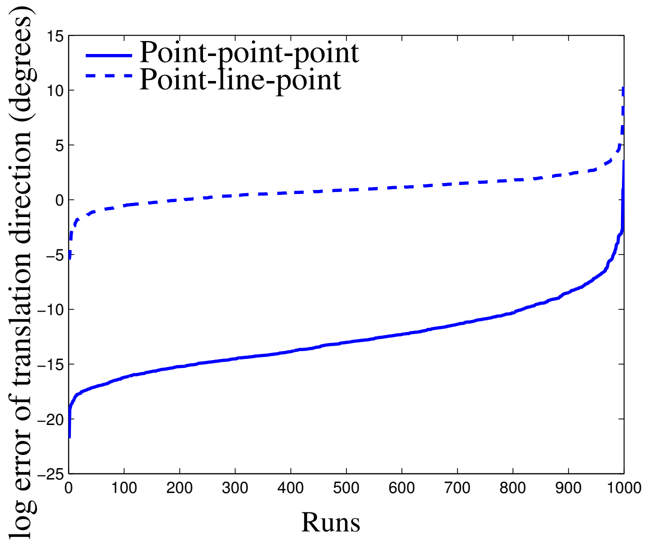
$\underline{A}^{1:9}$ and $\underline{A}^{1:9}$ denoting respectively the first 9 columns and the last 3 columns of the matrix $\underline{A}$ as explained in Section 2.3.1. $\overrightarrow{T}$ can then be determined using linear least squares as:

$$\overrightarrow{T} = -((\underline{A}^{10:12})^T\underline{A}^{10:12})^{-1}(\underline{A}^{10:12})^T\underline{A}^{1:9}\overrightarrow{R}. \tag{3.16}$$

Nevertheless, there is no guarantee that the closest rotation matrix to the originally determined $\underline{\hat{R}}$ satisfies properly the system $\underline{A}\overrightarrow{M}(t_1,t_3) = 0$. Therefore, we came up with a new approach to determine $\overrightarrow{M}(t_1,t_3)$ from $\overrightarrow{M}(t_1,t_2)$ and the projections at t_3. This approach is based on the five-points two views algorithm [88] and needs five points instead of six. It will be dubbed "essential matrices based" approach in the remainder of this chapter.



(a)



(b)

Figure 3.1: Relative performances of the point-line-point and point-point-point constraints over 1000 different runs. The errors have been sorted in ascending order and log-scale has been used for better visualization. The point-point-point constraint is clearly superior.

3.3.3 Five Points Essential Matrices Based Threading

Suppose that five points are matched across three views and that the motion $\underline{M}(t_1, t_2)$ between the first two of these three frames is known, and it is required to determine the motion $\underline{M}(t_1, t_3)$ across the first and the third frame in such a way to be consistent with $\underline{M}(t_1, t_2)$. The solution proposed herein starts by determining the essential matrices compatible with the five points matched in the first and third frames using the five-points algorithm [88]. Up to 10 matrices may be recovered. Then, the rotation and translation corresponding to each matrix are determined using SVD as in Section 2.4.3. In the calibrated setting, the only missing piece of information for the recovered $\underline{M}(t_1, t_3)$ and the available $\underline{M}(t_1, t_2)$ to be compatible is the scale of the translation vector $\overrightarrow{T}(t_1, t_3)$, which should be consistent with the trifocal tensor representing the motion across the three frames. Therefore, we can use the threading constraint of Equation (3.14) to determine this value. If the scaling of $\overrightarrow{T}(t_1, t_3)$ is unknown, and α is the unknown constant by which $\overrightarrow{T}(t_1, t_3)$ should be scaled so that it verifies the trifocal point-point-point incidence, then the last equation in the System (3.15) can be written as follows (omitting the time indices (t_1, t_3)):

$$\underline{A}^{10:12}\overrightarrow{T}\alpha = -\underline{A}^{1:9}\overrightarrow{R}, \tag{3.17}$$

which provides a least squares solution of α as:

$$\alpha = -\frac{\overrightarrow{T}^T(\underline{A}^{10:12})^T\underline{A}^{1:9}\overrightarrow{R}}{\overrightarrow{T}^T(\underline{A}^{10:12})^T\underline{A}^{10:12}\overrightarrow{T}}. \tag{3.18}$$

Once all the candidate trifocal tensors corresponding to all the candidate matrices are generated, the Sampson's approximation of the trifocal error (Equation (2.29)) is used to prune the candidate matrices that do not satisfy the trifocal constraint. In the general case, only one matrix satisfies this relation. There are two main advantages for this approach:

1. the errors in the estimated rotation and translation direction between t_1 and t_3 is independent of the errors in the estimated motion between t_1 and $_1$. Only the scale of the translation $\overrightarrow{T}(t_1, t_3)$ is dependent on those errors, while in the linear approach all of $\overrightarrow{M}(t_1, t_3)$ is affected by the errors in $\overrightarrow{M}(t_1, t_2)$. This has very important implications in the filtering as we will show later.

2. it uses only five points instead of six. This is especially important in the case of RANSAC [38] like approaches, where the likelihood of finding five points without outliers is higher than the likelihood of finding six, and the number of possible

minimal subsets with six points is much higher that the number of possible minimal subsets with five. For example for 50 points, there are 2 118 760 possible combinations of five points compared to 15 890 700 possible combinations of six points.

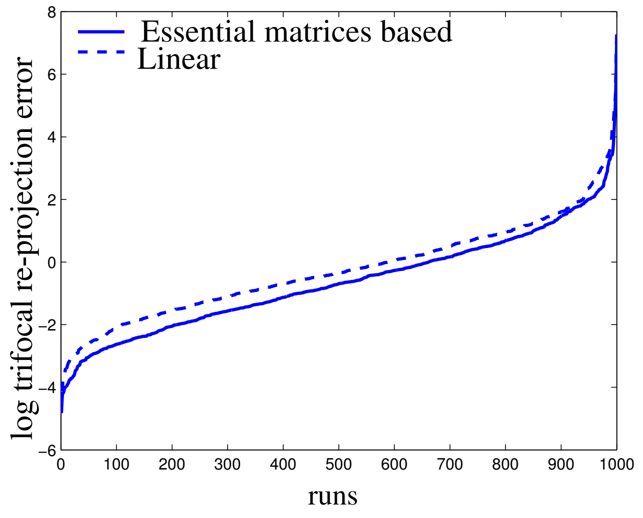
The obtained motion can be then refined using a non-linear optimization as explained in Section 3.3.5. Figure 3.2 shows that this essential matrices based threading approach outperforms the linear one introduced earlier in terms of the trifocal error both and after non-linear minimization. Also, we can extend this method to devise an approach to compute the trifocal tensor from five calibrated points across three views without triangulation. Earlier methods to compute the trifocal tensor involved six points for the projective case [101, 127].

3.3.4 Unique Trifocal Tensor from Five Correspondence

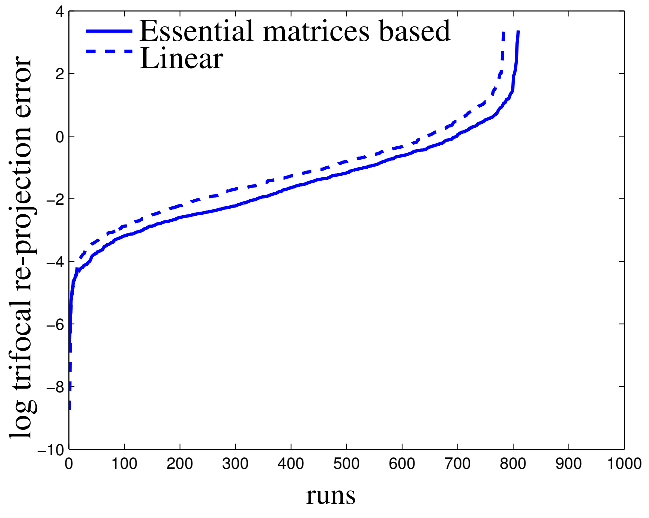
Not only does the above test based on the trifocal error allow us to prune the wrong essential matrices across the first and third views, but if the motion between the first and second views ($\underline{M}(t_1, t_2)$) is also unknown, it can be used to prune wrong matrices across the three views simultaneously. Therefore, our algorithm to determine the trifocal tensor (Algorithm 1) works as follows: we start by determining two sets of candidate essential matrices $\underline{E}^{(i)}(t_1, t_2)$ between the first and second views, and $\underline{E}^{(j)}(t_1, t_3)$ between the first and third views. The rotation matrices and translation vectors for each of the matrices in the two sets are then determined used SVD generating up to 20 motions each. Then for each couple of motions ($\underline{M}^{(i)}(t_1, t_2)$, $\underline{M}^{(j)}(t_1, t_3)$), Equation (3.18) is used to make the magnitude of the translation $\overrightarrow{T}^{(j)}(t_1, t_3)$ compatible with $\overrightarrow{T}^{(i)}(t_1, t_2)$. Then the trifocal error of both motions (Equation (2.29)) is calculated. The couple ($\underline{M}^{(i)}(t_1, t_2)$, $\underline{M}^{(j)}(t_1, t_3)$) that generates the smallest error is the one corresponding to the true trifocal tensor.

3.3.5 Non-linear Optimization

The above threading algorithms do not take into account the effect of the matching between the second sets of matches on the first motion $\overrightarrow{M}(t_1, t_2)$. To account for this, a non linear optimization of the trifocal tensor is performed by minimizing the trifocal error of the five points (or six points). Many approaches have suggested different parameterizations of the Trifocal tensor for the purpose of non-linear minimization [127]. However, in



(a) Before LM



(b) After LM

Figure 3.2: Relative performances (Trifocal error) of the linear and essential matrices based threading approaches, before and after 50 LM iterations. The essential-based approach outperforms the linear one.

Algorithm 1: Algorithm to determine the trifocal tensor from 5 points.

```
   Input: Set of five features matched in three frames
   Output: Trifocal tensor
1  Generate the essential matrices compatible with views 1-2;
2  Generate the corresponding rotation and translation of each matrix using SVD;
3  Generate the essential matrices compatible with views 1-3;
4  Generate the corresponding rotation and translation of each matrix using SVD;
5  for each matrix in the first set do
6      for each matrix in the second set do
7          Determine the scale of the second translation using Equation (3.18);
8          Determine the trifocal error of the resulting trifocal tensor for every point
           using Equation (2.29);
9      end
10 end
11 Select the trifocal tensor with the smallest error;
12 Perform LM optimization as in Section 3.3.5;
13 Output the resulting tensor;
```

the calibrated case, the best parameterization of the Trifocal tensor is to use directly the 11 parameters of the two motions. We encode the rotations using quaternions. Therefore, the vector to be optimized will be:

$$\overrightarrow{S} = [\overrightarrow{\Omega}(t_1, t_2); T_1(t_1, t_2); T_2(t_1, t_2); \overrightarrow{\Omega}(t_1, t_3); \overrightarrow{T}(t_1, t_3)].$$

$T_3(t_1, t_2)$ is kept fixed and not included in the state vector to fix the scale ambiguity. Replacing in Equation (3.10) the rotation matrices by their equivalent quaternions as in Equation (B.4), we obtain a system of non-linear equations in the elements of $\overrightarrow{S}$. The solution can be refined through a straightforward Levenberg-Marquardt minimization (Appendix D). The analytic Jacobian used in this LM minimization is provided in Appendix C.

3.4 Recursive Estimation Using the Threading Constraints

The threading constraints have two advantages that make them suitable for recursive filtering. Those two advantages are:

1. The ability to predict motion in a closed-form way between frames

2. The ability to do so without the determination of the depth which results in com-

putational benefits.

However, the use of these equations in recursive filtering is not straightforward for two reasons. First, the threading constraints use both the previous motion and the measurement and hence, this violates the statistical independence of the predicted motion from the measurements. The second one is that although threading constraints provide a better approximation than the random walk model, however with time, the system will drift because accumulation of errors. To solve this, this section presents a novel filter based on threading constraints and that does not rely on the Markovian predict-update paradigm. The filter instead draws on principles from Monte-Carlo simulations and random sample consensus. To introduce this threading based filter and the random sample consensus a formulation over three frames only is firstly presented and then extended to the case of multiple frames.

3.5 Estimation Over Three Frames

Assume a set of N image features $\overrightarrow{Z}(0)$, $\overrightarrow{Z}(1)$ and $\overrightarrow{Z}(2)$ are available at times 0, 1 and 2 respectively ($\overrightarrow{Z}(t) = [\overrightarrow{q}^0(t); ... \overrightarrow{q}^N(t)]$). Assume also that the motion $\overrightarrow{M}(0,1)$ is available. The goal is to obtain $\overrightarrow{M}(0,3)$. To tackle this problem, we start by identifying the information that can be obtained using the threading constraints:

- Every point i matched in the frames 0,1,2 provides two independent equations of the form 3.14 in $\overrightarrow{M}(0,2)$.

- Also, every five points, give a solution $\hat{\overrightarrow{M}}(0,2)$ as described in Section 3.3.3 . This solution can be seen as a set of 12 linear independent constraints on $\overrightarrow{M}(0,2)$ of the form $\underline{I}_{12}\overrightarrow{M}(0,2) = \hat{\overrightarrow{M}}(0,2)$ where $\underline{I}_{12}$ is a identity matrix of dimension 12×12.

The best solution in a least squares sense, is to stack all the constraints in one large linear system and solve it for $\overrightarrow{M}(0,2)$. However, this would be computationally expensive if N is large and most importantly it does not have a potential to be extended to multiple frames. Another way to do this is through the "Random Sample Consensus" principle [38]. This works by drawing random sets of the minimum number of constraints (6 in this case) from the pool of all the available constraints, estimate the motion corresponding to each set and then checking to what extent every hypothesis satisfies the image data. Satisfaction of the data is determined by the epipolar error across the two pairs of frames (0,2) and (1,2) using, for example, the Sampson's approximation of the epipolar errors (Equation (2.23)). Let $e^{i,(j)}(0,2)$ and $e^{i,(j)}(1,2)$ represent the error of the j^{th} hypothesis for the i^{th} feature across the two pairs of frames, then the average error of the j^{th} hypothesis can be defined as:

$$e_a^j = \frac{\sum_i^N (\rho(e^{i,(j)}(0,2)) + \rho(e^{i,(j)}(0,1)))}{N}, \tag{3.19}$$

where ρ is a robust function which limits the contribution of outliers. A discussion of such functions is provided by Huber [58]. The search is stopped when a hypothesis for which the average error is below a given threshold or when a fixed number of hypotheses is tested.

This can also be explained via the principle of *Importance Sampling* as explained in Section A.2.1. The desired distribution is the distribution $\pi(\overrightarrow{M}(0,2)|\overrightarrow{Z}(0), \overrightarrow{Z}(1), \overrightarrow{Z}(2))$. The proposal distribution in this case can be considered as the sample based distribution obtained by generating motion samples from random collections of the constraints considered above. Before considering their compliance to the whole image data, those samples can be considered as drawn from a uniform distribution. Then, a sample-based representation of the desired distribution can be obtained by weighing the samples with their likelihood given the data which can be considered, for the hypothesis j, as the inverse exponential of the average error e_a^j:

$$W(\overrightarrow{M}^{(j)}) = exp\left(\frac{-e_a^j}{\sigma^2}\right), \tag{3.20}$$

where σ controls the spread of the distribution. The solution can be taken as the Maximum A Posteriori (MAP) estimate defined as:

$$\begin{aligned}\hat{\overrightarrow{M}}(0,2)_{MAP} &= \text{argmax}_{\overrightarrow{M}(0,2)}(\pi(\overrightarrow{M}(0,2)|\overrightarrow{Z}(0:2))) \\ &= \text{argmax}_{\overrightarrow{M}(0,2)} W(\overrightarrow{M}^{(j)}(0,2)).\end{aligned} \tag{3.21}$$

So $\hat{\overrightarrow{M}}(0,2)_{MAP}$ is the estimate corresponding to the highest weight which is equivalent to the one with the lowest e_a as done previously. As an illustration of this, Figure 3.3 shows the distribution of the hypotheses generated for $M(0,2)$ (only the heading is shown) in a simulation example. A 1000 hypotheses are considered, with the 700 ones with the smallest error e_a displayed with a different color. Those 500 ones have a Gaussian like distribution around the true heading. The one with the smallest error is very close to the true heading.

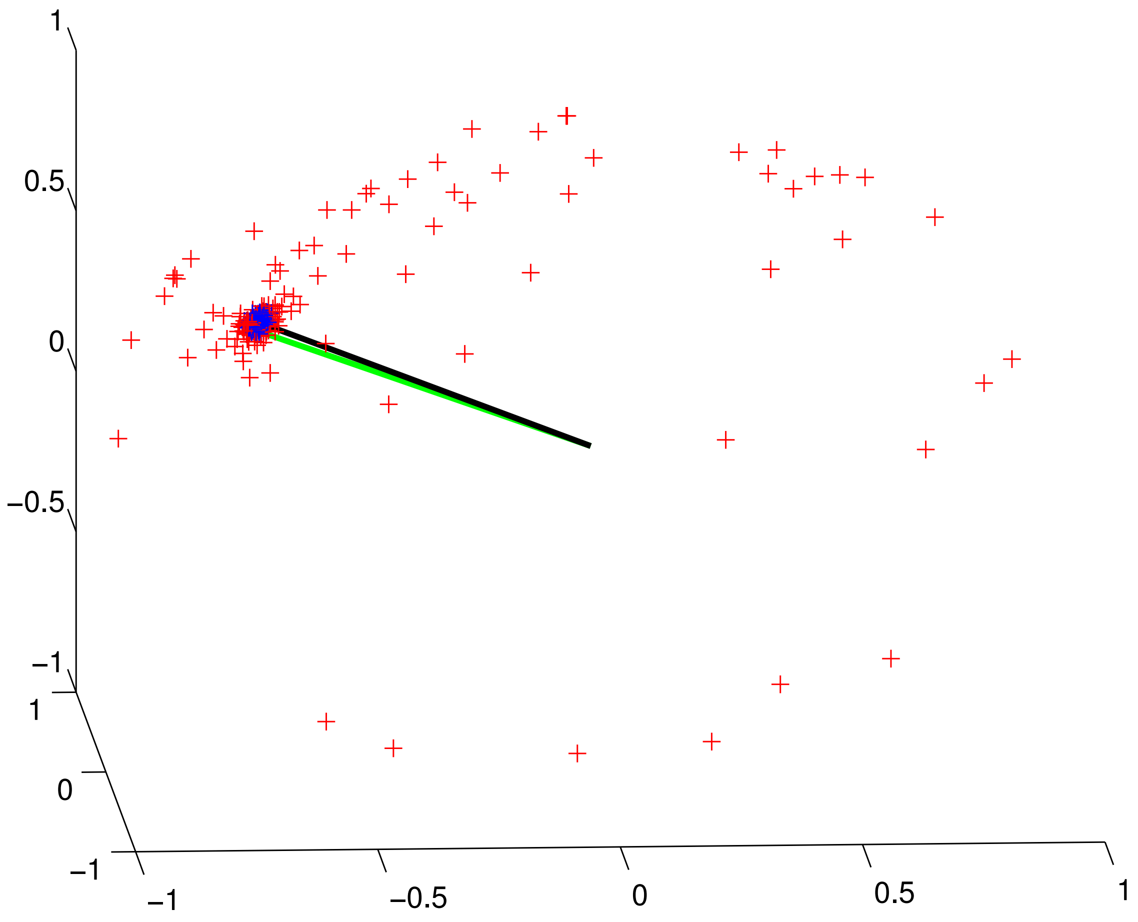


Figure 3.3: Distribution of the translation (heading) on the unit sphere. The green line in the middle points to the true heading and the black line points to the heading corresponding to the smallest error. The blue crosses represent the 500 hypotheses (out of 1000) with the lowest error. Those hypotheses seem to have a Gaussian distribution around the true heading.

The solution as presented in this section, shares many similarities with a RANSAC [39] procedure performed over three frames such as done by Nister *et al.* [90] but with replacing the use of absolute orientation algorithm (Section 2.4.5) by the threading based prediction. This is not novel by itself, but it is presented as a preamble for the multiple frames approach presented in the next section. This approach uses spatio-temporal sampling to provide a fast filtering scheme integrating information from all the previous frames.

3.6 Extension To Multiple Frames

In the case of multiple frames, a similar approach to the previous section is taken. However, in this case the pool of available constraints is much larger. To identify those constraints, assume that at every time instant t, the motions corresponding to all the previous time steps, 0 to $t-1$ are available. The first thing that follow from this, is that the relative motion between any two frames can be determined directly using the motion evolution equations (Equation (2.8)) as follows:

$$\begin{aligned} \overrightarrow{T}(i,j) &= \overrightarrow{T}(j) - \underline{R}(j)\underline{R}(i)^T\overrightarrow{T}(i) \\ \underline{R}(i,j) &= \underline{R}(j)\underline{R}(i)^T \end{aligned} \tag{3.22}$$

Now, substituting $\overrightarrow{T}(i,j)$ and $\underline{R}(i,j)$ in the threading constraint of Equation (3.14) results in an equation of the form:

$$\underline{A}(\overrightarrow{M}(i), \overrightarrow{M}(j), \overrightarrow{q}^k(i), \overrightarrow{q}^k(j), \overrightarrow{q}^k(t))\overrightarrow{M}(i,t) = 0 \tag{3.23}$$

Using the motion evolution equations, $\underline{R}(i,t)$ and $\overrightarrow{T}(i,t)$ can be written as:

$$\begin{aligned} \underline{R}(i,t) &= \underline{R}(t)\underline{R}^T(i) \\ \overrightarrow{T}(i,t) &= \overrightarrow{T}(t) - \underline{R}(t)\underline{R}^T(i)\overrightarrow{T}(i) \end{aligned} \tag{3.24}$$

which can be rearranged in the form $\overrightarrow{M}(i,t) = \underline{B}(\overrightarrow{M}(i))\overrightarrow{M}(t)$. Therefore, Equation (3.23) can be written as:

$$\underline{A}(\overrightarrow{M}(i), \overrightarrow{M}(j), \overrightarrow{q}^k(i), \overrightarrow{q}^k(j), \overrightarrow{q}^k(t))\underline{B}(\overrightarrow{M}(i))\overrightarrow{M}(t) = 0 \tag{3.25}$$

This equation means that as illustrated in Figure 3.4, every point matched in the frames i,j and t along with the camera poses at times i and j give a two-equations constraint on the motion $\overrightarrow{M}(t)$. This is very important because it leads to the following: a point matched in m frames other than the last frame yields $\frac{m(m-1)}{2}$ pairs (i,j), therefore, it gives rise to $m(m-1)$ equations in $\overrightarrow{M}(t)$. As $\overrightarrow{M}(t)$ is 12-dimensional, then $\overrightarrow{M}(t)$ can be determined from at least:

1. Only 1 point matched in 4 previous frames.
2. Two points matched in 3 previous frames each. The three frames in which those two points appear do not have to be the same.

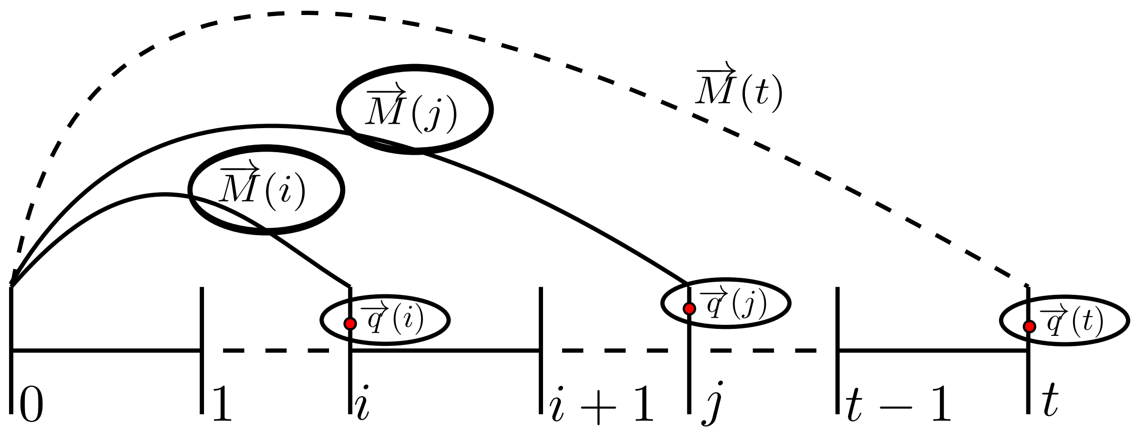


Figure 3.4: Information available at every time t: The vertical lines represent frames at different times. Every point matched in any two frames i and j and in the last frame t, with the motions $\vec{M}(i)$ and $\vec{M}(j)$ gives a 2-equations constraint on $\vec{M}(t)$

3. 1 point matched in 3 previous frames and 3 other points matched in two previous frames each.
4. 6 points matched in 2 previous frames each.

This result is new as it allows to determine the motion at time t using only 1 point matched in a set of previous frames. Also it has very important implications in filtering, as now multiple previous motions can contribute together in the determination of $\overrightarrow{M}(t)$.

Additionally, every set of five points, matched in any two frames i and j and time t give, via the essential matrix threading method using the motion $\overrightarrow{M}(i,t)$ determined as in Equations (3.24), an estimate $\hat{\overrightarrow{M}}(t)$ for $\overrightarrow{M}(t)$. This estimate can also be considered as a 12 equations constraint in $\overrightarrow{M}$ as mentionned earlier ($\underline{I}_{12}\overrightarrow{M}(t) = \hat{\overrightarrow{M}}(t)$).

From a maximum likelihood point of view the best estimate can be obtained as the one that minimizes the sum of all the constraints over all the previous frames and over all the features:

$$\begin{array}{rcl} \hat{\overrightarrow{M}}(t) & = & \operatorname{argmin}_{\overrightarrow{M}(t)}(c_1 + c_2) \\ c_1 & = & \sum_{i=0}^{t-2}\sum_{j=i+1}^{t-1}\sum_{k=1}^{N}(\underline{A}(\overrightarrow{M}(i), \overrightarrow{M}(j), \overrightarrow{q}^k(i), \overrightarrow{q}^k(j), \overrightarrow{q}^k(t))\underline{B}(\overrightarrow{M}(i))\overrightarrow{M}(t))^2 \\ c_2 & = & \sum_{i=0}^{t-2}\sum_{j=i+1}^{t-1}\sum_{l=1}^{\binom{N}{5}}(\underline{I}_{12}\overrightarrow{M}(t) - \hat{\overrightarrow{M}}(t))^2 \end{array} , \tag{3.26}$$

where $\binom{N}{5}$ is the number of all combinations of 5 features out of all the available N features. The solution of this problem can be determined by stacking all those linear constraints in one linear system and solving it for $\overrightarrow{M}(t)$. However, as shown in figure 3.4, having N points matched in m frames, the total number of constraints would be $m(m-1)N$. This means that in the case of 100 features matched in 100 frames more than 1000000 constraints need to be considered. This is too expensive computationally to be done online. Therefore, as done in the previous section, random sample consensus is used for the esimtation, however in this case, the constraints are generated via a spatio-temporal sampling process spanning both features and frames as described in the following.

3.6.1 Spatio-Temporal Random Sample Consensus

At every time t, hypotheses for the motion $\overrightarrow{M}(t)$ can be generated in one of three methods:

1. Repeat the following 6 times:

   - Chose at random two previous frames i and j, and then chose randomly a feature that is matched in i, j and t. Then, use equation 3.25 to obtain 2 equations in $\overrightarrow{M}(t)$.

   This would result in a system of 12 equations which can be solved to get a hypothesis of $\overrightarrow{M}(t)$.

2. Chose randomly two previous frames i and j, then chose randomly 5 features matched in i, j and t, then use the essential matrix based threading to determine a hypothesis for $\overrightarrow{M}(t)$.

3. Every solution from the second method results in a 12 equations constraint. By choosing some of those equations and adding them to equations generated as in the first method, a system of 12 equations is obtained and solved for $\overrightarrow{M}(t)$.

An important question here is what method to use. The first one is faster, but it leads to error accumulation (Section 3.3.3) especially when initializing from only one motion. The second method is more expensive, but it doesn't lead to errors accumulation. Therefore, we propose a scheme based on drawing hypotheses from both methods one and two. This also ensures more diversity in the generated hypotheses. As the first method is faster, more hypotheses are generated based on it. For instance, we used 200 hypotheses from the first method and 50 from the second one. (We haven't explored using hypotheses from the third method).

As in the previous case to select the best hypothesis, the hypotheses should be evaluated using the epipolar errors of the features between the last frame at time t and all previous frames sharing features with that frame. The average error of the j^t hypothesis is defined as:

$$e_a^j = \frac{\sum_i^N \sum_{k=f_i}^t \rho(e^{i,(j)}(k,t))}{N}, \tag{3.27}$$

where f_i is the first frame in which the i^{th} feature is detected. As the above error is too expensive to compute for all the hypotheses in real time, we approximate it by using, for every feature, the first frame in which it appeared and the last frame at t. The expression of the approximated error is then written as:

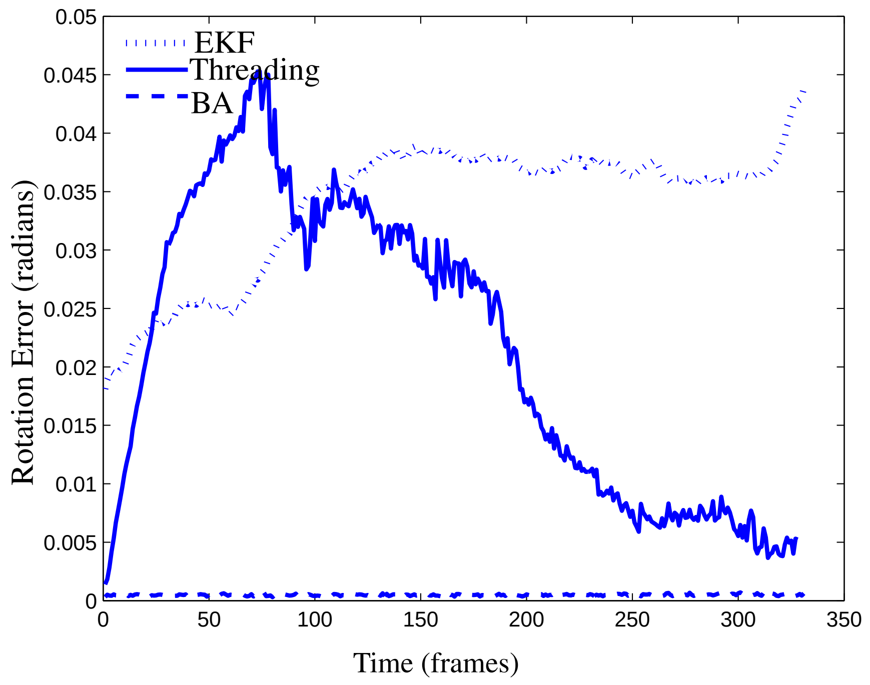
$$e_a^j = \frac{\sum_i^N \rho(e^{i,(j)}(f_i,t))}{N}. \tag{3.28}$$

3.7 Experimental Results

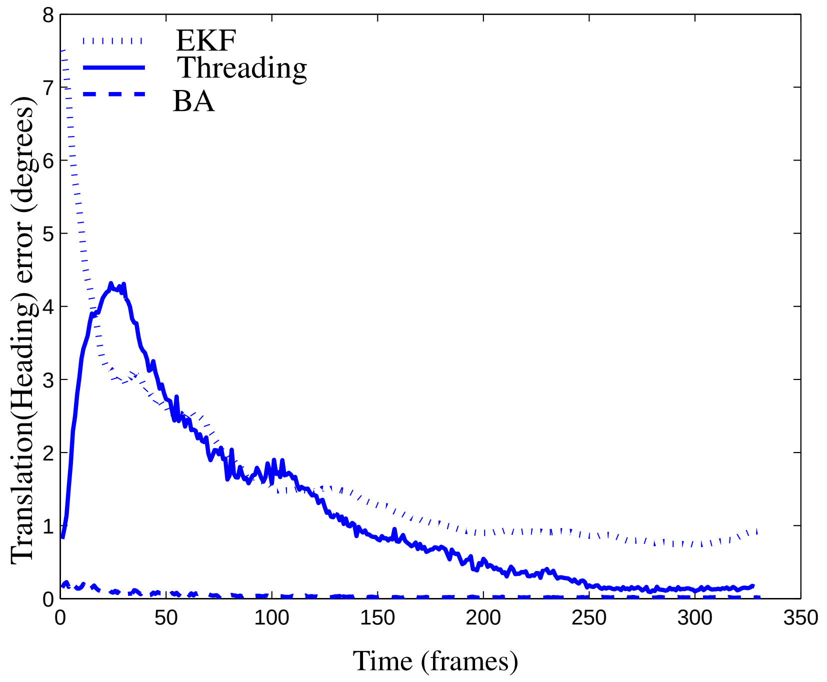
In this section simulated and real results are presented to show the performance of the proposed threading filter.

3.7.1 Simulated Data

We randomly generated $N = 50$ 3D points within a cube of size $4 \times 4 \times 4$ centered at $(0,\ 0,\ 3)$ (the unit is irrelevant). The camera is assumed to be at the origin of the inertial frame looking towards the $\overrightarrow{z}$ direction. Then a random motion is applied to the camera in such a way that it is always fixating at the cloud. Zero mean Gaussian noise is added to the projections of the 3D points on the camera frame. The results presented below correspond to the average of 50 runs of 400 frames each. For every run the results of the threading filter are compared with both the EKF and the optimal BA. Figure 3.5 shows the mean translation and rotation errors for the threading, EKF and BA filters over the 50 runs. In this figure, the translation error of the threading filter with respect to BA is expressed as the difference between the heading directions in degrees. The rotation error is determined as $||\hat{\underline{R}}\underline{R}^T - I_3||$ where $\hat{\underline{R}}$ is the estimated motion and $\underline{R}$ is the true rotation. This error combines both the error in the axis of rotation and in the



(a) Rotational error



(b) Translational Error

Figure 3.5: Performance of the proposed threading filter vs the EKF and BA. The plots show the mean errors over 50 runs with different motions and structures.

magnitude of rotation. In the case where both rotations are about the same axis, this error is be equivalent to the magnitude of the rotation difference in radians. For this reason “radians” will be used as a unit for this error to reflect the equivalent error if the directions of rotations were the same. The estimates of the threading filter are more than three times more accurate than the EKF. Therefore, the performance of the threading filter is similar or better than the performance achieved by the GLF filter of Eade and Drummond [34] as it was shown in Section 2.5.3 that the performance of the GLF is almost three times better than the EKF. Also, from a computational perspective, up to 200 hypotheses can be generated and tested within 30 ms on a Pentium(M) 2.13 GHZ (MATLAB implementation). This makes the filter very suitable for real time processing. The hypotheses can be processed in parallel and hence can highly benefit from multi-core machines and other parallel architectures. When increasing the rate of change of the camera velocity between successive frames, the EKF results tend to deteriorate while the threading filter retains the same performance. An important phenomenon that is observed in the results is that the error starts growing at first and then drops until it reaches convergence. This is due to fact that, at the beginning, the number of previous frames is small, and hence the temporal sampling is not very efficient since the pool to sample from is limited. Also, due to the fact that more hypotheses are generated from the linear approach, most of the motions at the first few frames would be determined using the linear approach and hence involve the accumulated error from the first motion.

3.7.2 Real Images

To test the threading filter on real images, sequences from the Rawseeds database [1] are used. Figures 3.6 and 3.7 show two samples from two sequences of this database. To show the ability of the threading filter to work with large baselines, the sequences were sub-sampled taking every 15^{th} frame only. The image points $\overrightarrow{q}$ are detected using the the FAST feature detector [103]. The detected features are matched between successive frames using the SURF feature descriptor [15]. The tracks of the features on the first frames of those sequences are shown in Figures 3.6 and 3.7. In the second sequence the motion varies more significantly between frames. Note the presence of many arrowheads which are the result of erroneous tracking.

The initial two motions $\overrightarrow{M}(0)$ and $\overrightarrow{M}(1)$ are determined for the first three frames using the five-point-algorithm [88] with the three points algorithm [43] within a RANSAC loop followed by non-linear minimization after rejection of the outliers. To give an idea about the quality of this initialization, Figure 3.8 shows the matching between the first two frames (a) and the re-projected estimates overlaid on top of them in red (b).

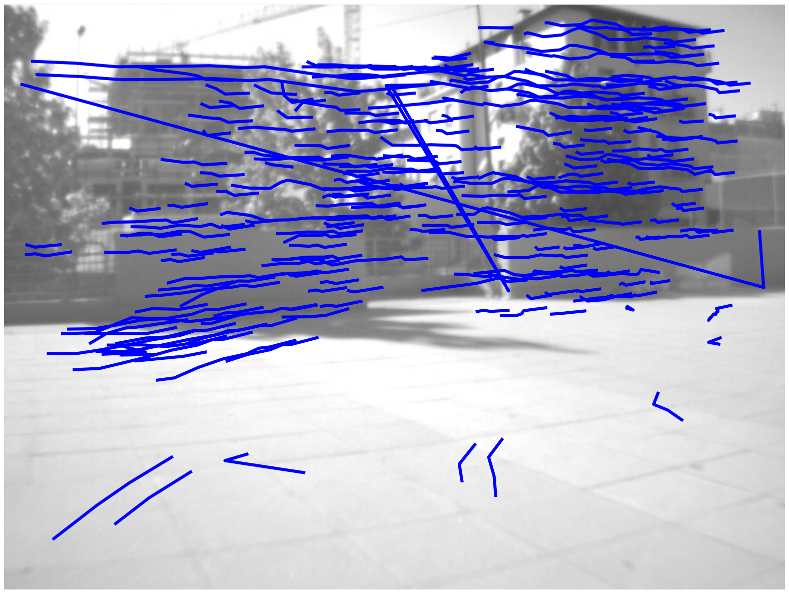


Figure 3.6: Tracks through the first few frames of the first sequence shown on the first frame of this sequence.

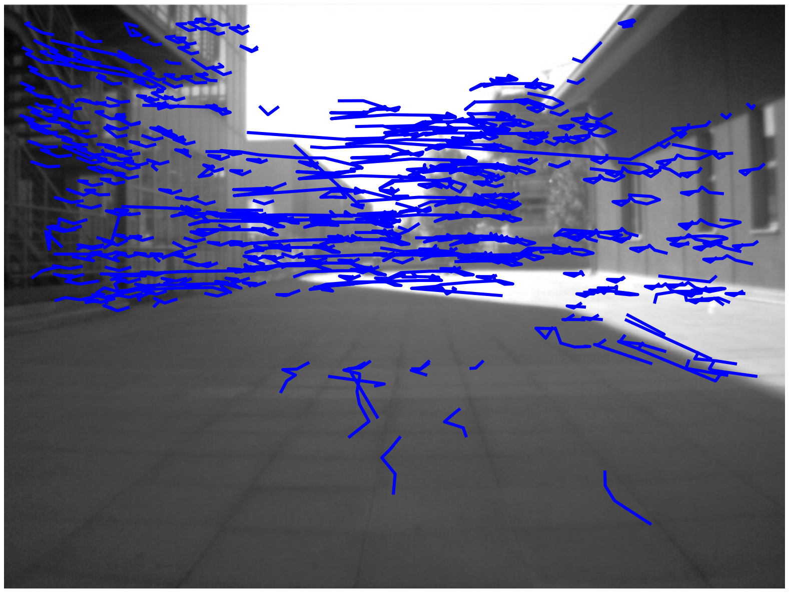
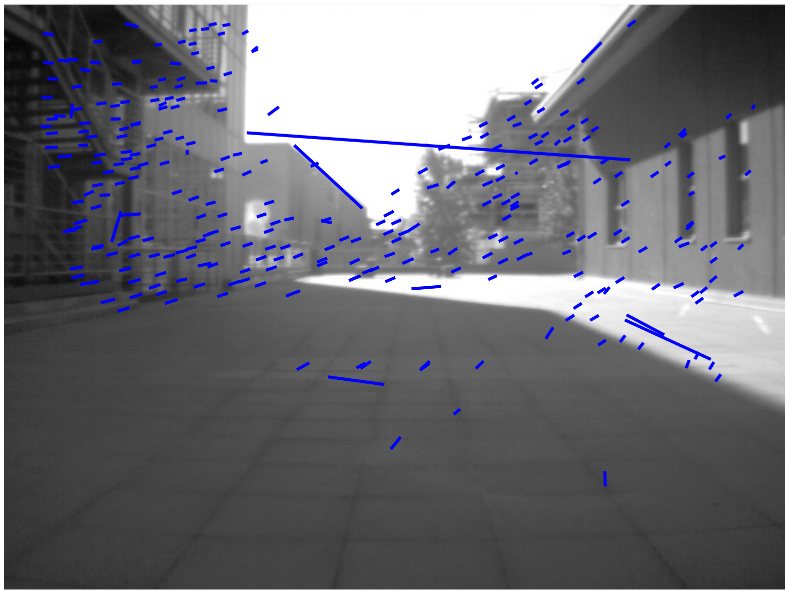
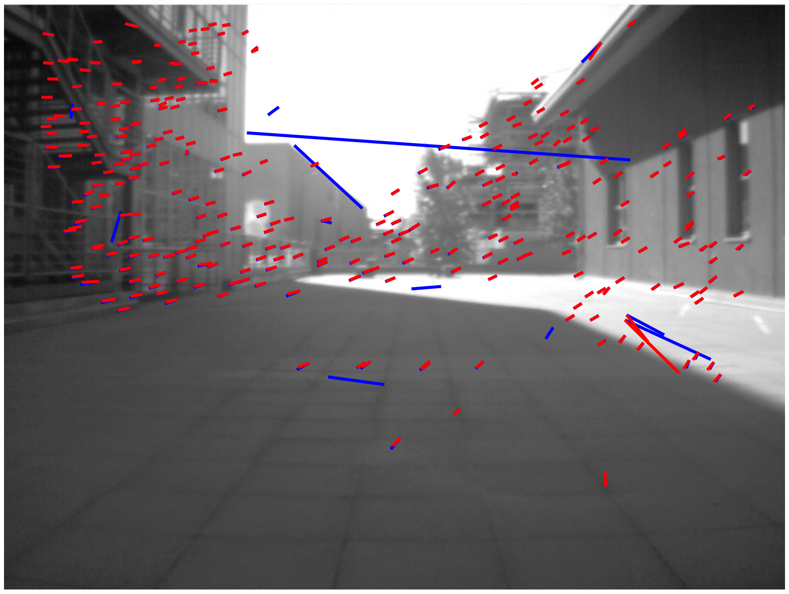


Figure 3.7: Tracks through the first few frames of the second sequence shown on the first frame of this sequence. The motion does not change smoothly between frames.

To assess the performance of the threading filter, and since no ground truth is available for those sequences, it is compared to the estimates of the optimal BA done over all the considered frames (200 frames). To give an idea about the accuracy of this BA solution we provide, in 3.9 and 3.10, the re-projection error (in pixels) of the estimates obtained by BA on all the frames in the two sequences. Notice that in the second se-



(a)



(b)

Figure 3.8: (a) Points tracked from the first frame to the second, the displacement is about 14 pixels on average. (b) The re-projected points from the first estimation overlaid on the measured ones.

quence, the re-projection error is larger due to the fact that the motion is not as smooth as in sequence one and hence the matching is the less accurate. The results of the threading filter on the two sequences are shown in Figures 3.11 and 3.12. The translation and rotation errors are expressed in terms of the distance from the BA estimates. The distance

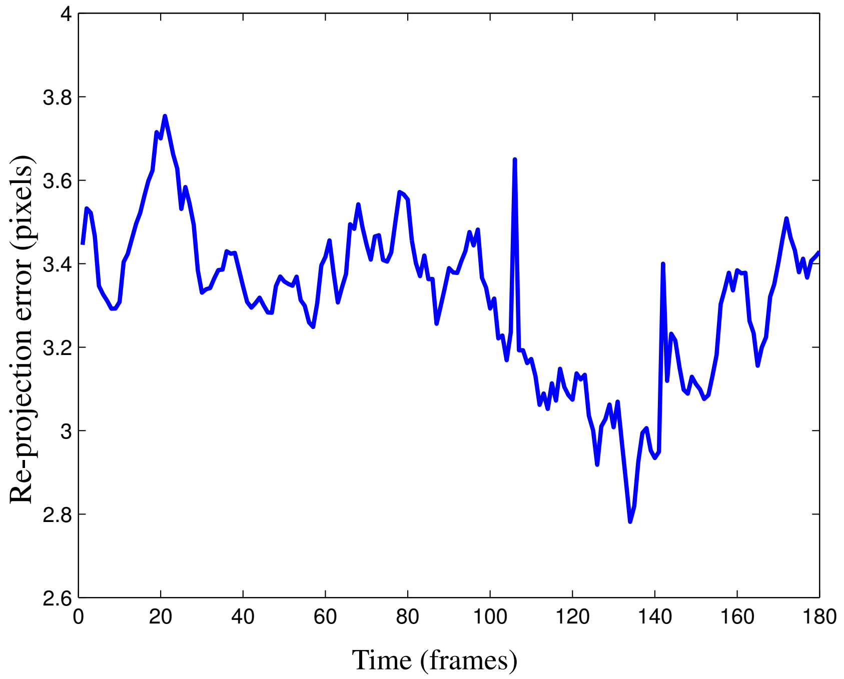


Figure 3.9: The BA re-projection error on the frames of the first sequence.

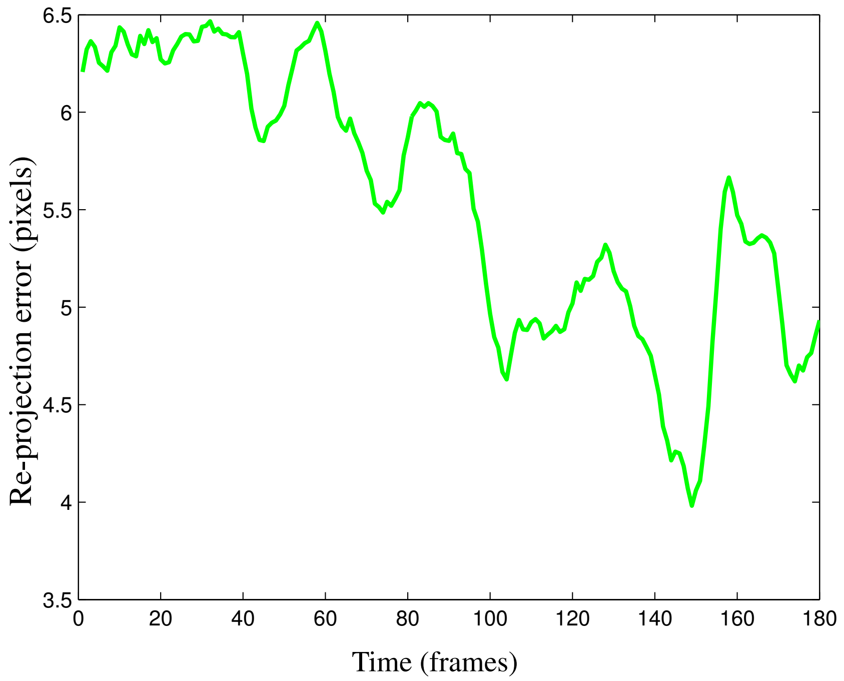
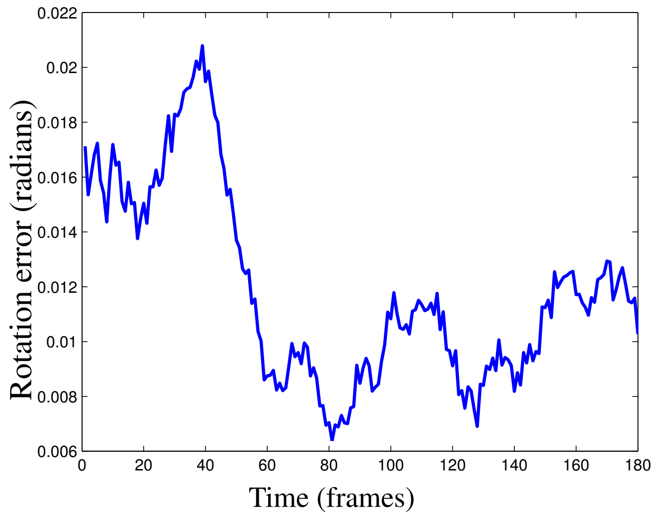
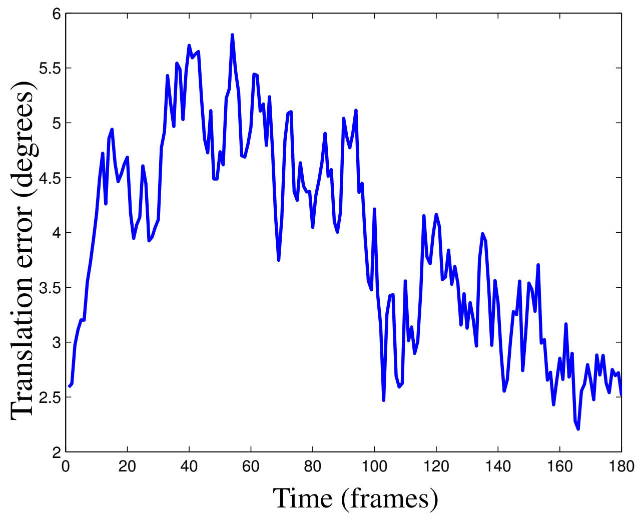


Figure 3.10: Re-projection errors of the BA for the second sequence.

in the heading between the threading filter and the bundle adjustement is about 3.5 degrees on average. The distance between the rotation is 0.014 radians on average or (0.8 degrees). This shows the closeness of the estimates to the BA estimates and validates the good accuracy achieved by the filter. The results follow the same pattern as the simula-



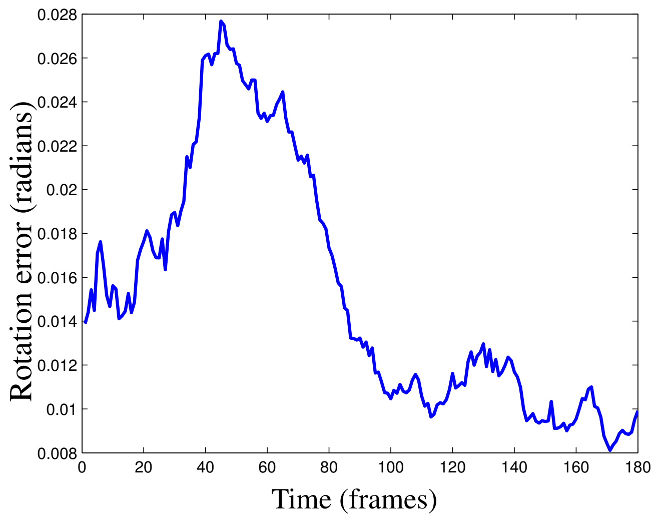
(a) Rotational error



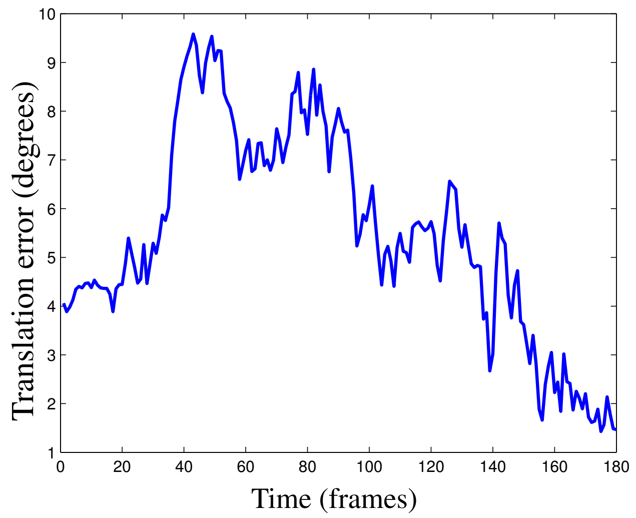
(b) Translational Error

Figure 3.11: Errors of our approach with respect to the BA solution for the first sequence. Those errors, being small, show that the estimates of the threading filter are close to the BA estimates which validates the conclusions drawn from the simulation results about the good accuracy achieved by the filter.

tion results. Again we see that the errors grows initially, however when the number of previous frames becomes larger, the error starts dropping until it stabilizes.



(a) Rotational error



(b) Translational Error

Figure 3.12: Errors of our approach with respect to the BA solution for the second sequence.

3.8 Conclusion

This chapter presented an approach capitalizing on the algebraic threading constraints for the estimation of the motion-only from a sequence of tracked features without need to determine the depth of the features. Linear constraints linking the motion at time t with any two previous motions at times i and j have been derived. We also devised a scheme based on spatio-temporal random sample consensus to efficiently capitalize on all the available constraints. The performance of this approach has been shown to perform at least as good as the state of the art GLF filtering approach [34]) in terms of motion estimates accuracy with better real-time characteristics and with a higher ability to deal with a large number of measurements and large baselines.

The presented approach can be enhanced in several ways. Currently, samples are drawn totally randomly. As we are only drawing a limited set out of a very large number of constraints, it would be better to come up with a more clever way to select the most diverse and accurate set of features. One approach would be for example to consider only three frames for every features and to chose those as to maximize the signal to noise ratio. Under the assumption that the SNR is maximal for the widest displacement the best three frames to chose for every point would be the first, last and middle frame. Another approach would be to try to come up, for each feature, with one artificial constraint that is equivalent to all the constraints involving this feature. This is can be done by iteratively adjusting the projections of a given feature in all the frames untill all the constraints involving this feature become equivalent.

Also, another vertical for improvement is to maintain for every frame, not only a single estimate of the motion, but a Gaussian distribution represented by a mean vector and a covariance matrix. Assuming that we know the noise distribution in the feature matches and the covariance of the motions up to $t-1$, Then when performing the inference of $\overrightarrow{M}(t)$ using two randomly selected frames i and j (as in Section 3.6), the covariance of $\overrightarrow{M}(t)$ can be also inferred using covariance propagation techniques (Section A.4). This is a simple operation and requires only the computation of the Jacobian of Equation (3.10).

Another advantage of the presented approach is that it can be very easily integrated with other motion sensors, whether they provide absolute position measurements such as GPS, or relative measurements such as inertial sensors or odometers. Those measurements can be included easily as in equation 3.25 the motions $\overrightarrow{M}(i)$ and $\overrightarrow{M}(j)$ do not have to be vision estimates, but can directly use the estimates of other sensors. In this case, depending on the relative uncertainty of those sensors with respect to vision, a portion of the generated hypothesis would be using previous motions from those sensors.

Chapter 4

Efficient Augmentation of the Analytic SFM Estimators with Frame-to-Frame Features

Two categories of features are used in SFM estimation (Figure 4.1): features that are tracked for an extended number of frames (referred to as "tracked features" in this chapter) and features that are matched in pairs of consecutive frames only (referred to as frame-to-frame features). While tracked features provide more information and are more useful for the integration of the 3D estimates over time, the frame-to-frame features still carry important information about the incremental motion (velocity) and hence about all the other estimates. That is why they have been used in BA [136] and in some particle filtering approaches [33], in conjunction with the tracked features, to improve the accuracy of the estimates. However, in analytic filters, the frame-to-frame features have not been exploited, principally because the cost would be too high and also because different filtering approaches would need to address them differently. In this chapter, we provide a filtering approach that allows the information from the frame-to-frame features to be incorporated within any analytic filter in a very computationally efficient way. This approach involves two contributions:

- Providing a new way to incorporate the frame-to-frame information via a separate additional filtering step that can be easily added to any analytic filter with minimal change.
- Reducing the complexity of this additional filtering step by orders of magnitude so as to be able to process more than 200 frame-to-frame features in less than 5 ms.

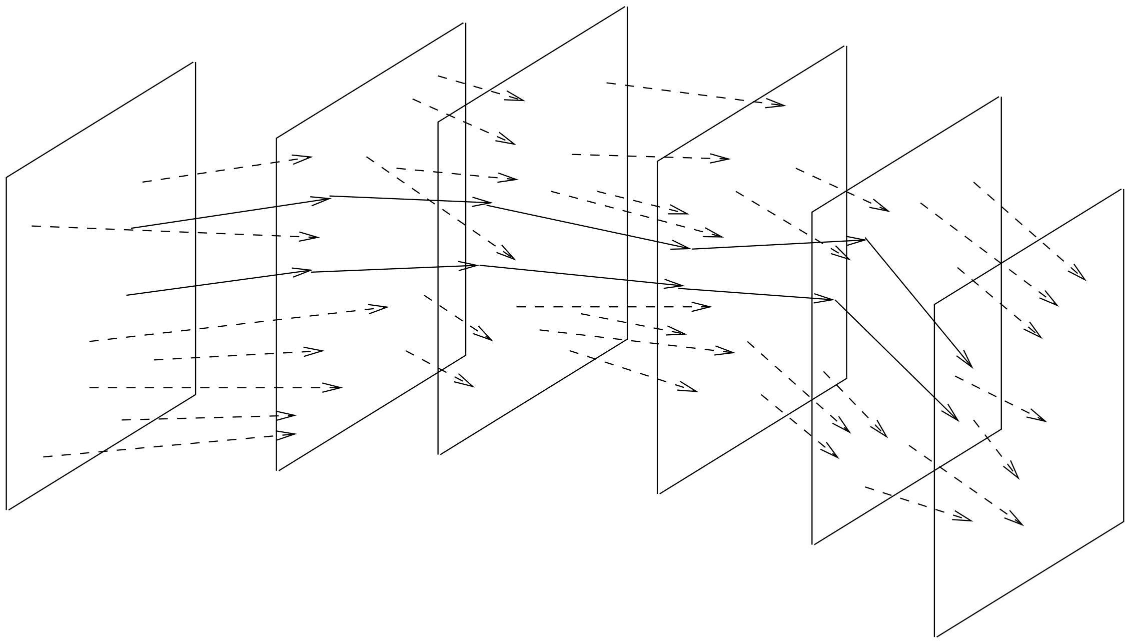


Figure 4.1: Two types of image measurements for SFM estimation: Features tracked over many frames (solid lines). Frame-to-Frame features (Dashed arrows).

4.1 Introduction

This chapter is concerned with augmenting the analytic SFM filters using features tracked in several frames, with frame-to-frame features. Frame-to-frame features are features that are matched only in two consecutive frames. They have been incorporated earlier in the context of BA by Zhang [136] and have been also used in Particle Filters by Eade and Drummond [33] but only through weighing the particles. The rationale for using the frame-to-frame features is that a large number of points can be matched between consecutive frames and the success of odometry based approaches using a large number of points to compute the incremental motion [70,71,75,89]. Also, another main motivation is that the features between adjacent frames can be tracked using optical flow techniques such as [78] while tracking over many frames requires matching of features detected at every frame and has generally higher noise levels in the features localization.

In the context of analytic SFM filters (i.e., filters using an analytic representation of the distribution of the state vector as opposed to particles), we are not aware of any approach that adds frame-to-frame features to filters using tracked features. There have been some approaches using frame-to-frame features only (not as an addition to the tracked features) such as the "essential" filter of Soatto and Perona [115] using the epipolar constraint as the measurement equation and the subspace filter also by Soatto and Perona [118] using the subspace method of Jepson and Heeger [61] based on optical flow as measurement mechanism. The problem of using frame-to-frame features only is that the translation magnitude between different frames cannot be estimated relative to a common gauge. Also those filters have cubic computational complexity in the terms of the frame-to-frame features and hence are not able to run in real-time with a large number of features.

The frame-to-frame features can be considered as differential measurements in the sense that they give information about the change in the pose between every two frames, in contrast to the tracked features which provide absolute measurement for the pose and the depth. Absolute here is defined with respect to a fixed gauge. Section 4.2 presents a toy problem illustrating through a simple example the effect of adding differential measurements to a filter using absolute measurements. For the SFM problem, casted as a dynamical estimation problem, adding the frame-to-frame measurements results in an additional measurement equation per frame-to-frame feature. This is somehow problematic as the additional frame-to-frame measurement equation is an implicit measurement equation (defined as $h(\overrightarrow{S}, \overrightarrow{Z}) = 0$ instead of $\overrightarrow{Z} = h(\overrightarrow{S})$. This requires a specific treatment that varies with the type of filtering adopted. For instance, with the EKF and GLF a solution such as a first order Taylor expansion about the measurements is required, while the UKF and particle filtering can handle the implicit measurements directly. To avoid this problem, we propose to incorporate the frame-to-frame features in an extra filtering step done right after the main filtering step (using the tracked features). In theory, given two independent sets of observations, performing a filtering using both sets simultaneously is exactly equivalent to filtering using one of the sets then filtering using the other. However, in the case of SFM, an important problem needs to be addressed. The frame-to-frame features do not provide a way to fix the Euclidean similarity to the same Gauge used by the state vector as they don't provide any observation for the parameters that can be used to fix this gauge (see Section 4.3). This requires extra processing to bring the filtered state vector back to its gauge before the filtering. We propose a solution to this problem which enables us then to include the frame-to-frame features in an separate filtering step to which will be referred to as "extra" filtering while we refer to the filter with the tracked features by the term "main filter". A flowchart of a filtering iteration is shown in Figure 4.2. The benefits of having a separate filtering are numerous:

- It can be added to any estimator as long as it maintains a mean vector and a covariance matrix. It can be added to existent implementations with minimal coding

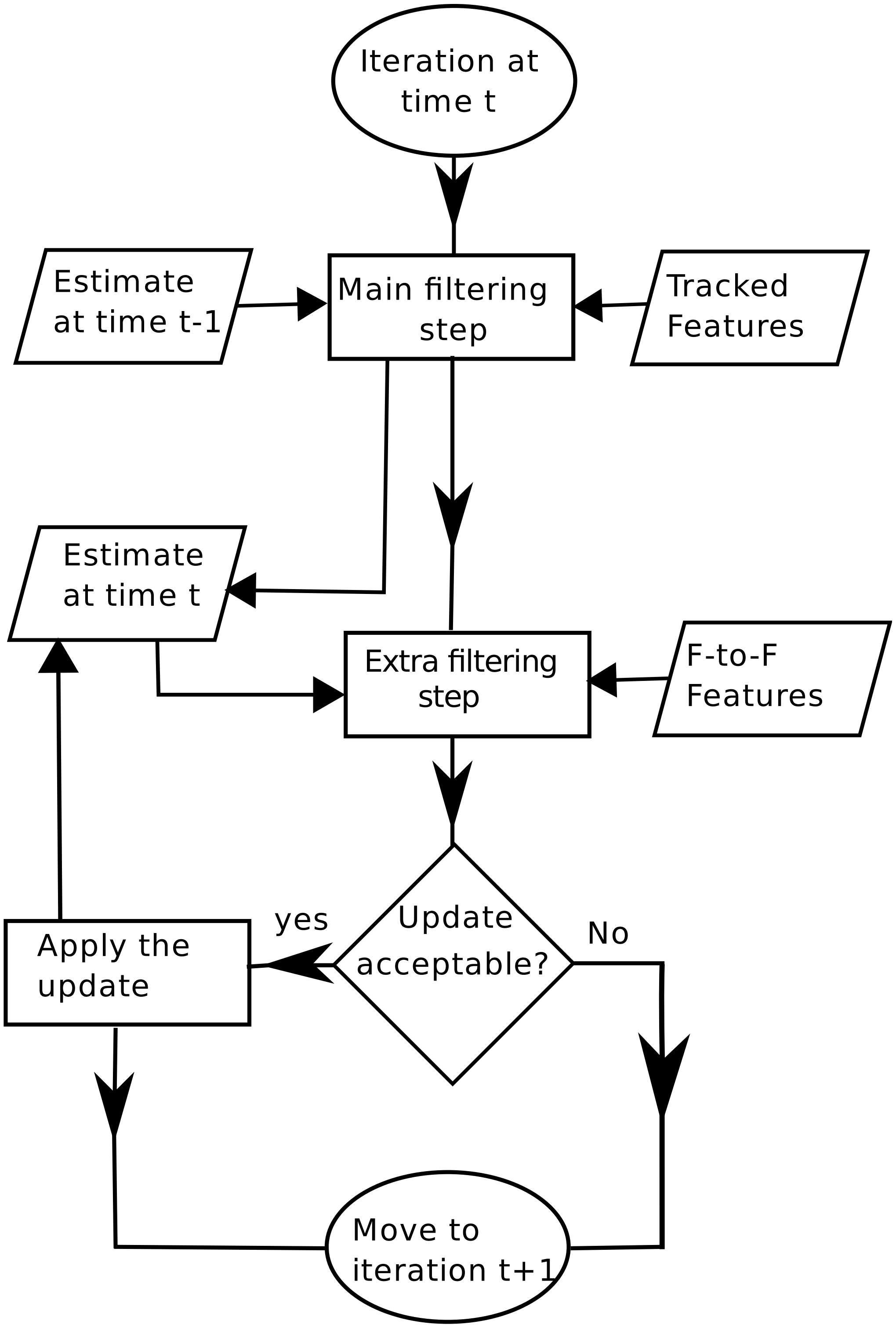


Figure 4.2: The filtering step using frame-to-frame features does not interfere with the internal operation of the main filter. It only takes the output of the main filter between two iterations and modifies it to account for the Frame-to-Frame features information.

changes. Actually it can be coded as a generic function and adding it to a filter would require only creating an interface for this function and then calling it after every iteration of the main filter.

- The results of the filtering can be accepted or rejected based on some criteria such as the epipolar error of the frame-to-frame features, the number of outliers or the extent of change in the state vector.
- It can be divided into several independent steps, which allows the use of Robust techniques based on RANSAC [38].
- It allows to reduce the computational cost of the filtering with frame-to-frame features.

The cost of the extra filtering is initially cubic in the number of added frame-to-frame features (and in the size of the filtered state vector also). However, we rely on two facts to achieve a reduction of this cost by order of magnitudes: (1) The noise vectors in different frame-to-frame features are assumed to be statistically independent and hence their covariance matrix is bloc diagonal. (2) The frame to frame features only affect directly a small part of the state vector which is the velocity. Their effect on the other part of the state vector is only through the covariance matrix of the state vector. The approach presented to capitalize on this is to update first the velocity estimates using the frame-to-frame features. We introduce a method to efficiently perform this update by using the Sherman-Morrison-Woodbury formula for inverting sums of matrices [53]. Then, using the covariance matrix of the state vector, the update is propagated to the remaining elements of the state vector.

The remainder of this chapter is as follows: Section 4.2 presents a toy problem illustrating the effect of using differential measurements with absolute measurements. Section 4.3 presents the frame-to-frame features constraint, and the proposed solution (including dealing with the gauge problem). Section 4.4 addresses the problem of complexity reduction. Finally, Section 4.5 presents some experimental results.

4.2 Toy Problem with Absolute and Differential Measurements

To illustrate the usefulness of adding frame-to-frame features we present a toy problem consisting of a linear estimation problem (in which the Kalman filter is optimal). In this toy problem, the aim is to estimate the value of a scalar variable $x(t)$ from two types of observation. The first one, $m(t)$, is an observation of $x(t)$ itself (simulating the tracked features) and the second are multiple observations $p_i(t)$ of the rate of change of x. Therefore, a dynamical system is formulated as follows:

$$\begin{aligned} x(t) &= x(t-1) + v(t) && (4.1) \\ v(t) &= v(t-1) + n_v, \quad n_v =\sim N(0, \sigma_v) && (4.2) \end{aligned}$$

The measurements for this system are provided by the equations:

$$\begin{aligned} m(t) &= x(t), \quad , n_m = \mathcal{N}(0, \sim_m) && (4.3) \\ p_i(t) &= v(t) + n_{pi}, \quad , n_{pi} =\sim N(0, \sigma_{pi}) && (4.4) \end{aligned}$$

This system can be solved optimally with straightforward Kalman Filtering. A simulation with a 100 different runs yields the results shown in Figure 4.3. This figure displays the root mean square errors of: 1) The measurements without filtering, 2) filtering using the absolute measurements only, 3) filtering using absolute measurements and 10 differential measurements. 4) filtering with absolute measurements and 20 differential measurements. The results show that the use of differential data increases the accuracy and the more differential data is used, the better is the result. In this example, we assumed the same level of noise in both the discrete and differential data. In the case of SFM this is not quite accurate. The noise in the discrete feature matches is usually larger than the differential data, because the matching of the discrete features is done between far apart frames, while the differential data consists of features matched in adjacent frames and hence the noise in the matching is much less. If the noise in the differential data is reduced to half in our toy problem, we get the results shown in Figure 4.4. This figure, shows the filtering with discrete data only, the filtering with discrete and differential data with the same noise level and filtering with discrete and differential data with the noise level in the differential data equal to half of the noise level in the discrete data. The differential data with the lower noise level reduces the total estimation error by almost half.

4.3 SFM with Frame-To-Frame Constraints

The frame-to-frame information can be considered as K discrete features $\overrightarrow{q}^j(t)$, $\overrightarrow{q}^j(t+1)$ $j = 1, ..., K$ matched between every two consecutive frames t and $t+1$, or as K

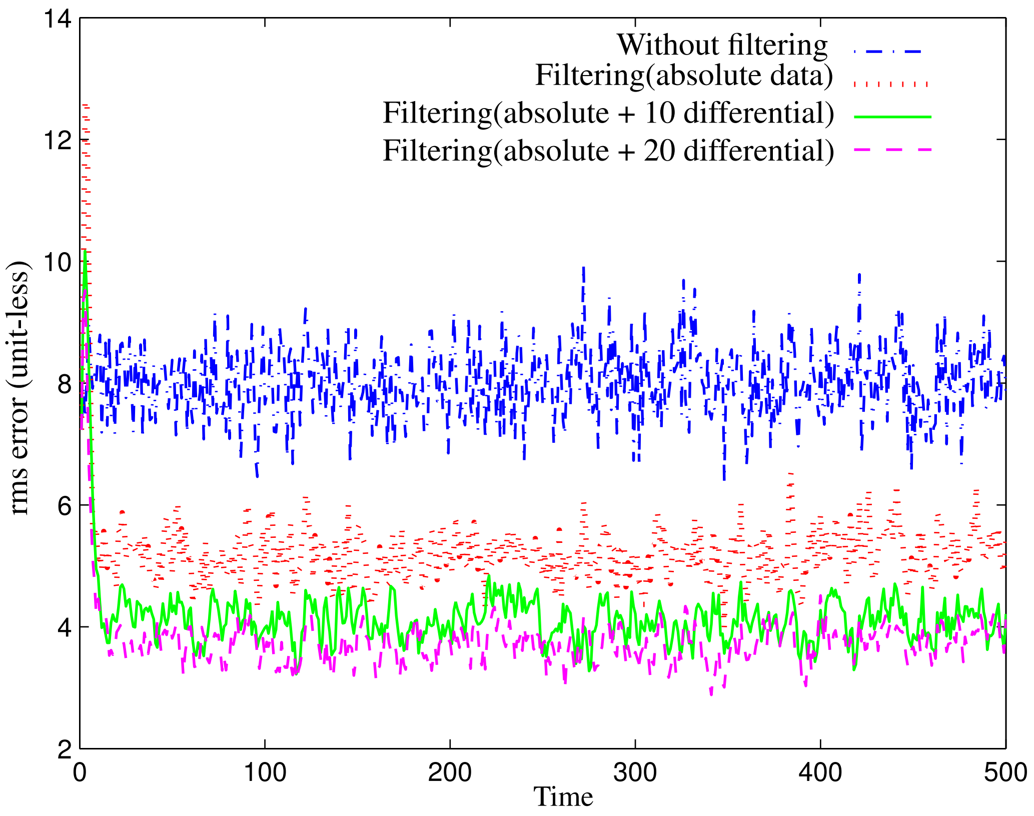


Figure 4.3: Toy problem representing the use of a large number of differential measurements with absolute measurements. The effect of the differential data in reducing the estimation error is clear.

optical flow vectors $\dot{\overrightarrow{q}}^j(t)$ estimated at the frame t. For these features, we are only interested in the information they carry about the incremental motion (velocity) $[\overrightarrow{\omega}; \overrightarrow{V}]$. Therefore, the depth of these points should be eliminated to obtain a constraint that links the incremental motion between $t-1$ and t to the frame-to-frame correspondences. The Sampson's approximation to the epipolar error introduced in Equation (2.23) provides such constraint for the discrete case. Similarly, the geometric constraint in Equation (2.25) is a good choice for the differential constraint. In either cases, the constraint can be written in the form of an objective function h^2:

$$h^2(\overrightarrow{S}(t), \overrightarrow{Z}^2(t)) = 0, \tag{4.5}$$

where $\overrightarrow{S}(t) = [\overrightarrow{\Omega}(t); \overrightarrow{T}(t); \overrightarrow{P}^1; ...; \overrightarrow{P}^N; \overrightarrow{\omega}(t); \overrightarrow{V}(t)]$ is the state vector, and $\overrightarrow{Z}^2(t)$ is a vector containing the discrete or continuous frame-to-frame features. Conversely, $\overrightarrow{Z}^1(t)$ will be containing the N features $\overrightarrow{q}^i(t+1)$ $i \in \{1, .., N\}$ tracked over many frames.

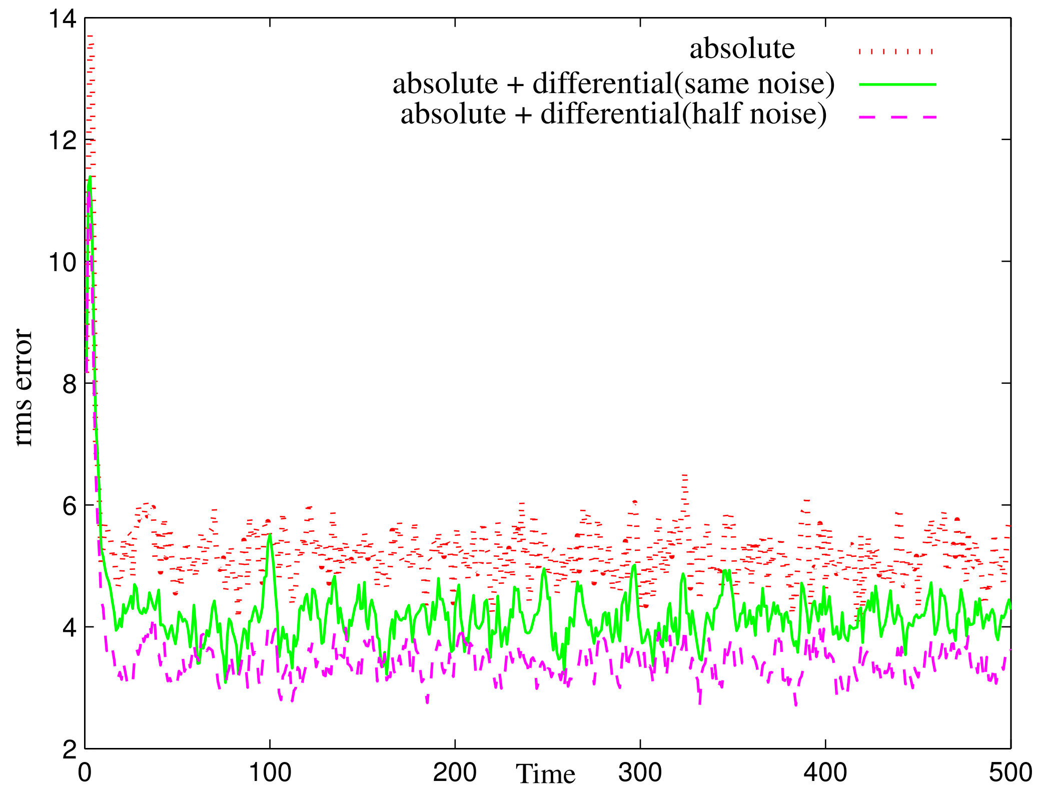


Figure 4.4: Improvement due to differential measurements with different noise levels. The data with the lower noise level results in twice the improvement.

Those features are related to the state vectors in a equation of the form:

$$\overrightarrow{Z}^1(t)) = h^1(\overrightarrow{S}(t)), \tag{4.6}$$

where the function h^1 represents the discrete SFM constraint (Equation (2.12)).

4.3.1 Dynamical System with Tracked Features and frame-to-frame features

A dynamical system for SFM with Equations (4.6) and (4.5) can be written as follows:

$$\left\{ \begin{array}{ll} \overrightarrow{S}(t+1) = f\left(\overrightarrow{S}(t)\right) + \overrightarrow{n}_{\overrightarrow{S}}(t) & \overrightarrow{n}_{\overrightarrow{S}}(t) \sim \mathcal{N}(0, \Sigma_{\overrightarrow{S}}) \\ \overrightarrow{Z}^1(t) = h^1\left(\overrightarrow{S}(t)\right) + \overrightarrow{n}_{\overrightarrow{Z}^1}(t) & \overrightarrow{n}_{\overrightarrow{Z}^1}(t) \sim \mathcal{N}(0, \Sigma_{\overrightarrow{Z}^1}) \\ h^2\left(\overrightarrow{S}(t), \overrightarrow{Z}^2(t) - \overrightarrow{n}_{\overrightarrow{Z}^2}(t)\right) + & \overrightarrow{n}_{\overrightarrow{Z}^2}(t) \sim \mathcal{N}(0, \Sigma_{\overrightarrow{Z}^2}) \end{array} \right. \tag{4.7}$$

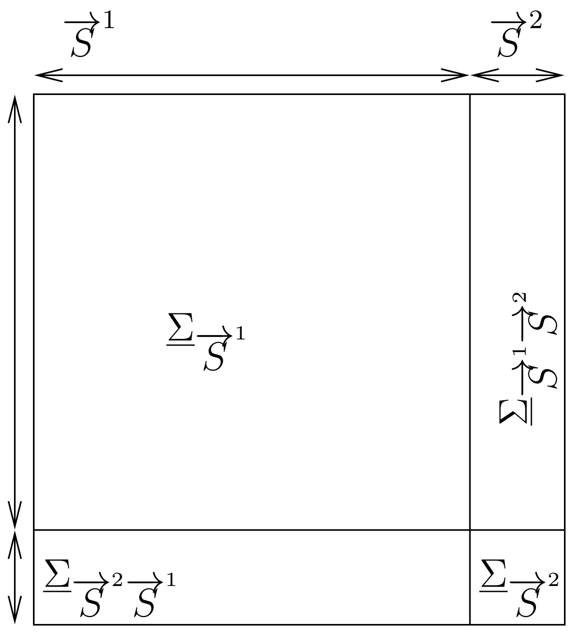
The function f is defined by the following system:

$$\begin{cases} P_1^i(t+1) &=& P_1^i(t) & i=2,..,N \\ P_2^i(t+1) &=& P_2^i(t) & i=2,..,N \\ P_3^i(t+1) &=& P_3^i(t) & i=4,..,N \\ \overrightarrow{T}(t+1) &=& \mathbb{R}^3toSO_3(\overrightarrow{\omega}(t))\overrightarrow{T}(t)+\overrightarrow{V}(t) & \\ \overrightarrow{\Omega}(t+1) &=& SO_3to\mathbb{R}^3\left(\mathbb{R}^3toSO_3(\overrightarrow{\omega}(t))\mathbb{R}^3toSO_3(\overrightarrow{\Omega}(t))\right) & \\ \overrightarrow{V}(t+1) &=& \overrightarrow{V}(t)+\overrightarrow{a}_V(t) & \\ \overrightarrow{\omega}(t+1) &=& \overrightarrow{\omega}(t)+\overrightarrow{a}_\omega(t) & \end{cases} \tag{4.8}$$

where as explained in Section 2.3.2 $\mathbb{R}^3toSO_3$ and $SO_3to\mathbb{R}^3$ represent the maps between the rotation matrix representation and the 3-vector (angle-axis or quaternions) representation of rotations. Note that P_1^i and P_2^i (the x, y coordinates of the $3D$ features) start with the index $i=2$ while P_3^i (depth) starts from $i=4$, which means that all the coordinates of the first feature $\overrightarrow{P}^1$ and the depth of the second and third features are kept fixed and not included in the state vector. These 5 parameters are used as Gauge constraints to fix the similarity ambiguity. This method to fix the similarity has been used by Chiuso et. al [26]. $\overrightarrow{a}_V(t)$ and $\overrightarrow{a}_\omega(t)$ are the translational and rotational accelerations. In the absence of any information on the dynamics of the motion these accelerations can be modeled as random walks. $\Sigma_{\overrightarrow{S}}$, called the process noise covariance matrix, represents the uncertainty on the evolution of the $3D$ parameters. $\Sigma_{\overrightarrow{Z}^1}$ and $\Sigma_{\overrightarrow{Z}^2}$ are block diagonal covariance matrices composed of the covariance matrices of the noise in the tracked features and the frame-to-frame features respectively. In the remainder of this chapter we will be using the notations $\overrightarrow{S}^1$ and $\overrightarrow{S}^2$ to represent the following sub-vectors of $\overrightarrow{S}(t)$.

$$\begin{gathered} \overrightarrow{S}(t)=[\overrightarrow{S}^1(t);\overrightarrow{S}^2(t)] \\ \overrightarrow{S}^1(t)=[\overrightarrow{\Omega}(t);\overrightarrow{T}(t);P_1^2;P_2^2;P_1^3;P_2^3;\overrightarrow{P}^4;...;\overrightarrow{P}^N], \\ \overrightarrow{S}^2(t)=[\overrightarrow{\omega}(t);\overrightarrow{V}(t)] \end{gathered} \tag{4.9}$$

The dimension of $\overrightarrow{S}^1(t)$ is $3N+1$ (6 parameters for the motion and $3N-5$ structure, the (-5) is for the elements used to fix the gauge). The dimension of $\overrightarrow{S}^2(t)$ is 6. Similarly, $\Sigma_{\overrightarrow{S}^1}(t)$, $\Sigma_{\overrightarrow{S}^2}(t)$, $\Sigma_{\overrightarrow{S}^1\overrightarrow{S}^2}(t)$ and $\Sigma_{\overrightarrow{S}^2\overrightarrow{S}^1}(t)$ are the sub-matrices of the covariance matrix corresponding to these vectors (Figure 4.5). The system in Equation (4.7) can be solved using any filtering technique. However, as discussed earlier this induces two problems: (1) because Equation (4.6) is an implicit, a different solution might be required for different types of filters, and (2) the computational cost is too high for real-time processing.



$\overrightarrow{S}^1$
$\overrightarrow{S}^2$
$\underline{\Sigma}_{\overrightarrow{S}^1}$
$\underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2}$
$\underline{\Sigma}_{\overrightarrow{S}^2\overrightarrow{S}^1}$
$\underline{\Sigma}_{\overrightarrow{S}^2}$

Figure 4.5: Structure of the covariance matrix of the state vector showing the sub-matrices corresponding to the sub-vectors used in the formulation.

Also, we proposed a solution for this based on separating the filtering with frame-to-frame features from the main filter and we mentioned that this separate filtering does not constrain the estimates to respect the gauge set by $\overrightarrow{P}^1$, P_3^2 and P_3^3, and that an extra processing is required to fix this problem. In the next section we provide a solution to this issue.

4.3.2 Adjusting the Gauge

As described in Section 4.3, the gauge is fixed by the first $3D$ feature and the depth of the second and third. Assuming that the extra filtering results in an extra change of gauge represented by the similarity transform $(\alpha^t, \underline{R}^t, \overrightarrow{T}^t)$, and that the updated rotation and translation are denoted by $\underline{R}^u$ and $\overrightarrow{T}^u$ respectively, then to keep the gauge parameters fixed, a few conditions need to hold. First of all we should have:

$$\underline{R}^u\overrightarrow{P}^1 + \overrightarrow{T}^u = \underline{R}\overrightarrow{P}^1 + \overrightarrow{T}, \tag{4.10}$$

since $\overrightarrow{P}^1$ should remain constant. From this equation we deduce that $\underline{R}^u = \underline{R}$ and $\underline{T}^u = \underline{T}$, then

$$\begin{aligned} \underline{R}^t &= \underline{R}^u \underline{R}^T \\ \overrightarrow{T}^t &= \overrightarrow{T}^u - \underline{R}^u \underline{R}^T \overrightarrow{T} \end{aligned}. \tag{4.11}$$

Now to determine the scaling α^t, two other conditions that need to hold are used, and those being that the depth of the second and third features (P_3^2 and P_3^3) should stay constant. We can write

$$\begin{aligned} \overrightarrow{P}^{2u} &= \alpha^t(\underline{R}^t \overrightarrow{P}^2 + \overrightarrow{T}^t) \\ \overrightarrow{P}^{3u} &= \alpha^t(\underline{R}^t \overrightarrow{P}^3 + \overrightarrow{T}^t) \end{aligned} \tag{4.12}$$

Each of the above two equations is a three dimensional equation. We only use the last equation of each one corresponding to P_3^{2u} and P_3^{3u} and determine α^t by setting $P_3^{2u} = P_3^2$ and $P_3^{3u} = P_3^3$. Once the similarity transformation $(\alpha^t, \underline{R}^t, \overrightarrow{T}^t)$ is determined, its effect is removed by applying its inverse to all the points $\overrightarrow{P}^i$. The computational cost of this step involves $4N^2 + 4N + 131$ multiplication operations.

4.3.3 Filtering Equations

Let $\overrightarrow{S}(t)$ and $\underline{\Sigma}_{\overrightarrow{S}}(t)$ represent the output of the main filter at time t, we will denote the output of the extra filtering step by the superscript u. The filtering equations used to determine $\overrightarrow{S}^u(t)$ and $\underline{\Sigma}^u_{\overrightarrow{S}}(t)$ from $\overrightarrow{S}(t)$ and $\underline{\Sigma}_{\overrightarrow{S}}(t)$ and the frame-to-frame features are based on an implicit EKF update, since the measurement equation pertinent to the frame-to-frame features is implicit. The derivations of this filter are provided in Appendix A.

Let $\underline{H}_{\overrightarrow{S}}(t)$ represent the Jacobian matrix of $h^2(\overrightarrow{S}, \overrightarrow{Z}^2)$ with respect to $\overrightarrow{S}$, and $\underline{H}_{\overrightarrow{Z}^2}(t)$ its Jacobian matrix with respect to $\overrightarrow{Z}^2(t)$, evaluated at the current estimate of $\overrightarrow{S}$ provided by the main filter:

$$\begin{cases} \underline{H}_{\overrightarrow{S}}(t) = \frac{\partial h^2(\overrightarrow{S}(t), \overrightarrow{Z}^2(t))}{\partial \overrightarrow{S}} & K \times (3n+5) \text{ matrix} \\ \underline{H}_{\overrightarrow{Z}^2}(t) = \frac{\partial h^2(\overrightarrow{S}(t), \overrightarrow{Z}^2(t))}{\partial (\overrightarrow{Z}^2)} & K \times K \text{ bi-diagonal matrix} \end{cases} \tag{4.13}$$

The matrix $\underline{H}_{\overrightarrow{S}}$ has a dimension of $K \times (3N + 7)$. It consists of two parts: the first $K \times (3N + 1)$ corresponds to the Jacobian of h^2 with respect to $\overrightarrow{S}^1$ and is zero; the second part is the Jacobian matrix $\underline{H}_{\overrightarrow{S}^2}$ of h with respect to $\overrightarrow{S}^2$ and which is a $K \times 6$

matrix. Therefore only this part needs to be computed and stored. In the discrete case, the derivation of the matrix $\underline{H}_{\overrightarrow{S}^2}$ is done using the chain rule:

$$\underline{H}_{\overrightarrow{S}^2} = \frac{\partial h}{\partial \overrightarrow{S}^2} = \frac{\partial h}{\partial \underline{E}} \frac{\partial \underline{E}}{\partial([\overrightarrow{V}; \overrightarrow{\omega}])}, \tag{4.14}$$

where $\underline{E}$ is the essential matrix corresponding to the motion $[\overrightarrow{V}; \overrightarrow{\omega}]$,(i.e.,$\underline{E} = [\overrightarrow{V}]_\times \underline{R}_\omega$ with $\underline{R}_\omega = \mathbb{R}^3 to SO_3(\overrightarrow{\omega})$). $\frac{\partial h}{\partial \underline{E}}$ is computed using Maple, $\frac{\partial \underline{E}}{\partial([\overrightarrow{V}; \overrightarrow{\omega}])}$ is computed also using the chain rule:

$$\frac{\partial \underline{E}}{\partial([\overrightarrow{V}; \overrightarrow{\omega}])} = \frac{\partial \underline{E}}{\partial[\overrightarrow{R}_\omega; \overrightarrow{V}]} \frac{\partial[\overrightarrow{R}_\omega; \overrightarrow{V}]}{\partial([\overrightarrow{V}; \overrightarrow{\omega}])}$$

$\frac{\partial \underline{E}}{\partial[\overrightarrow{R}_\omega; \overrightarrow{V}]}$ can be computed manually by determining element by element the entries of the Jacobian of $[\overrightarrow{V}]_\times \underline{R}_\omega$, and $\frac{\partial[\overrightarrow{R}_\omega; \overrightarrow{V}]}{\partial([\overrightarrow{V}; \overrightarrow{\omega}])}$ can be computed from the derivatives of $SO_3 to \mathbb{R}^3$ (see Appendix B).

The matrix $\underline{H}_{\overrightarrow{Z}^2}$ is block diagonal with every row containing the 4 elements of $\frac{\partial h}{\partial[\overrightarrow{q}(t); \overrightarrow{q}(t+1)]}$. Let $\underline{\Sigma}_{h(Z^2)}$ be the matrix defined as:

$$\underline{\Sigma}_{h(Z^2)} \overset{\text{def}}{=} \underline{H}_{\overrightarrow{Z}^2}(t) \underline{\Sigma}_{\overrightarrow{Z}^2}(t) \underline{H}^T_{\overrightarrow{Z}^2}(t) \tag{4.15}$$

which is a $K \times L$ diagonal matrix representing the uncertainty in h due to the uncertainty in $\overrightarrow{Z}^2$. The computation of $\underline{\Sigma}_{h(Z^2)}$ can be done in $O(K)$ time since $\underline{H}_{\overrightarrow{Z}^2}$ is bi-diagonal and $\Sigma_{\overrightarrow{Z}^2}$ is diagonal and then we only need to compute the diagonal elements of $\underline{\Sigma}_{h(Z^2)}$.

The update equations are then written as:

$$\begin{aligned} \overrightarrow{S}^u(t) &= \overrightarrow{S}(t) + \underline{L}(t) h^2(\overrightarrow{S}(t), \overrightarrow{Z^2}(t)) \\ \underline{\Sigma}^u_{\overrightarrow{S}}(t) &= \underline{\Gamma}(t) \underline{\Sigma}_{\overrightarrow{S}} \underline{\Gamma}^T(t) + \underline{L}(t) \underline{\Sigma}_{h(\overrightarrow{Z}^2)} \underline{L}(t)^T \end{aligned} \tag{4.16}$$

where

$$\begin{aligned} \underline{L}(t) &= -\underline{\Sigma}_{\overrightarrow{S}}(t) \underline{H}^T_{\overrightarrow{S}} \underline{\Lambda}^{-1}(t) \\ \underline{\Lambda}(t) &= \underline{H}_{\overrightarrow{S}}(t) \underline{\Sigma}_{\overrightarrow{S}}(t) \underline{H}^T_{\overrightarrow{S}}(t) + \underline{\Sigma}_{h(\overrightarrow{Z}^2)}(t) \\ \underline{\Gamma}(t) &= \underline{I}_{3N+7} - \underline{L}(t) \underline{H}_{\overrightarrow{S}}(t). \end{aligned} \tag{4.17}$$

4.4 Complexity Reduction

The filtering equations provided in the previous section (Equations (4.16) and (4.17)) are of cubical complexity in terms of the number of frame-to-frame features included K. In this section we will use the notation N to represent the total number of parameters in the state vector (i.e., for N_t tracked features the total number of parameters in the State vector would be $N = 3N_t + 12 - 5 = 3N_t + 7$. The "12" is the number of motion parameters (rotation, translation, rotational and translational velocities) and the "5" is the number of parameters used to fix the gauge as explained earlier.

Two points are exploited to reduce the computational complexity: (1) the noise vectors in the frame-to-frame features are statistically independent (which means that their covariance matrix is diagonal) and (2) the frame-to-frame features provide direct measurements only for a small part of the state vector (velocity $\overrightarrow{T}$ and $\overrightarrow{\omega}$)). There are numerous ways to capitalize on these two points. To illustrate how we reached to the proposed solution and why it is so efficient, we will analyze several of the possible reduction strategies. We use the terms $c_i(N, K)$ to refer to the computational cost (as number of multiplication operations) of each strategy with c_0 being the cost of evaluating Equations (4.16) and (4.17) directly. For the sake of simplicity we will also use c_i to refer to the strategy corresponding to the cost c_i. The following is a summary of the analysis done in this section.

- Firstly the cost $c_0(N, K)$ of the straightforward evaluation of Equations (4.16) and (4.17) is determined.
- Using the fact that the K measurements are independent, we use the techniques of inverting sums of matrices to come up with a general speedup scheme when K is greater N. This method generates a cost $c_1(N, K)$ that is low compared to c_0 when $K >> N$. However, when K is close to N the speedup is negligible.
- Capitalizing on the zero parts in the Jacobian of h^2, many of the operations involved can be done with lower cost by avoiding the multiplications involving the zero parts. This way we modify c_0 to obtain c_2 and c_1 to obtain c_3. We show that c_3 is better than c_2 for high K, however for low values of K, c_2 is lower than c_3.
- We devise a new strategy by simultaneously capitalizing on the zero parts of $H_{\overrightarrow{S}}$ and on the independence of the measurements. This method (c_4) outperforms both c_2 and c_3 and generates a speedup of up to 20 times for the case $(K = 200, N_t =$

$50)$. We are specifically interested in this case, because those are the values we use in our simulation results. Nevertheless, c_4 is still not very convenient of real time operation as for the latter case it requires about $35ms$.

- As c_1 is very efficient when K is much higher than N, we propose a method that updates only the velocity using the method c_1 (in this case N is only 12) and then propagate the update to the remaining elements of the state vector using the covariance matrix $\underline{\Sigma}_{\overrightarrow{S}}$. This method enables us to process the case of $(K = 200, N_t = 50)$ in about $5ms$.

4.4.1 Cost of the Straightforward Evaluation

The total number of multiplication operations involved in evaluating Equations (4.16) and (4.17) is determined as follows:

- $\underline{H}_{\overrightarrow{S}}(t)\underline{\Sigma}_{\overrightarrow{S}}(t)\underline{H}^T_{\overrightarrow{S}}(t)$ requires $KN^2 + K^2N$ multiplications.
- $\underline{\Lambda}^{-1}$ requires K^3 multiplications.
- $\underline{L}$ requires NK^2 by using the same $\underline{\Sigma}_{\overrightarrow{S}}(t)\underline{H}^T_{\overrightarrow{S}}(t)$ that was computed for $\underline{\Lambda}$.
- $\underline{\Gamma}$ requires N^2K multiplications.
- $\underline{\Sigma}_{\overrightarrow{S}}(t)$ requires $2N^3$ + NK+N^2K multiplications.
- $\overrightarrow{S}^u(t)$ requires N multiplications.

Therefore the total number of multiplications (including the gauge update) is:

$$c_0(N, K) = K^3 + 3NK^2 + 2(N + N^2)K + N^3 + 4N^2 + 4N + 131. \qquad (4.18)$$

This equation shows that this cost is not only cubic in the number of added frame-to-frame features (K) but also in the number of parameters in the state vector (N).

4.4.2 Reduction Based on Independent Measurements

Assume in a given problem that a state vector of size N is being updated using the information of K independent observations. Theoretically, adding the K frame-to-frame features at once is equivalent to adding them one by one since they are independent.

Hence it can be linear in K, however, since every addition is cubic in N also, adding them one by one will be also very heavy computationally as it would involve KN^3 computations. Nevertheless, we can, through using the inversion of sum of matrices identity and capitalizing on the fact that $\underline{\Sigma}_{h^2(\overrightarrow{S})}$ is diagonal, reduce this cost to become linear in K while involving a $5N^3$ term instead of KN^3. We start from the computation of the matrix $\underline{\Lambda}^{-1}$ using the Sherman-Morrison-Woodbury formula [53]:

$$(\underline{A}+\underline{UBV})^{-1} = \underline{A}^{-1} - \underline{A}^{-1}\underline{U}(\underline{C}^{-1} + \underline{VA}^{-1}\underline{U})^{-1}\underline{VA}^{-1}, \tag{4.19}$$

where $\underline{\text{A}}$, $\underline{\text{U}}$, $\underline{\text{B}}$ and $\underline{\text{V}}$ all denote matrices of the correct size. Using this formula $\underline{\Lambda}^{-1}$ is expanded as follows:

$$\begin{gathered}\underline{\Lambda}^{-1} = (\underline{H}_{\overrightarrow{S}}\underline{\Sigma}_{\overrightarrow{S}}\underline{H}^T_{\overrightarrow{S}} + \underline{\Sigma}_{h(\overrightarrow{Z})})^{-1} = \\ \underline{\Sigma}^{-1}_{h(\overrightarrow{Z})} - \underline{\Sigma}^{-1}_{h(\overrightarrow{Z})}\underline{H}_{\overrightarrow{S}}(\underline{I}_6 + \underline{\Sigma}_{\overrightarrow{S}}\underline{H}^T_{\overrightarrow{S}}\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)}\underline{H}_{\overrightarrow{S}})^{-1}\underline{\Sigma}_{\overrightarrow{S}}\underline{H}^T_{\overrightarrow{S}}\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)}\end{gathered}, \tag{4.20}$$

Note that this inversion is useful only because the measurements are independent and hence $\underline{\Sigma}_{h(\overrightarrow{Z})}$ is diagonal and can be inverted in only K divisions. Then $\underline{L}$ can be written as:

$$\begin{gathered}\underline{L} = \\ -\underline{\Sigma}_{\overrightarrow{S}}\underline{H}^T_{\overrightarrow{S}}(\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)} - \underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)}\underline{H}_{\overrightarrow{S}}(\underline{I}_6 + \underline{\Sigma}_{\overrightarrow{S}}\underline{H}^T_{\overrightarrow{S}}\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)}\underline{H}_{\overrightarrow{S}})^{-1}\underline{\Sigma}_{\overrightarrow{S}}\underline{H}^T_{\overrightarrow{S}}\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)})\end{gathered}. \tag{4.21}$$

$\underline{\Sigma}_{\overrightarrow{S}}\underline{H}^T_{\overrightarrow{S}}$ is a multiplication of a $N\times N$ matrix by a $N\times K$ matrix and can be done in N^2K multiplication operations. $\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)}\underline{H}_{\overrightarrow{S}}$ can be done in NK multiplication operations. Define the two matrices $\underline{G}^0$ and $\underline{G}^1$:

$$\begin{aligned}\underline{G}^0 &= \underline{\Sigma}_{\overrightarrow{S}}\underline{H}^T_{\overrightarrow{S}}\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)} \\ \underline{G}^1 &= \underline{\Sigma}_{\overrightarrow{S}}\underline{H}^T_{\overrightarrow{S}}\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)}\underline{H}_{\overrightarrow{S}}\end{aligned}, \tag{4.22}$$

$\underline{G}^0$ can be computed in NK multiplications and $\underline{G}^1$ in N^2K using $\underline{G}^0$. Now $\underline{L}$ and $\underline{LH}_{\overrightarrow{S}}$ can be written as:

$$\begin{aligned}\underline{L} &= -\underline{G}^0 + \underline{G}^1(\underline{I}_6 + \underline{G}^1)^{-1}\underline{G}^0 \\ \underline{LH}_{\overrightarrow{S}} &= -\underline{G}^1 + \underline{G}^1(\underline{I}_6 + \underline{G}^1)^{-1}\underline{G}^1\end{aligned}. \tag{4.23}$$

The reason for writing $\underline{LH}_{\overrightarrow{S}}$ this way is that it can be computed much faster than multiplying $\underline{L}$ by $\underline{H}_{\overrightarrow{S}}$. Table 4.1 shows the number of multiplications operations involved in

$\underline{\Sigma}_{\vec{S}}\underline{H}_{\vec{S}}^{T}$	N^2K
$\underline{\Sigma}_{h(\vec{Z}^2)}^{-1}\underline{H}_{\vec{S}}$	NK
$\underline{G}^0$	NK
$\underline{G}^1$	N^2K
$\underline{L}$	$N^2K + N^3$
$\underline{LH}_{\vec{S}}$	$2N^3$
$\underline{\Sigma}_{\vec{S}}^{u}$	$N(N+1)K + 2N^3$
$\vec{U}^{0u}$	NK
Total	$4(N^2+N)K + 5N^3$

Table 4.1: Number of multiplication operations involved in the filtering based on the independence of the features.

the update. The total number of multiplications is:

$$c_1(N,K) = 4(N^2+N)K + 5N^3 + 4N^2 + 4N + 131 \tag{4.24}$$

Again the $4N^2 + 4N + 131$ comes from the gauge adjustment step. To evaluate the improvement due to this modification, we evaluate $c_0(N,K)/c_1(N,K)$ for different values of N and K. Figure 4.6 shows the speedup achieved. For high enough K relative to N the speedup is good (11 times for K=500 and N=82 corresponding to $N_t = 25$ tracked features). However, this would take about 265ms on a Pentium M 2.13 GHz processor. For lower values of K and higher values of N the speedup is much lower. For example for 50 tracked features ($N = 157$) and $K = 200$ frame-to-frame features the speedup would be only 1.03 times. The efficiency increases cubically when K increases for con-

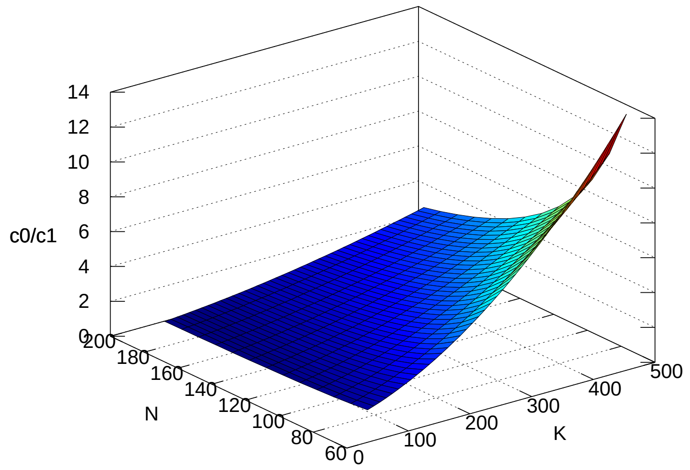


Figure 4.6: c_0/c_1 for different values of K and N. The speedup is significant only when $K >> N$.

stant N. The increase is more pronounced when N is small, for example for $N = 12$ and $K = 300$ the speedup is about 154 times. We will rely on this to come up with the method c_5 described later.

4.4.3 Capitalizing on the Jacobian Structure

Since the frame-to-frame features measurement equation involves only the velocities ($\overrightarrow{V}$ and $\overrightarrow{\omega}$), the Jacobian of h^2 with respect to $\overrightarrow{S}$ is a $K \times N$ matrix with only the last 6 columns non-zero (Figure 4.7). This results in a few of the transitional matrices in the filtering equations with significant zero parts. By avoiding multiplying by those parts a considerable speedup can be obtained.

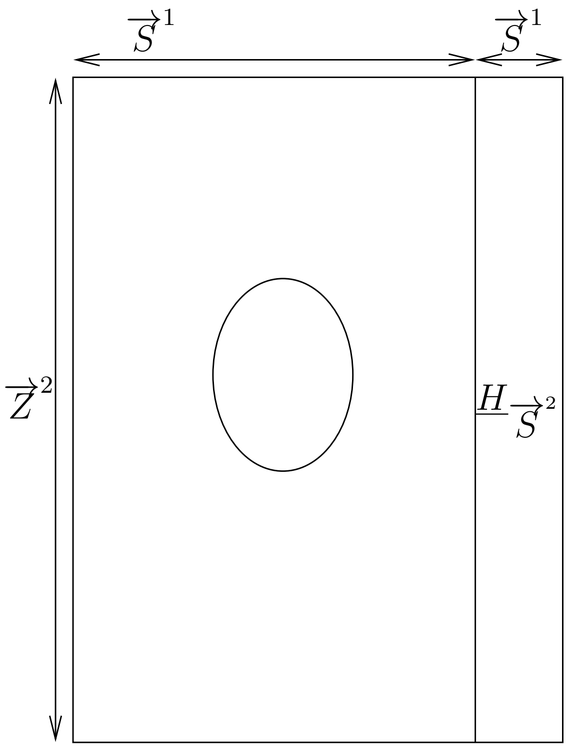


Figure 4.7: Structure of the Jacobian matrix. Only 6 out of $(3N + 7)$ columns are non zero.

Speedup for the Straightforward Equations

Avoiding the multiplication by the zero parts of the matrices in Equations (4.16) and (4.17), a new cost can be determined as follows:

- $H\underline{\Sigma}_{\overrightarrow{S}}$ and $H\underline{\Sigma}_{\overrightarrow{S}}H^T$ require $12KN$ multiplications
- $\underline{\lambda}^{-1}$ requires K^3 multiplications
- $\underline{L}$ requires $6K^2 + 6NK$ multiplications
- Γ would require $6NK$
- $\underline{\Sigma}^u_{\overrightarrow{S}}$ requires $12K^2 + 20N^2 + 36K + 12N + 72$ multiplications
- $\overrightarrow{S}^u$ requires KN multiplications

Therefore the total cost is:

$$c_2(N, K) = K^3 + 18K^2 + 25KN + 36K + 12N + 72 + 4N^2 + 4N + 131 \qquad (4.25)$$

The speedup $c_0(N, K)/c_2(N, K)$ obtained from this operation is shown in Figure 4.8. When K is small and N is large, the speedup is large and reaches about 15 times. This is because the speedup in this method targets the N factors in the cost. When K is large or N is small, the contribution of N to the cost is minimal and reducing this contribution does not affect the cost much.

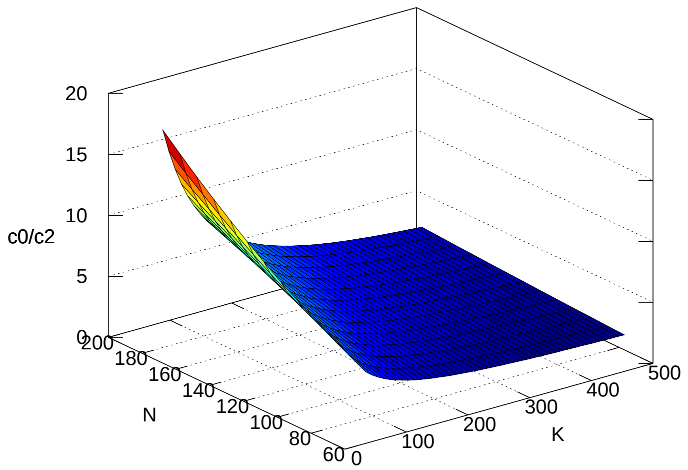


Figure 4.8: c_0/c_2 for different values of K and N. The speedup is significant only for low K and high N which is not a practical case.

Speeding-up c_1 by considering the structure of $\underline{H}_{\vec{S}}$

Taking into account the zero parts of the Jacobian in performing the operations of c_1 shown in Table 4.1, a new computational cost can be achieved:

$$c_3(N, K) = N^3 + 32N^2 + N^2K + 14NK + 12K^2 + 12N + 36K + 72 + 4N^2 + 4N + 131. \tag{4.26}$$

The speedup of c_3 versus c_0 is shown in Figure 4.9. This speedup is on average two times better than c_1. It is also, most of the times better than c_2, however, for small values of K the speedup of c_3 is lower than the speedup of c_2 (Figure 4.9). Actually, for the case of ($N_t = 50$ and $K = 200$) c_2 is more efficient ($c_2/c_3 = 0.9$). The conclusion from this is that using the zero parts of the Jacobian with the already reduced equations based on the independence of the features, does not accumulate the speedup of both. This is also shown by comparing Figures 4.8 and 4.11; the effect of relying on the zero parts of $\underline{H}$ is much stronger on c_0 than on c_1.

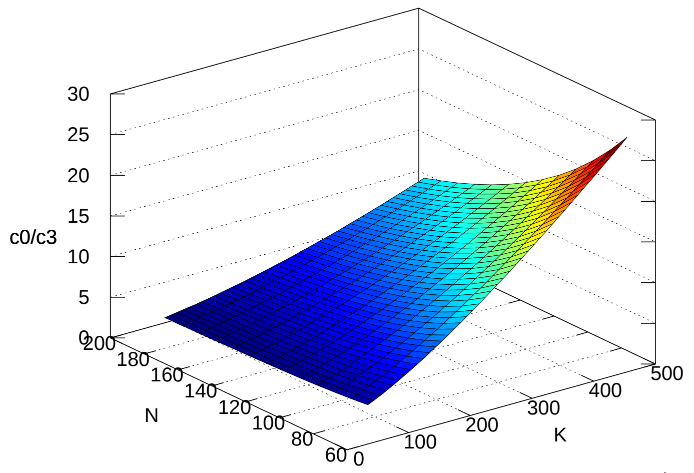


Figure 4.9: c_0/c_3 for different values of K and N. Compared to c_0/c_1, considering the zero parts of the Jacobian results in about 2 times speedup.

4.4.4 Using the Two Reduction Points Simultaneously

Based on the conclusion of the previous section, we aim here to rely simultaneously on the zero parts of $\underline{H}_s$ and on the independence of the frame-to-frame features in an effort to harness the speedups of both. Again, we start from the computation of $\underline{\lambda}$. The

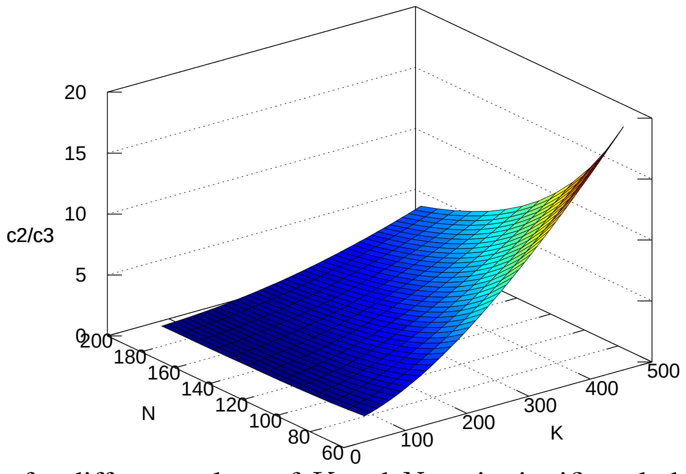


Figure 4.10: c_2/c_3 for different values of K and N. c_3 is significantly better than c_2 only for small values of N and large values of K.

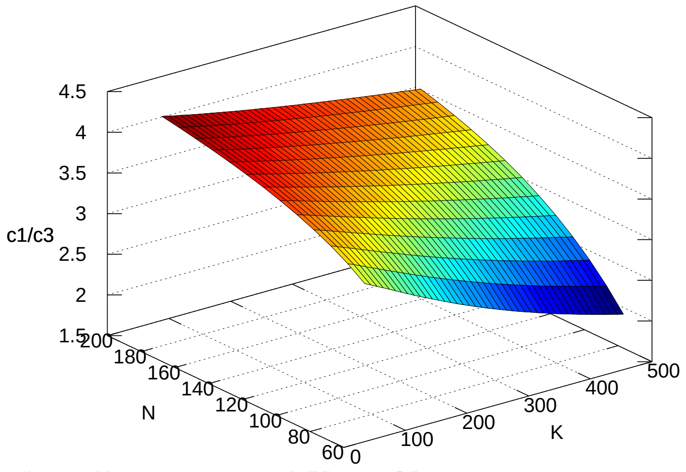


Figure 4.11: c_1/c_3 for different values of K and N. About two times speedup is achieved by capitlizing on the structure of the structure of the Jacobian. This means that c_3 does not fully exploit the potential of speedup offered by the Jacobian structure.

Sherman-Morrison-Woodbury [53] formula is applied to the inversion of $\underline{\Lambda}$:

$$\begin{aligned}\underline{\Lambda}^{-1} &= (\underline{H}_{\overrightarrow{S}}\underline{\Sigma}_{\overrightarrow{S}}\underline{H}_{\overrightarrow{S}}^{T} + \underline{\Sigma}_{h(\overrightarrow{Z}^2)})^{-1} \\ &= \underline{\Sigma}_{h(\overrightarrow{Z}^2)}^{-1} - \underline{\Sigma}_{h(\overrightarrow{Z}^2)}^{-1}\underline{H}_{\overrightarrow{S}}(I_{3N+7} + \underline{\Sigma}_{\overrightarrow{S}}\underline{H}_{\overrightarrow{S}}^{T}\underline{\Sigma}_{h(\overrightarrow{Z}^2)}^{-1}\underline{H}_{\overrightarrow{S}})^{-1}\underline{\Sigma}_{\overrightarrow{S}}\underline{H}_{\overrightarrow{S}}^{T}\underline{\Sigma}_{h(\overrightarrow{Z}^2)}^{-1}\end{aligned}$$

Hence, $\underline{L}$ can be written as:

$$\underline{L} = -\underline{\Sigma}_{\vec{S}}\underline{H}^T_{\vec{S}}(\underline{\Sigma}^{-1}_{h(\vec{Z}^2)} - \underline{\Sigma}^{-1}_{h(\vec{Z}^2)}\underline{H}_{\vec{S}}(I_{3N+7} + \underline{\Sigma}_{\vec{S}}\underline{H}^T_{\vec{S}}\underline{\Sigma}^{-1}_{h(\vec{Z}^2)}\underline{H}_{\vec{S}})^{-1}\underline{\Sigma}_{\vec{S}}\underline{H}^T_{\vec{S}}\underline{\Sigma}^{-1}_{h(\vec{Z}^2)})$$

Define the two matrices $\underline{G}^1$ and $\underline{G}^2$

$$\begin{aligned}\underline{G}^1 &= \underline{\Sigma}^{-1}_{h(\vec{Z}^2)}\underline{H}_{\vec{S}} \\ \underline{G}^2 &= \underline{\Sigma}_{\vec{S}}\underline{H}^T_{\vec{S}} = [\underline{\Sigma}_{\vec{S}^1\vec{S}^2};\underline{\Sigma}_{\vec{S}^2}]\underline{H}^T_{\vec{S}^2}\end{aligned} \tag{4.27}$$

Note that $\underline{G}^1$ has the same structure as $\underline{H}_{\vec{S}}$ and same dimensions. Therefore, only the non-zero part in it which is equal to $\underline{\Sigma}^{-1}_{h(\vec{Z}^2)}\underline{H}_{\vec{S}^2}$ needs to be computed. We refer to this part as $\underline{G}^0$. The computation of $\underline{G}_0$ involves only $6K$ multiplications since $\underline{\Sigma}^{-1}_{h(\vec{Z}^2)}$ is $K \times K$ diagonal. Now $(\underline{I}_{3N+7} + \underline{\Sigma}_{\vec{S}}\underline{H}^T_{\vec{S}}\underline{\Sigma}^{-1}_{h(\vec{Z}^2)}\underline{H}_{\vec{S}})^{-1}$ in $\underline{L}$ can be written as:

$$(\underline{I}_{3N+7} + \underline{\Sigma}_{\vec{S}}\underline{H}^T_{\vec{S}}\underline{\Sigma}^{-1}_{h(\vec{Z}^2)}\underline{H}_{\vec{S}})^{-1} = (\underline{I}_{3N+5} + [\underline{\Sigma}_{\vec{S}^1\vec{S}^2};\underline{\Sigma}_{\vec{S}^2}]\underline{H}^T_{\vec{S}^2}\underline{G}^1)^{-1}.$$

Using again another variant of the Sherman-Morrison-Woodbury formula which was introduced in [51] and gives the inverse of a matrix of the form $(\underline{A} + \underline{UBV})$ as:

$$(\underline{A} + \underline{UBV})^{-1} = \underline{A}^{-1} - \underline{A}^{-1}\underline{U}(\underline{I} + \underline{BVA}^{-1}\underline{U})^{-1}\underline{BVA}^{-1}, \tag{4.28}$$

we can write:

$$\begin{aligned}&(\underline{I}_{3N+7} + [\underline{\Sigma}_{\vec{S}^1\vec{S}^2};\underline{\Sigma}_{\vec{S}^2}]\underline{H}^T_{\vec{S}^2}\underline{G}^1)^{-1} = \\ &\underline{I}^{-1}_{3N+7} - \underline{I}^{-1}_{3N+7}[\underline{\Sigma}_{\vec{S}^1\vec{S}^2};\underline{\Sigma}_{\vec{S}^2}](\underline{I}_6 + \underline{H}^T_{\vec{S}^2}\underline{G}^1\underline{I}^{-1}_{3N+7}[\underline{\Sigma}_{\vec{S}^1\vec{S}^2};\underline{\Sigma}_{\vec{S}^2}])^{-1}\underline{H}^T_{\vec{S}^2}\underline{G}^1\underline{I}^{-1}_{3N+7} \\ &= \underline{I}_{3N+7} - [\underline{\Sigma}_{\vec{S}^1\vec{S}^2};\underline{\Sigma}_{\vec{S}^2}](\underline{I}_6 + \underline{H}^T_{\vec{S}^2}\underline{G}^1[\underline{\Sigma}_{\vec{S}^1\vec{S}^2};\underline{\Sigma}_{\vec{S}^2}])^{-1}\underline{H}^T_{\vec{S}^2}\underline{G}^1.\end{aligned}$$

Recalling that $\underline{G}^1$ has the same form as $\underline{H}^T_{\vec{S}}$ in Figure 4.7 with its non-zero part equal $\underline{\Sigma}^{-1}_{h(\vec{Z}^2)}\underline{H}_{\vec{S}^2}$ then we can define $\underline{G}^3$ as:

$$\underline{G}^3 = \underline{G}^1[\underline{\Sigma}_{\vec{S}^1\vec{S}^2};\underline{\Sigma}_{\vec{S}^2}] = \underline{\Sigma}^{-1}_{h(\vec{Z}^2)}\underline{H}_{\vec{S}^2}\underline{\Sigma}_{\vec{S}^2} = \underline{G}^0\underline{\Sigma}_{\vec{S}^2}.$$

$\underline{G}^3$ is a $K \times 6$ matrix whose computation using $\underline{G}^0$ requires $36K$ multiplications. This allows us to write:

$$\begin{aligned}(\underline{I}_{3N+7} + [\underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2};\underline{\Sigma}_{\overrightarrow{S}^2}]\underline{H}^T_{\overrightarrow{S}^2}\underline{G}^1)^{-1} &= \\ = \underline{I}_{3N+7} - [\underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2};\underline{\Sigma}_{\overrightarrow{S}^2}](\underline{I}_6 + \underline{H}^T_{\overrightarrow{S}^2}\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)}\underline{H}_{\overrightarrow{S}^2}\underline{\Sigma}_{\overrightarrow{S}^2})^{-1}\underline{H}^T_{\overrightarrow{S}^2}\underline{G}^1\end{aligned}$$

Define $\underline{G}^4$ as

$$\underline{G}^4 = \underline{H}^T_{\overrightarrow{S}^2}\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)}\underline{H}_{\overrightarrow{S}^2}\underline{\Sigma}_{\overrightarrow{S}^2} = \underline{H}^T_{\overrightarrow{S}^2}\underline{G}^3 \tag{4.29}$$

$\underline{G}^4$ is a 6×6 matrix which can be computed in $36K$ multiplication from $\underline{G}^3$. Then,

$$\begin{aligned}(\underline{I}_{3N+7} + [\underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2};\underline{\Sigma}_{\overrightarrow{S}^2}]\underline{H}^T_{\overrightarrow{S}^2}\underline{G}^1)^{-1} &= \\ = \underline{I}_{3N+7} - [\underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2};\underline{\Sigma}_{\overrightarrow{S}^2}](\underline{I}_6 + \underline{G}^4)^{-1}\underline{H}^T_{\overrightarrow{S}^2}\underline{G}^1\end{aligned}$$

Now $\underline{L}$ can be written as:

$$\begin{aligned}\underline{L} &= -\underline{G}^2(\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)} - \underline{G}^1(\underline{I}_{3N+7} - [\underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2};\underline{\Sigma}_{\overrightarrow{S}^2}](\underline{I}_6 + \underline{G}^4)^{-1}\underline{H}^T_{\overrightarrow{S}^2}\underline{G}^1)\underline{G}^2\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)} \\ &= -\underline{G}^2(\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)} - (\underline{G}^1 - \underline{G}^1[\underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2};\underline{\Sigma}_{\overrightarrow{S}^2}](\underline{I}_6 + \underline{G}^4)^{-1}\underline{H}^T_{\overrightarrow{S}^2}\underline{G}^1)\underline{G}^2\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)}) \\ &= -\underline{G}^2(\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)} - (\underline{G}^1 - \underline{G}^3(\underline{I}_6 + \underline{G}^4)^{-1}\underline{H}^T_{\overrightarrow{S}^2}\underline{G}^1)\underline{G}^2\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)})\end{aligned}$$

Note also that $\underline{H}^T_{\overrightarrow{S}^2}\underline{G}^1$ can be written as:

$$\underline{H}^T_{\overrightarrow{S}^2}\underline{G}^1 = [\underline{0}_{6\times(3N+1)} \;\; \underline{H}^T_{\overrightarrow{S}^2}\underline{G}^0] = [\underline{0}_{6\times(3N+1)} \;\; \underline{G}^5],$$

with $\underline{G}^5 = \underline{H}^T_{\overrightarrow{S}^2}\underline{G}^0$ with is a 6×6 matrix and $\underline{0}_{6\times(3N+1)}$ is a zero matrix of size $6 \times (3N+1)$. Therefore, $\underline{G}^3(\underline{I}_6 + \underline{G}^4)^{-1}\underline{H}^T_{\overrightarrow{S}^2}\underline{G}^1$ can be written as:

$$\underline{G}^3(\underline{I}_6 + \underline{G}^4)^{-1}\underline{H}^T_{\overrightarrow{S}^2}\underline{G}^1 = [\underline{0}_{6\times(3N+1)} \;\; \underline{G}^3(\underline{I}_6 + \underline{G}^4)^{-1}\underline{G}^5],$$

and since $\underline{G}^1 = [\underline{0}_{6\times(3N+1)} \;\; \underline{G}^0]$ then

$$\underline{L} = -\underline{G}^2(\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)} - [\underline{0}_{6\times(3N+1)} \;\; \underline{G}^0 - \underline{G}^3(\underline{I}_6 + \underline{G}^4)^{-1}\underline{G}^5]\underline{G}^2\underline{\Sigma}^{-1}_{h(\overrightarrow{Z}^2)}),$$

and we have:

$$
\begin{aligned}
[\underline{0}_{6\times(3N+1)} \;\; \underline{G}^0 - \underline{G}^3(\underline{I}_6 + \underline{G}^4)^{-1}\underline{G}^5]\underline{G}^2 &= \\
[\underline{0}_{6\times(3N+1)} \;\; \underline{G}^0 - \underline{G}^3(\underline{I}_6 + \underline{G}^4)^{-1}\underline{G}^5][\underline{\Sigma}_{\vec{S}^1\vec{S}^2}; \underline{\Sigma}_{\vec{S}^2}]\underline{H}^T_{\vec{S}^2} &= \\
(\underline{G}^0 - \underline{G}^3(\underline{I}_6 + \underline{G}^4)^{-1}\underline{G}^5)\underline{\Sigma}_{\vec{S}^2}\underline{H}^T_{\vec{S}^2}&,
\end{aligned}
$$

and $\underline{L}$ can be re-written as:

$$\underline{L} = -\underline{G}^2(\underline{\Sigma}^{-1}_{h(\vec{Z}^2)} - (\underline{G}^0 - \underline{G}^3(\underline{I}_6 + \underline{G}^4)^{-1}\underline{G}^5)\underline{\Sigma}_{\vec{S}^2}\underline{H}^T_{\vec{S}^2}\underline{\Sigma}^{-1}_{h(\vec{Z}^2)})$$

Note that $\underline{\Sigma}_{\vec{S}^2}\underline{H}^T_{\vec{S}^2}\underline{\Sigma}^{-1}_{h(\vec{Z}^2)}$ is equal to $(\underline{G}^3)^T$, then $\underline{L}$ can be written finally as:

$$\underline{L} = -\underline{G}^2(\underline{\Sigma}^{-1}_{h(\vec{Z}^2)} - (\underline{G}^0 - \underline{G}^3(\underline{I}_6 + \underline{G}^4)^{-1}\underline{G}^5)(\underline{G}^3)^T)$$

The sequence of operations to compute L is then summarized as follows:

$$
\begin{aligned}
\underline{G}^0 &= \underline{\Sigma}_{h(\vec{Z}^2)}\underline{H}_{\vec{S}^2} \\
\underline{G}^2 &= [\underline{\Sigma}_{\vec{S}^1\vec{S}^2}; \underline{\Sigma}_{\vec{S}^2}]\underline{H}^T_{\vec{S}^2} \\
\underline{G}^3 &= \underline{G}^0\underline{\Sigma}_{\vec{S}^2} \\
\underline{G}^4 &= \underline{H}^T_{\vec{S}^2}\underline{G}^3 \\
\underline{G}^5 &= \underline{H}^T_{\vec{S}^2}\underline{G}^0 \\
\underline{L} &= -\underline{G}^2(\underline{\Sigma}^{-1}_{h(\vec{Z}^2)} - (\underline{G}^0 - \underline{G}^3(\underline{I}_6 + \underline{G}^4)^{-1}\underline{G}^5)(\underline{G}^3)^T)
\end{aligned}
$$

Note that $\underline{H}_{\vec{S}}$ is never fully stored or used. Table 4.2 shows the computational complexity of all the involved operations in computing $\underline{L}$. Computing Γ requires an additional $6NK$. The computation of $\underline{\Sigma}^u_{\vec{S}}(t)$ requires the following: $\underline{\Gamma}(t)\underline{\Sigma}_{\vec{S}}\underline{\Gamma}^T(t)$ can be computed in $12N^2 + (2(N-6)^2 = 14N^2 - 24N + 72$ multiplications, and $\underline{L}(t)\underline{\Sigma}_{h(\vec{Z}^2)}\underline{L}^T(t)$ can be computed in $12K^2 + 36K + 36N + 6N^2$ multiplications. Therefore, $\underline{\Sigma}^u_{\vec{S}}(t)$ can be computed in $12K^2 + 20N^2 + 36K + 12N + 72$ multiplications. The computation of $\vec{S}^u(t)$ requires KN multiplications. The total cost c_4 is:

$$c_4(N,K) = 24K^2 + (14N + 186)K + 20N^2 + 12N + 540 + 4N^2 + 4N + 131 \quad (4.30)$$

The speedup of c_4 versus s_0 is shown in Figure 4.12. It's clear that this speedup outperforms both c_2 and c_3. What is very important about this case is that the largest speed up happens for a given K when N is larger. For the case of 50 tracked features and 200

$\underline{G}^0$	$6K$	$O(k)$
$\underline{G}^2$	$6KN$	$O(k)$
$\underline{G}^3$	$36K$	$O(k)$
$\underline{G}^4$	$36K$	$O(k)$
$\underline{G}^5$ $(I_6 + \underline{G}^4)^{-1}\underline{G}^5$	$36K$ 432	$O(k)$ Constant
$\underline{G}^0 - \underline{G}^3(I_6 + \underline{G}^4)^{-1}\underline{G}^5$	$36K$	$O(K)$
$(\underline{G}^0 - \underline{G}^3(I_6 + \underline{G}^4)^{-1}\underline{G}^5)(\underline{G}^3)^T$	$6K^2$	$O(K^2)$
$\underline{L}$	$7NK + 6K^2$	$O(K^2)$

Table 4.2: Computational Complexity of all the operations involved in computing $\underline{L}$ in the method c_4. The total operation has a quadratic computational complexity

frame-to-frame features, the processing time is $35ms$. This is close to real time operation if the frame rate is low. Figure 4.13 shows the values of c_4/c_3 for different N and K.

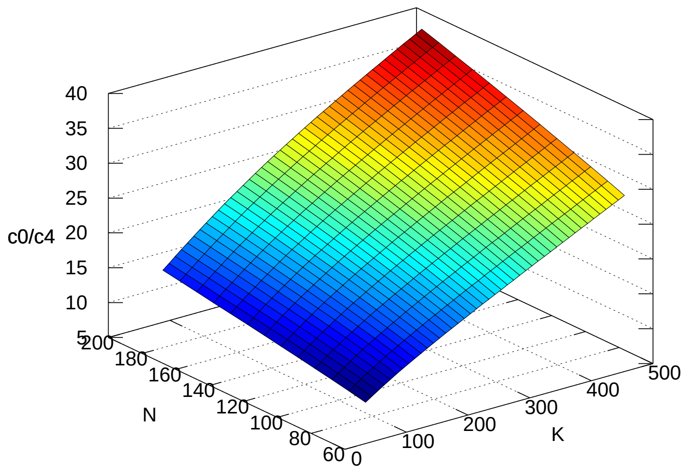


Figure 4.12: c_0/c_4 for different values of K and N. The speedup of c_4 is better than both then c_3 and c_2.

Only for large values of K and very small values of N, c_3 tends to become better than c_4.

4.4.5 Cost Reduction Using Partial Filtering

In the previous presented methods, the best speed up achieved for the case $(K = 200, N_t = 50)$ was $35ms$ which is still a little bit beyond the real-time requirements. In this section, we propose a totally different approach to significantly decrease this cost. The main idea rests on the fact that the frame-to-frame features only affect directly the incremental

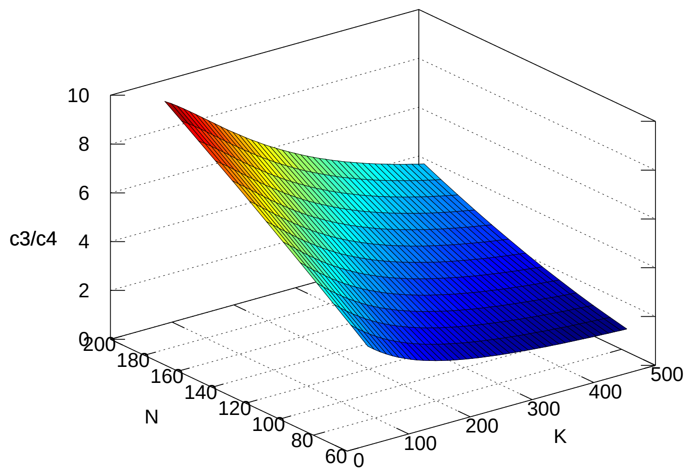


Figure 4.13: c_3/c_4 for different values of K and N. Only for large values of K and very small values of N, c_3 tends to become better than c_4.

motion ($\overrightarrow{V}$ and $\overrightarrow{\omega}$). The other parameters are only affected through their covariance with ($\overrightarrow{V}$ and $\overrightarrow{\omega}$). Therefore, we can first update the incremental motion ($\overrightarrow{V}$) and ($\overrightarrow{\omega}$) using the Frame-To-Frame features, then use the covariance matrix of $\overrightarrow{S}$ to propagate this update to the other $3D$ parameters. The flowchart of the proposed filtering method is shown in Figure 4.14 and the steps involved are explained below.

Transforming To Shperical Coordinates

An important issue to address is that since the frame-to-frame features do not provide any observation for the magnitude of the translational inter-frame motion $\overrightarrow{V}$, a new representation of $\overrightarrow{V}$ consisting of the orientation only is required. We use the spherical coordinates θ, ϕ and ρ for this representation. The magnitude of the translation ρ is not updated directly in the filter, however it should be updated afterwards since its covariant with θ and ρ. to be used after the partial update. Define the vector $\overrightarrow{U}$:

$$\overrightarrow{U} = \begin{pmatrix} \theta = atan(\frac{V_2}{V_1}) \\ \phi = acos(\frac{V_3}{\sqrt{V_1^2+V_2+V_3}}) \\ \rho = \sqrt{V_1^2+V_2+V_3} \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix}$$

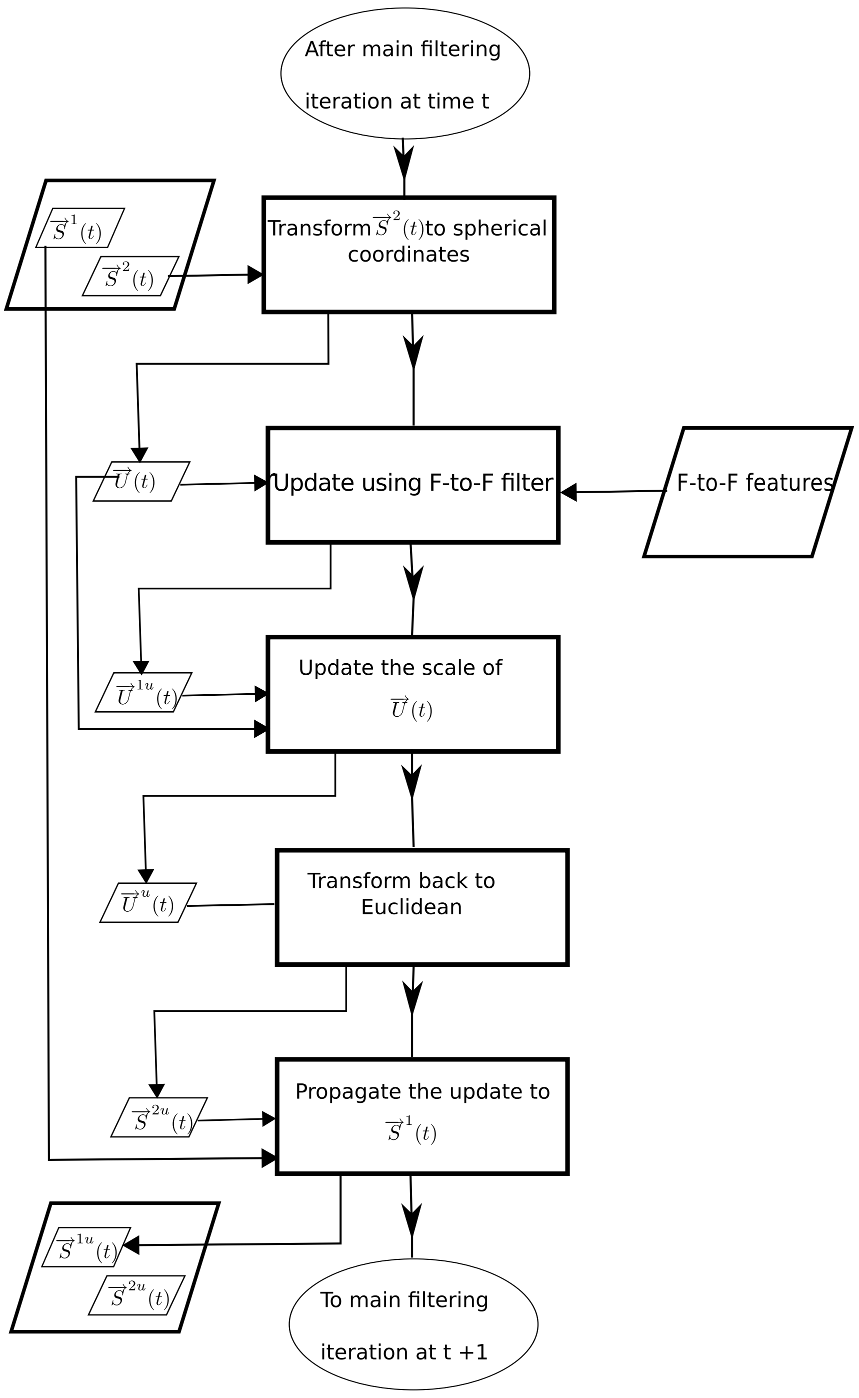

After main filtering iteration at time t
Transform $\overrightarrow{S}^2(t)$ to spherical coordinates
$\overrightarrow{S}^1(t)$
$\overrightarrow{S}^2(t)$
$\overrightarrow{U}(t)$
Update using F-to-F filter
F-to-F features
$\overrightarrow{U}^{1u}(t)$
Update the scale of $\overrightarrow{U}(t)$
$\overrightarrow{U}^{u}(t)$
Transform back to Euclidean
$\overrightarrow{S}^{2u}(t)$
Propagate the update to $\overrightarrow{S}^1(t)$
$\overrightarrow{S}^{1u}(t)$
$\overrightarrow{S}^{2u}(t)$
To main filtering iteration at t +1

Figure 4.14: Flowchart of the extra filtering with partial update of the velocity followed by an update of the remaining parametes.

The covariance matrix $\underline{\Sigma}_{\overrightarrow{U}}$ of $\overrightarrow{U}$ can be obtained from $\underline{\Sigma}_{\overrightarrow{S}^2}$ using a first order covariance propagation (Appendix A):

$$\underline{\Sigma}_{\overrightarrow{U}} = \left(\frac{\partial \overrightarrow{U}}{\partial \overrightarrow{S}^2}\right) \underline{\Sigma}_{\overrightarrow{S}^2} \left(\frac{\partial \overrightarrow{U}}{\partial \overrightarrow{S}^2}\right)^T \tag{4.31}$$

The magnitude of the translation ρ is not observable directly from the frame-to-frame features, therefore it is eliminated from the state vector $\overrightarrow{U}$, hence defining a new five dimensional state vector $\overrightarrow{U}^0$ equal to $\overrightarrow{U}$ stripped from the magnitude of translation:

$$\overrightarrow{U}^0 = \begin{pmatrix} \theta = atan(\frac{V_2}{V_1}) \\ \phi = acos(\frac{V_3}{\sqrt{V_1^2+V_2+V_3}}) \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix}$$

The covariance matrix $\underline{\Sigma}_{\overrightarrow{U}^0}$ of $\overrightarrow{U}^0$ is obtained from $\underline{\Sigma}_{\overrightarrow{U}}$ by removing the row and column corresponding to ρ (i.e., the third row and the third column). Although ρ is not included in the state vector and not updated with $\overrightarrow{U}^0$, it is covariant with it and therefore, it will be affected when $\overrightarrow{U}^0$ is changed. We discuss how to deal with this issue in Section 4.4.5. In the same way the orientation of the (constant) acceleration $\overrightarrow{a}_v$ is transformed to spherical coordinates to get a_ϕ and a_θ. The computational cost of the operations involved in determining $\overrightarrow{U}^0$ and its covariance matrix is equivalent to 474 multiplications.

Filtering

The filtering of $\overrightarrow{U}^0$ is based on the following measurement equations:

$$\left\{ h(\overrightarrow{U}^0(t), \overrightarrow{Z}^2(t)) = 0 \right. . \qquad (4.32)$$

This equation has a slight abuse of notation, since when h is evaluated, $\overrightarrow{U}^0$ is assumed to be augmented with a $\rho = 1$. The Jacobian matrix $\underline{H}_{\overrightarrow{U}^0}$ is computed as follows:

$$\underline{H}_{\overrightarrow{U}^0} = \underline{H}_{\overrightarrow{S}^2} \frac{\partial \overrightarrow{S}^2}{\partial \overrightarrow{U}^0}, \qquad (4.33)$$

$\underline{H}_{\overrightarrow{U}^0}$ is a $K \times 6$ matrix and its computation involves $180K$ multiplications. The matrix $\underline{\Sigma}_{h(\overrightarrow{Z}^2)}$ retains the same expression as in the previous section, however it is evaluated at a magnitude $\rho = 1$. Now for this case, we have $N = 12$. The method that gives the best speedup for $N = 12$ is c_1. Therefore, we use c_1 for this step and, with N equal to N the cost of this step will be $168K + 1080 + 180K = 348K$.

Adjusting the magnitude of $\vec{V}$

After updating $\overrightarrow{U}^0$ (the state vector with un-scaled translation), it has to be re-scaled back by ρ to obtain the updated $\overrightarrow{U}^u$. Since the scale of the translation ρ is covariant with the rest of $\overrightarrow{U}^0$ (through the covariance $\underline{\Sigma}_{\overrightarrow{U}}$), when $(\overrightarrow{U}^0)$ is updated, ρ needs to be updated accordingly. The variance of ρ in $\underline{\Sigma}_{\overrightarrow{U}}$ is the element at position (3,3) i.e., $\underline{\Sigma}_{\overrightarrow{U}15}$ (15^{th} element in row-major order). Denoting by $\overrightarrow{\rho}$ the 1-vector containing the single element ρ, the covariance $\underline{\Sigma}_{\overrightarrow{\rho}\overrightarrow{U}^0}$ is the third row of $\underline{\Sigma}_{\overrightarrow{U}}$ without its third element. Let $\underline{W}$ be the matrix (row vector) defined as $\underline{W} = \underline{\Sigma}_{\overrightarrow{\rho}\overrightarrow{U}^0}\underline{\Sigma}_{\overrightarrow{U}^0}^{-1}$, ρ, its variance and $\underline{\Sigma}_{\overrightarrow{\rho}\overrightarrow{U}^0}$ can be updated as follows(See Appendix A):

$$\begin{cases} \rho^u = \rho + \underline{W}(\overrightarrow{U}^{0u} - \overrightarrow{U}^0) \\ \underline{\Sigma}^u_{\overrightarrow{\rho}\overrightarrow{U}^0} = \underline{W}\underline{\Sigma}^u_{\overrightarrow{U}^0} \\ \underline{\Sigma}^u_{\overrightarrow{U}15} = \underline{\Sigma}_{\overrightarrow{U}15} - \underline{W}(\underline{\Sigma}_{\overrightarrow{U}^0} - \underline{\sigma}^u_{\overrightarrow{U}^0})\underline{W}^T \end{cases} . \tag{4.34}$$

However, we want the scale of the incremental translation to stay fixed at ρ instead of ρ^u and hence a rescaling should be done as follows:

$$\begin{cases} \underline{\Sigma}^u_{\overrightarrow{\rho}\overrightarrow{U}^0} = \frac{\rho}{\rho^u}\underline{\Sigma}^u_{\overrightarrow{\rho}\overrightarrow{U}^0} \\ \underline{\Sigma}^u_{\overrightarrow{U}15} = \left(\frac{\rho}{\rho_u}\right)^2 \underline{\Sigma}^u_{\overrightarrow{U}15} \\ \rho^u = \rho \end{cases} . \tag{4.35}$$

The updated vector $\overrightarrow{U}^u$ and its covariance matrix $\underline{\Sigma}^u_{\overrightarrow{U}}$ are obtained by augmenting $\overrightarrow{U}^{0u}$ with ρ^u, and $\underline{\Sigma}_{\overrightarrow{U}^0}$ with $\underline{\Sigma}^u_{\overrightarrow{\rho}\overrightarrow{U}^0}$ and $\underline{\Sigma}^u_{\overrightarrow{U}15}$. 557 multiplication operations are involved in the scale adjustment step.

Reverting Back To Euclidean

After obtaining $\overrightarrow{U}_u$ it needs to be reverted back to the Euclidean representation. The updated Euclidean vector $\overrightarrow{S}^{2u}$ is determined as:

$$\overrightarrow{S}^{2u} = \begin{pmatrix} U_3^u cos(U_1^u) sin(U_2^u) \\ U_3^u sin(U_1^u) sin(U_2^u) \\ U_3^u cos(U_2^u) \\ U_1^u \\ U_2^u \\ U_3^u \end{pmatrix}$$

The corresponding updated covariance matrix $\underline{\Sigma}^u_{\overrightarrow{S}^2}$ is:

$$\underline{\Sigma}^u_{\overrightarrow{S}^2} = \left(\frac{\partial \overrightarrow{S}^{2u}}{\partial \overrightarrow{U^u}}\right) \underline{\Sigma}^u_{\overrightarrow{U}} \left(\frac{\partial \overrightarrow{S}^{2u}}{\partial \overrightarrow{U^u}}\right)^T \tag{4.36}$$

The computational cost invloved in those operations is equivalent to 454 multiplications.

Updating the Remaining State Variables

After updating $\overrightarrow{S}^2$ we need to update the unobserved part $\overrightarrow{S}^1$ of $\overrightarrow{S}$. This is done using the covariance matrix $\underline{\Sigma}_{\overrightarrow{S}}$ which acts as a string probabilistically connecting $\overrightarrow{S}^2$ and $\overrightarrow{S}^1$, through the covariance submatrix $\underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2}$. The updated $\overrightarrow{S}^{1u}$ and its covariance are $\underline{\Sigma}^u_{\overrightarrow{S}^1}$ given by the following equations (Appendix B):

$$\begin{cases} \underline{W} = \underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2}\underline{\Sigma}^{-1}_{\overrightarrow{S}^2} \\ \overrightarrow{S}^{1u} = \overrightarrow{S}^1 + \underline{W}(\overrightarrow{S}^{2u} - \overrightarrow{S}^2) \\ \underline{\Sigma}^u_{\overrightarrow{S}^1\overrightarrow{S}^2} = \underline{W}\underline{\Sigma}^u_{\overrightarrow{S}^2} \\ \underline{\Sigma}^u_{\overrightarrow{S}^1} = \underline{\Sigma}_{\overrightarrow{S}^1} - \underline{W}(\underline{\Sigma}_{\overrightarrow{S}^2} - \underline{\Sigma}^u_{\overrightarrow{S}^2})\underline{W}^T \end{cases} . \tag{4.37}$$

The cost of this update can be evaluated as $12N^2 - 99N + 360$ given that $\underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2}$ is an $N - 6$ by 6 matrix.

Therefore the total cost of this update can be defined as follows:

$$c_5 = 348K + 12N^2 - 99N + 2925 + 4N^2 + 4N + 131. \tag{4.38}$$

This cost is linear in K and quadratic in N. The speedup with respect to c_0 is shown in Figure 4.15. This figure shows up that the speedup is dramatically larger than the methods presented in the previous sections. For the case $(K = 200, N_t = 50)$ the speedup is 90.089 times and the time required is about $7ms$. Another important thing about this method is that for a moderate N about (N_t about 50 features) the main computational cost is due to N. Therefore, K can be increased significantly while incurring a very low increase in the total cost. Figure 4.16 shows the increase of the cost for N fixed to 200 while increasing K from 100 to 2000. The cost for including $K = 1000$ frame-to-frame features is only 1.7 times larger than the cost needed for 100.

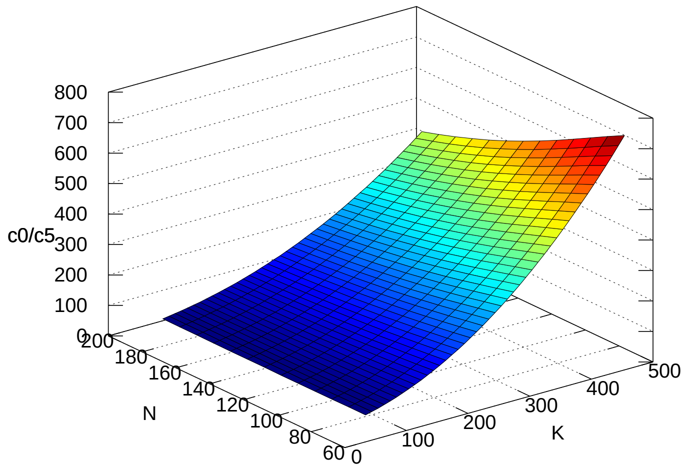


Figure 4.15: Speedup of the partial filtering method c_5 for different values of K and N. The reduced computational cost is hundreds of times smaller than the original cost.

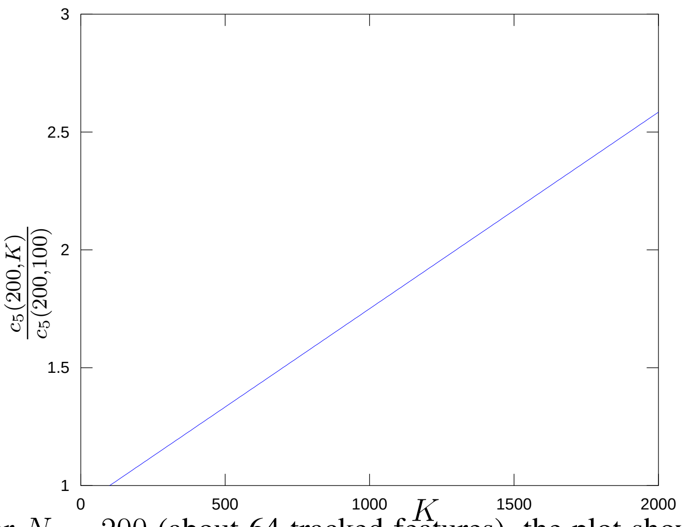


Figure 4.16: For $N = 200$ (about 64 tracked features), the plot shows, when the number of tracked features increases from 100 to 2000, the increase in the cost versus the cost at 100. The addition of 2000 frame-to-frame features requires only 2.6 times more computation than the addition of 100.

4.5 Experimental Results

In this section, we study the impact of the incorporation of the frame-to-frame features on the accuracy of the results. The EKF filter is chosen as the main filter (the filter to which the frame-to-frame information is to be added) as it is the most widely used filter. Nevertheless, the conclusions generalize to any analytic filter.

4.5.1 Simulation Results

To assess the effect of the extra update step using the frame to frame feature on the EKF, we performed numerous Monte-Carlo simulations as follows. We randomly generated $N = 50$ 3D points within a cube of size $4m^3$ centered at $(0,\ 0,\ 5m)$ (the units are not really relevant but “meters” are used to better illustrate the extent of the errors in the results). Then a sequence of random small motions are applied to a virtual camera in such a way that it is always fixating at the centroid of the cloud. For every motion in the sequence, zero-mean Gaussian noise is added to the projections of the 3D points on the corresponding camera frame. Also, at every frame another random cloud of $K = 200$ points with the same dimensions is generated, and its projections on the last two frames (augmented with noise) are used to simulate the frame-to-frame features.

The results presented below correspond to the average of 50 runs. For each run, both the regular EKF, using feature tracked across many frames only, and our filter involving frame-to-frame features, are used to estimate the $3D$ parameters. The time required to perform the update given the 200 frame-to-frame features is generally less than $8ms$, on an Intel Pentium Pentium M 2.13 GHz with a C++ implementation. And since the time required is linear in the number of features, it is easy to determine the number of features that can be used within the available time budget to maintain a real-time performance.

4.5.2 3D Parameters Errors

The error in the translation and translational velocity is determined as the angle in degrees between the true and estimated directions. The error in rotation is taken as $R\hat{R}^T - I_3$ as described in Section 3.7.1. The error in rotational velocity and 3D feature positions is taken as the Root Mean Square (RMS) error between the true and estimated vectors. We also look at the re-projection errors of the last estimate of the 3D features in all the previous images. The means of those errors for the 50 runs are shown in Figures 4.17,4.18,4.19,4.20,4.21 and 4.22 respectively.

The main observation regarding the results is that using the frame-to-frame features in the extra update step improves the accuracy of all the 3D parameters. The errors in most of these error is reduced by almost a factor of two. The improvement in the $3D$ parameters accuracy is reflected in a reduction of the re-projection error as shown in (Figure 4.22).

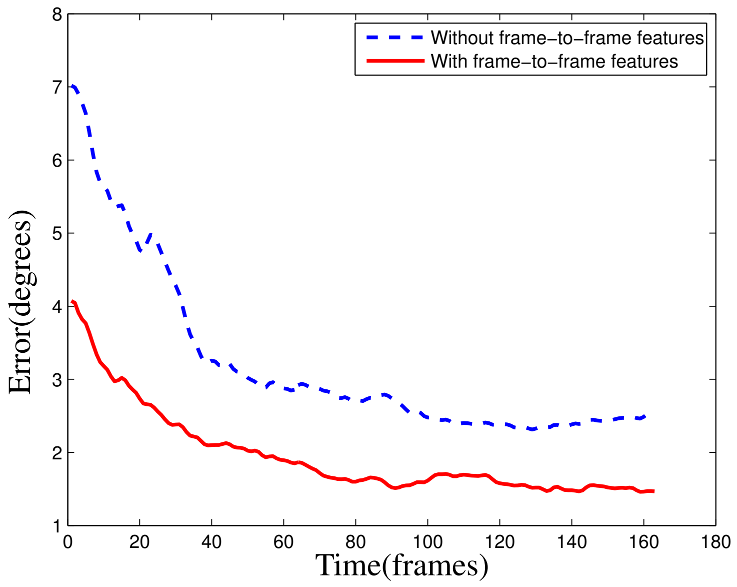


Figure 4.17: Translation direction error in degrees. The first frames before convergence are truncated. The introduction of the frame-to-frame features reduces the error by a factor of 2.

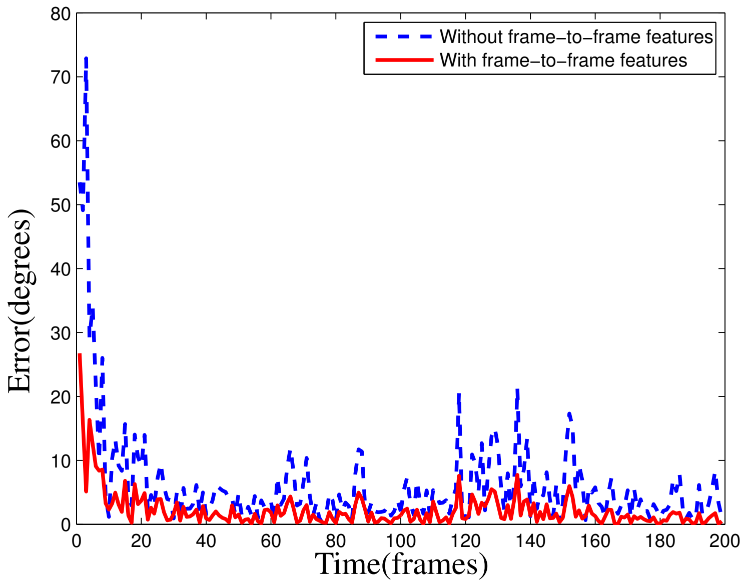


Figure 4.18: Translational velocity error in degrees. The velocity error is significantly reduced when using the frame-to-frame features

4.5.3 Motion Consistency

To study the effect of the extra update step on the consistency of the filter, the Normalized Estimation Error Square (NEES) [11] is used to characterize the consistency of the

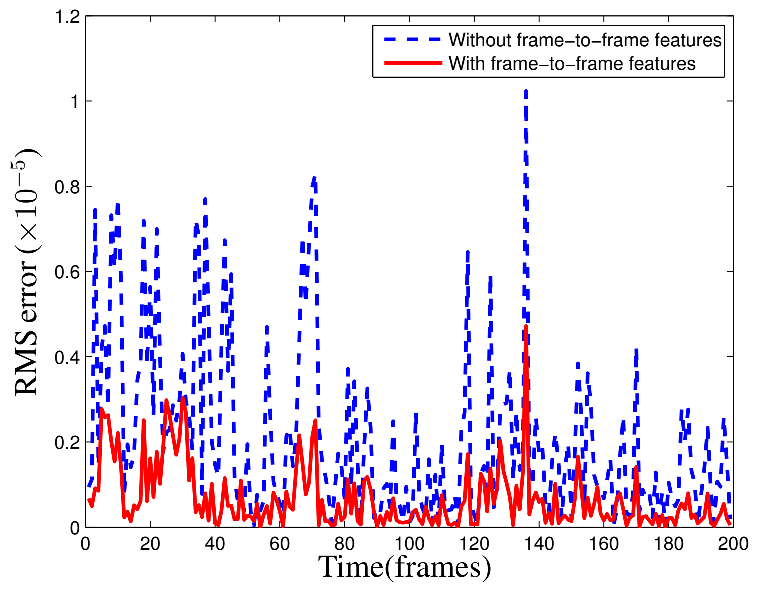


Figure 4.19: Rotational velocity error. Significant reduction of the error with the use of frame-to-frame features.

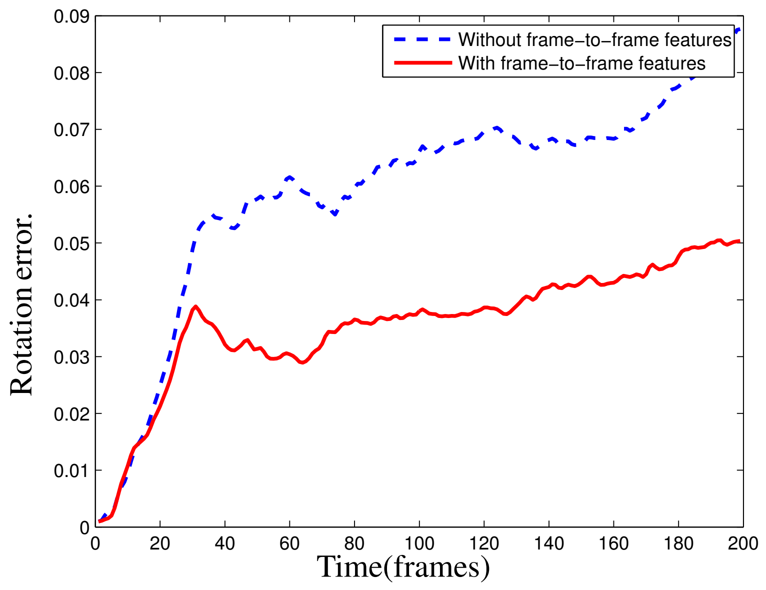


Figure 4.20: Rotational error. The introduction of the frame-to-frame features reduces the error almost by half.

filtering:

$$\epsilon(t) = (\overrightarrow{S}(t) - \hat{\overrightarrow{S}}(t))^T \Sigma_{\overrightarrow{S}}^{-1}(t)(\overrightarrow{S}(t) - \hat{\overrightarrow{S}}(t)) \tag{4.39}$$

A measure of filter consistency is found by examination of the average NEES over many Monte Carlo runs of the filter. Assuming that the filter is consistent and approx-

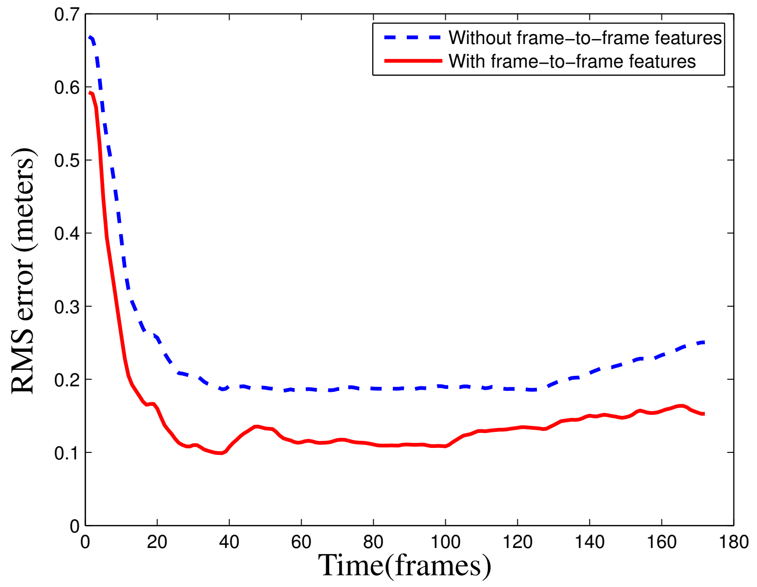


Figure 4.21: Depth error (meters). The frame-to-frame features help in obtaining a much accurate depth.

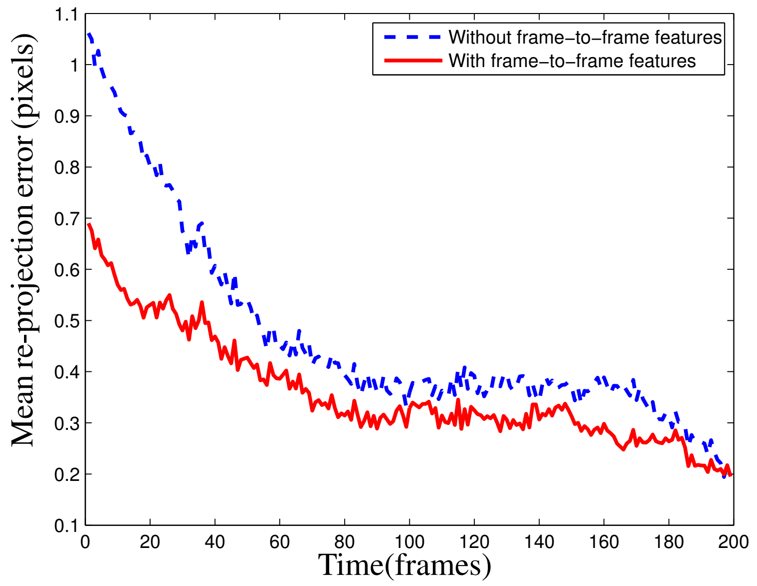


Figure 4.22: Mean re-projection error of the final estimate of the $3D$ points on all the previous frames(pixels). The improvements in the $3D$ parameters estimates is reflected in a smaller re-projection error.

imately linear-Gaussian, then ϵ_t should be χ^2 (chi-square) distributed with $dim(\overrightarrow{S}(t))$ degrees of freedom. Over N_c runs, the average value of ϵ_t is computed as:

$$\bar{\epsilon}_t = \frac{1}{N_c} \sum \epsilon_t, \tag{4.40}$$

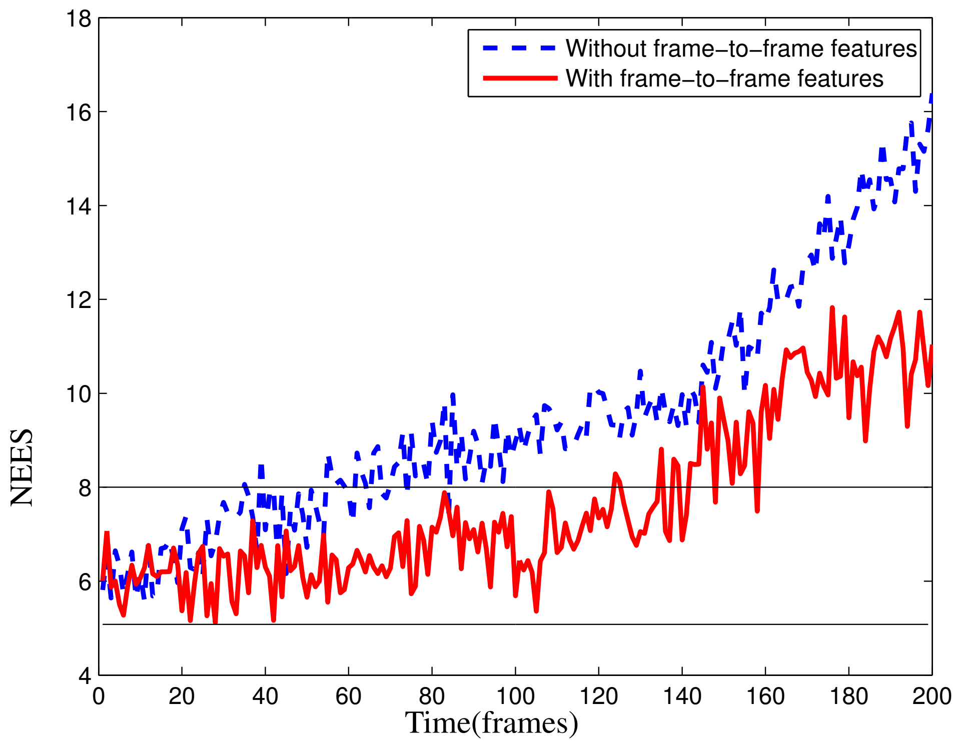


Figure 4.23: Average NEES (Normalized Estimation Error Square) of the camera motion over 50 Monte Carlo runs. The horizontal lines mark the 95% probability concentration region for the 6-dimensional state vector. The use of frame-to-frame features renders the filter more consistent.

therefore, $N_c\bar{\epsilon}_t$ is χ^2 (chi-square) distributed with $N_c \times \dim(\overrightarrow{S}(t))$ degrees of freedom [11]. For the 6 dimensional motion with $N_c = 50$, the 95% probability concentration region for $\bar{\epsilon}_t$ is bounded by the interval $[5.08, 7]$. If $\bar{\epsilon}_t$ rises significantly higher than the upper bound, the filter is optimistic, if it tends below the lower bound, the filter is conservative. Figure 4.23 shows the average NEES over 50 runs for the regular EKF, and for the augmented one with frame-to-frame features. This figure shows that the consistency of the filter is greatly improved by the introduction of the frame-to-frame features.

4.6 Results on Real Image Sequences

We performed tests on images taken in our lab with a calibrated camera moved by hand. Figure 4.24 shows a frame from this sequence, the trajectories of the image projections while the camera is being moved. Also the frame-to-frame features are shown as arrows (not to scale) pointing to the direction of the matches in the next frame. The features were detected using Shi and Tomasi's method [107] and tracked using a pyramidal implementation of the KLT tracker [18]. 50 tracked features were maintained at every time-step and 200 frame-to-frame features were determined between every two consecutive frames. The improvement due to the incorporation of the frame-to-frame features is assessed by the re-projection error of the last $3D$ estimates on all the previous frames. As in the case of the simulated data, Figures 4.25 and 4.26 show that introducing the frame-to-frame features reduces significantly the re-projection error, and hence leads to better $3D$ estimates.

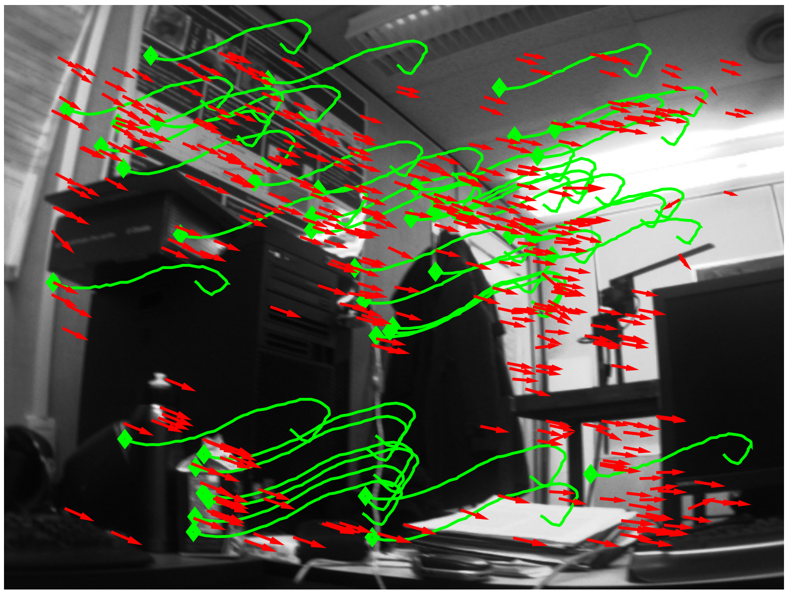


Figure 4.24: Trajectories of features tracked over many frames and frame-to-frame features (arrows not to scale) shown on the first frame.

4.7 Conclusions

This chapter presented an approach to fold the information from frame-to-frame features in analytical SFM filters. The approach can be applied to any filter as long as it maintains a mean vector and a covariance matrix of the estimates. It is accomplished through an extra filtering step that requires virtually no coding change in the main filter and is computationally economic to be able to accommodate hundreds of frame-to-frame features with a state vector of 50 tracked features in about $10ms$. We provided also experimental results in which the extra filtering step is added to the EKF filter and showed significant improvement.

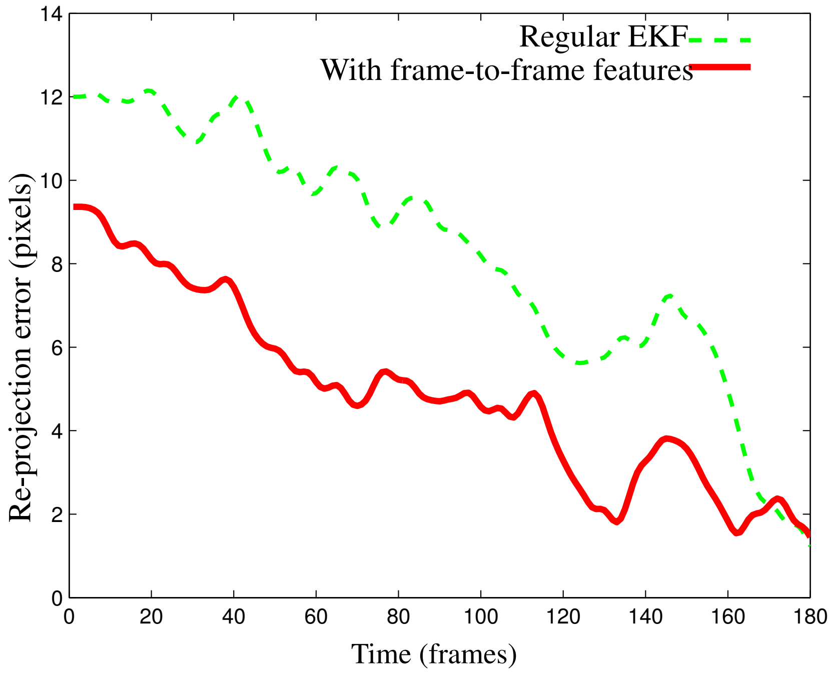


Figure 4.25: Improvement due to the incorporation of the frame-to-frame features shown in terms of the re-projection error of the final state vector on all the previous frames of the first sequence.

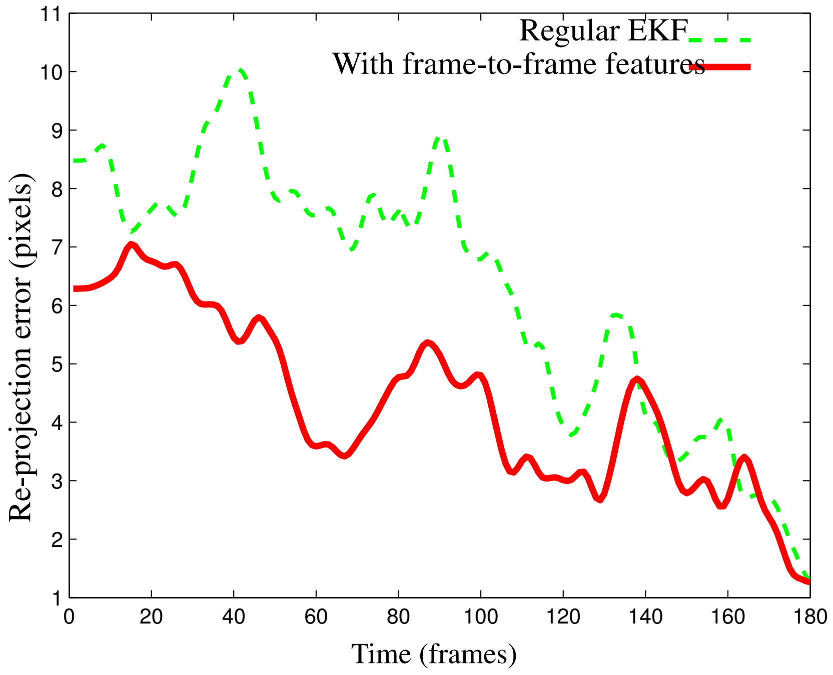


Figure 4.26: Improvement due to the incorporation of the frame-to-frame features shown in terms of the re-projection error of the final state vector on all the previous frames of the second sequence.

When using real data, the issue of outliers in the frame-to-frame features arises. A way to overcome this is to do many filtering steps using different small subsets of frame-to-frame features in each step. During those steps only the velocity is updated, and when an update is found to satisfy the largest number of frame to frame features, this update is propagated to the remaining elements of the state vector.

The presented approach, although designed for filter based SFM, can be also used to augment non-filter based approaches such as the PTAM approach [68] since this approach performs a non-linear optimization and maintains a Gaussian distribution of the estimates.

Also, the presented solution, by accommodating both implicit and explicit measurement equations (explicit equations can be seen as a subset of implicit equations) extends naturally to any types of frame-to-frame measurements other than point-wise features. For example, curves and edges can be used as long as one can formulate a measurement equation relating those frame-to-frame measurements to the camera velocity.

Chapter 5

Scalable Accurate Multi-Hypothesis SFM Estimation

This chapter deals with the issue of SFM scalability to accommodate a large number of features while maintaining a real-time performance. The state of the art filter in terms of scalability is the RBPF filter introduced in the SLAM community as the FASTSLAM2.0 [85] and extended to the visual case by Eade and Drummond [33]. However, in terms of accuracy the RBPF does not fare as well as the EKF. The filter introduced in this chapter achieves the same scalability as the RBPF but with an accuracy matching that of the GLF which is the state of the art filter in accuracy.

5.1 Introduction

One of the main problems of the EKF-SFM (and some of the state-of-the art filters such as the GLF) is that they don't scale efficiently to accommodate large number of features. The computational cost growing quadratically with the number of features in the state-vector (and cubically with the number of features observed at at every frame), quickly becomes un-suitable for real-time operations. The RBPF has been introduced to solve this issue, firstly in SLAM with the FastSLAM2.0 filter [85] and then extended to visual SLAM by Eade and Drummond [33] and Sim *et al.* [108]. The RBPF achieved a very good scalability, with a computational cost that is almost linear in the number of features. However, as mentioned in Chapter 2, the accuracy of the RBPF is lower than the EKF (Section 2.5.3). The main issue behind this is the fact that, in order to preserve the statistical independence between the prediction and update phases of the filter, the update of the structure parameters relies on a motion estimate that is only predicted from the last time step and that does not involve any information from the new measurements. Hence, the performance of the filter is highly influenced by the quality of this update. In the cases where the motion varies rapidly, this results a detrimental effect on the filter. This chapter addresses this problem and introduces a filtering approach with the same scalability of the RBPF but with a performance matching the GLF. The proposed approach bears similarity to the RBPF [33] (and hence the similar scalability) and to the Random Sample Consensus (RANSAC) principle [39]. The state vector is divided into two parts. The first part contains the motion (pose and velocity) of the camera plus a small number of $3D$ features and the second part contains the remaining features. The first part is estimated using a low dimensional EKF and every feature in the second part is estimated conditionnally on the first part using an individual EKF. Multiple hypotheses are considered such that, for each hypothesis, the $3D$ features in the first part of the hypothesis are chosen randomly and hence the similarity with RANSAC (since only those features will be involved in the motion estimaton). This as discussed later (Section 5.3.1) is one of the factors behind the increased accuracy of the presented approach. The hypotheses are similar to the particles in the RBPF but with major differences: (1)the motion is estimated with a full covariance matrix that is propagated over time, (2)the $3D$ feature EKFs are based on a motion that takes into consideration not only the prediction from the motion estimate at the previous time step but also the most recent measurements, and (3) those EKFs are conditioned on the distribution of the motion not only on its mean. Also instead of computing weights for the particles and then performing a resampling based on those weights, the hypotheses in the presented approach are tested by computing an approximation to the re-projection error of the state-vector (See Section 2.4.1 for a definition of the re-projection error) on all the frames. Testing the hypotheses with this error is very successful in selecting one of the best hypotheses.

In the next section we provide a detailed description of the RBPF and pinpoint the problem that reduces its efficiency. Then, we introduce a solution to this problem outlining how it contrasts with the RBPF approach. A full formulation of the proposed approach is presented subsequently.

5.2 The RBPF-SFM and Description of the Problem

The RBPF uses the important characteristic of SFM and SLAM that features are independent given the camera motion. Then, a particle filter can be used in which each particle is composed of a motion sample and N low dimensional filter EKFs, one for each of the N features. Based on the following decomposition of the state vector,

$$\begin{cases} \overrightarrow{S} = [\overrightarrow{S}^1; \overrightarrow{S}^2] \\ \overrightarrow{S}^1 = [\overrightarrow{\Omega}(t); \overrightarrow{T}(t); \overrightarrow{\Omega}(t); \overrightarrow{V}] \\ \overrightarrow{S}^2 = [\overrightarrow{P}^i(t); , ...] \qquad\qquad i \in \{1, ..., N\} \end{cases}, \tag{5.1}$$

the probability distribution of the state vector can be written as:

$$\pi(\overrightarrow{S}|\overrightarrow{Z}) = \pi(\overrightarrow{S}^1|\overrightarrow{Z})\pi(\overrightarrow{S}^2|\overrightarrow{S}^1, \overrightarrow{Z}). \tag{5.2}$$

Since the $3D$ features in $\overrightarrow{S}^2$ are independent given the motion in $\overrightarrow{S}^1$, the above distribution can be further factored as:

$$\begin{aligned} \pi(\overrightarrow{S}|\overrightarrow{Z}) &= \pi(\overrightarrow{S}^1|\overrightarrow{Z})\pi(\overrightarrow{P}^1|\overrightarrow{S}^1, \overrightarrow{q}^1)...\pi(\overrightarrow{P}^N|\overrightarrow{S}^1, \overrightarrow{q}^N) \\ &= \pi(\overrightarrow{S}^1|\overrightarrow{Z})\Pi_{i=1}^N \pi(\overrightarrow{P}^i|\overrightarrow{S}^1, \overrightarrow{q}^i) \end{aligned} \tag{5.3}$$

In a sequential setting the above equation can be written in the form of Equation (A.5) (in Appendix A):

$$\begin{gathered} \pi(\overrightarrow{S}(t)|\overrightarrow{Z}(0:t)) = \\ \frac{\pi(\overrightarrow{S}^1(t)|\overrightarrow{Z}(0:t-1))\pi(\overrightarrow{Z}(t)|\overrightarrow{S}^1(t))}{\pi(\overrightarrow{Z}(t)|\overrightarrow{Z}(0:t))} \times \\ \Pi_{i=1}^N \frac{\pi(\overrightarrow{P}^i(t)|\overrightarrow{S}^1(t), \overrightarrow{q}^i(0:t-1))\pi(\overrightarrow{q}^i(t)|\overrightarrow{S}^1(t), \overrightarrow{P^i}(t))}{\pi(\overrightarrow{q}^i(t)|\overrightarrow{q}^i(0:t))} \end{gathered}. \tag{5.4}$$

Now every one of the factors

$$\frac{\pi(\overrightarrow{P}^i(t)|\overrightarrow{S}^1(t), \overrightarrow{q}^i(0:t-1))\pi(\overrightarrow{q}^i(t)|\overrightarrow{S}^1(t), \overrightarrow{P^i}(t))}{\pi(\overrightarrow{q}^i(t)|\overrightarrow{q}^i(0:t))}$$

in Equation (5.4) can be determined conditionally on $\overrightarrow{S}^1$ using a low dimensional EKF as explained in Section A.2.1. However, the first factor

$$\frac{\pi(\overrightarrow{S}^1(t)|\overrightarrow{Z}(0:t-1))\pi(\overrightarrow{Z}(t)|\overrightarrow{S}^1(t))}{\pi(\overrightarrow{Z}(t)|\overrightarrow{Z}(0:t))}$$

in Equation (5.4) can not be determined using an EKF since evaluating the likelihood of $\overrightarrow{S}^1$ given $\overrightarrow{Z}$ involves using the 3D points $\overrightarrow{P}^i$ which would then violate the con-

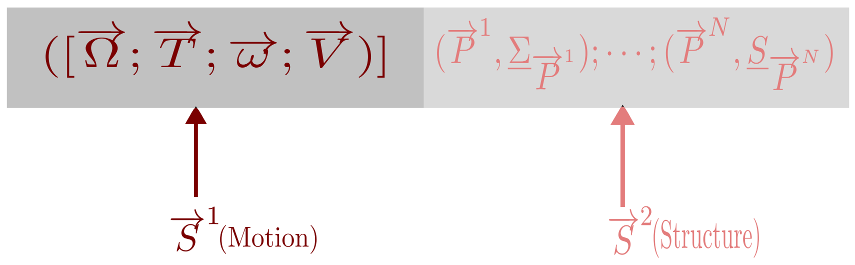

$([\vec{\Omega};\vec{T};\vec{\omega};\vec{V})]$
$(\vec{P}^1,\Sigma_{\vec{P}^1});\cdots;(\vec{P}^N,S_{\vec{P}^N})$
$\vec{S}^1$(Motion)
$\vec{S}^2$(Structure)

Figure 5.1: RBPF particle structure. It consists of a motion sample, and the Gaussian representations (mean and covariance) of every $3D$ point conditioned on the motion sample.

ditional independence in Equation (5.3). That is why in the RBPF the motion $\vec{S}^1$ is estimated using particle filtering, and for each particle the $\vec{P}^i s$ are estimated using individual EKFs. Therefore, a particle (Figure 5.1) will be represented by a motion sample $\vec{S}^{1,(j)}$ and N Kalman filters that estimate the N $3D$ points conditioned on the motion, with $j \in \{1,..,N_s\}$ and N_s is the number of particles. Earlier versions of the RBPF [84] used the prior distribution as importance sampling function (see Section A.2.2). At every time step, the motion in every particle $\vec{S}^{1,(j)}(t)$ is predicted from $\vec{S}^{1,(j)}(t-1)$ using the system evolution equation (Equation (2.17)), then all the $3D$ points distributions in this particle $(\vec{P}^{i,(j)}(t), \Sigma^{(j)}_{\vec{P}^i}(t))$ are updated using individual EKFs. Finally a new weight is computed for the particle based on Equation (A.23):

$$W^{(j)}(t) \propto \pi(\vec{Z}(t)|\vec{S}^{1,(j)}(t)). \tag{5.5}$$

Since this likelihood can not be determined directly without the $3D$ points, a marginalization over all the $3D$ points is used:

$$\begin{aligned} W^{(j)}(t) &\propto \pi(\vec{Z}(t)|\vec{S}^{1,(j)}(t)) \\ &= \sum_{i=1}^{N} \pi(\vec{Z}(t)|\vec{S}^{1,(j)}(t), \vec{P}^{i,(j)}(t))\pi(\vec{P}^{i,(j)}(t)|\vec{S}^{1,(j)}(t-1), \vec{Z}(t-1)). \end{aligned} \tag{5.6}$$

Note that the marginalization in Equation (5.6) is done over the predicted values of the $3D$ points not their updated values in order to keep the statistical independence. The two distributions in the above equation are Gaussian with the first one resulting directly from the projection equation and the second is the predicted distribution of the $3D$ points.

The RBPF as described above suffers from an obvious inefficiency arising from the proposal distribution used. The motion is sampled in accordance to the prediction and does not consider the most recent measurement $\vec{Z}(t)$ acquired at time t; instead, the measurement is incorporated through resampling. This approach is problematic especially when the motion is large, because newly sampled motions will be mostly falling in regions of low measurement likelihood and hence, getting discarded in the resampling step with high probability; therefore, a large number of samples would be required. To deal with that problem, Montemerlo [85] proposed using the optimal importance function (Equation (A.19)) which takes into consideration not only the previous estimate but the most recent measurements. In this approach, adopted also in [33, 108], the motion is sampled from a Gaussian distribution of $\overrightarrow{S}^1(t)$ obtained by combining in an EKF style, the predicted distribution $\pi(\overrightarrow{S}^{1,(j)}, \overrightarrow{S}^{1,(j)}(t-1))$ with the measurement equation $h(\overrightarrow{S}^1(t), f(\overrightarrow{S}^2(t-1)), \overrightarrow{Z}(t))$ (Figure 5.2). The resulting Gaussian distribution has a covariance matrix $\underline{\Sigma}^{(j)}_{\overrightarrow{S}^1}(t)$ and a mean $\overrightarrow{\mu}^{(j)}_{\overrightarrow{S}^1}(t)$ given by:

$$\begin{aligned}\underline{\Sigma}^{(j)}_{\overrightarrow{S}^1}(t) &= (\underline{H}^T_{\overrightarrow{S}^1}\underline{\Sigma}^{-1}_{h(\overrightarrow{Z},\overrightarrow{S}^2)}\underline{H}_{\overrightarrow{S}^1} + \underline{\Sigma}_f^{-1})^{-1} \\ \overrightarrow{\mu}^{(j)}_{\overrightarrow{S}^1}(t) &= \underline{\Sigma}^{(j)}_{\overrightarrow{S}^1}(t)\underline{H}^T_{\overrightarrow{S}^1}\underline{\Sigma}^{-1}_{h(\overrightarrow{Z},\overrightarrow{S}^2)}(\overrightarrow{Z}(t) - \hat{\overrightarrow{Z}}^{(j)}) + f(\overrightarrow{S}^{(j)}(t)\end{aligned}, \tag{5.7}$$

where,

$$\underline{\Sigma}_{h(\overrightarrow{Z},\overrightarrow{S}^2)} = \underline{\Sigma}_{\overrightarrow{Z}} + \underline{H}_{\overrightarrow{S}^2}\underline{\Sigma}^{(j)}_{\overrightarrow{S}^2}(t-1)\underline{H}^T_{\overrightarrow{S}^2} \tag{5.8}$$

$\underline{H}_{\overrightarrow{S}^1}$ and $\underline{H}_{\overrightarrow{S}^2}$ are the Jacobian matrices of h with respect to $\overrightarrow{S}^1$ and $\overrightarrow{S}^2$ respectively. $\hat{\overrightarrow{Z}}^{(j)}$ is the vector of the re-projected features by the predicted particle $[\overrightarrow{S}^{1,(j)}; \overrightarrow{S}^{2,(j)}]$.

5.2.1 The RBPF Problem

Even with using the optimal importance function as explained above, the RBPF still has a major shortcoming: when the motion is sampled, it uses the $3D$ features vector predicted from the previous time step. After this step, each $3D$ feature in $\overrightarrow{S}^2$ is updated in an individual EKF. This EKF can not use the newly sampled motion in $\overrightarrow{S}^1(t)$ in order to keep the statistical independence from the measurements. Therefore, the predicted motion $f(\overrightarrow{S}^{1,(j)}(t-1))$ is used instead (Figure 5.2). This shows that although the optimal importance function is being used, the way samples are drawn from this importance function is not optimal in the sense of providing particles that lie in the regions of highest probability of the posterior. This is because the predicted motion (especially when dealing with fast motions, involves an uncertainty that spreads out the particle instead of being concentrated around the mode of the posterior). To deal with that, some research [35, 67, 108] used mixed proposal distributions that combine hypotheses from the motion model (nominal particles) and from the observation models (dual particles). One shortcoming of such approaches is the need of another modality beside vision such as wheel encoders [67], stereo [35], or the need to use the $3D$ points in the particle in generating the dual hypothesis [108] which means that those $3D$ points don't get updated using the image measurements at the times when the particle they belong to is used as a dual hypothesis.

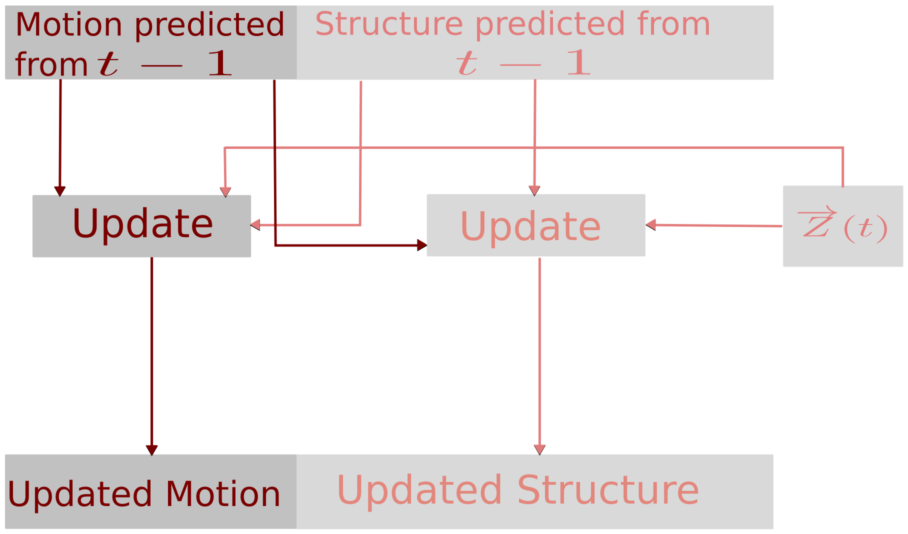


Figure 5.2: RBPF update: The motion is sampled from a distribution combining the predicted motion, the predicted structure and the current measurements. The structure is updated using the predicted motion, the predicted structure and the current measurement.

5.2.2 Proposed Approach

The Rao-Blackwellization principle based on dividing the state-vector into two parts, one to be estimated using particle filtering and the other analytically is a very good way to achieve scalability. However, dividing the state-vector into a structure part and a motion part is neither the only way nor the optimal way to proceed. In this chapter we propose another division based on introducing a small random number of features with the motion in $\vec{S}^1$. The rationale behind this is clarified later in this section and the subsequent one. Therefore, we construct multiple hypotheses where the state vector in every hypothesis is divided into two parts: a first part that consists of the camera motion and a small number N_p of random $3D$ features and another part containing the remaining $3D$ features conditioned on the first part (Figure 5.3). The N_p features that are associated with the camera motion are chosen randomly for every particle by drawing putative sets from the features at the first frame. Since the motion in every hypothesis has some $3D$ features

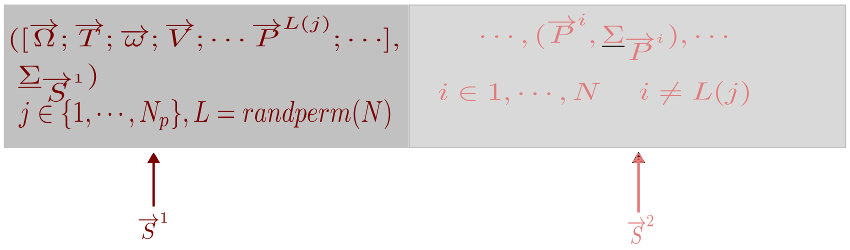

$([\vec{\Omega};\vec{T};\vec{\omega};\vec{V};\cdots\vec{P}^{L(j)};\cdots],$
$\Sigma_{\vec{S}^1})$
$j\in\{1,\cdots,N_p\}, L=randperm(N)$
$\cdots,(\vec{P}^i,\Sigma_{\vec{P}^i}),\cdots$
$i\in 1,\cdots,N \quad i\neq L(j)$
$\vec{S}^1$
$\vec{S}^2$

Figure 5.3: Hypothesis structure in our approach. The first part consists of a motion sample and a few number of $3D$ features with one covariance matrix. The second part consists of the Gaussian representations(mean and covariance) of every $3D$ point conditioned on the first part.

associated with it and sharing with it a common covariance matrix, the motion can be updated using those $3D$ features and their image projections in an EKF, and this very updated motion, which takes into consideration the most recent image measurement, is used to update the remaining 3D features in the second part of the sample. This leads to an efficient update (Figure 5.4) of the samples especially in cases where the motion is changing rapidly. Furthermore, since the motion samples in different hypotheses are updated using different random samples, the presented approach is robust to outliers and is able to achieve a high accuracy as discussed in Section 5.3.1. The following section provides a detailed formulation of the proposed approach.

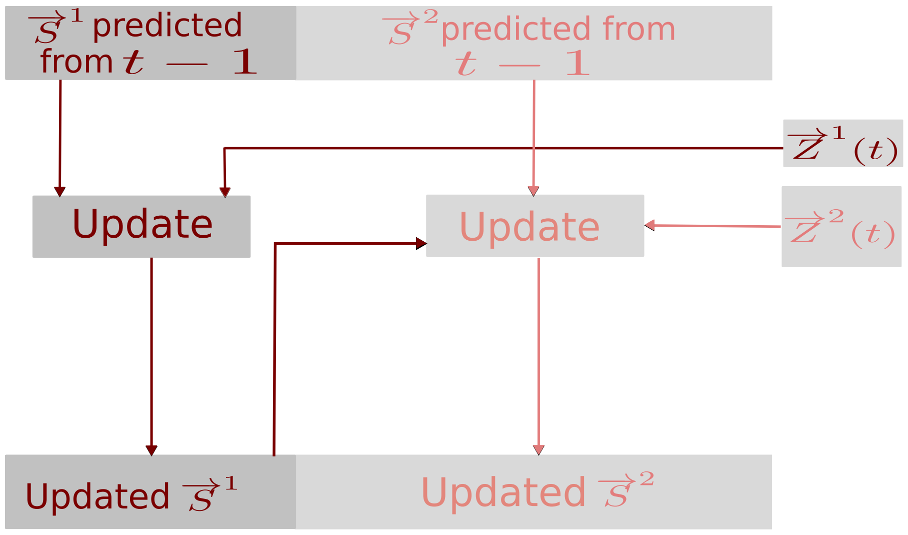


Figure 5.4: Update style in the proposed approach. Notice that the structure in every hypothesis gets updated based on an updated motion.

5.3 Formulation

Assume that the state vector is divided into two parts: A part $\overrightarrow{S}^1$ containing the camera motion plus a few number N_p of features, chosen randomly from the N $3D$ features in the state vector, and another part $\overrightarrow{S}^2$ consisting of the remaining features:

$$\begin{cases} \overrightarrow{S} = [\overrightarrow{S}^1; \overrightarrow{S}^2] & \\ \overrightarrow{S}^1 = [\overrightarrow{\Omega}(t); \overrightarrow{T}(t); \overrightarrow{P}^{L(l)}(t); , ...], & L = randperm(N),\ l \in [1, k] \\ \overrightarrow{S}^2 = [\overrightarrow{P}^i(t); , ...], & i = 1, ..., N,\ \ i \neq L(l), \end{cases} \tag{5.9}$$

where $randperm(N)$ is a random permutation of the numbers from 1 to N. Since the elements ($3D$ feature locations) in the second part $\overrightarrow{S}^2$ are independent given $\overrightarrow{S}^1$, the desired probability distribution can therefore be written as follows (dropping the time indices):

$$\begin{aligned} \pi(\overrightarrow{S}|\overrightarrow{Z}) &= \pi(\overrightarrow{S}^1, \overrightarrow{S}^2|\overrightarrow{Z}^1, \overrightarrow{Z}^2) \\ &= \pi(\overrightarrow{S}^1|\overrightarrow{Z}^1, \overrightarrow{Z}^2)\pi(\overrightarrow{S}^2|\overrightarrow{S}^1, \overrightarrow{Z}^1, \overrightarrow{Z}^2) \\ &= \pi(\overrightarrow{S}^1|\overrightarrow{Z}^1, \overrightarrow{Z}^2)\pi(\overrightarrow{S}^2|\overrightarrow{S}^1, \overrightarrow{Z}^2) \\ &= \pi(\overrightarrow{S}^1|\overrightarrow{Z}^1, \overrightarrow{Z}^2)\Pi_{i=1, i\neq L(j)}^{N}\pi(\overrightarrow{P}^i|\overrightarrow{S}^1, \overrightarrow{q}^i) \end{aligned} \tag{5.10}$$

where $\overrightarrow{Z}^1$ are the image projections of the 3D features in $\overrightarrow{S}^1$ and $\overrightarrow{Z}^2$ are the image projections of the 3D features in $\overrightarrow{S}^2$.

The rationale of dividing the state vector in the presented way is that, since $\overrightarrow{S}^2$ is independent of $\overrightarrow{Z}^1$ given $\overrightarrow{S}^1$, the desired distribution can be approximated as follows:

$$\pi(\overrightarrow{S}|\overrightarrow{Z}) = \pi(\overrightarrow{S}^1|\overrightarrow{Z}^1)\Pi_{i=1, i\neq L(j)}^{N}\pi(\overrightarrow{P}^i|\overrightarrow{S}^1, \overrightarrow{q}^i) \tag{5.11}$$

The approximation comes from the fact that $\overrightarrow{S}^1$ is now dependent only on the first part of the visual measurements $\overrightarrow{Z}^1$ which is a set of few N_p features, instead of being dependent on all the features ($\overrightarrow{Z}^1$ and $\overrightarrow{Z}^2$). Therefore, the motion gets to be estimated using the projections of N_p features only. The advantage, on the other side, is that $\overrightarrow{S}^1$ can be estimated using an EKF and every $3D$ feature in $\overrightarrow{S}^2$ can be estimated conditioned on $\overrightarrow{S}^1$ using its own EKF.

To deal with the fact that $\overrightarrow{S}^1$ is only estimated from few features, we form multiple different hypotheses by selecting a different random set of features to be in $\overrightarrow{S}^1$ of every hypothesis. This is somehow similar to the RANSAC principle: Instead of getting an estimate using all the available data, generate putative sets of minimal required number and determine the different hypotheses that fit those samples and select the one that fits the highest number of measurements. However, unlike RANSAC we are not trying to avoid outliers, instead we are trying to find a good hypothesis. Although our system shares some similarity with the RBPF, we note the following major differences

- In the presented filter, a mixture of Gaussians representation of the camera motion is maintained instead of just multiple samples. $\overrightarrow{S}^1$ in every hypothesis has its full covariance matrix and gets estimated using a regular EKF. This has crucial effects when estimating the individual $3D$ features based on those motions since the uncertainty of the motion is also used and not only its mean value.
- The estimation of the $3D$ features uses the motion updated given the most recent measurements instead of the motion predicted from the previous one only. This makes the system more accurate than the RBPF and guaranteed to converge even with one hypothesis.
- The RBPF involves a resampling step after each iteration. In the proposed, fitness of the hypotheses to all the data is tested based on an approximation of the accumulated re-projection error on all the previous images. The bad hypotheses (with high error) are thrown away and new hypotheses are forked from the good ones.
- The system provides extra robustness over the RBPF since the motion of every hypothesis is determined using a different set of image features. Also, this same fact results in much increased accuracy as described hereafter.

5.3.1 Increased Accuracy

Besides the gain in accuracy obtained from updating the structure using an update motion, another improvement in the accuracy is due to the fact of using several EKF filters with different small sets of features each. This stems from the following observation: considering more measurements in a Kalman Filter (or an extended one) usually results in more accurate estimates. However, if the addition of the new measurements results in an increase in the state vector (i.e., when the additional measurement involves an extra unknown) the addition might result either in an improvement, a deterioration of the estimates. This is similar to the SFM problem. Performing the estimation with a few number of features might give estimates that are either better or worse than the EKF using all the features. This means that if multiple filters with different sets of features are considered, some of those filters will generate better results than the solution with all the features and some will be worse. The following toy-problem illustrates this observation.

Let x, v, and p_i with $i \in \{1, ..., N\}$ be a set of scalar variables with the following dynamics:

$$\begin{cases} x(t+1) &= x(t) + v(t+1) \\ v(t+1) &= v(t) + n_v(t), \quad n_v(t) \sim \mathcal{N}(0, \Sigma_{n_v}) \\ p_i(t+1) &= p_i(t) \end{cases} . \tag{5.12}$$

Assume that we are interested in estimating $x(t)$ from a set of observation of the form:

$$m_i(t) = x(t) + p_i(t) + n_m(t), \quad n_m(t) \sim \mathcal{N}(0, \Sigma_{n_m}). \tag{5.13}$$

As can be seen, any additional measurement equation involves an additional unknown p_i that needs to be added to the state vector and estimated. The above problem can be solved directly using Kalman Filter with gives the optimal online solution for this linear problem. Figure 5.5 shows a typical simulation of this problem with $N = 50$ measurements, in which x is estimated by using all the measurement equations of the 50 points, and by using 20 different sets of 10 chosen randomly from 50 points. The blue line corresponds to the root mean square error of the estimate of x considering all the 50. The red lines correspond to the errors of the 20 estimates using 10 points each. Some of those filters have better estimates than the filter considering all the points, while other have worse estimates. Interestingly, the one with the lowest error is significantly lower than the full filter. The same phenomenon happens in the presented approach. Section 5.3.4 presents a technique to select one of the best hypotheses.

The steps involved in the filtering procedure are as follows (Algorithm 2): At every time step, every hypothesis is updated by first updating $\overrightarrow{S}_1$ using EKF filtering (Section 5.3.2), and then every 3D feature in $\overrightarrow{S}_2$ is updated in a 3-dimensional EKF conditioned on $\overrightarrow{S}_1$ (Section 5.3.3). Then, the weights associated with the hypotheses are evaluated (Section 5.3.4). If the weight of a given hypothesis drops below a given threshold, the hypothesis is discarded and a new hypothesis is forked from the existent hypotheses (Section 5.3.6).

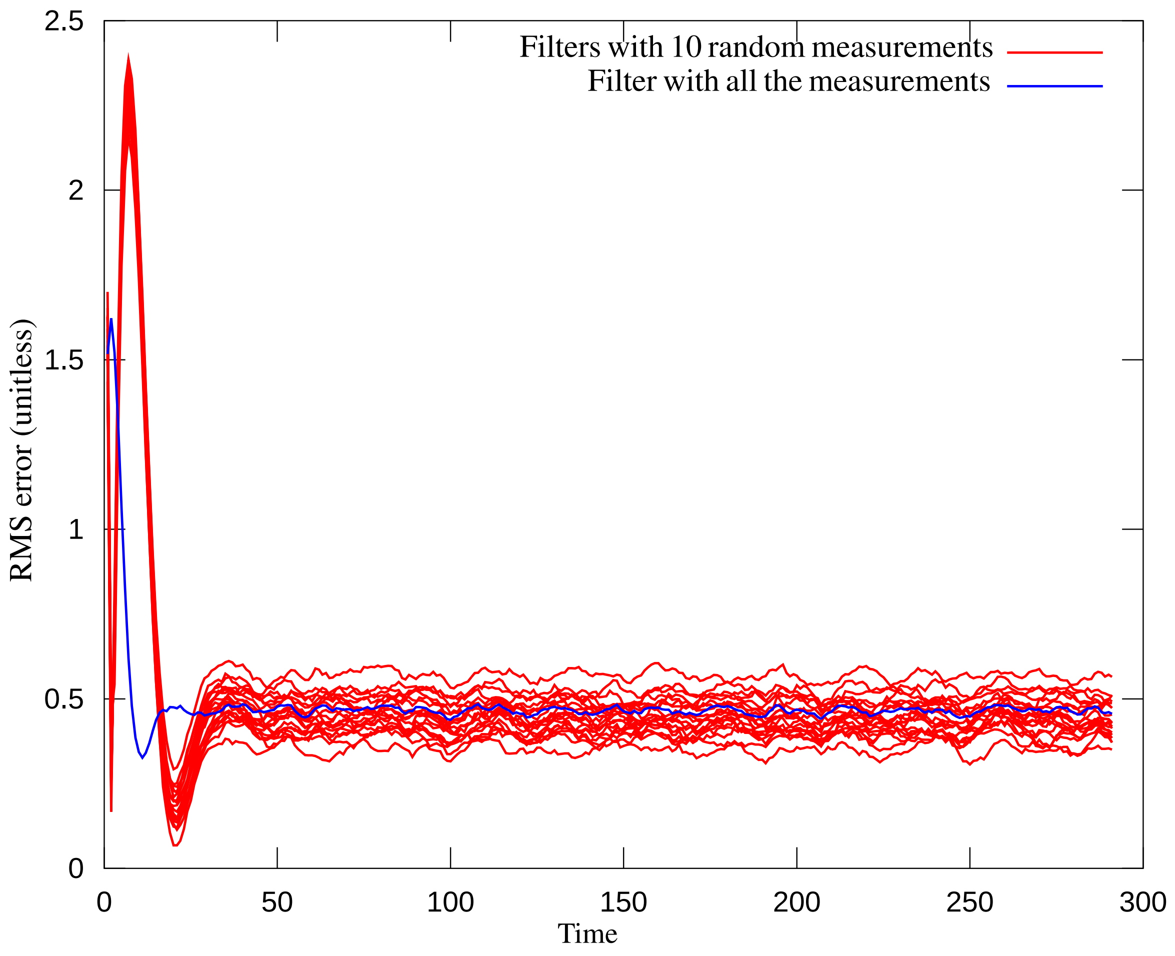


Figure 5.5: Toy problem: estimation errors using 50 measurements versus estimations errors using random sets of 10 measurements.

5.3.2 Filtering Part1 ($\vec{S}^1$) in Every Hypothesis

For each hypothesis $\overrightarrow{S}^{1,(j)}$ we have a system of the form (dropping the index j):

$$\begin{cases} \overrightarrow{S}^1(t) = f(\overrightarrow{S}^1(t-1)) + \overrightarrow{n}_f(t), & \overrightarrow{n}_f(t) \sim \mathcal{N}(0, \Sigma_f) \\ \overrightarrow{Z}^1(t) = h(\overrightarrow{S}^1) + \overrightarrow{n}_{\overrightarrow{Z}^1}(t), & \overrightarrow{n}_{\overrightarrow{Z}^1}(t) \sim \mathcal{N}(0, \Sigma_m), \;\; l = 1, ..., k. \end{cases} \tag{5.14}$$

where f and h represent a dynamical system and measurement equations of the forms presented in Equations (2.34) and (2.35) respectively.

To update $\overrightarrow{S}^{1,(j)}$ from $t-1$ to t, using the standard notations of dynamical systems filtering where $\overrightarrow{S}(t|t-1)$ refers to an estimate of $\overrightarrow{S}$ predicted from the previous estimate and $\overrightarrow{S}(t|t)$ refers to an estimate of $\overrightarrow{S}$ that uses the previous estimate and the most recent measurement, the filtering equations will be (Appendix A):

Algorithm 2: Summary of the proposed filtering.

```
   Input: Features Tracked at every frame
   Output: Motion and Structure at every instant
1  for $i = 0 : N_s$ do
2      Generate a set of 10 random features;
3      Initialize an EKF based on the generated set of features;
4  end
5  while New frames are being captured do
6      Track the features to the new frame;
7      if Number of features in the new frame<threshold then
8          Detect new features;
9      end
10     for $j = 1 : N_s$ do
11         Update the part $\overrightarrow{S}^1$ of the hypothesis $j$ as in Section 5.3.2;
12         Update the part $\overrightarrow{S}^2$ of the hypothesis $j$ as in Section 5.3.3;
13         Compute the error of the hypothesis $j$ as in section 5.3.4;
14     end
15     Discard hypotheses with high re-projection error;
16     Fork new hypotheses from hypotheses with low re-projection error;
17     Output the motion with the lowest error;
18     if New Features are available then
19         Insert them into the filter as in Section 5.3.5;
20     end
21 end
```

Prediction

$$\begin{cases} \overrightarrow{S}^1(t|t-1) & = f(\overrightarrow{S}^1(t-1|t-1)) \\ \underline{\Sigma}_{\overrightarrow{S}^1}(t|t-1) & = \underline{F}_{\overrightarrow{S}^1}(t-1)\underline{\Sigma}_{\overrightarrow{S}^1}(t-1|t-1)\underline{F}^T_{\overrightarrow{S}^1}(t-1) + \underline{\Sigma}_{\overrightarrow{Z}^1} \end{cases}. \tag{5.15}$$

Update

$$\begin{cases} \overrightarrow{S}^1(t|t) = \overrightarrow{S}^1(t|t-1) + \underline{L}(t)(\overrightarrow{q}(t) - h(\hat{\overrightarrow{S}}_1(t|t-1)) \\ \underline{\Sigma}_{\overrightarrow{S}^1}(t|t) = \underline{\Gamma}(t)\underline{\Sigma}_{\overrightarrow{S}^1}(t|t-1)\underline{\Gamma}^T(t) + \underline{L}(t)\Sigma_{\overrightarrow{S}^1}\underline{L}^T(t). \end{cases} \tag{5.16}$$

Gain

$$\begin{cases} \underline{\Lambda}t = \underline{H}_{\overrightarrow{S}^1}(t)\underline{\Sigma}_{\overrightarrow{S}^1}(t)\underline{H}^T_{\overrightarrow{S}^1}(t) + \underline{\Sigma}_{\overrightarrow{Z}^1} \\ \underline{L}(t) = \underline{\Sigma}_{\overrightarrow{S}^1}(t|t-1)\underline{H}^T_{\overrightarrow{S}^1}\underline{\Lambda}^{-1}(t) \\ \Gamma(t) = \underline{I} - \underline{L}(t)\underline{H}_{\overrightarrow{Z}^1}(t). \end{cases} \tag{5.17}$$

Linearization

$$\begin{cases} \underline{F}_{\overrightarrow{S}^1}(t) = \frac{\partial f}{\partial \overrightarrow{S}^1}(\overrightarrow{S}^1(t-1|t-1)) \\ \underline{H}_{\overrightarrow{S}^1}(t) = \frac{\partial h}{\partial \overrightarrow{S}^1}(\overrightarrow{S}^1(t|t-1)). \end{cases} \tag{5.18}$$

For the details of the calculations of the above matrices refer to Appendix D.

5.3.3 Estimation of $\vec{S}^2$

For every feature $\overrightarrow{P}^{i,(j)}$ of $\overrightarrow{S}^{2,(j)}$ we have the following dynamical system:

$$\begin{cases} \overrightarrow{P}^{i,(j)}(t) = \overrightarrow{P}^{i,(j)}(t-1) + \overrightarrow{n}_{\overrightarrow{p}}(t), & \overrightarrow{n}_{\overrightarrow{p}}(t) \sim \mathcal{N}(0, \underline{\Sigma}_{\overrightarrow{p}}) \\ \overrightarrow{q}^i(t) = proj(\underline{R}^{(j)}(t|t)\overrightarrow{P}^{i,(j)}(t) + \hat{\overrightarrow{T}}^{(j)}(t|t)) + \overrightarrow{n}_{\overrightarrow{q}^i}(t), & \overrightarrow{n}_{\overrightarrow{q}^i}(t) \sim \mathcal{N}(0, \underline{\Sigma}_{\overrightarrow{q}^i}). \end{cases} \tag{5.19}$$

Note that in the above equation the measurement model is formulated in terms of $\underline{R}^{(j)}(t|t)$ and $\overrightarrow{T}^{(j)}(t|t)$ which represent the updated motion and not as in the RBPF in terms of $\underline{R}^{(j)}(t|t-1)$ and $\overrightarrow{T}^{(j)}(t|t-1))$ which represent the motion predicted from the previous time step. Also, a major difference with the RBPF is that we have the covariance matrix $\underline{\Sigma}_{\overrightarrow{S}^1}$ of the motion and not just the mean. To incorporate the effect of this covariance in the filtering we use a first order Taylor expansion. Let $\underline{H}_{\overrightarrow{M}}$ be the Jacobian of the measurement equation with respect to $\underline{R}(t)$ and $\overrightarrow{T}(t)$ at $(R^{(j)}(t), \overrightarrow{T}^{(j)}(t))$ ($\underline{H}_{\overrightarrow{M}}$ is a submatrix of $\underline{H}_{\overrightarrow{S}^1}$). Then, $\overrightarrow{P}^{i,(j)}$ can be updated using the standard EKF based on the System 5.19. However, instead of using $\underline{\Sigma}_{\overrightarrow{q}^i}$ we replace it by $\underline{\Sigma}_{\overrightarrow{q}^i} + \underline{\Sigma}_{h(\overrightarrow{M})}$ where $\underline{\Sigma}_{h(\overrightarrow{M})}$ is defined as follows:

$$\underline{\Sigma}_{h(\overrightarrow{M})} = \underline{H}_{\overrightarrow{M}}\underline{\Sigma}_{\overrightarrow{M}}\underline{H}^T_{\overrightarrow{M}}. \tag{5.20}$$

$\underline{\Sigma}_{h(\overrightarrow{M})}$ represents the covariance of the uncertainty in the measurement equation (second equation in System (5.19)) due to the uncertainty in $R^{(j)}(t)$ and $\overrightarrow{T}^{(j)}(t)$

5.3.4 Evaluating the Hypotheses

At every time instant we rank each of the hypotheses based on its fitness to the available measurements. Ideally to test a given hypothesis j at time t, we have to determine its re-projection error $e(j,t)$ on all the frames from 0 to t. The re-projection at time t of the i^{th} $3D$ point of the j^{th} hypothesis on the k^{th} frame and its covariance matrix are given by:

$$\begin{aligned}\hat{\overrightarrow{q}}^{i,(j)}(t,k) &= proj(R^{(j)}(k)\overrightarrow{P}^{i,(j)}(t) + \overrightarrow{T^{(j)}}(k)) \\ \underline{\Sigma}^{(j)}_{\hat{\overrightarrow{q}}^i}(t,k) &= \underline{H}^{(j)}_{\overrightarrow{M}}(k)\underline{\Sigma}^{(j)}_{\overrightarrow{M}}(k)\underline{H}^{(j)T}_{\overrightarrow{M}} + \underline{H}^{(j)}_{\overrightarrow{P}^i}(k)\underline{\Sigma}^{(j)}_{\overrightarrow{P}^i}(k)\underline{H}^{(j)T}_{\overrightarrow{P}^i}\end{aligned} \tag{5.21}$$

where $\underline{H}^{(j)}_{\overrightarrow{P}^i}(k)$ is the Jacobian matrix of the projection equation h on the K^{th} with respect to $\overrightarrow{P}^{i,(j)}$ using the motion $\overrightarrow{M}^{(j)}(k)$ of the j^{th} sample. The error can be written then as:

$$\begin{aligned}e(j,t) = \frac{1}{t+1}\sum_{k=0}^{t}\sum_{i=1}^{N}\rho((&\hat{\overrightarrow{q}}^{i,(j)}(t,k) - \overrightarrow{q}^i(k))^T \\ &(\underline{\Sigma}^{(j)}_{\hat{\overrightarrow{q}}^i}(t,k) + \underline{\Sigma}_{\overrightarrow{q}^i}(k))^{-1} \\ &(\hat{\overrightarrow{q}}^{i,(j)}(t,k) - \overrightarrow{q}^i(k))\end{aligned} \tag{5.22}$$

where ρ is a robust function which we simply define as $\rho(e) = e$ if $0 \leq e < th$ and $\rho(e) = th$ if $e \geq th$, where th is a threshold. The errors are expressed in terms of mean pixel error per feature. However. this approach would require us to keep track of all the image projections and to recompute the re-projection errors of the state vector on all the frames, at every time step, which is too expensive computationally. Therefore, we use an approximation of this error which can be computed recursively at every frame as follows:

$$\begin{aligned}\hat{e}(j,t) = \hat{e}(j,t-1)\frac{t}{t+1} + \\ \frac{\sum_{i=1}^{N}\rho((\hat{\overrightarrow{q}}^{i,(j)}(t,t) - \overrightarrow{q}^i(t))^T(\underline{\Sigma}^{(j)}_{\hat{\overrightarrow{q}}^i}(t,t) + \underline{\Sigma}_{\overrightarrow{q}^i}(t))^{-1}(\hat{\overrightarrow{q}}^{i,(j)}(t,t) - \overrightarrow{q}^i(t))}{t+1}\end{aligned} \tag{5.23}$$

In this case, instead of computing the re-projection error of the state vector $\overrightarrow{S}(t)$ on all the previous frames k, $k = 0, ..., t$, we are computing for every frame k the re-projection error of $\overrightarrow{S}(k)$ on this frame, which would be equivalent to the following:

$$\begin{aligned}\hat{e}(j,t) = \frac{1}{t+1}\sum_{k=0}^{t}\sum_{i=1}^{N}\rho((\hat{\overrightarrow{q}}^{i,(j)}(k,k) - \overrightarrow{q}^i(k))^T(\underline{\Sigma}^{(j)}_{\hat{\overrightarrow{q}}^i}(k,k) + \underline{\Sigma}_{\overrightarrow{q}^i}(k))^{-1} \\ (\hat{\overrightarrow{q}}^{i,(j)}(k,k) - \overrightarrow{q}^i(k))\end{aligned}$$

If the error of a given hypothesis goes above a certain threshold, the hypothesis is thrown away and a new hypothesis is forked from the best available hypothesis, by interchanging landmarks between its $\overrightarrow{S}^1$ and $\overrightarrow{S}^2$.

5.3.5 Initialization and New Features

Upon starting our system, a number of features are detected in the first image and then tracked through subsequent frames. We randomly sample N_s putative sets of $N_p = 10$ features each. For every set an EKF is initialized to estimate the motion and the N_p $3D$ features. This initialization is done as in [26]. After a number of frames when the EKFs are assumed to have converged, for every particle we triangulate the depth of the features in its second part using the optimal method of Oliensis [92]. The covariance of every feature is determined using the unscented transform [63] and the covariance of the motion in the first part as well as the covariance of the detected features. The triangulated $3D$ features and their covariances are used as an initialization of the individual EKFs corresponding to those features.

A similar procedure is used to insert new features. When a new feature is detected, it is tracked for a certain number of frames say from t_1 until t_2. Then, for each one of the available particles, the $3D$ location of the newly detected feature is determined by triangulation using the projection matrices corresponding to the times t_1 and t_2 and then used as an initial values for the EKF of this feature in the considered particle.

When one of the N_p features in the part $\overrightarrow{S}^1$ of a given sample disappears or gets occluded, it is replaced by another feature from the part $\overrightarrow{S}^2$ of the particle, and this latter feature is chosen as the one with the lowest variance. The replacement is done by eliminating the disappeared feature from the state vector and substituting its mean by the mean of the new feature. Also the lines and rows in the covariance matrix corresponding to the old feature are set to zero and only the part corresponding to the autocovariance of the old feature is replaced by the covariance of the new feature.

5.3.6 Forking New Hypotheses

When the re-projection error of a given hypothesis gets higher than a given threshold the hypothesis is discarded and is replaced by a new hypothesis generated from one of the good hypotheses (with low accumulated re-projection error) as follows; Let $(\overrightarrow{S}^{1,(j)}, \overrightarrow{S}^{2,(j)})$ be the hypothesis to fork from. The new hypothesis is generated by interchanging the feature having the largest variance in $\overrightarrow{S}^{1,(j)}$ with the feature having the lowest variance in $\overrightarrow{S}^{2,(j)}$.

5.4 Experimental Results

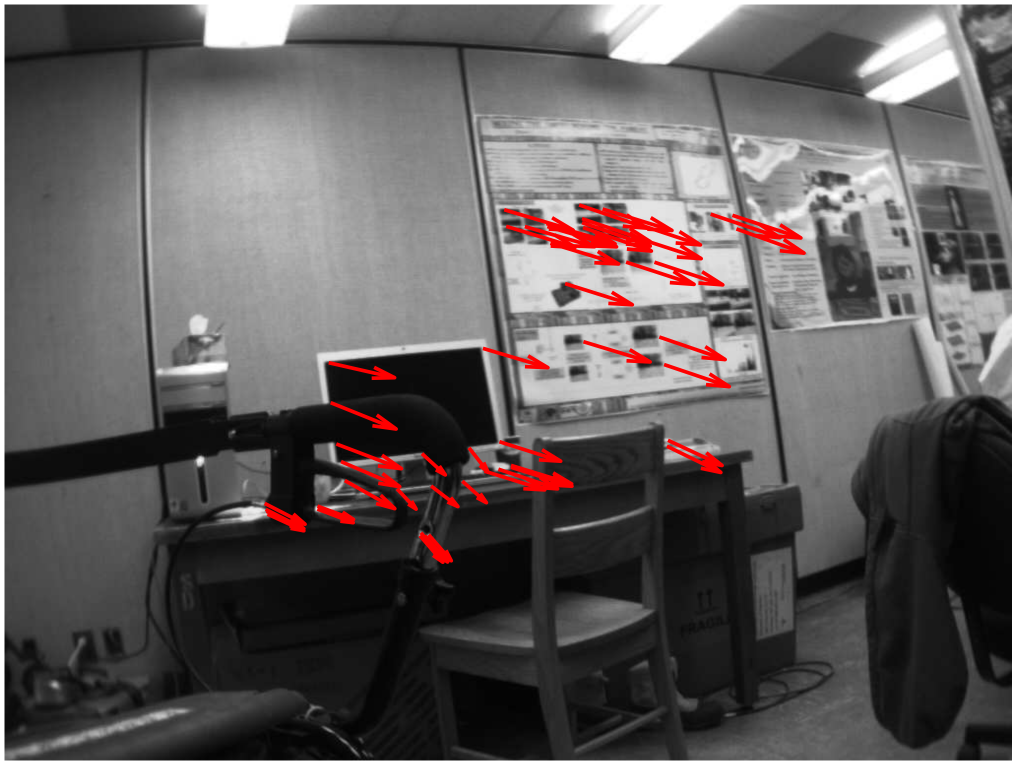


Figure 5.6: A frame from sequence 2 and the directions of the optical flow used to track the features. Mean feature displacement is 3 pixels per frame.

To test the performance of the presented filter, a camera with known calibration parameters is used. As the aim of recursive filtering is to determine the motion that correspond to the least re-projection error of the state vector on all the filtered frames, we use this error as a measure of performance. We compare the performance of our filter to the performance of the EKF and the optimal bundle adjustment and rely on those to assess the performance relative to other filters. We did many tests on different sequences taken inside our lab. In some of the sequences the camera is fixed on a pan-tilt unit and in other sequences the camera is moved by hand. We report hereafter the results on three sequences. In sequence 1 the camera is fixed on a pan-tilt unit and undergoing a smooth pan and tilt motion. Although the camera is rotating, the motion involves a translation due to the arm of the pan tilt unit. The average pixel displacement or optical flow is about 1 pixel per frame. In the second sequence the camera is hand held and moved around the scene with an average displacement of about 3 pixels per frame. In sequence three, the camera is also moved by hand, but the motion is more jerky and the average displacement is about 6 pixels per frame. We used 20 particles in both cases. Figure 5.6 shows an instance of sequence 2 with the directions of optical flow (Not to scale) detected in this frame. In Figure 5.7 are presented the re-projection errors of the state vector at time 160 of all the samples on all the frames from 10 until 160 for sequence 1. The blue lines

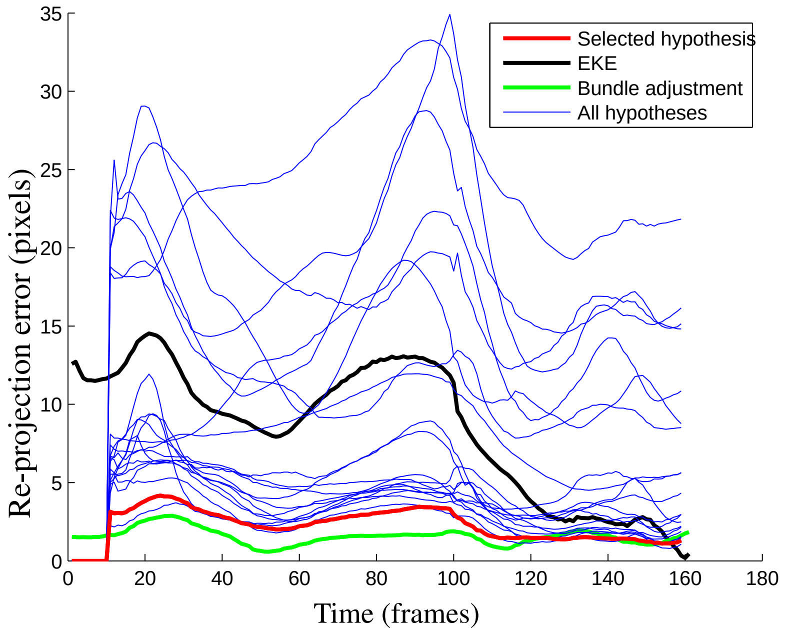


Figure 5.7: Mean re-projection errors of the State Vector at frame 160 on all the previous frames of sequence 1. The best hypothesis (very close to BA) has been successfuly selected by the proposed algorithm.

correspond to all the hypotheses considered in our filter, the black line to the regular EKF SFM, the red line to the selected hypothesis (hypothesis with the highest weight) and the green line to the optimal bundle adjustment solution. Two main observations can be made:

- The particle with the lowest re-projection error in all the frames has been selected successfully by our filter, which means that the approximation of the error as in Section 5.3.4 works well.
- The selected hypothesis performance approaches the performance of bundle adjustment. Also, the re-projection error of this hypothesis is less than three times that of the EKF. This suggests that the performance of the presented filter is similar or better than the performance of the GLF filter. A further improvement is expected if the presented filter is combined with the GLF by replacing the filtering in the local nodes of the GLF with the presented filter.

Figure 5.7 shows the same re-projection error for all the hypotheses, however the red line

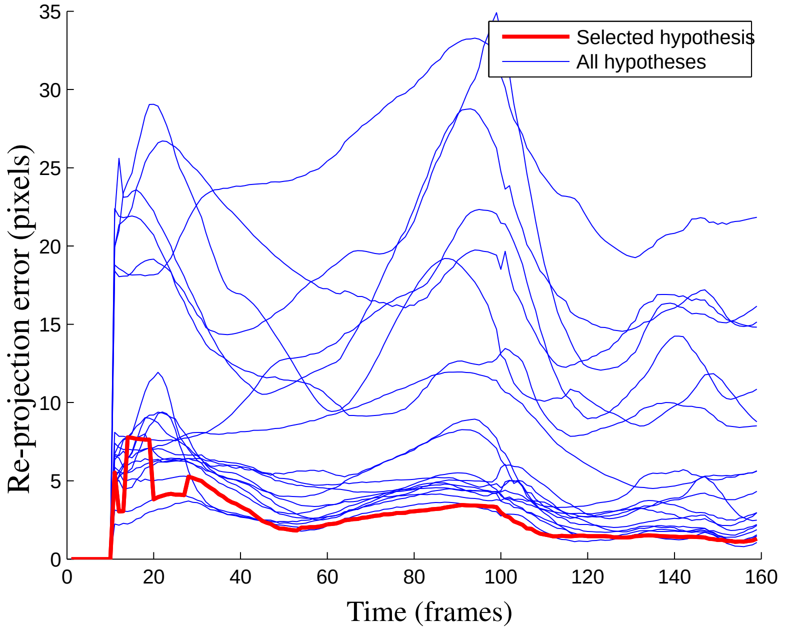


Figure 5.8: Mean re-projection errors of the State Vector of all the hypotheses at frame 160 on all the previous frames of sequence 1. The red line shows the hypothesis that was selected at each time instant. After fluctuating for the first few frames, the best hypothesis is chosen.

now shows the hypothesis that was selected at each time step (instead of the hypothesis selected at last time step as in the previous figure). This figure shows that after few fluctuations,the right hypothesis is selected. Figures 5.9 and 5.10 represent the same results as Figures 5.9 and 5.10 but for sequence 2. The same observations can be made, but this time the error is a little bit higher due to the less smooth motion.

Figures 5.11 and 5.12 correspond to sequence 3. In this case the errors are high all the filters. Also, at the last time step the hypothesis selected by our system is not the best one, however it is not a bad one either and it is still much better than the EKF. Furthermore, as shown in Figure 5.12 (showing the evolution of the selected hypothesis), at most of the time steps, the selected hypothesis was one of the best hypotheses. Only at the last few frames, the not so good one has been chosen. This means that had the filtering been continued a little further, a better hypothesis would have been probably selected. The results on those latter two sequences show the consistency of our results and fortify the conclusions deduced from the first sequence.

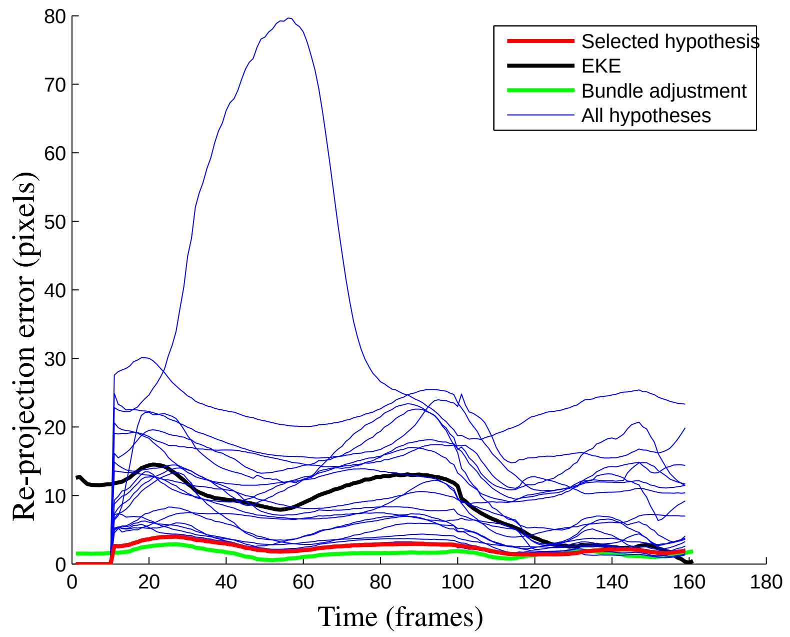


Figure 5.9: Mean re-projection errors of the State Vector at frame 160 on all the previous frames of sequence 2. Again the best hypothesis is chosen.

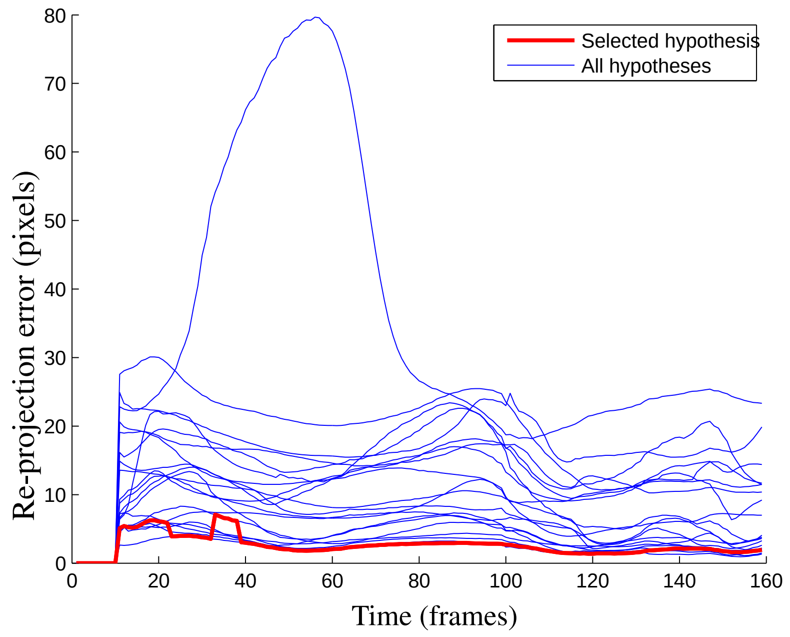


Figure 5.10: Mean re-projection errors of the State Vector of all the hypotheses at frame 160 on all the previous frames of sequence 2. The red line shows the hypothesis that was selected at each time instant.

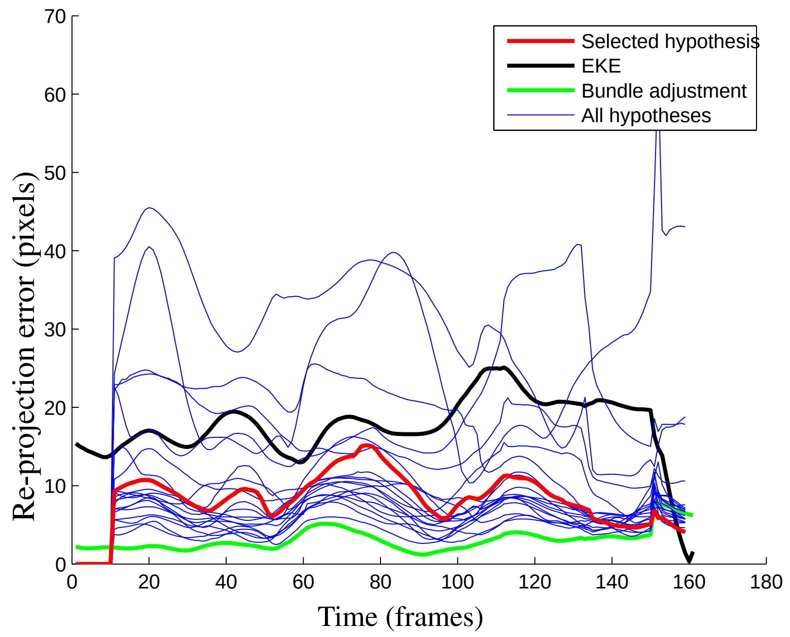


Figure 5.11: Mean re-projection errors of the State Vector at frame 160 on all the previous frames of sequence 3. The selected hypothesis is not the best, however as shown in Figure 5.12, this hypothesis has been chosen only toward at the end of the sequence.

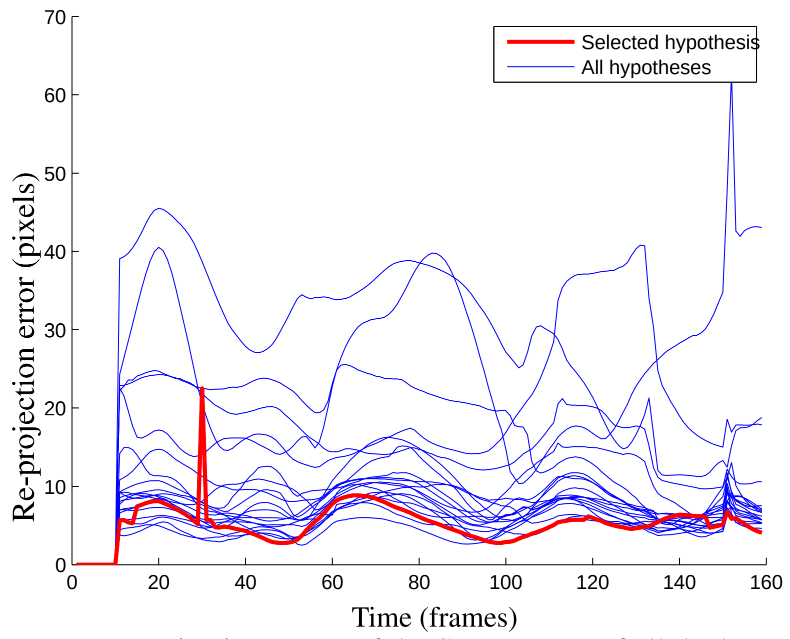


Figure 5.12: Mean re-projection errors of the State Vector of all the hypotheses at frame 160 on all the previous frames of sequence 3. The red line shows the hypothesis that was selected at each time instant. The not so good hypothesis was selected only at the end of the sequence. Before that the best one was selected at almost all the times.

5.4.1 Recovered Depth

Figure 5.13 shows the recovered depth of a set of features. The relative depth of every feature is displayed right beside the feature. A qualitative inspection of the depth suggests a good performance of the filter. The same can be concluded from Figure 5.14 where a 3D tesselation of the recovered points and the trajectory of the camera are shown.

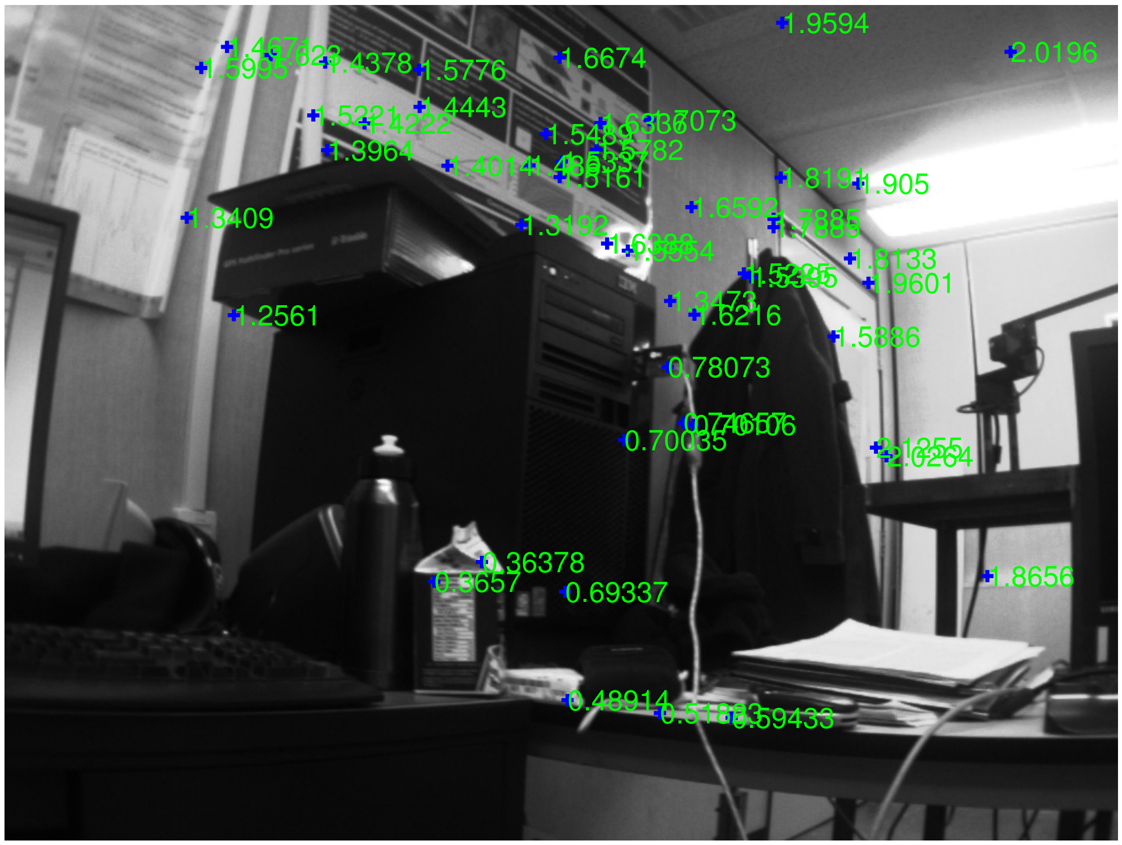
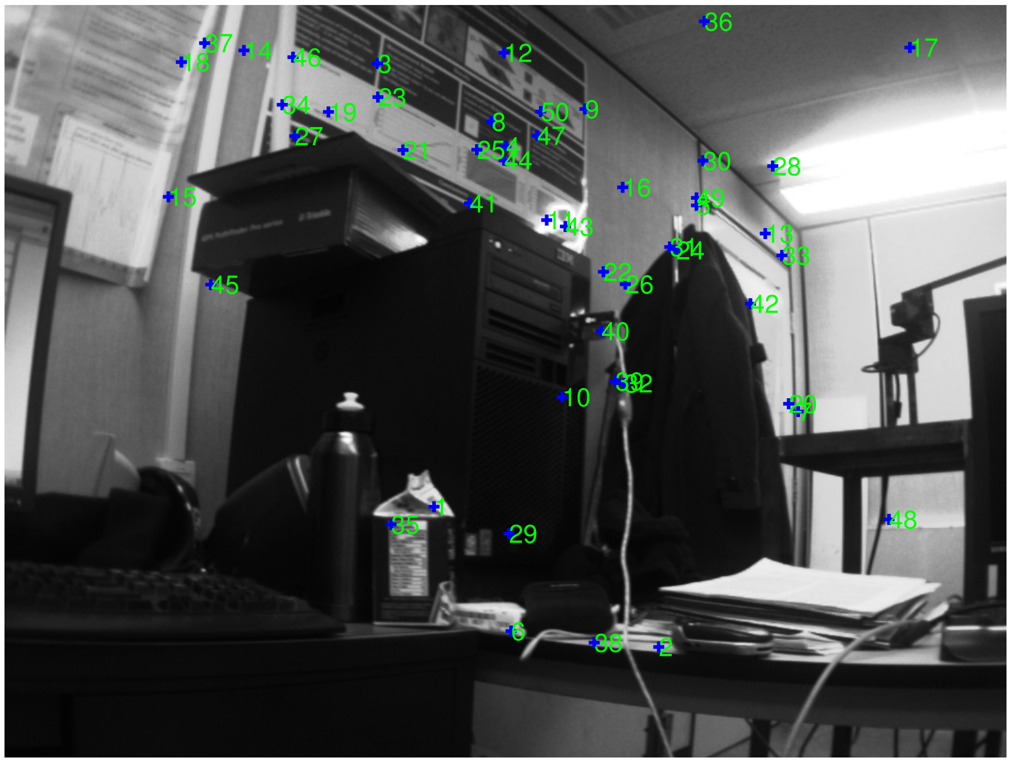


Figure 5.13: Recovered depth showed on one of the frames. The depth (relative) is displayed beside every detected feature. The camera is handheld by a person seated beside the keyboard.

5.5 Conclusion

This chapter presented an approach to solve the scalability issue in recursive SFM estimation. The results showed that we can achieve a comparable accuracy to the GLF while having the same scalability of the RBPF.

The main factor that affects the performance of our approach is the choice of the features in the first part of every hypothesis. If the number of features is not very large,


(a)

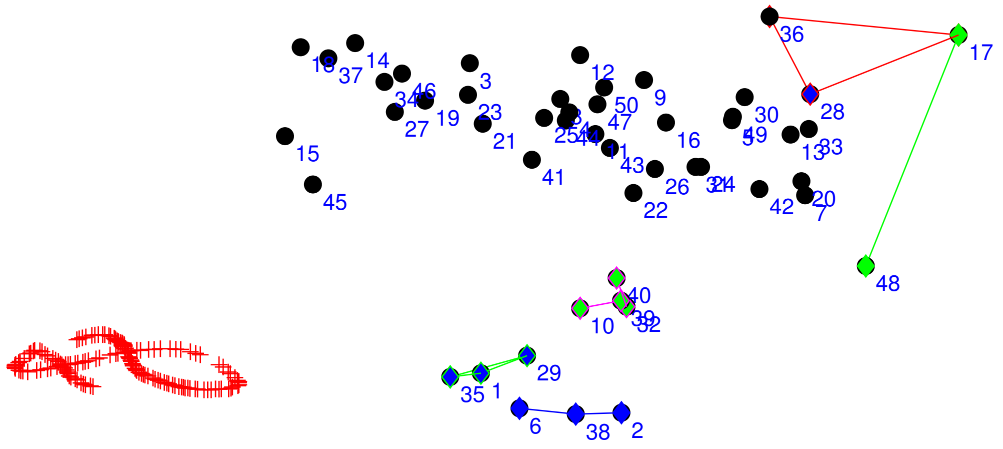

(b)

Figure 5.14: Tessellation of the recovered 3D points with indices to correspond each 3D features to its location in the image. The red crosses represent the camera trajectory. Notice the 3D positions of the points on the edge of the table, the points on the roof, the points on the milk box and the points that are on the wall. Those points seem to be a faithful recovery of the 3D structure of the scene.

choosing the features randomly in each hypothesis works pretty well. However with a very large number of features, this might be far from optimal and therefore a more systematic way should be used in selecting those features. One way is to start with a large number of hypotheses, estimating only the first parts of each hypothesis, which is not very expensive computationally, then the estimation can carry on with the best N_s hypotheses yielding the lowest epipolar error for all the features. Another more interesting way would be to bootstrap the filter with a regular RBPF and then use an approach similar to what Martinec and Pajdla [80] do in their batch optimization approach. Their method consists in selecting four points (can be virtual) that best represent the set of matches in all the images prior to bundle adjustment. Those four points are selected using the so called *rescaled measurement matrix* which is equal to the projection matrix multiplied with all the 3D points. The four points are selected such as their corresponding columns in the latter matrix represent the four dimensional subspace spanned by all the columns of the matrix. The bootstrap estimate determined by the RBPF can be used in a similar fashion to construct a rescaled measurement matrix from which a set of most representative points can be recovered. If such approach is successful, then the filtering can be done with only one hypothesis having in its first part ($\overrightarrow{S}^1$) the set of representative points.

The filtering of the first part in each hypothesis, does not have to be based on an EKF, it can instead use any filtering approach or even key-frame bundle adjustment. This key-frame bundle adjustment might have a real-time potential since the first part of every hypothesis is low dimensional. Also, it might result in an improved accuracy. Substantiating these claims is left for future research.

Chapter 6

Wide Baseline Prediction For Filter-Based SFM

Filter-based approaches work usually in two steps: prediction and update. Prediction is the process of determining a prior estimate for the state vector at time $t + 1$ from the previous estimate at time t. Update is the process of modifying the predicted estimate so it complies with the available measurements. A key issue in those two processes is that the prediction and update should be statistically independent and hence no "common" data can be used in both of them. This is problematic in SFM in the case where the baseline between successive frames (i.e., the displacement between the camera centers) is wide, because, the previous estimate of the motion at t does not allow to solely determine estimate of the motion at $t+1$ accurate enough for the filtering because of the significant change of the motion between the two frames across the wide baseline. Therefore, in the absence of external sources of information, image matching constraints involving the last frame need to be used. However, these constraints can not involve the projections of the 3D features that are in the state vector because this would jeopardize the independence of the prediction from the update. In the case of stereo images, such constraints can be derived from the features matched in the last two frames only. This solution has been used earlier in some previous approaches [35,64]. However, for monocular images, the features matched in the last two frames only, do not provide a measurement of the translation magnitude and therefore, the estimate obtained from those two frames can not be consistent with the gauge at previous time instants. In this chapter, we provide a probabilistic solution to this problem by using features that are matched in the last three frames only. We show that this solution provides reliable prediction of the motion across large baselines.

6.1 Introduction

One of the unsolved problems in filter-based SFM, is the problem of predicting the motion at time $t + 1$ from the motion at time t, across a large baseline without using any other external source of information. This problem can be described as follows: Assume that we have an SFM filter in which the estimated state vector at time t contains, among other parameters, the absolute motion $\overrightarrow{M}(t)$. Assume also that the camera undergoes a relatively large motion between t and $t + 1$. When the frame at $t + 1$ is captured, for the filtering to proceed, $\overrightarrow{M}(t + 1)$ needs to be predicted from $\overrightarrow{M}(t)$ and then adjusted to conform with the visual information of the new frame. However, because of the large baseline, $\overrightarrow{M}(t + 1)$ can not be predicted decently using a random walk model. Some approaches such as Karlsson's [67] rely on wheel encoders or other modality to deal with this. However, in a purely visual approach, another visual information not involving the 3D features in the state vector or their projections needs to be used. The most obvious way to do this is to rely on features matched in the last two frames only. Those feature provide an estimate for the incremental motion $\overrightarrow{M}(t, t + 1)$ which can be combined with $\overrightarrow{M}(t)$ using the motion evolution equations (Equations (2.8)) to result in the desired motion $\overrightarrow{M}(t + 1)$. Such solution has been used in earlier approaches such as the EKF stereo approach of Jung and Lacroix [64] and the stereo $\sigma-$SLAM RBPF-based algorithm of Elinas et. al [35]. While this solution works perfectly in the stereo case where the scale ambiguity is fixed by the distance between the two cameras, it can not be extended to the monocular case, because the translation magnitude in $\overrightarrow{M}(t)$ and the translation magnitude in $\overrightarrow{M}(t, t + 1)$ wouldn't be consistent.

This chapter proposes to solve the issue of motion prediction across a wide monocular baseline by using the features that are shared only between the most recent three frames at every time-step (and hence they are not the projections of any previously used features). One direct way to do this is to use the available motion across the frames $t - 1$ and t, along with the features matched in $t - 1$, t and $t + 1$, to determine a consistent motion across t and $t + 1$. This can be done using either an absolute orientation algorithm (Section 2.4.5) or using the threading constraints described in Chapter 3. However, this would be too expensive computationally, especially in the case of particle filters and multi-hypothesis filters as it needs to be done separately for every particle or hypothesis. Conversly, the proposed method determines the relative motion across those three frames and its covariance using a non-linear minimization of the re-projection errors (done only once in the case of multiple particles). This results, as shown in Figure 6.1, in two motion distributions, the current one in the state vector $[\overrightarrow{M}(t); \overrightarrow{M}(t-1, t)]$ and the newly computed one $[\overrightarrow{M}(t-1,t);\overrightarrow{M}(t,t+1)]$, both of them having different beliefs about the value and the uncertainty of $\overrightarrow{M}(t-1,t)$. The approach presented in this chapter capitalizes on this fact to determine a distribution of $\overrightarrow{M}(t)$ and $\overrightarrow{M}(t,t+1)$ consistent with both distributions and hence to predict the distribution of $\overrightarrow{M}(t+1)$. This latter processing is done separately for every particle in the case of a multi-particle system. However as it is very cheap computationally (involving only manipulation of at most 12×12 matrices), it can be carried out in real time.

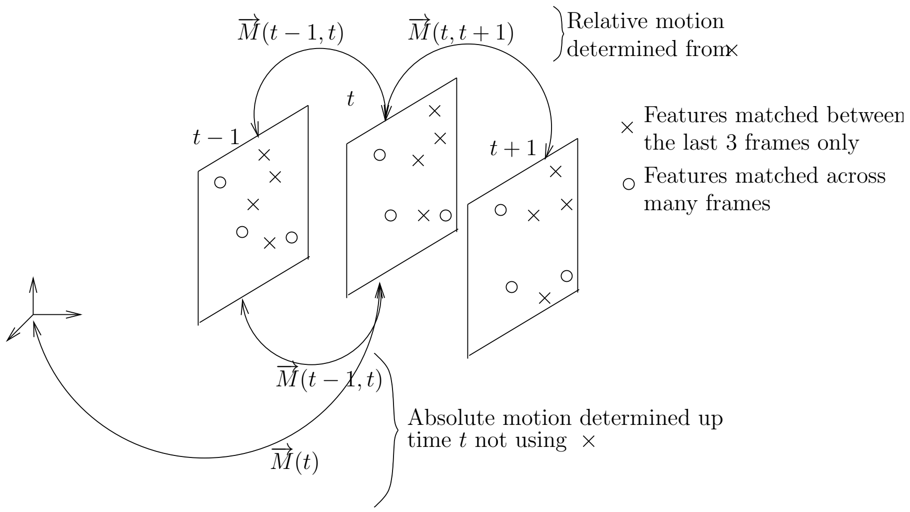

$\overrightarrow{M}(t-1,t)$
$\overrightarrow{M}(t,t+1)$
Relative motion determined from ×
t
$t-1$
$t+1$
× Features matched between the last 3 frames only
○ Features matched across many frames
$\overrightarrow{M}(t-1,t)$
Absolute motion determined up time t not using ×
$\overrightarrow{M}(t)$

Figure 6.1: Prediction over large baselines: The features shared only between the last three frames provide a distribution of $[M(t-1,t);M(t,t+1)]$. The current state vector estimated up to time t provides a distribution of $[M(t);M(t-1,t)]$. These two sources of information, have different (independent) distributions of $M(t-1,t)$. This allows us to determine what the current state vector predicts the distribution of $M(t,t+1)$ to be.

6.2 Methodology

As mentioned in the background chapter (Section 2.5.3), filter-based dynamical system are driven by the velocity and hence the state vecotr consists of the absolute motion $\overrightarrow{M}(t) = [\overrightarrow{\Omega}(t);\overrightarrow{T}(t)]$, the velocity $[\overrightarrow{\omega}(T);\overrightarrow{V}(t)]$ and the $3D$ positions of some features. In the case of wide baseline, we replace the velocity with the incremental motion between the frames $t-1$ and t ($\overrightarrow{M}(t-1,t) = [\overrightarrow{\Omega}(t-1,t);\overrightarrow{T}(t-1,t)]$). For the sake of simplicity, we will take the $3D$ features out of the picture while formulating our approach although we implicitly assume their existence. We assume that the filter maintains at every time step t a Gaussian distribution of the state vector represented by a mean vector

$\overrightarrow{\mu}_{[\overrightarrow{M}(t);\overrightarrow{M}(t-1,t)]}$ and a covariance matrix $\Sigma_{[\overrightarrow{M}(t);\overrightarrow{M}(t-1,t)]}$. The state evolution equations can be written as follows:

$$\overrightarrow{S}(t+1) = f(\overrightarrow{S}(t)) =$$
$$\begin{cases} \overrightarrow{\Omega}(t+1) = SO_3to\mathbb{R}^3\left(\mathbb{R}^3toSO_3(\overrightarrow{\Omega}(t,t+1))\mathbb{R}^3toSO_3(\overrightarrow{\Omega}(t))\right) \\ \overrightarrow{T}(t+1) = \mathbb{R}^3toSO_3(\overrightarrow{\Omega}(t,t+1))\overrightarrow{T}(t) + \overrightarrow{T}(t,t+1) \\ \overrightarrow{\Omega}(t,t+1) = ? \\ \overrightarrow{T}(t,t+1) = ? \end{cases}, \tag{6.1}$$

where the sign "?" reflects the fact that it is not trivial to predict $[\overrightarrow{M}(t,t+1)]$ from $[\overrightarrow{M}(t);\overrightarrow{M}(t-1,t)]$ when the baseline between consecutive camera positions is large. As mentioned in the introduction, our solution for this is to use the features shared only between the last three frames. When using those features, two distributions will be available at every time step:

- The output of the filter at time t represented by a Gaussian distribution of the absolute motion $\overrightarrow{M}(t)$ and the incremental motion $\overrightarrow{M}(t-1,t)$:

$$\pi^0([\overrightarrow{M}(t);\overrightarrow{M}(t-1,t)]) = \mathcal{N}(\overrightarrow{\mu}^0_{[\overrightarrow{M}(t);\overrightarrow{M}(t-1,t)]}, \Sigma^0_{[\overrightarrow{M}(t);\overrightarrow{M}(t-1,t)]}). \tag{6.2}$$

- The three frames at times $t-1$, t and $t+1$ provide a Gaussian distribution of the two incremental motions $\overrightarrow{M}(t-1,t)$ and $\overrightarrow{M}(t,t+1)$:

$$\pi^1([\overrightarrow{M}(t-1,t);\overrightarrow{M}(t,t+1)]) = \mathcal{N}(\overrightarrow{\mu}^n_{[\overrightarrow{M}(t-1,t);\overrightarrow{M}(t,t+1)]}, \Sigma^n_{[\overrightarrow{M}(t-1,t);\overrightarrow{M}(t,t+1)]}). \tag{6.3}$$

Those two distributions are conditioned on two independent sources of information and are independent of the projections of the 3D features in the state vector. Therefore, they can be used to provide a prediction of $\overrightarrow{M}(t+1)$. To illustrate how to do this, we think of the covariance matrices as springs connecting the different estimates as shown in Figure 6.2: To determine $\overrightarrow{M}(t+1)$, we need estimates of $\overrightarrow{M}(t,t+1)$ and $\overrightarrow{M}(t)$. But, the estimates available for those quantities come from different distributions. Getting them to comply to the same distribution is equivalent to having a spring connecting them in Figure 6.2, which can be done first by pulling the two nodes of $\overrightarrow{M}(t-1,t)$ to a place that satisfies both distributions. This will trigger a change in the positions of $\overrightarrow{M}(t,t+1)$ and $\overrightarrow{M}(t)$ so that they both agree with new position of $\overrightarrow{M}(t-1,t)$. In terms of probability distributions those steps can be done by first merging the two distributions of $\overrightarrow{M}(t-1,t)$ and then propagating the change to the distributions of $\overrightarrow{M}(t,t+1)$ and $\overrightarrow{M}(t)$.

h!

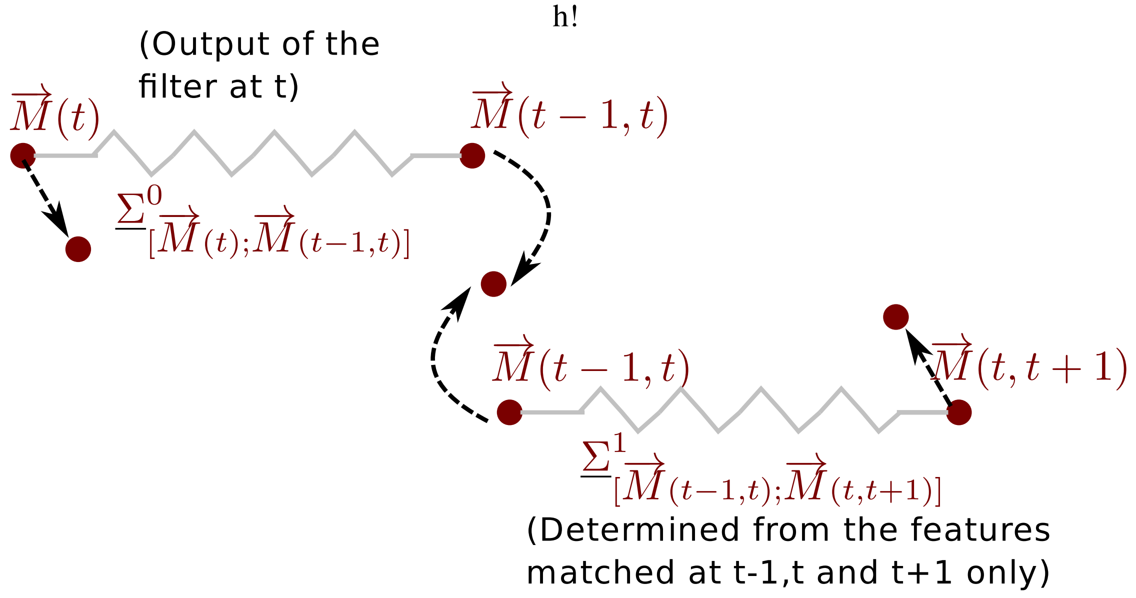

(Output of the filter at t)
$\overrightarrow{M}(t)$
$\overrightarrow{M}(t-1,t)$
$\Sigma^0_{[\overrightarrow{M}(t);\overrightarrow{M}(t-1,t)]}$
$\overrightarrow{M}(t-1,t)$
$\overrightarrow{M}(t,t+1)$
$\Sigma^1_{[\overrightarrow{M}(t-1,t);\overrightarrow{M}(t,t+1)]}$
(Determined from the features matched at t-1,t and t+1 only)

Figure 6.2: The covariance matrices act as springs connecting the different motions. The prediction task requires a connection between $\vec{M}(t)$ and $\vec{M}(t,t+1)$. This is done by moving $\vec{M}(t-1,t)$ to a place that satisfies both distributions and then changing $\vec{M}(t)$ and $\vec{M}(t,t+1)$ accordingly.

Finally those two distributions are used to determine a distribution of the absolute motion $\overrightarrow{M}(t+1)$ which can be used as prediction in the filtering. An important thing to be noted here is that the magnitude of the translation $\vec{\mu}^1_{\overrightarrow{T}(t-1,t)}$ is different than the magnitude of $\vec{\mu}^0_{\overrightarrow{T}(t-1,t)}$, and hence $\pi^1([\overrightarrow{M}(t-1,t);\overrightarrow{M}(t,t+1)])$ needs to be modified in such a way that the magnitude of the translation $\vec{\mu}^1_{\overrightarrow{T}(t-1,t)}$ becomes equal to its magnitude mean in π^0. The flowchart of the prediction process is shown in Figure 6.3 with all the steps and the input and output of each step. A detailed description of each step is provided in the following sections.

6.2.1 Determine The Three Frames Relative Motion

The three-frames motions $\overrightarrow{M}(t-1,t)$ and $\overrightarrow{M}(t,t+1)$ are determined from a set of K features $\vec{q}^{\,i}(t)$, $\vec{q}^{\,i}(t-1)$ and $\vec{q}^{\,i}(t+1)$ with $i \in \{1,...,K\}$. Nister's five point algorithm [88] is employed to determine the essential matrix between $t-1$ and t, and the three-frames absolute orientation algoritm of Fischler and Bolles [43] is used to determine the relative motion between t and $t+1$ within a RANSAC methodology [38]. This

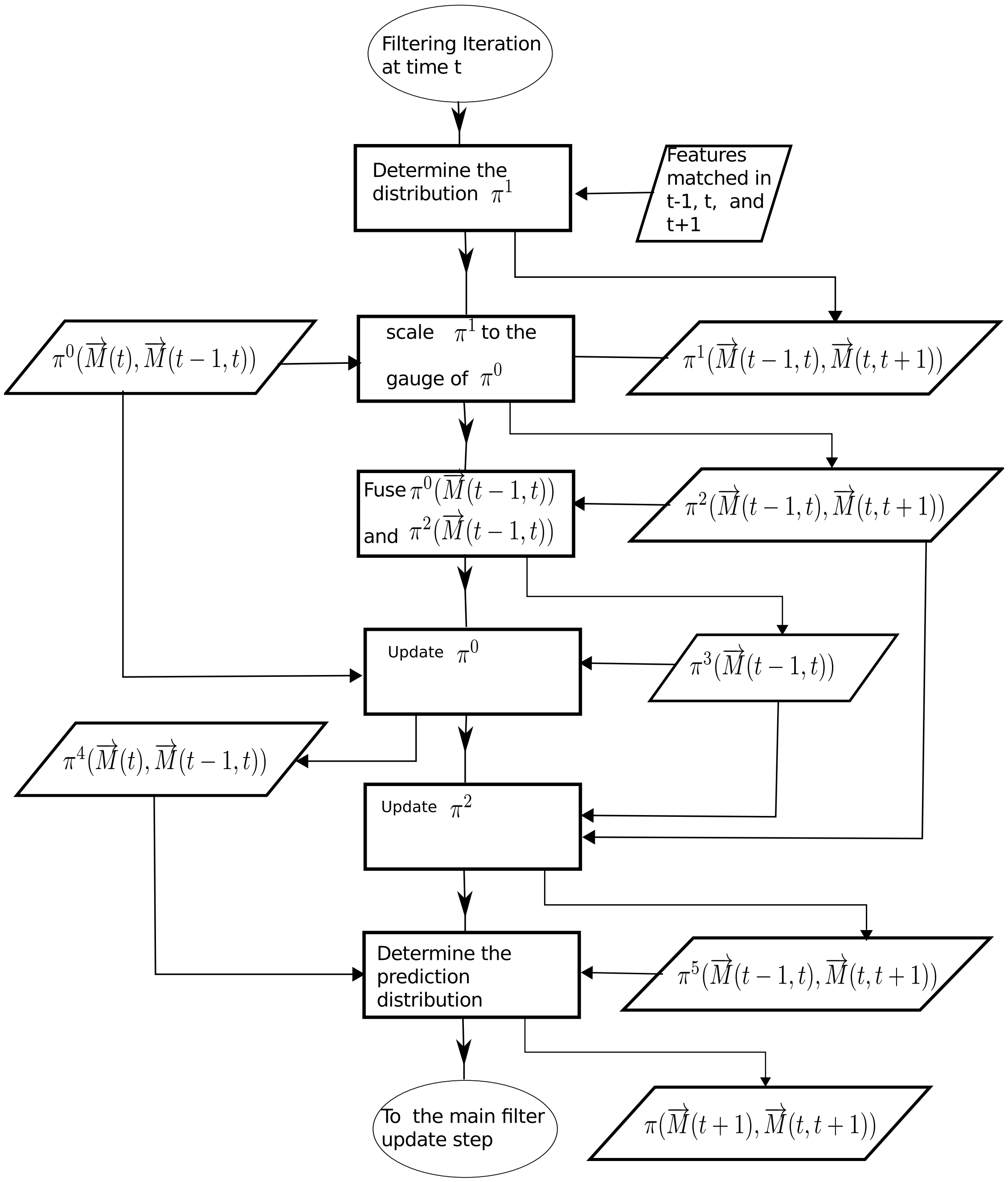

Filtering Iteration at time t
Determine the distribution π^1
Features matched in t-1, t, and t+1
scale π^1 to the gauge of π^0
$\pi^0(\overrightarrow{M}(t), \overrightarrow{M}(t-1,t))$
$\pi^1(\overrightarrow{M}(t-1,t), \overrightarrow{M}(t,t+1))$
Fuse $\pi^0(\overrightarrow{M}(t-1,t))$ and $\pi^2(\overrightarrow{M}(t-1,t))$
$\pi^2(\overrightarrow{M}(t-1,t), \overrightarrow{M}(t,t+1))$
Update π^0
$\pi^3(\overrightarrow{M}(t-1,t))$
$\pi^4(\overrightarrow{M}(t), \overrightarrow{M}(t-1,t))$
Update π^2
Determine the prediction distribution
$\pi^5(\overrightarrow{M}(t-1,t), \overrightarrow{M}(t,t+1))$
To the main filter update step
$\pi(\overrightarrow{M}(t+1), \overrightarrow{M}(t,t+1))$

Figure 6.3: Flowchart of the prediction filter showing the output and input at every stage.

RANSAC procedure results in an estimate for the vector $\overrightarrow{S}^1 = [\overrightarrow{M}(t-1,t);\overrightarrow{M}(t,t+1);\overrightarrow{Q}^1;...;\overrightarrow{Q}^K]$ where $\overrightarrow{Q}^i$ is the $3D$ points corresponding to the image feature $\overrightarrow{q}^i$. The estimate $\overrightarrow{S}^1$ is refined subsequently by minimizing the re-projection error in the three frames. The nonlinear minimization is a series of Levenberg-Marquardt iterations where the incremental update $\delta\overrightarrow{S}^1$ of $\overrightarrow{S}^1$ is given by:

$$\underline{H}^T_{\overrightarrow{S}^1}\underline{W}\underline{H}_{\overrightarrow{S}^1}\delta\overrightarrow{S}^1 = -\underline{H}^T_{\overrightarrow{S}^1}\underline{W}\delta\overrightarrow{Z}, \tag{6.4}$$

where $\overrightarrow{Z}$ is a vector containing all the image projections in the three frames, $\underline{H}_{\overrightarrow{S}^1}$ is the Jacobian of the projection equations of the K $3D$ points with respect to $\overrightarrow{S}^1$. $\delta\overrightarrow{Z}$ is the reprojected error (i.e., the distance between the re-projection of $\overrightarrow{S}^1$ and the measured image projections $\overrightarrow{Z}$). $\underline{W}$ is a block diagonal matrix containing the weight (inverse variance) of each image projection. The matrix $\underline{H}^T_{\overrightarrow{S}^1}\underline{W}\underline{H}_{\overrightarrow{S}^1}$ is an approximation of the Hessian matrix which, on convergence, from a Maximum Likelihood point of view, represents the inverse covariance of $\overrightarrow{S}^1$. Note that, because of SFM scale ambiguity, during the minimization the scale of $\overrightarrow{T}(t-1,t)$ is kept fixed to unity. The largest part of the Hessian matrix (the one corresponding to the $3D$ features) is block diagonal. We capitalize on this fact in the LM iterations and use the Schurr complement to invert the Hessian matrix in a computationally efficient way.

6.2.2 Re-scaling the Incremental Motion

As mentioned in the previous section, the scale of $\overrightarrow{\mu}^1_{\overrightarrow{T}(t-1,t)}$ is fixed to 1 during LM optimization, and is hence different than the magnitude of $\overrightarrow{\mu}^0_{\overrightarrow{T}(t-1,t)}$. Therefore, $\overrightarrow{\mu}^1_{[\overrightarrow{M}(t-1,t);\overrightarrow{M}(t,t+1)]}$ and the 12×12 covariance matrix $\underline{\Sigma}^1_{[\overrightarrow{M}(t-1,t);\overrightarrow{M}(t,t+1)]}$ need to be rescaled such that $||\overrightarrow{\mu}^1_{\overrightarrow{T}(t-1,t)}||$ becomes equal to $||\overrightarrow{\mu}^0_{\overrightarrow{T}(t-1,t)}||$. Let $\rho = ||\overrightarrow{\mu}^0_{\overrightarrow{T}(t-1,t)}||$, then the rescaled distribution $\pi^2([\overrightarrow{M}(t-1,t);\overrightarrow{M}(t,t+1)])$ is determined as follows:

$$\overrightarrow{\mu}^2_{[\overrightarrow{M}(t-1,t);\overrightarrow{M}(t,t+1)]} = \begin{bmatrix} \overrightarrow{\mu}^1_{\overrightarrow{\Omega}(t-1,t)} \\ \rho\overrightarrow{\mu}^1_{\overrightarrow{T}(t-1,t)} \\ \overrightarrow{\mu}^1_{\overrightarrow{\Omega}(t,t+1)} \\ \rho\overrightarrow{\mu}^1_{\overrightarrow{T}(t,t+1)} \end{bmatrix}, \tag{6.5}$$

and

$$\Sigma^2_{[\overrightarrow{M}(t-1,t);\overrightarrow{M}(t,t+1)]} =$$
$$\left(\begin{array}{cc|cc}
\Sigma^1_{\overrightarrow{\Omega}(t-1,t)} & \rho\Sigma^1_{\overrightarrow{\Omega}(t-1,t)\overrightarrow{T}(t-1,t)} & \Sigma^1_{\overrightarrow{\Omega}(t-1,t)\overrightarrow{\Omega}(t,t+1)} & \rho\Sigma^1_{\overrightarrow{\Omega}(t-1,t)\overrightarrow{T}(t,t+1)} \\
\rho\Sigma^1_{\overrightarrow{T}(t-1,t)\overrightarrow{\Omega}(t-1,t)} & \rho^2\Sigma^1_{\overrightarrow{T}(t-1,t)} & \rho\Sigma^1_{\overrightarrow{T}(t-1,t)\overrightarrow{\Omega}(t,t+1)} & \rho^2\Sigma^1_{\overrightarrow{T}(t-1,t)\overrightarrow{T}(t,t+1)} \\
\hline
\Sigma^1_{\overrightarrow{\Omega}(t,t+1)\overrightarrow{\Omega}(t-1,t)} & \rho\Sigma^1_{\overrightarrow{\Omega}(t,t+1)\overrightarrow{T}(t-1,t)} & \Sigma^1_{\overrightarrow{\Omega}(t,t+1)} & \rho\Sigma^1_{\overrightarrow{\Omega}(t,t+1)\overrightarrow{T}(t,t+1)} \\
\rho\Sigma^1_{\overrightarrow{T}(t,t+1)\overrightarrow{\Omega}(t-1,t)} & \rho^2\Sigma^1_{\overrightarrow{T}(t,t+1)\overrightarrow{T}(t-1,t)} & \rho\Sigma^1_{\overrightarrow{T}(t,t+1)\overrightarrow{\Omega}(t,t+1)} & \rho^2\Sigma^1_{\overrightarrow{T}(t,t+1)}
\end{array}\right). \tag{6.6}$$

6.2.3 Fusing the Common Incremental Motion

$\pi^0(\overrightarrow{M}(t-1,t))$ and $\pi^2(\overrightarrow{M}(t-1,t))$ now represent two independent Gaussian probability distributions of the same vector $\overrightarrow{M}(t-1,t)$. Therefore, they can be fused together to obtain a new Gaussian distribution $\pi^3(\overrightarrow{M}(t-1,t))$ whose mean vector and covariance matrix are:

$$\begin{gathered}
\overrightarrow{\mu}^3_{\overrightarrow{M}(t,t-1)} = \\
\Sigma^2_{\overrightarrow{M}(t,t-1)}(\Sigma^0_{\overrightarrow{M}(t,t-1)} + \Sigma^2_{\overrightarrow{M}(t,t-1)})^{-1}\overrightarrow{\mu}^0_{\overrightarrow{M}(t,t-1)} + \Sigma^0_{\overrightarrow{M}(t,t-1)}(\Sigma^0_{\overrightarrow{M}(t,t-1)} + \Sigma^2_{\overrightarrow{M}(t,t-1)})^{-1}\overrightarrow{\mu}^2_{\overrightarrow{M}(t,t-1)} \\
\Sigma_{\overrightarrow{M}(t,t-1)} = \\
\Sigma^2_{\overrightarrow{M}(t,t-1)}(\Sigma^0_{\overrightarrow{M}(t,t-1)} + \Sigma^2_{\overrightarrow{M}(t,t-1)})^{-1}\Sigma^0_{\overrightarrow{M}(t,t-1)} \\
= \Sigma^0_{\overrightarrow{M}(t,t-1)}(\Sigma^0_{\overrightarrow{M}(t,t-1)} + \Sigma^2_{\overrightarrow{M}(t,t-1)})^{-1}\Sigma^2_{\overrightarrow{M}(t,t-1)}
\end{gathered} \tag{6.7}$$

The above equations can be determined using the conditional distribution equations and the expressions of π^0 and π^2 as Gaussian distribution functions. They can be alternatively derived using Kalman filtering equations by casting one of the distributions as the prior distribution and the other as the measurement distribution. This fusion will result in a distribution with a mean vector in which the scale of $\overrightarrow{\mu}_{\overrightarrow{T}(t-1,t)}$ is different than ρ. Therefore, it should be rescaled to ρ and its covariance matrix should be changed accordingly. This is done as in Section 6.2.2.

6.2.4 Update the New Incremental Motion

Once $\pi^2(\overrightarrow{M}(t-1,t))$ is updated to $\pi^3(\overrightarrow{M}(t-1,t))$, this update is propagated to the other parts of $\pi^2([\overrightarrow{M}(t-1,t);\overrightarrow{M}(t,t+1)])$ to obtain a new updated Gaussian distribution

$\pi^3([\overrightarrow{M}(t-1,t);\overrightarrow{M}(t,t+1)])$ incorporating the information acquired from the state-vector distribution π^0. The mean and covariance of the new distribution are computed as follows, where $\underline{C}$ is an auxiliary matrix:

$$\begin{cases} \underline{C} = \underline{\Sigma}^2_{\overrightarrow{M}(t,t+1)\overrightarrow{M}(t-1,t)}(\underline{\Sigma}^2_{\overrightarrow{M}(t-1,t)})^{-1} \\ \overrightarrow{\mu}^3_{\overrightarrow{M}(t,t+1)} = \overrightarrow{\mu}^2_{\overrightarrow{M}(t,t+1)} + \underline{C}(\overrightarrow{\mu}^3_{\overrightarrow{M}(t-1,t)} - \overrightarrow{\mu}^2_{\overrightarrow{M}(t-1,t)}) \\ \underline{\Sigma}^3_{\overrightarrow{M}(t,t+1)\overrightarrow{M}(t-1,t)} = \underline{C}\underline{\Sigma}^3_{\overrightarrow{M}(t-1,t)} \\ \underline{\Sigma}^3_{\overrightarrow{M}(t,t+1)} = \underline{\Sigma}^2_{\overrightarrow{M}(t,t+1)} - \underline{C}(\underline{\Sigma}^2_{\overrightarrow{M}(t-1,t)} - \underline{\Sigma}^3_{\overrightarrow{M}(t-1,t)})\underline{C}^T \end{cases}. \tag{6.8}$$

The above equations can be determined using the Schurr complement and conditional probability equations (Section A.3). Also after this update a rescaling needs to be done such that the magnitude of $\overrightarrow{\mu}^3_{\overrightarrow{T}(t,t+1)}$ becomes equal to its value before the update.

6.2.5 Update the Old Absolute Motion

Similarly, the change from $\pi^0(\overrightarrow{M}(t-1))$ to $\pi^3(\overrightarrow{M}(t-1))$ is propagated to the other part of $\pi^0([\overrightarrow{M}(t);\overrightarrow{M}(t-1,t)])$ to include in it the information from the three frames features. The new updated distribution $\pi^3([\overrightarrow{M}(t);\overrightarrow{M}(t-1,t)])$ is determined by a set of equations similar to Equations (6.8):

$$\begin{cases} \underline{C} = \underline{\Sigma}^0_{\overrightarrow{M}(t)\overrightarrow{M}(t-1,t)}(\underline{\Sigma}^0_{\overrightarrow{M}(t-1,t)})^{-1} \\ \overrightarrow{\mu}^3_{\overrightarrow{M}(t)} = \overrightarrow{\mu}^0_{\overrightarrow{M}(t)} + \underline{C}(\overrightarrow{\mu}^3_{\overrightarrow{M}(t-1,t)} - \overrightarrow{\mu}^0_{\overrightarrow{M}(t-1,t)}) \\ \underline{\Sigma}^3_{\overrightarrow{M}(t)\overrightarrow{M}(t-1,t)} = \underline{C}\underline{\Sigma}^0_{\overrightarrow{M}(t-1,t)} \\ \underline{\Sigma}^3_{\overrightarrow{M}(t)} = \underline{\Sigma}^0_{\overrightarrow{M}(t)} - \underline{C}(\underline{\Sigma}^0_{\overrightarrow{M}(t-1,t)} - \underline{\Sigma}^3_{\overrightarrow{M}(t-1,t)})\underline{C}^T \end{cases}. \tag{6.9}$$

Note that the update in this step is also applied to the $3D$ features in the state vector. After the update a rescaling procedure should be done to keep the same gauge as before the update. This procedure is the same as the one performed in Chapter 4 (Section 4.3.2).

6.2.6 Infer the Prediction Distribution

The two distributions $\pi^3(\underline{M}(t,t+1))$ and $\pi^3(\underline{M}(t))$ are now consistent with each other, Hence, the predicted distribution at time $t+1$, $\pi([\underline{M}(t+1);\underline{M}(t,t+1)])$ can be determined using the motion evolution equations (Equation (6.1)). The mean is given by:

$$
\begin{aligned}
\overrightarrow{\mu}_{[\overrightarrow{M}(t+1);\overrightarrow{M}(t,t+1)]} &= g(\overrightarrow{\mu}^3_{\overrightarrow{M}(t)}, \overrightarrow{\mu}^3_{\overrightarrow{M}(t,t+1)}) \\
&= \begin{bmatrix} \overrightarrow{\mu}^3_{\overrightarrow{T}(t+1)} \\ \overrightarrow{\mu}^3_{\overrightarrow{\Omega}(t+1)} \\ \overrightarrow{\mu}^3_{\overrightarrow{T}(t,t+1)} \\ \overrightarrow{\mu}^3_{\overrightarrow{\Omega}(t,t+t)} \end{bmatrix} \\
&= \begin{bmatrix} \mathbb{R}^3toSO_3(\overrightarrow{\mu}^3_{\overrightarrow{\Omega}(t,t+1)})\overrightarrow{\mu}^3_{\overrightarrow{T}(t)} + \overrightarrow{\mu}^3_{\overrightarrow{T}(t,t+1)} \\ \mathbb{R}^3toSO_3(\overrightarrow{\mu}^3_{\overrightarrow{\Omega}(t,t+1)})\mathbb{R}^3toSO_3(\overrightarrow{\mu}^3_{\overrightarrow{\Omega}(t)}) \\ \overrightarrow{\mu}^3_{\overrightarrow{T}(t,t+1)} \\ \overrightarrow{\mu}^3_{\overrightarrow{\Omega}(t,t+1)} \end{bmatrix},
\end{aligned}
\tag{6.10}
$$

and the variance is determined using non-linear covariance propagation (First order Taylor expansion as described in Section A.4):

$$
\underline{\Sigma}_{[\overrightarrow{M}(t+1);\overrightarrow{M}(t,t+1)]} = \underline{G}_{\overrightarrow{M}(t)}\underline{\Sigma}_{\overrightarrow{M}(t)}\underline{G}^T_{\overrightarrow{M}(t)} + \underline{G}_{\overrightarrow{M}(t,t+1)}\underline{\Sigma}_{\overrightarrow{M}(t,t+1)}\underline{G}^T_{\overrightarrow{M}(t,t+1)}, \tag{6.11}
$$

where $\underline{G}_{\overrightarrow{M}(t)}$ and $\underline{G}_{\overrightarrow{M}(t,t+1)}$ are the Jacobian matrices of the function g with respect to $\overrightarrow{M}(t)$ and $\overrightarrow{M}(t+1)$ computed at $\overrightarrow{\mu}^3_{\overrightarrow{M}(t)}$ and $\overrightarrow{\mu}^3_{\overrightarrow{M}(t,t+1)}$ respectively.

$\overrightarrow{\mu}_{[\overrightarrow{M}(t+1);\overrightarrow{M}(t,t+1)]}$ and $\underline{\Sigma}_{[\overrightarrow{M}(t+1);\overrightarrow{M}(t,t+1)]}$ now represent a distribution of $[\overrightarrow{M}(t+1); \overrightarrow{M}(t, t+1)]$ that can be used as a prediction for any filtering technique.

6.3 Experimental results

To test the validity of the presented prediction approach, experimental tests are performed on the two sequences used in Chapter 3 (Figures 3.6 and 3.7). In a first set of experiments, we used the proposed prediction approach as the only estimation mechanism. No other filtering of the motion or the 3D features is used. The performance is tested at every frame using the re-projection error of all the previous features with image projections in the considered frame. The results on both sequences are shown by the blue solid lines in Figures 6.4 and 6.6. Those figures show that the proposed prediction by itself provides good motion estimates for a certain amount of time however, for an extended number of frames, errors accumulate and it starts diverging. It is important to mention that a part of this error is due to the fact that the $3D$ points are triangulated over three frames only and

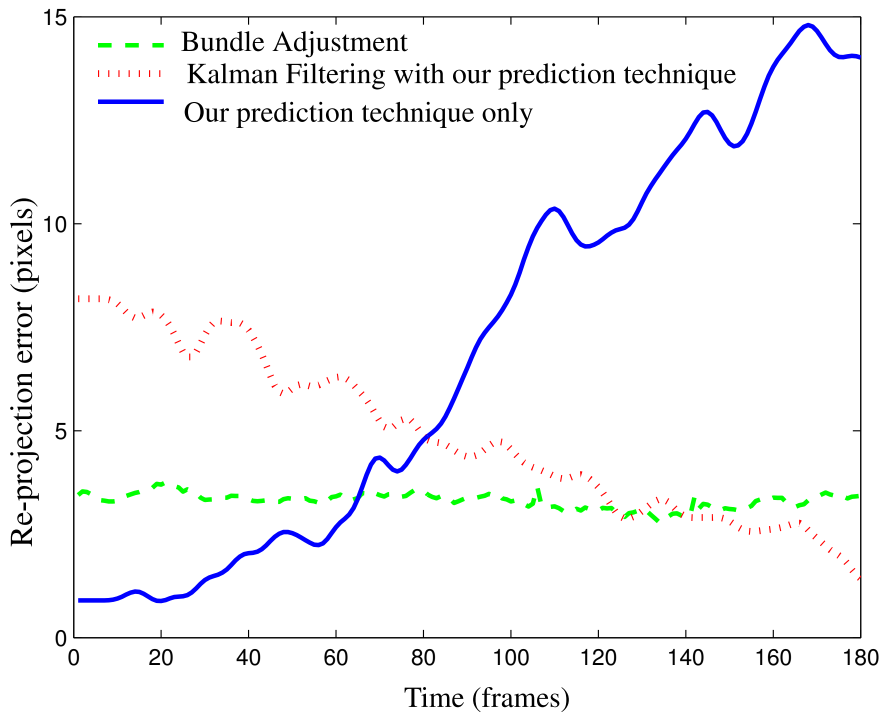


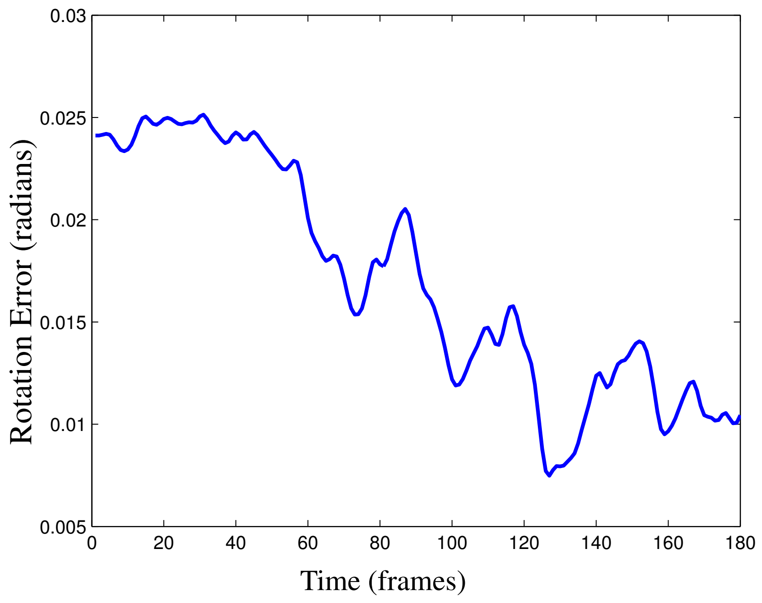
Figure 6.4: The re-projection errors achieved by the proposed technique on the first sequence. Notice that using the prediction only provides some level of filtering but it tends to diverge as the number of frames increases due to error accumulation. When Kalman filtering is performed using the presented prediction scheme, the errors are reduced significantly.

therefore are not very reliable.

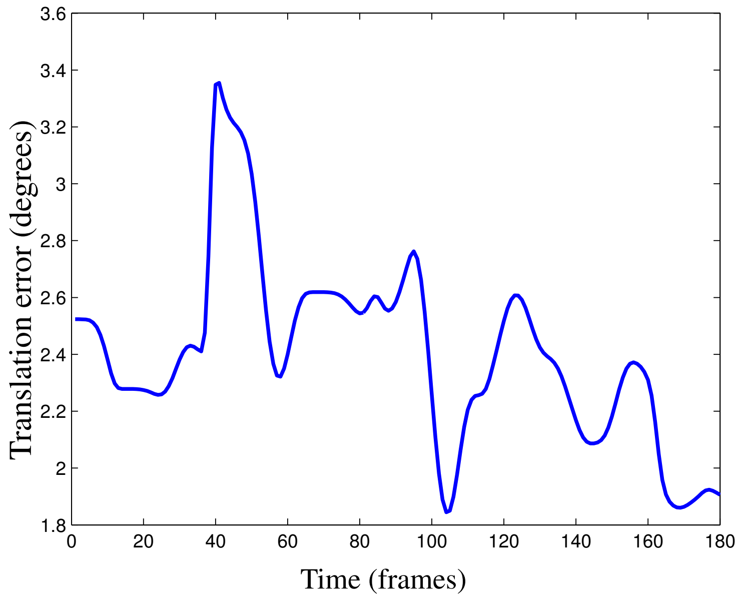
In a second set of experiments, the presented prediction scheme is used within an EKF. The results of this filtering are shown in Figures 6.4 and 6.6 where the red dotted lines show the re-projection errors of the EKF filtering using the proposed prediction scheme and the green lines show the re-projection errors of BA. Also, Figures 6.5 and 6.7 show the distance between the estimates of this EKF filtering and the and BA estimates. With the proposed approach as its prediction mechanism, the EKF performed nicely and provided good results compared to the optimal BA. Without this prediction mechanism, the EKF fails to produce acceptable results on those sequences.

6.4 Conclusion

This chapter presented a solution to the problem of motion prediction over large baseline in the case of monocular images. This solution is based on using the features that are matched only in the last three frames, and are not consequently the projections of



(a) Rotational error



(b) Translational Error

Figure 6.5: Errors of the EKF using the presented prediction approach with respect to BA for the first sequence. The estimates are close to BA estimates which shows that this approach succeeds in providing the EKF with good prediction distributions at every time step.

any feature in the state vector. Therefore, their use in the prediction does not jeopardize the statistical independence of the predicted distribution and the measurements. We demonstrated the performance of this technique using tests on real images.

An important aspect of the presented solution is that, not only it allows a reliable

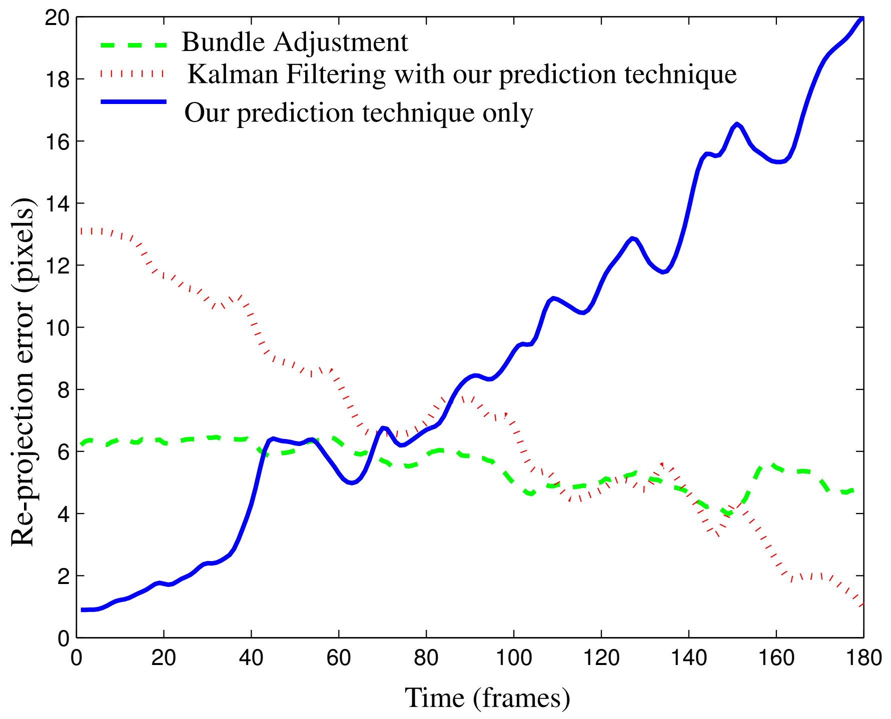
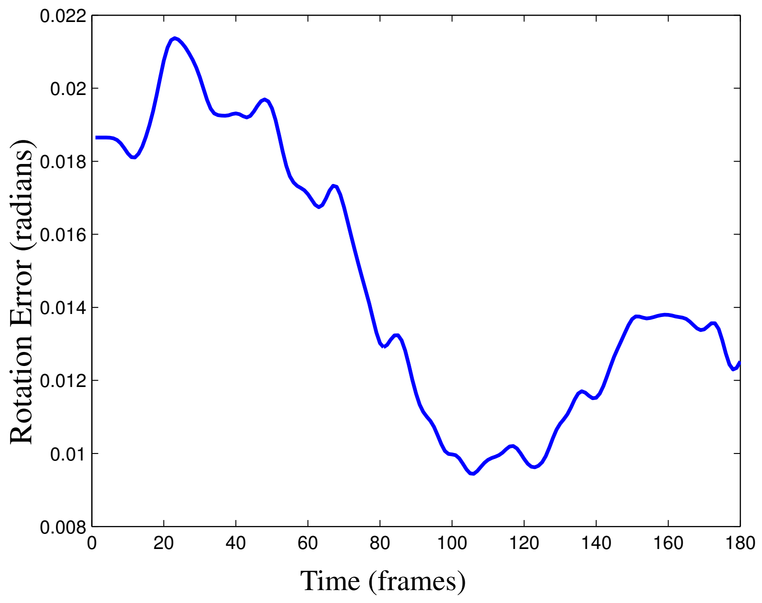


Figure 6.6: The re-projection errors achieved by the presented prediction approach on the second sequence. The same observation as in Figure 6.4 can be made.

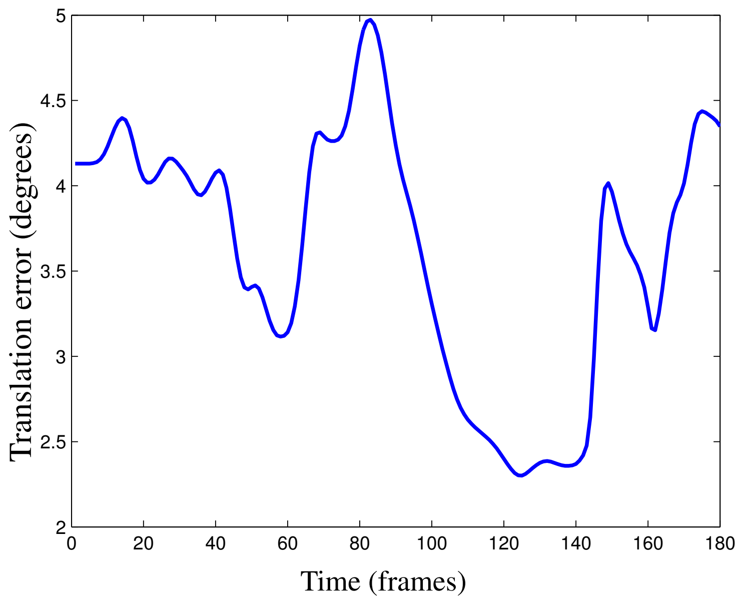
prediction, but it also fuses the information from the features that are matched in every three frames only in the filter which results in added accuracy. This adds to the filter some of the benefits of odometry-based approaches.

Another nice property of the presented solution, is that, in the case of filters based on multiple particles or hypotheses, the computation of the motion across the three frames, which is the most computationally demanding stage, needs to be done only once for all the particles.

The determination of the motion across the most recent frames does not have to be done necessarily as described in Section 6.2.1. Actually, any approach providing a Gaussian distribution of the motion over those three frames would work. Of interest here, is approaches based on optical flow such as the method of Zhang and Tomasi [135]. This is useful not in cases of large baselines, but cases of motion blur when the camera experiences sudden motions. Motion blur has a very detrimental effect on the matching of features, however, its effect on optical flow is not as bad. Actually, some approaches such as the optical flow estimation of Rekleitis [100] use motion blur to estimate the optical flow. Therefore, in cases of motion blur such methods can be used to determine the motion distribution across the last three frames with can be used then as an input to



(a) Rotational error



(b) Translational Error

Figure 6.7: Errors of the EKF using our prediction approach with respect to bundle adjustment for the second sequence. The same observation as in Figure 6.5 can be made.

our approach.

Another application of the presented solution outside providing prediction for filtering approaches, is for example with the PTAM approach [68] described in Section 2.5.4. In this approach, when the number of estimated 3D features with projection in the new image is low, tracking a new motion using an absolute orientation algorithm is not reliable. Therefore, the proposed approach can be used over the last three frames to provide a more reliable tracking.

Chapter 7

Conclusion

This chapter revisits the contributions of the thesis, discusses their impact on the state of the art and identifies potential future research and ways of improvements. The chapter starts with a brief summary of the contributions, then a discussion of the relations between them, their differences and how they complement each other. After that, the impact of the contributions on the general framework presented in the background (Section 2.5.5) is described. A system that capitalizes on the contributions and uses vision in conjunction with inertial and GPS measurements is subsequently proposed. Finally some miscellaneous comments are provided.

7.1 Contributions Summary

The contributions of this thesis are as follows:

- A "threading filter" (Chapter 3) intended principally for motion only estimation is presented. This filter achieves an accuracy that is more than three times higher than the EKF and hence it matches the accuracy of the state of the art GLF filter. However, this threading filter, unlike Bayesian filters, possesses the very important property that it can work regardless of the size of the baseline between consecutive camera centers. Furthermore, it is computationally efficient and resilient to outliers since it is based on random sample consensus.
- A "frame-to-frame filter" (Chapter 4) is introduced to augment any analytic filter with frame-to-frame information. This approach is very computationally efficient

and is able to process hundreds of frame-to-frame features in about $10ms$. Experimental results have shown that this approach enhances the EKF, and therefore similar enhancements (probably to different extents) are expected with other filters.

- To deal with the issue of scalability, a “multi-hypothesis filter” (Chapter 5) scalable to accommodate large numbers of features is presented. This filter possesses the same scalability properties of the RBPF (which is the most scalable filter so far), but with an accuracy similar to the GLF (about three times higher than the EKF).
- A solution for the problem of motion prediction across large monocular baselines in Bayesian filters is provided (Chapter 6). This solution provides reliable prediction distribution without using the projections of the features that are already in the state vector.

7.2 Relations between the Contributions

Both the threading filter (Chapter 3) and the multi-hypothesis filter (Chapter 5) can be augmented with the frame-to-frame filter to increase their accuracy.

- For the threading filter, the frame-to-frame filter requires that the main estimator into which the frame-to-frame information is to be included, to output a Gaussian distribution of the estimates. In the conclusion of Chapter 3 we discussed how to modify the threading filter so that it provides also a covariance matrix of the estimates. Therefore, it can be easily augmented with the frame-to-frame filter.
- For the multi-hypothesis filter, augmenting it with the frame-to-frame filter is pretty straightforward since each hypothesis is based on a small dimensional EKF. It is important to note here that the frame-to-frame filtering of every hypothesis should be done before updating the second parts as in Section 5.3.3 so that the update of the features in individual EKF uses the motion that’s already updated with the frame-to-frame information.

The approach provided in Chapter 6 is intended to extend the range of operation of Bayesian filters that follow the predict-update paradigm. Therefore, it is ideal to work with the multi-hypothesis filter of Chapter 5. The threading filter does not require such addition since it naturally works with large baselines. On another side, in the presented large baseline predictor, instead of the method presented in Section 6.2.1, the developed threading constraint in Section 3.3.2 followed by a non-linear optimization as in Section 3.3.5 can be used in the determination of the motion across the three-frames. This would result in a considerable speed-up since the determination of the motion across the three-frames is the main computational burden in the large baseline predictor.

The last point to address in this section is the relation between the threading filter (Chapter 3) and the multi-hypothesis filter (Chapter 5). The threading filter achieves the filtering at every time step in a consensus or voting way. The multi-hypothesis filter achieves filtering based on multi-hypothesis Bayesian EKF filters. However, as discussed in the conclusion of Chapter 5, the core filter (estimating the first part) in every hypothesis, does not have to be an EKF. Therefore, we can capitalize on the strengths of both approaches, by replacing the EKFs in some of the hypotheses in the multi-hypothesis filter with threading filters. The resulting filter would be as follows:

- Hypothesis structure: a hypothesis divides the state vector into two parts: $\overrightarrow{S}^1$ containing the motion and a set of random N_p features (N_p is set to 9 in the thesis) and $\overrightarrow{S}^2$ containing the remaining 3D feature. The hypothesis consists then of a Gaussian distribution ($\overrightarrow{\mu}_{\overrightarrow{S}^1(t)}$, $\underline{\Sigma}_{\overrightarrow{S}^1(t)}$) of $\overrightarrow{S}^1$ and a set of Gaussian distributions ($\overrightarrow{\mu}_{\overrightarrow{P}^i}$, $\underline{\Sigma}_{\overrightarrow{P}^i}$), one for every $3D$ feature in $\overrightarrow{S}^2$.
- Filtering: The filtering consists of three steps.
  1. The first step is to filter the first part ($\overrightarrow{S}^1$) of every hypothesis which we refer to as "core" filtering. This core filtering is low dimensional and can be done either using an EKF or the threading filter. The threading filter in this case needs to be modified as described in Section 3.8 to provide the covariance of the estimates in addition to their mean values. Then, at every time step, a portion of the hypotheses will have their core filtering carried out using the threading filter while the rest is performed using the EKF. To determine the portion of hypotheses processed with each filter, experimental tests need to be performed to identify the relative performance of the two filters. Note that in Chapter 3 the presented results showed that the threading filter is clearly much superior to the EKF. However, in the case of very low dimensional filters this might not be case as the threading filter owes its performance to its ability to select good data samples (spatially and temporally). However, with low dimensional state-vectors, the pool of spatial data is too small which might reduce the performance of the threading filter versus the EKF.

2. The second step is to filter the second part of every hypothesis based on the distribution of the first part and this is done as in Section 5.3.3.
3. The last step is to determine the new weight of every hypothesis as described in Section 5.3.4.

Furthermore, the frame-to-frame filter can be used with each one of the hypotheses (whether it is core filtering is done using EKF or threading. For the hypotheses whose core filter is based on EKF, the large-baseline predictor can be used whenever the baseline is large. The resulting filter would harness the strengths of the four contributions which results in a scalable robust filtering with additional accuracy.

7.2.1 Integrating with Parallel Tracking and Mapping

In the review provided in Chapter 2 it was argued, based on the most recent results published in the literature, that the best sequential SFM system would be, as shown in Figure 7.1, a system separating the tracking from the mapping into a front-end and a back-end processes running separately in parallel. The back end is a bundle adjuster processing key frames, while the front end is an SFM filter that tracks beyond the last key frame. Whenever a key-frame is selected (based on criteria such as the distance from the last key frame), this frame is added to the back-end bundle adjuster. The back-end does not have to run in real-time, however, it has to keep up with the tracker in the sense that the back-end should be able to finish its optimization by the time the tracker selects a new key-frame. This setup shows that the local filtering is very crucial for the overall performance of the estimator. The more accurate the filtering, the less the chance of the back-end optimizer to fall in local minima, the faster its convergence and hence the higher the number of key-frames that could be considered. The contributions of the thesis open up the opportunity for many improvement in the front-end:

- First of all, the contributions yield filtering results that increase the accuracy and speed of filtering. The approach combining the contributions as discussed in the previous section is an ideal choice for the local filtering front-end.
- The threading filter can be used as a fast and accurate front end tracker of the motion only. Then, this motion is used in the back-end to perform a guided matching for the features already in the state vector of the back end, and for new features between the key-frames. The guided matching (i.e., looking for every feature in an

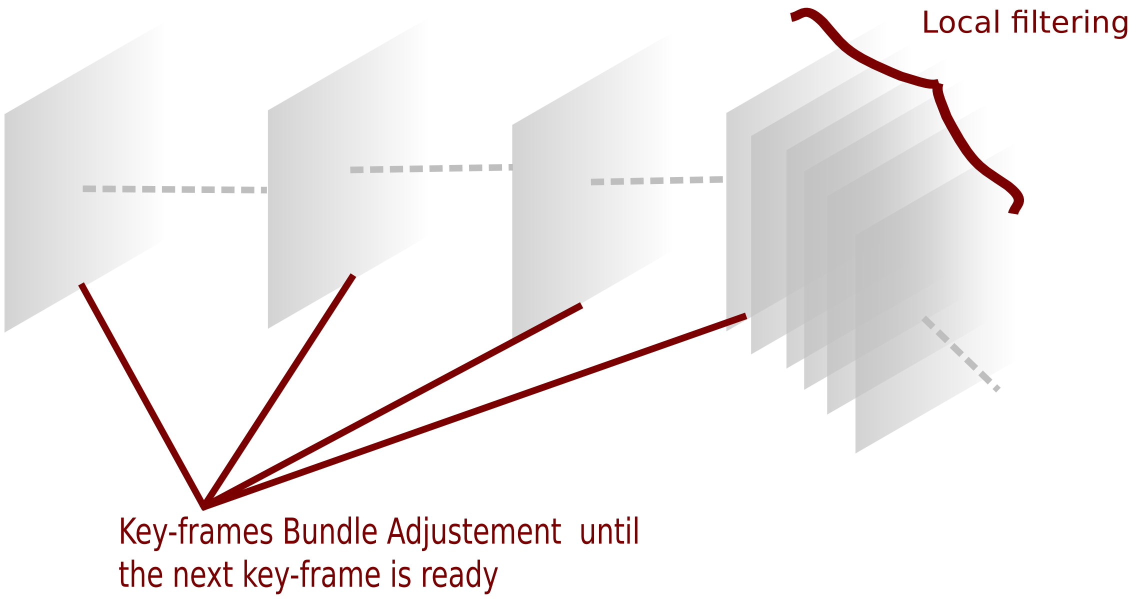

Local filtering
Key-frames Bundle Adjustement until the next key-frame is ready

Figure 7.1: Best sequential SFM setup based on state of the art results: A separate thread performs bundle adjustment on key-frames, while a local filtering thread selects the key frames and computes the relative motion to the last key-frame. The local filtering performance affects the performance of the whole system.

elliptical region determined from its estimate in the state vector and its uncertinty) is very important when the features are tracked for large distances because it can be much more accurate than regular matching. Another case where the threading filter is important for the front-end is when the motion only is estimated by the whole system. In this case, the back-end would be an optimizer enforcing motion consistency across loops such as the one in FrameSLAM [70].

- Even if tracking based an on absolute orientation algorithm (Section 2.4.5) is chosen instead of filtering for the front-end as in the approach of Klein and Murray [68], the frame-to-frame feature filter can still be used to provide additional accuracy for the tracking estimates, as it can augment any estimator providing Gaussian distribution of the estimates.

- Finally, in the cases of motion blur, the same solution proposed to extend filter based SFM to large baselines, coupled with optical flow estimation techniques across the last three frames, can be very useful in providing reliable tracking estimates as discussed in the conclusion of Chapter 6.

7.3 Integration With GPS and Inertial Systems

This section focuses on the integration of the presented contributions with inertial and GPS sensors for long range navigation and mapping. Inertial measurements have been used in conjunction with vision [24,62,71,99,104] to achieve a performance better than the performance of either one of them. The combination of vision and inertial sensors is often used to extend the operation of GPS in regions of signal shortages [8,41,56,132]. With GPS, inertial and magnetometer readings, the following equations are added to the dynamic system of structure of motion (Equation (2.34)):

$$\begin{cases} \overrightarrow{T}_{GPS}(t) &= \underline{C}_{GPS}\overrightarrow{T}(t) + \overrightarrow{B}_{GPS} + \overrightarrow{n}_{GPS} \\ \overrightarrow{a}_{ins}(t) &= R^T_{ins}[\overrightarrow{a}(t) + \underline{R}^T(t)\overrightarrow{g} + \ddot{\underline{R}}(t)\overrightarrow{T}_{ins}] + \overrightarrow{a}_{bias} + \overrightarrow{n}_{ins,acc} \\ \overrightarrow{\omega_{ins}}(t) &= R^T_{ins}\overrightarrow{\omega}(t) + \overrightarrow{\omega}_{bias} + \overrightarrow{n}_{ins,gyro} \\ \overrightarrow{\Omega}_{mag}(t) &= \overrightarrow{\Omega}(t) + \overrightarrow{n}_{mag} \end{cases} \tag{7.1}$$

$\underline{C}_{GPS}$ is a matrix representing the rotation between the navigation frame and the ECEF (Earth Centered Earth Fixed) frame, and $\overrightarrow{B}_{GPS}$ is a vector from the earth center to the origin of the navigation frame. R_{ins} is the rotation between the navigation frame and the Inertial Measurement Unit (IMU) frame and $\overrightarrow{T}_{ins}$ is the vector from the IMU frame to the navigation frame. R_{ins} and $\overrightarrow{T}_{ins}$ can be determined using $\overrightarrow{M}(t)$ and the trnas-formation between the IMU frame and the camera frame. $\overrightarrow{n}_{GPS}$ $(0, \Sigma_{\overrightarrow{n}_{GPS}})$ is the noise from uncertainty in the GPS measurement, and can be obtained from the Dilution of Precision (DOP) values provided by the GPS unit. $\overrightarrow{n}_{ins,acc}$ $(0, \Sigma_{\overrightarrow{n}_{ins,acc}})$ and $\overrightarrow{n}_{ins,gyro}$ $(0, \Sigma_{\overrightarrow{n}_{ins,gyro}})$ are the noise from uncertainties in the accelerometers and gyroscopes of the inertial sensor. $\overrightarrow{a}_{bias}$ and $\overrightarrow{\omega}_{bias}$ are the biases for the accelerometers and gyroscopes given in the hardware specifications. $\overrightarrow{n}_{mag}$ $(0, \Sigma_{mag})$ is the noise from uncertainty in the magnetometer measurements.

Equations (7.1) show that taking care of these measurements in the multi-hypothesis filter is straightforward as the core filtering is done using EKFs. The GPS and Inertial measurements result in additional measurements equations in the core filtering. Therefore, we can easily integrate GPS and inertial measurements in the approach combining the contributions of the thesis discussed earlier. An important issue to consider is that, although the combination of different sensors measurements in general leads to better results than using every measurement by itself, in some situations if one of the sensors measurements is really bad or if the uncertainties in the measurements are not well known, the combination might not be fruitful. Therefore, as a measure against that, we diversify the core filters in the hypotheses in the system, by having hypotheses combining both vision and inertial sensors and other hypotheses based on each sensor by itself. Therefore, the system would have three different types of hypotheses:

- Hypotheses in which the core filter uses purely vision measurements (N_p features) without any inertial measurements. This can be done using either EKF or threading.
- Hypotheses in which the core filter is EKF integrating both vision and inertial sensors.
- One hypothesis corresponding to inertial measurements (plus magnetometer) without vision.

At every time step the estimation of all the hypotheses can be done in a GPS mode or in a non-GPS mode depending on the availability of the GPS signal. When GPS measurement are available, they are used to fix the similarity ambiguity. When the GPS signal disappears, the system resorts back to vision for that purpose and fix some structure parameters as explained in Section 4.3 to deal with this ambiguity. At every time step, an error similar to the one derived in Section 5.3.4 and extended to consider the discrepancy with the inertial and GPS measurements is used to test the hypotheses and select the best one.

We anticipate, based on our vision only results, that such system would achieve better results than systems fusing vision and inertial measurements directly in a Kalman Filtering Style [62].

7.4 Miscellaneous Notes

The issue of occlusions has not been addressed explicitly in the thesis. However, the presented approaches have some capabilities to deal with occlusions. If the feature matcher is able to identify when features become occluded, they can be just discarded as we don't keep descriptors of the features as done in SLAM. Occlusions that are not detected by the feature matcher, result in outliers. The threading filter, being based on random sample consensus can easily avoid outliers. Also, the multi-hypothesis filter is made robust to outliers through using the robust function ρ in Section 5.3.4. The only case when outliers would seriously affect the performance of this filter is when most or all the hypotheses contain outliers in their first part.

We have not touched upon the problem of dense structure estimation, nor the exploration of spatial constraints between features (such as between nearby features or features belonging to same planes).

Also, only static scenes with no independently moving objects were considered. The case where there are multiple different motions in the scene with respect to the camera has not been addressed. It would be interesting to investigate how the contributions presented could be extended to deal with this case.

Finally, in this thesis, we considered the SFM estimation problem totally separated from the feature detection and estimation problem. This is a controversial issue as, while considering the two problems as totally independent results in algorithms that can work with any feature detection and matching technique, casting the two problems as one problem has obvious benefits since the $3D$ parameters generated by the SFM estimator are very useful for feature matching.

Appendix A

Statistical Inference

This Appendix presents a brief overview of the main inference mechanisms used in the thesis.

A.1 Combining the Estimates of Two Gaussian Vectors

Let $\hat{\overrightarrow{s}}^1$, $\underline{\Sigma}_{\overrightarrow{S}^1}$ and $\hat{\overrightarrow{s}}^2$, $\underline{\Sigma}_{\overrightarrow{S}^2}$ be two estimates of a random vector $\overrightarrow{s}$ obtained from two independent measurements $\overrightarrow{z}^1$ and $\overrightarrow{z}^2$. Then the estimate of $\overrightarrow{s}$ given the two measurements can be determined as follows:

$$\hat{\overrightarrow{s}} = \underline{\Sigma}_{\overrightarrow{S}^2}(\underline{\Sigma}_{\overrightarrow{S}^1} + \underline{\Sigma}_{\overrightarrow{S}^2})^{-1}\hat{\overrightarrow{s}}^1 + \underline{\Sigma}_{\overrightarrow{S}^1}(\underline{\Sigma}_{\overrightarrow{S}^1} + \underline{\Sigma}_{\overrightarrow{S}^2})^{-1}\hat{\overrightarrow{S}}^2 \tag{A.1}$$

$$\underline{\Sigma}_{\hat{\overrightarrow{S}}} = \underline{\Sigma}_{\overrightarrow{S}^2}(\underline{\Sigma}_{\overrightarrow{S}^1} + \underline{\Sigma}_{\overrightarrow{S}^2})^{-1}\underline{\Sigma}_{\overrightarrow{S}^2} = \underline{\Sigma}_{\overrightarrow{S}^1}(\underline{\Sigma}_{\overrightarrow{S}^1} + \underline{\Sigma}_{\overrightarrow{S}^2})^{-1}\underline{\Sigma}_{\overrightarrow{S}^2} \tag{A.2}$$

The above equations can be derived from the fact that since $\overrightarrow{z}^2$ and $\overrightarrow{z}^2$ are independent then:

$$\pi(\overrightarrow{s}|\overrightarrow{z}^1, \overrightarrow{z}^2) = \frac{\pi(\overrightarrow{s}|\overrightarrow{z}^1)\pi(\overrightarrow{s}|\overrightarrow{z}^2)}{\pi(\overrightarrow{s})} \propto \pi(\overrightarrow{s}|\overrightarrow{z}^1)\pi(\overrightarrow{s}|\overrightarrow{z}^2), \tag{A.3}$$

then using the Gaussian distribution expressions of $\pi(\overrightarrow{s}|\overrightarrow{z}^1)$ and $\pi(\overrightarrow{s}|\overrightarrow{z}^2)$ we obtain the expression of $\pi(\overrightarrow{s}|\overrightarrow{z}^1, \overrightarrow{z}^2)$ as a Gaussian distribution with mean and covariance as in equations A.1. Note that the same result could be obtained by casting the problem as the special case of a Kalman Filter with $\hat{\overrightarrow{s}}^1$, $\underline{\Sigma}_{\overrightarrow{S}^1}$ representing the predicted vector and $\hat{\overrightarrow{s}}^1$, $\underline{\Sigma}_{\overrightarrow{S}^2}$ the measurement.

A.2 Bayesian Filtering

Given the following dynamical system:

$$\left\{ \begin{array}{lcll} \overrightarrow{S}(t+1) & = & f(\overrightarrow{S}(t)) + \overrightarrow{n}_f(t) & \overrightarrow{n}_f(t) \sim \mathcal{N}(0, \underline{\Sigma}_f) \\ \overrightarrow{Z}(t) & = & h(\overrightarrow{S}(t)) + \overrightarrow{n}_{\overrightarrow{Z}}(t) & \overrightarrow{n}_{\overrightarrow{Z}}(t) \sim \mathcal{N}(0, \underline{\Sigma}_{\overrightarrow{Z}}) \end{array} \right. \tag{A.4}$$

Bayesian filtering relies on the famous Bayes rule and on Markov assumption to determine the probability distribution $\pi(\overrightarrow{S}(t)|\overrightarrow{Z}(0:t))$:

$$\begin{aligned} \pi(\overrightarrow{S}(t)|\overrightarrow{Z}(0:t)) &= \frac{\pi(\overrightarrow{S}(t)|\overrightarrow{Z}(0:t-1)) \times \pi(\overrightarrow{Z}(t)|\overrightarrow{S}(t))}{\int \pi(\overrightarrow{S}(2)|\overrightarrow{Z}(0:t-1)) \times \pi(\overrightarrow{Z}(t)|\overrightarrow{S}(t)) d\overrightarrow{S}} \\ &= \frac{\pi(\overrightarrow{S}(t)|\overrightarrow{Z}(0:t-1)) \times \pi(\overrightarrow{Z}(t)|\overrightarrow{S}(t))}{\pi(\overrightarrow{Z}(t)|\overrightarrow{Z}(0:t))} \end{aligned} \tag{A.5}$$

The denominator in the above equation is a normalized constant. Calling this constant k we can write:

$$\pi(\overrightarrow{S}(t)|\overrightarrow{Z}(0:t)) = k\pi(\overrightarrow{S}(t)|\overrightarrow{Z}(0:t-1))\pi(\overrightarrow{Z}(t)|\overrightarrow{S}(t)). \tag{A.6}$$

Via the Chapman-Kolmogorov equation $\pi(\overrightarrow{S}(t)|\overrightarrow{Z}(0:t-1)$ can be written as:

$$\pi(\overrightarrow{S}(t)|\overrightarrow{Z}(0:t-1) = \int_{\overrightarrow{S}(t-1)} \pi(\overrightarrow{S}(t)|\overrightarrow{S}(t-1))\pi(\overrightarrow{S}(t-1)|\overrightarrow{Z}(0:t-1), \tag{A.7}$$

then

$$\pi(\overrightarrow{S}(t)|\overrightarrow{Z}(0:t)) = k\pi(\overrightarrow{Z}(t)|\overrightarrow{S}(t)) \int_{\overrightarrow{S}(t-1)} \pi(\overrightarrow{S}(t)|\overrightarrow{S}(t-1))\pi(\overrightarrow{S}(t-1)|\overrightarrow{Z}(0:t-1). \tag{A.8}$$

Equation (A.8) constitutes the basis of all bayesian recursive filters. We will describe hereafter the standard KF, EKF and Particle Filters.

A.2.1 Kalman Filtering

The Kalman Filter assumes that the posterior density at every time step is Gaussian and, hence parametrized by a mean and a covariance. It also assumes that f and h are linear function and can be re-written as:

$$f(\overrightarrow{S}(t)) = \underline{F}(t)\overrightarrow{S}(t) h(\overrightarrow{Z}(t)) = \underline{H}(t)\overrightarrow{Z}(t), \tag{A.9}$$

where the $\underline{H}$ and $\underline{F}$ are known matrices representing the actions of h and f respectively. The Kalman filter is derived using Equations (A.5) and (A.7) and relying on the fact that the product of two Gaussian distribution is a Gaussian distribution. $\overrightarrow{S}(t)$ and its covariance are then determined in two steps: Prediction:

$$\begin{array}{c} \overrightarrow{S}(t|t-1) = \underline{F}(t)\overrightarrow{S}(t-1) \\ \underline{\Sigma}_{\overrightarrow{S}}(t|t-1) = \underline{F}(t)\underline{\Sigma}_{\overrightarrow{S}}(t-1)\underline{F}(t) + \underline{\Sigma}_f \end{array}. \tag{A.10}$$

Update:

$$\begin{array}{c} \overrightarrow{S}(t) = \overrightarrow{S}(t|t-1) + \underline{L}(t)(\overrightarrow{Z}(t) - \underline{H}(t)\overrightarrow{S}(t|t-1)) \\ \underline{\Sigma}_{\overrightarrow{S}}(t) = (\underline{I} - \underline{L}(T)\underline{H}(t))\underline{\Sigma}_{\overrightarrow{S}}(t|t-1) \end{array} \tag{A.11}$$

where

$$\begin{array}{r} \underline{\Lambda} = \underline{H}(t)\underline{\Sigma}_{\overrightarrow{S}}(t|t-1)\underline{H}^T(t) + \underline{\Sigma}_{\overrightarrow{Z}}(t) \\ \underline{L}(t) = \underline{\Sigma}_{\overrightarrow{S}}(t|t-1)\underline{H}^T(t)\underline{\Lambda}^{-1}(t) \end{array}$$

Extended Kalman Filtering

In the case, where f and h are non-linear, a first order Taylor expansion is used about the predicted estimates and the Jacobians $\underline{F}_{\overrightarrow{S}}$ and $\underline{H}_{\overrightarrow{S}}$ of f and h respectively are used. In this case the prediction and update steps are as follows: Prediction:

$$\begin{array}{c} \overrightarrow{S}(t|t-1) = f(\overrightarrow{S}(t-1)) \\ \underline{\Sigma}_{\overrightarrow{S}}(t|t-1) = \underline{F}_{\overrightarrow{S}}(t)\underline{\Sigma}_{\overrightarrow{S}}(t-1)\underline{F}_{\overrightarrow{S}}(t) + \underline{\Sigma}_f \end{array}. \tag{A.12}$$

Update:

$$\begin{array}{c} \overrightarrow{S}(t) = \overrightarrow{S}(t|t-1) + \underline{L}(t)(\overrightarrow{Z}(t) - h(\overrightarrow{S}(t|t-1))) \\ \underline{\Sigma}_{\overrightarrow{S}}(t) = (\underline{I} - \underline{L}(T)\underline{H}_{\overrightarrow{S}}(t))\underline{\Sigma}_{\overrightarrow{S}}(t|t-1) \end{array} \tag{A.13}$$

where

$$\begin{aligned}\underline{\Lambda} &= \underline{H}_{\overrightarrow{S}}(t)\underline{\Sigma}_{\overrightarrow{S}}(t|t-1)\underline{H}^T_{\overrightarrow{S}}(t) + \underline{\Sigma}_{\overrightarrow{Z}}(t)\\ \underline{L}(t) &= \underline{\Sigma}_{\overrightarrow{S}}(t|t-1)\underline{H}^T_{\overrightarrow{z}}(t)\underline{\Lambda}^{-1}(t)\end{aligned}$$

Implicit Extended Kalman Filtering

In the case of an implicit measurement equation of the form $h(\overrightarrow{S}, \overrightarrow{Z})$ instead of the regular form $\overrightarrow{Z} = h(\overrightarrow{S})$, the solution is to use another Taylor expansion of h but this time about $\overrightarrow{Z}$ and define the new measurement noise covariance $\underline{\Sigma}_{h(\overrightarrow{Z})}$ of the linearized equation as:

$$\underline{\Sigma}_{h(\overrightarrow{Z})} = \underline{H}_{\overrightarrow{Z}}\underline{\Sigma}_{\overrightarrow{Z}}\underline{H}^T_{\overrightarrow{Z}}. \tag{A.14}$$

Then, the new equations of the gain $\underline{L}$ can be shown to be as follows:

$$\begin{aligned}\underline{\Lambda} &= \underline{H}_{\overrightarrow{S}}(t)\underline{\Sigma}_{\overrightarrow{S}}(t|t-1)\underline{H}^T_{\overrightarrow{S}}(t) + \underline{\Sigma}_{h(\overrightarrow{Z})}(t)\\ \underline{L}(t) &= -\underline{\Sigma}_{\overrightarrow{S}}(t|t-1)\underline{H}^T_{\overrightarrow{z}}(t)\underline{\Lambda}^{-1}(t)\end{aligned}$$

A.2.2 Particle Filtering

In particle filtering we recursively approximate the posterior $\pi(\overrightarrow{S}(t)|\overrightarrow{Z}(t))$ as a set of N weighted samples $\{\overrightarrow{S}^{(i)}(t), W^{(i)}(t)\}_{i=1}^N$, where $W(t)^{(i)}$ is the weight for the particle $\overrightarrow{S}^{(i)}$. Given this approximate representation, we obtain a Monte Carlo approximation of the Bayes filtering distribution:

$$\pi(\overrightarrow{S}(t)|\overrightarrow{Z}(0:t)) = k\pi(\overrightarrow{Z}(t)|\overrightarrow{S}(t))\sum_i W^{(i)}\pi(\overrightarrow{S}(t)|\overrightarrow{S}^{(i)}(t-1)). \tag{A.15}$$

let $\{\overrightarrow{S}^{(i)}(t), W^{(i)}(t)\}_{i=1}^N$ denote a *random measure* that characterizes the posterior probability distribution $\pi(\overrightarrow{S}(t)|\overrightarrow{Z}(t))$ where $\overrightarrow{S}^{(i)}$, $i \in 1,..,N$ is a set of samples drawn from $\pi(\overrightarrow{S}(t)|\overrightarrow{Z}(t))$ and $W(t)^{(i)}$ are associated weights normalized such that they add up to one. Then, the posterior density at can be approximated as:

$$\pi(\overrightarrow{S}(t)|\overrightarrow{Z}(t)) \approx \sum_i W^{(i)}(t)\delta(\overrightarrow{S}^{(i)}(t) - \overrightarrow{S}(t)), \tag{A.16}$$

where δ is the Kronecker function. The weights are chosen based on the *importance sampling* principle: The importance sampling principle allows us to obtain samples from a desired distribution π from which we can not directly sample but we can determine $\pi(\overrightarrow{S})$ for given $\overrightarrow{S}$. To do this, another distribution r called proposal distribution from which it is easy to sample is used. The importance sampling technique consists of placing different importance on each sample, depending on how likely it was for it to have been generated by the distribution π, rather than the actual sampling distribution, r. The closer r is to π the less samples are required. The weights are chosen then as follows:

$$W^{(i)}(t) = \frac{\pi(\overrightarrow{S}^{(i)})}{r(\overrightarrow{S}^{(i)})} \tag{A.17}$$

Now, in the sequential case, it can be shown [4], that if at each time step, the samples representing $\pi(\overrightarrow{S}(t-1)|\overrightarrow{Z}(0:t-1))$ are updated in such a way that they represent $r(\overrightarrow{S}(t)|\overrightarrow{S}(t-1), \overrightarrow{Z})$ then the weights can be updated as follows:

$$W^{(i)}(t) = W^{(i)}(t-1)\frac{\pi(\overrightarrow{Z}(t)|\overrightarrow{S}^{(i)}(t))\pi(\overrightarrow{S}^{(i)}(t)|\overrightarrow{S}^{(i)}(t-1))}{r(\overrightarrow{S}^{(i)}(t)|\overrightarrow{S}^{(i)}(t-1), \overrightarrow{Z}(0:t))}, \tag{A.18}$$

so that $\{\overrightarrow{S}^{(i)}(t), W^{(i)}(t)\}_{i=1}^{N}$ becomes a representation of $\pi(\overrightarrow{S}(t)|\overrightarrow{Z}(0:t))$. The choice of $r(\overrightarrow{S}^{(i)}(t)|\overrightarrow{S}^{(i)}(t-1), \overrightarrow{Z}(0:t))$ is crucial. It has been shown [32] that the optimal importance function r (minimizing the variance of the true weights) is:

$$r(\overrightarrow{S}^{(i)}(t)|\overrightarrow{S}^{(i)}(t-1), \overrightarrow{Z}(t)) = \pi(\overrightarrow{S}^{(i)}(t)|\overrightarrow{S}^{(i)}(t-1), \overrightarrow{Z}(t)). \tag{A.19}$$

In this case Equation (A.18) reduces to:

$$W^{(i)}(t) = W^{(i)}(t-1)\pi(\overrightarrow{Z}(t)|\overrightarrow{S}^{(i)}(t-1)). \tag{A.20}$$

Usually this optimal importance function is not used because of two problems: (1) In general it is not possible to sample from $\pi(\overrightarrow{S}^{(i)}(t)|\overrightarrow{S}^{(i)}(t-1), \overrightarrow{Z}(0:t))$ and (2) computing $)pi(\overrightarrow{Z}(t)|\overrightarrow{S}^{(i)}(t-1))$ generally requires integrating over all the $\overrightarrow{S}^{(i)}(t)$. A sub-optimal importance function that is most commonly used is the prediction distribution:

$$r(\overrightarrow{S}^{(i)}(t)|\overrightarrow{S}^{(i)}(t-1), \overrightarrow{Z}(0:t)) = \pi(\overrightarrow{S}^{(i)}(t)|\overrightarrow{S}^{(i)}(t-1), \overrightarrow{Z}(t)). \tag{A.21}$$

The prediction distribution is easy to sample from (using the system evolution equations), and the weight update in this case would be:

$$W^{(i)}(t) \propto W^{(i)}(t-1)\pi(\overrightarrow{Z}(t)|\overrightarrow{S}^{(i)}(t)). \tag{A.22}$$

$\pi(\overrightarrow{Z}(t)|\overrightarrow{S}^{(i)}(t))$ is generally easy to determine as the likelihood of the i^{th} sample with respect to the measurements. The particle filtering just described known as Sequential Importance Sampling (SIS) suffers from the degeneracy problem, where after a few iterations most particles would have a negligible weight. Resampling is commonly used to deal with this problem. the basic idea of resampling is to eliminate particles with large weights. The resampling step involves generating a new set $\{\overrightarrow{S}^{(i)}\}_{i=1}^{K}$ of unweighted samples by resampling (with replacement) K samples from the set of weighted samples representing $\pi(\overrightarrow{S}(t)|\overrightarrow{Z}(0:t)$ such that the probability of $\overrightarrow{S}^{(i)}$ to be drawn is equal to $W^{(i)}(t)$. In this case the new weights are directly proportional to the likelihood:

$$W^{(i)}(t) \propto \pi(\overrightarrow{Z}(t)|\overrightarrow{S}^{(i)}(t)). \tag{A.23}$$

A.3 Covariance-Based Update Propagation

Let $\overrightarrow{S}$ be a vector partitioned into two parts $\overrightarrow{S}^1$ and $\overrightarrow{S}^2$. The mean and covariance of $\overrightarrow{S}$ can then be written as follows:

$$\overrightarrow{\mu}_{\overrightarrow{S}} = \begin{bmatrix} \overrightarrow{\mu}_{\overrightarrow{S}^1} \\ \overrightarrow{\mu}_{\overrightarrow{S}^2} \end{bmatrix}. \tag{A.24}$$

$$\underline{\Sigma}_{\overrightarrow{S}} = \begin{bmatrix} \underline{\Sigma}_{\overrightarrow{S}^1} & \underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2} \\ \underline{\Sigma}_{\overrightarrow{S}^2\overrightarrow{S}^1} & \underline{\Sigma}_{\overrightarrow{S}^2} \end{bmatrix}. \tag{A.25}$$

If we know that the mean of $\overrightarrow{S}^2$ is updated to $\overrightarrow{\mu}^u_{\overrightarrow{S}^2}$, then, the distribution of $\overrightarrow{S}^1$ can be updated to the probability distribution of $\overrightarrow{S}^1$ given $\overrightarrow{S}^2 = \overrightarrow{\mu}^u_{\overrightarrow{S}^2}$, which has the following mean and covariance matrix:

$$\overrightarrow{\mu}^u_{\overrightarrow{S}^1} = \overrightarrow{\mu}_{\overrightarrow{S}^1} + \underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2}\underline{\Sigma}^{-1}_{\overrightarrow{S}^2}(\overrightarrow{\mu}^u_{\overrightarrow{S}^2} - \overrightarrow{\mu}_{\overrightarrow{S}^2}). \tag{A.26}$$

$$\underline{\Sigma}^u_{\overrightarrow{S}^1} = \underline{\Sigma}_{\overrightarrow{S}^1} - \underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2}\underline{\Sigma}^{-1}_{\overrightarrow{S}^2}\underline{\Sigma}_{\overrightarrow{S}^2\overrightarrow{S}^1}. \tag{A.27}$$

Furthermore, if both the mean and covariance of $\overrightarrow{S}^2$ are updated then, $\overrightarrow{\mu}_{\overrightarrow{S}^1}$, $\underline{\Sigma}_{\overrightarrow{S}^1}$, and $\underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2}$ need to be updated as follows:

$$\overrightarrow{\mu}^u_{\overrightarrow{S}^1} = \overrightarrow{\mu}_{\overrightarrow{S}^1} + \underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2}\underline{\Sigma}^{-1}_{\overrightarrow{S}^2}(\overrightarrow{\mu}^u_{\overrightarrow{S}^2} - \overrightarrow{\mu}_{\overrightarrow{S}^2}). \tag{A.28}$$

$$\underline{\Sigma}^u_{\overrightarrow{S}^1} = \underline{\Sigma}_{\overrightarrow{S}^1} - \underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2}\underline{\Sigma}^{-1}_{\overrightarrow{S}^2}(\underline{\Sigma}_{\overrightarrow{S}^2} - \underline{\Sigma}^u_{\overrightarrow{S}^2})(\underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2}\underline{\Sigma}^{-1}_{\overrightarrow{S}^2})^T. \tag{A.29}$$

$$\underline{\Sigma}^u_{\overrightarrow{S}^1\overrightarrow{S}^2} = \underline{\Sigma}_{\overrightarrow{S}^1\overrightarrow{S}^2}\underline{\Sigma}^{-1}_{\overrightarrow{S}^2}\underline{\Sigma}^u_{\overrightarrow{S}^2}. \tag{A.30}$$

Proofs for those latter formulas can be found in Appendix E of [73].

A.4 Nonlinear Covariance propagation

Let $\overrightarrow{S} \in \mathbb{R}^n$ be a random vector with mean $\overrightarrow{\mu}_{\overrightarrow{S}}$ and covariance matrix $\underline{\Sigma}_{\overrightarrow{S}}$, and let $h : \mathbb{R}^n \to \mathbb{R}^m$ be a nonlinear function. Up to first-order approximation, $\overrightarrow{z} = f(\overrightarrow{\mu}_{\overrightarrow{S}}) + \underline{H}(\overrightarrow{S} - \overrightarrow{\mu}_{\overrightarrow{S}})$, where $\underline{H}_{\overrightarrow{S}}$ is the Jacobian matrix of h evaluated at $\overrightarrow{S}$. If h is approximately affine in the region about the mean of the distribution, then this approximation is reasonable and the random vector $\overrightarrow{z} \in \mathbb{R}^m$ has mean $\overrightarrow{\mu}_Z \approx h(\overrightarrow{\mu}_{\overrightarrow{S}})$ and covariance $\underline{\Sigma}_{\overrightarrow{Z}} \approx \underline{H}_{\overrightarrow{S}}\underline{\Sigma}_{\overrightarrow{S}}\underline{H}_{\overrightarrow{S}}$. The covariance can also be propagated through a non-linear function using the Unscented Transform [63].

A.5 Gauss-Newton and Levenberg Marquardt Optimization

The Gauss-Newton algorithm (GNA) [17] is a method used to solve non-linear least squares problems. It can be seen as a modification of Newton's method for finding a minimum of a function. Unlike Newton's method, the Gauss-Newton algorithm can only be used to minimize a sum of squared function values, but it has the advantage that second derivatives, which can be challenging to compute, are not required.

Gauss Newton can be used to refine the solution of problems of the form

$$\overrightarrow{Z} = f(\overrightarrow{S}).$$

Starting with an initial guess $\overrightarrow{S}^{(0)}$ for the minimum, the method proceeds by the iterations

$$\overrightarrow{S}^{(i+1)} = \overrightarrow{S}^{(i)} + \delta\overrightarrow{S},$$

where the increment $\delta\overrightarrow{s}$ is determined in terms of the Jacobian of the function f, as

$$\underline{H}^T_{\overrightarrow{S}} H_{\overrightarrow{S}} \delta\overrightarrow{S} = H^T_{\overrightarrow{S}} \delta\overrightarrow{z},$$

where $H_{\overrightarrow{S}}$ is the Jacobian of f with respect to $\overrightarrow{S}$ and $\delta\overrightarrow{Z} = \overrightarrow{z} - f(\overrightarrow{s}^{(i)})$. A major problem with the GNA is that if the initial solution is not very close to the minimum, the algorithm might diverge. A solution to this is provided by the Levenberg Marquardt algorithm (LMA) [86]. The LMA interpolates between the methods of GNA and gradient descent through introducing the parameter λ, and modifying the equation used to find the incremental update as follows:

$$(H^T H + \lambda I)\delta\overrightarrow{S} = H^T \delta\overrightarrow{Z}.$$

If λ is much greater than 1, this equation will be equivalent to a gradient descent iterations:

$$\delta\overrightarrow{S} \propto H^T \delta\overrightarrow{Z}.$$

When λ is close to zero, LMA will be equivalent to GNA. LMA starts with a certain value of λ, then after every iteration, if the error $\delta\overrightarrow{z}$ increases, λ is increased by a certain factor, if the error decreases λ is decreased. This makes LMA more robust than the GNA as it can find a solution even if it starts from a point far from the minimum.

Appendix B

Rotation Representations

Three dimensional rotations can be represented in different ways. The most common way is a rotation matrix $\underline{R}$ belonging to the special group of orthogonal matrices $SO(3)$ (Lie group of rotations in $\mathbb{R}^3$). However, the rotation matrix is not suitable for optimization and estimation purposes since it is a 9 parameters representation of a 3 dimensional entity. Two more convenient representations are the angle-axis representation and the quaternion representation. Both representations are minimal and can be coded in a 3-vector which we refer to in this thesis as $\overrightarrow{\Omega}$. This appendix describes those two representation and the maps between them and the rotation matrix representation.

B.1 Angle-axis Representation

The angle axis representation, is a minimal representation that represents the rotation, by an axis of rotation $\overrightarrow{r}$ (2 parameters) and an angle of rotation θ around the axis (1 parameter). Therefore, the rotation is represented as a 3-vector $\overrightarrow{\Omega} = \theta\overrightarrow{r}$.This representation is suitable for small rotations (in $[-\pi, \pi]$ since it presents jumps on the boundaries of this interval). The transformation from and to rotation matrix is given by the log and exponential maps.

The exponential map from $\mathbb{R}^3$ to $SO(3)$ is obtained using the Rodrigues formula [19], which is derived via a taylor expansion of the exponential of the skew matrix of a rotation vector Ω :

$$\begin{aligned}\underline{R} &= \mathbb{R}^3 to SO_3(\overrightarrow{\Omega}) \\ &= I_3 + [\overrightarrow{r}]_\times \sin(\theta) + [\overrightarrow{r}]_\times^2 (1 - \cos(\theta))\end{aligned}$$

where $\theta = ||\overrightarrow{\Omega}||$, $||.||$ being the L2-norm, and I_3 is a 3×3 identity matrix. The log map is defined as follows:

$$\overrightarrow{\Omega} = SO_3 to\mathbb{R}^3(\underline{R}) = \log(\underline{R}) = \begin{cases} 0 & \text{if } \theta = 0 \\ \frac{\theta}{2\sin(\theta)}(R - R^\top) & \text{if } \theta \neq 0 \text{ and } \theta \in (-\pi, \pi) \end{cases}$$

where $\theta = \arccos\left(\frac{\text{trace}(R)-1}{2}\right)$

B.2 Quaternions Representation

A quaternion is a 4-tuple $\in \mathbb{R}^4$, the four-dimensional vector space over the real numbers. It can be thought of as a complex number with three different imaginary parts and can be written as: $\mathring{q} = q_0 + iq_1 + jq_2 + kq_3$.

Quaternions multiplication is not commutative. To multiply quaternions, the following 2 rules are used: $i^2 = j^2 = k^2 = -1$ and $ij = -ji = k$, $jk = -kj = i$, $ki = -ik = j$. Then if $\mathring{r} = \mathring{p}\mathring{q}$, we get

$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{B.1}$$

The conjugate of a quaternion is defined by:

$$\mathring{q}^* = (q_0 + iq_1 + jq_2 + kq_3)^* = q_0 - iq_1 - jq_2 - kq_3.$$

A unit quaternion, $q_o + iq_1 + jq_2 + kq_3$, can be used to represent a rotation by θ about a unit vector $\overrightarrow{r}$ as follows:

$$\begin{aligned} p_0 &= cos((\theta)/2) \\ \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} &= \overrightarrow{r} sin(\theta/2) \end{aligned} \tag{B.2}$$

To rotate a vector $\overrightarrow{v}$ by a quaternion $\mathring{q}$, the rotated vector $\overrightarrow{u}$ is given by:

$$\begin{bmatrix} 0 \\ \overrightarrow{u} \end{bmatrix} = \mathring{q} \begin{bmatrix} 0 \\ \overrightarrow{v} \end{bmatrix} \mathring{q}^* \tag{B.3}$$

Since in a unit quaternion we have $q_0 = \sqrt{q_1^2 + q_2^2 + q_3^2}$, we represent unit quaternions as a 3-vector $\overrightarrow{\Omega}$, such that $\Omega_1 = q_1$, $\Omega_2 = q_2$, and $\Omega_3 = q_3$. The map between the quaternion representation and the rotation matrix is defined as follows:

$$\overrightarrow{\Omega} = SO_3to\mathbb{R}^3(\underline{R}) = \begin{bmatrix} \frac{R_8 - R_6}{2\sqrt{1+R_1+R_5+R_9}} \\ \frac{R_3 - R_7}{2\sqrt{1+R_1+R_5+R_9}} \\ \frac{R_4 - R_2}{2\sqrt{1+R_1+R_5+R_9}} \end{bmatrix} \tag{B.4}$$

$$\underline{R} = \mathbb{R}^3toSO_3(\overrightarrow{\Omega}) = \begin{bmatrix} 1 - 2\Omega_2^2 - 2\Omega_3^2 & 2\Omega_1\Omega_2 - 2cos(\frac{\theta}{2})\Omega_3 & 2\Omega_1\Omega_3 + 2cos(\frac{\theta}{2})\Omega_2 \\ 2\Omega_1\Omega_2 + 2cos(\frac{\theta}{2})\Omega_3 & 1 - 2\Omega_1^2 - 2\Omega_3^2 & 2\Omega_2\Omega_3 - 2cos(\frac{\theta}{2})\Omega_1 \\ 2\Omega_1\Omega_3 - 2\Omega_2 & 2\Omega_2\Omega_3 + 2cos(\frac{\theta}{2})\Omega_1 & 1 - 2\Omega_1^2 - 2\Omega_2^2 \end{bmatrix} \tag{B.5}$$

where $cos(\frac{\theta}{2}) = \sqrt{1 - \Omega_1^2 + \Omega_2^2 + \Omega_3^2}$.

Appendix C

Trifocal Tensor Constraint

In chapter two, the point-point-point trifocal incidence equation (Equation (2.29)) has been rearranged as:

$$\underline{K}\overrightarrow{\mathcal{T}} = \overrightarrow{0}.$$

This appendix provides the expression of $\underline{K}$ and the Jacobian of Equation (3.10). $\underline{K}$ is a 9×27 matrix, therefore, in order to fit within the width of the page, we divide it into three chunks $\underline{K}^{1:9}$, $\underline{K}^{10:18}$ and $\underline{K}^{19:27}$. The entries of $\underline{K}$ are trilinear products of the image projections $\overrightarrow{q}(t_1)$, $\overrightarrow{q}(t_2)$ and $\overrightarrow{q}(t_3)$. Let

$$\begin{aligned} u_1 &= q_1(t_1) \\ v_1 &= q_2(t_1) \\ u_2 &= q_1(t_2) \\ v_2 &= q_2(t_2) \\ u_3 &= q_1(t_3) \\ v_3 &= q_2(t_3) \end{aligned} \tag{C.1}$$

then,

$$\underline{K}^{1:9} =$$

$$\begin{bmatrix}
0 & 0 & 0 & 0 & -u_1 & u_1v_3 & 0 & u_1v_2 & -u_1v_2v_3 \\
0 & 0 & 0 & u_1 & 0 & -u_1u_3 & -u_1v_2 & 0 & u_1u_3v_2 \\
0 & 0 & 0 & -u_1v_3 & u_1u_3 & 0 & u_1v_2v_3 & -u_1v_2u_3 & 0 \\
0 & u_1 & -u_1v_3 & 0 & 0 & 0 & 0 & -u_1u_2 & u_1u_2v_3 \\
-u_1 & 0 & u_1u_3 & 0 & 0 & 0 & u_1u_2 & 0 & -u_1u_2u_3 \\
u_1v_3 & -u_1u_3 & 0 & 0 & 0 & 0 & -u_1u_2v_3 & u_1u_2u_3 & 0 \\
0 & -u_1v_2 & u_1v_2v_3 & 0 & u_1u_2 & -u_1u_2v_3 & 0 & 0 & 0 \\
u_1v_2 & 0 & -u_1v_2u_3 & -u_1u_2 & 0 & u_1u_2u_3 & 0 & 0 & 0 \\
-u_1v_2v_3 & u_1v_2u_3 & 0 & u_1u_2v_3 & -u_1u_2u_3 & 0 & 0 & 0 & 0
\end{bmatrix}, \tag{C.2}$$

$$\underline{K}^{10:18} =$$

$$\begin{bmatrix}
0 & 0 & 0 & 0 & -v_1 & v_1v_3 & 0 & v_1v_2 & -v_1v_2v_3 \\
0 & 0 & 0 & v_1 & 0 & -v_1u_3 & -v_1v_2 & 0 & v_1u_3v_2 \\
0 & 0 & 0 & -v_1v_3 & v_1u_3 & 0 & v_1v_2v_3 & -v_1v_2u_3 & 0 \\
0 & v_1 & -v_1v_3 & 0 & 0 & 0 & 0 & -v_1u_2 & v_1u_2v_3 \\
-v_1 & 0 & v_1u_3 & 0 & 0 & 0 & v_1u_2 & 0 & -v_1u_2u_3 \\
v_1v_3 & -v_1u_3 & 0 & 0 & 0 & 0 & -v_1u_2v_3 & v_1u_2u_3 & 0 \\
0 & -v_1v_2 & v_1v_2v_3 & 0 & v_1u_2 & -v_1u_2v_3 & 0 & 0 & 0 \\
v_1v_2 & 0 & -v_1v_2u_3 & -v_1u_2 & 0 & v_1u_2u_3 & 0 & 0 & 0 \\
-v_1v_2v_3 & v_1v_2u_3 & 0 & v_1u_2v_3 & -v_1u_2u_3 & 0 & 0 & 0 & 0
\end{bmatrix}, \tag{C.3}$$

and

$$\underline{K}^{19:27} =$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & -1 & v_3 & 0 & v_2 & -v_2v_3 \\ 0 & 0 & 0 & 1 & 0 & -u_3 & -v_2 & 0u & v_3v_2 \\ 0 & 0 & 0 & -v_3 & u_3 & 0 & v_2v_3 & -v_2u_3 & 0 \\ 0 & 1 & -v_3 & 0 & 0 & 0 & 0 & -u_2 & u_2v_3 \\ -1 & 0 & u_3 & 0 & 0 & 0 & u_2 & 0 & -u_2u_3 \\ v_3 & -u_3 & 0 & 0 & 0 & 0 & -u_2v_3 & u_2u_3 & 0 \\ 0 & -v_2 & v_2v_3 & 0 & u_2 & -u_2v_3 & 0 & 0 & 0 \\ v_2 & 0 & -v_2u_3 & -u_2 & 0 & u_2u_3 & 0 & 0 & 0 \\ -v_2v_3 & v_2u_3 & 0 & u_2v_3 & -u_2u_3 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{(C.4)}$$

Writing Equation (3.10) in terms of the quaternion elements corresponding to $\underline{R}(t_1, t_2)$ and $\underline{R}(t_1, t_3)$ we obtain a system of non-linear equation in the 12 elements of $[\overrightarrow{T}(t_1, t_2); \overrightarrow{\Omega}(t_1, t_2); \overrightarrow{T}(t_1, t_3); \overrightarrow{\Omega}(t_1, t_3)]$. The Jacobian of that system is a 27×9 matrix, which we obtain in the following manner:

$$\begin{aligned} \frac{\partial \underline{K}\overrightarrow{\mathcal{T}}}{\partial \overrightarrow{\Omega}(t_1, t_2)} &= \frac{\partial \underline{K}\overrightarrow{\mathcal{T}}}{\partial \mathcal{T}} \frac{\partial \mathcal{T}}{\partial \underline{R}(t_1, t_2)} \frac{\partial \underline{R}(t_1, t_2)}{\partial \overrightarrow{\Omega}(t_1, t_2)} \\ &= \underline{K} \frac{\partial \mathcal{T}}{\partial \underline{R}(t_1, t_2)} \frac{\partial \underline{R}(t_1, t_2)}{\partial \overrightarrow{\Omega}(t_1, t_2)} \end{aligned} \quad \text{(C.5)}$$

$\frac{\partial \mathcal{T}}{\partial \underline{R}(t_1,t_2)}$ can be determined easily from Equation (3.11), while $\frac{\partial \underline{R}(t_1,t_2)}{\partial \overrightarrow{\Omega}(t_1,t_2)}$ is provided in Appendix D.

Appendix D

Jacobians Computations

In this appendix we represent the Jacobian of a matrix $\underline{A}$ with respect to a matrix $\underline{B}$ as

$$\frac{\partial \underline{A}}{\partial \underline{B}} = \frac{\partial vec(\underline{A})}{\partial vec(\underline{B})^T} \tag{D.1}$$

D.1 Derivatives of $SO_3 to \mathbb{R}^3$ and $\mathbb{R}^3 to SO_3$

In the case of the angle-axis representation, the Jacobian of $SO_3 to \mathbb{R}^3$ and $\mathbb{R}^3 to SO_3$ are computed by elementary means from the Rodrigues formula [19] and its inverse:

$$\begin{gathered}\frac{\partial \mathbb{R}^3 to SO_3(\overrightarrow{\Omega})}{\partial \overrightarrow{\Omega}} = \frac{sin(\theta)}{\theta}\frac{\partial [\overrightarrow{\Omega}_n]_\times}{\partial \overrightarrow{\Omega}_n^T} + \frac{1-cos(\theta)}{\theta}\frac{\partial ([\overrightarrow{\Omega}_n]_\times)^2}{\partial \overrightarrow{\Omega}_n^T} + \\ vec([\overrightarrow{\Omega}_n]_\times)(cos(\theta) - \frac{sin(\theta)}{\theta}) + vec(([\overrightarrow{\Omega}_n]_\times)^2)(sin(\theta) - 2\frac{1-cos(\theta)}{\theta})\overrightarrow{\Omega}_n^T\end{gathered} \tag{D.2}$$

For $SO_3 to \mathbb{R}^3(\underline{R})$, the derivative of the diagonal elements is

$$\frac{\partial \overrightarrow{\omega}}{\partial \mathbb{R}_{ii}} = \frac{\theta cos(\theta) - sin(\theta)}{2 sin(\theta)^2}\overrightarrow{r}$$

and the derivative of the off diagonal elements is

$$\frac{\partial \overrightarrow{\omega}}{\partial \mathbb{R}_{ij}} = \frac{\theta}{2 sin(\theta)}\begin{bmatrix} d_{i3}d_{j2} - d_{i2}d_{j3} \\ d_{i1}d_{j3} - d_{i3}d_{j1} \\ d_{i1}d_{j2} - d_{i2}d_{j1} \end{bmatrix}$$

In the quaternions case, the Jacobians are obtained from the derivations of Equations (B.4) and (B.5). The derivative of $\mathbb{R}^3toSO_3(\overrightarrow{\Omega})$ is

$$\frac{\partial \mathbb{R}^3toSO_3(\overrightarrow{\Omega})}{\partial \overrightarrow{(}\Omega)} = \begin{bmatrix} 0 & -4\Omega_2 & -4\Omega_3 \\ 2\Omega_2 & 2\Omega_1 & 2cos(\frac{\Theta}{2}) \\ 2\Omega_3 & 2cos(\frac{\Theta}{2}) & 2\Omega_1 \\ 2\Omega_2 & 2\Omega_1 & 2cos(\frac{\Theta}{2}) \\ -4\Omega_1 & 0 & -4\Omega_3 \\ -2cos(\frac{\Theta}{2}) & 2\Omega_3 & 2\Omega_2 \\ 2\Omega_3 & -2 & 2\Omega_1 \\ 2cos(\frac{\Theta}{2}) & 2\Omega_3 & 2\Omega_2 \\ -4\Omega_1 & -4\Omega_2 & 0 \end{bmatrix} \tag{D.3}$$

Let $\alpha_1 = \frac{-(R_3-R_6)}{4(1+R_1+R_5+R_9)^{3/2}}$, $\alpha_2 = \frac{-(R_3-R_7)}{4(1+R_1+R_5+R_9)^{3/2}}$, and $\alpha_3 = \frac{-(R_4-R_2)}{4(1+R_1+R_5+R_9)^{3/2}}$, and $\alpha_4 = \frac{1}{2}(1+R_1+R_5+R_9)^{-1/2}$. Then the derivative of $SO_3to\mathbb{R}^3(\underline{R})$ is

$$\frac{\partial SO_3to\mathbb{R}^3(\underline{R})}{\partial \underline{R}} = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \alpha_1 & \alpha_2 & \alpha_4-\alpha_3 \\ \alpha_1 & -\alpha_4-\alpha_2 & \alpha_3 \\ \alpha_1 & \alpha_2 & \alpha_4-\alpha_3 \\ \alpha_1 & \alpha_2 & \alpha_3 \\ -\alpha_4-\alpha_1 & \alpha_2 & \alpha_3 \\ \alpha_1 & -\alpha_4-\alpha_2 & \alpha_3 \\ \alpha_4-\alpha_1 & \alpha_2 & \alpha_3 \\ \alpha_1 & \alpha_2 & \alpha_3 \end{bmatrix}^T \tag{D.4}$$

D.2 Derivatives of the Motion Evolution Equations

The motion evolution equations are defined as:

$$\begin{aligned} \underline{R}(t+1) &= \underline{R}(t,t+1)\underline{R}(t) \\ \overrightarrow{T}(t+1) &= \underline{R}(t,t+1)\overrightarrow{T}(t) + \overrightarrow{T}(t,t+1) \end{aligned} \tag{D.5}$$

which can be written as follows in terms of $\overrightarrow{\Omega}(t)$

$$\overrightarrow{\Omega}(t+1) = SO_3to\mathbb{R}^3(\underline{R})(\mathbb{R}^3toSO_3(\overrightarrow{\Omega}(t,t+1))\mathbb{R}^3toSO_3(\overrightarrow{\Omega}(t)) \tag{D.6}$$

$$\overrightarrow{T}(t+1) = (\mathbb{R}^3toSO_3(\overrightarrow{\Omega}(t,t+1))\overrightarrow{T}(t) + \overrightarrow{T}(t,t+1) \tag{D.7}$$

Five derivatives are needed: $\frac{\partial\overrightarrow{\Omega}(t+1)}{\partial\overrightarrow{\Omega}(t,t+1)}, \frac{\partial\overrightarrow{\Omega}(t+1)}{\partial\overrightarrow{\Omega}(t)}, \frac{\partial\overrightarrow{T}(t+1)}{\partial\overrightarrow{\Omega}(t,t+1)}, \frac{\partial\overrightarrow{T}(t+1)}{\partial\overrightarrow{T}(t,t+1)}, \frac{\partial\overrightarrow{T}(t+1)}{\partial\overrightarrow{T}(t)}$.

Let $\underline{R}(t+1) = (\mathbb{R}^3toSO_3(\overrightarrow{\Omega}(t,t+1))(\mathbb{R}^3toSO_3(\overrightarrow{\Omega})(t)) = (\mathbb{R}^3toSO_3(\overrightarrow{\Omega}(t+1))$ and $\underline{R}(t) = (\mathbb{R}^3toSO_3(\overrightarrow{\Omega}(t)))$, then:

1. $\frac{\partial\overrightarrow{\Omega}(t+1)}{\partial\overrightarrow{\Omega}(t,t+1)}$ is determined by the chain rule:

$$\begin{gathered}\frac{\partial\overrightarrow{\Omega}(t+1)}{\partial\overrightarrow{\Omega}(t,t+1)} = \\ \frac{\partial\overrightarrow{\Omega}(t+1)}{\partial\underline{R}(t+1)}\frac{\partial\underline{R}(t+1)}{\partial\mathbb{R}^3toSO_3(\overrightarrow{\Omega}(t,t+1))}\frac{\partial\mathbb{R}^3toSO_3(\overrightarrow{\Omega}(t,t+1))}{\partial\overrightarrow{\Omega}(t,t+1)},\end{gathered} \tag{D.8}$$

with:

- $\frac{\partial\overrightarrow{\Omega}(t+1)}{\partial\underline{R}(t+1)}$ is determined by the inverse Rodrigues formula derivatives and is a 3×9 matrix.
- Let $\underline{R}(t) = \mathbb{R}^3toSO_3(\overrightarrow{\Omega})(t)$ then $\frac{\partial\underline{R}(t+1)}{\partial\mathbb{R}^3toSO_3(\overrightarrow{\Omega}(t,t+1))} = \frac{\partial(\mathbb{R}^3toSO_3(\overrightarrow{\Omega}(t,t+1))\underline{R}(t))}{\partial\mathbb{R}^3toSO_3(\overrightarrow{\Omega}(t,t+1))}$ which is equal to the following 9×9 block diagonal matrix:

$$\begin{gathered}\frac{\partial(\mathbb{R}^3toSO_3(\overrightarrow{\Omega})(t,t+1))\underline{R}(t))}{\partial\mathbb{R}^3toSO_3(\overrightarrow{\Omega})(t,t+1)} = \\ \begin{bmatrix} R_1(t) & 0 & 0 & R_4(t) & 0 & 0 & R_7(t) & 0 & 0 \\ 0 & R_1(t) & 0 & 0 & R_4(t) & 0 & 0 & R_7(t) & 0 \\ 0 & 0 & R_1(t) & 0 & 0 & R_4(t) & 0 & 0 & R_7(t) \\ R_2(t) & 0 & 0 & R_5(t) & 0 & 0 & R_8(t) & 0 & 0 \\ 0 & R_2(t) & 0 & 0 & R_5(t) & 0 & 0 & R_8(t) & 0 \\ 0 & 0 & R_2(t) & 0 & 0 & R_5(t) & 0 & 0 & R_8(t) \\ R_3(t) & 0 & 0 & R_6(t) & 0 & 0 & R_9(t) & 0 & 0 \\ 0 & R_3(t) & 0 & 0 & R_6(t) & 0 & 0 & R_9(t) & 0 \\ 0 & 0 & R_3(t) & 0 & 0 & R_6(t) & 0 & 0 & R_9(t) \end{bmatrix}\end{gathered}$$

- $\frac{\partial\mathbb{R}^3toSO_3(\overrightarrow{\Omega}(t,t+1))}{\partial\overrightarrow{\Omega}(t,t+1)}$ is determined by the Rodrigues formula derivatives and is a

9×3 matrix.

2. $\frac{\partial \overrightarrow{\Omega}(t+1)}{\partial \overrightarrow{\Omega}(t)}$ is similarly determined as :

$$\frac{\partial \overrightarrow{\Omega}(t+1)}{\partial \overrightarrow{\Omega}(t)} =$$
$$\frac{\partial \overrightarrow{\Omega}(t+1)}{\partial \underline{R}(t+1)} \frac{\partial \underline{R}(t+1)}{\partial \mathbb{R}^3 to SO_3(\overrightarrow{\Omega}(t))} \frac{\partial \mathbb{R}^3 to SO_3(\overrightarrow{\Omega}(t))}{\partial \overrightarrow{\Omega}(t)},$$

- $\frac{\partial \underline{R}(t+1)}{\partial \mathbb{R}^3 to SO_3(\overrightarrow{\Omega}(t))} = \frac{\partial (\underline{R}(t,t+1) \mathbb{R}^3 to SO_3(\overrightarrow{\Omega}(t))}{\partial \mathbb{R}^3 to SO_3(\overrightarrow{\Omega}(t))}$ which is equal to the following 9×9 block diagonal matrix:

$$\frac{\partial (\underline{R}(t,t+1) \mathbb{R}^3 to SO_3(\overrightarrow{\Omega}(t))}{\partial \mathbb{R}^3 to SO_3(\overrightarrow{\Omega}(t))} = \begin{bmatrix} \underline{R}(t,t+1) & 0 & 0 \\ 0 & \underline{R}(t,t+1) & 0 \\ 0 & 0 & \underline{R}(t,t+1) \end{bmatrix} \tag{D.9}$$

3. $\frac{\partial \overrightarrow{T}(t+1)}{\partial \overrightarrow{\Omega}(t,t+1)}$ can be written as follows:

$$\frac{\partial \overrightarrow{T}(t+1)}{\partial \overrightarrow{\Omega}(t,t+1)} = \frac{\partial \mathbb{R}^3 to SO_3(\overrightarrow{\Omega}(t,t+1)) \overrightarrow{T}(t)}{\partial \mathbb{R}^3 to SO_3(\overrightarrow{\Omega}(t,t+1))} \frac{\partial \mathbb{R}^3 to SO_3(\overrightarrow{\Omega}(t,t+1))}{\partial \overrightarrow{\Omega}(t,t+1)} \tag{D.10}$$

where $\frac{\partial \mathbb{R}^3 to SO_3(\overrightarrow{\Omega}(t,t+1)) \overrightarrow{T}(t)}{\partial \mathbb{R}^3 to SO_3(\overrightarrow{\Omega}(t,t+1))}$ is equal to

$$\begin{bmatrix} T_1(t) & 0 & 0 & T_2(t) & 0 & 0 & T_3(t) & 0 & 0 \\ 0 & T_1(t) & 0 & 0 & T_2(t) & 0 & 0 & T_3(t) & 0 \\ 0 & 0 & T_1(t) & 0 & 0 & T_2(t) & 0 & 0 & T_3(t) \end{bmatrix}$$

4. $\frac{\partial \overrightarrow{T}(t+1)}{\partial \overrightarrow{T}(t)}$ is equal to $\mathbb{R}^3 to SO_3(\overrightarrow{\Omega}(t,t+1))$.

5. $\frac{\partial \overrightarrow{T}(t+1)}{\partial \overrightarrow{T}(t,t+1)}$ is equal to I_3

References

[1] The rawseeds dataset. `http://www.rawseeds.com`.

[2] G. Adiv. Inherent ambiguities in recovering 3-d motion and structure from a noisy flow field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):477–489, 1989.

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

[4] M.S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, D. Sci, T. Organ, and S.A. Adelaide. A tutorial on particle filters for online nonlinear/non-GaussianBayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188, 2002.

[5] S. Avidan and A. Shashua. Tensor embedding of the fundamental matrix. *3D Structure from Multiple Images of Large-Scale Environments*, pages 47–62, 1998.

[6] S. Avidan and A. Shashua. Threading fundamental matrices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):73–77, 2001.

[7] A. Azarbayejani and A.P. Pentland. Recursive estimation of motion, structure and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):562–575, 1995.

[8] L. Bai and Y. Wang. Fusing image, GPS and GIS for road tracking using multiple condensation particle filters. In *2008 IEEE Intelligent Vehicles Symposium*, pages 162–167, 2008.

[9] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

[10] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the EKF-SLAM algorithm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3562–3568, 2006.

[11] Y. Bar-Shalom, X.R. Li, and T. Kirubarajan. *Estimation with applications to tracking and navigation*. Wiley-Interscience, 2001.

[12] J.L. Barron and R. Eagleson. Recursive estimation of time-varying motion and structure parameters. *Pattern Recognition*, 29(5):797–818, 1996.

[13] J.L. Barron, A.D. Jepson, and J.K. Tsotsos. The feasibility of motion and structure from noisy time-varying image velocity information. *International Journal of Computer Vision*, 5(3):239–269, 1990.

[14] J.L. Barron, W.K.J. Ngai, and H. Spies. Quantitative depth recovery from time-varying optical flow in a kalman filter framework. *LNCS 2616 Theoretical Foundations of Computer Vision: Geometry, Morphology, and Computational Imaging*, 2616:344–355, 2003.

[15] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Computer Vision–ECCV 2006*, pages 404–417, 2006.

[16] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Computer VisionECCV'92*, pages 237–252. Springer, 1992.

[17] Å. Bjorck. *Numerical methods for least squares problems*. Society for Industrial Mathematics, 1996.

[18] J.Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm. *Intel Corporation, Microprocessor Research Labs*, 2000.

[19] R. Brockett. Robotic manipulators and the product of exponentials formula. In *Mathematical theory of networks and systems*, pages 120–129. Springer, 1984.

[20] T.J. Broida, S. Chandrashekhar, and R. Chellappa. Recursive 3-d motion estimation from a monocular image sequence. *IEEE Transactions on Aerospace and Electronic Systems*, 26(4):639–656, 1990.

[21] T.J. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):90–99, 1986.

[22] A.R. Bruss and B.K.P. Horn. Passive navigation. *Computer Vision, Graphics, and Image Processing*, 21:3–20, 1983.

[23] D. Chekhlov, M. Pupilli, W. Mayol-Cuevas, and A. Calway. Real-time and robust monocular SLAM using predictive multi-resolution descriptors. *Advances in Visual Computing*, pages 276–285, 2006.

[24] M. Chikuma, Z. Hu, and K. Uchimura. Fusion of Vision, GPS and 3D-Gyro in Solving Camera Global Registration Problem for Vision-based Road Navigation. *IEIC Technical Report (Institute of Electronics, Information and Communication Engineers)*, 103(643):71–76, 2004.

[25] A. Chiuso, P. Favaro, H. Jin, and S. Soatto. 3-d motion and structure from 2-d motion causally integrated over time: Implementation. In *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part II*, pages 734–750, London, UK, 2000. Springer-Verlag.

[26] A. Chiuso, P. Favaro, H. Jin, and S. Soatto. Structure from Motion Causally Integrated Over Time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 523–535, 2002.

[27] A. Chiuso and S. Soatto. MFm: 3-D Motion From 2-D Motion Causally Integrated Over Time-Part I: Theory. In *Eur. Conf. Computer Vision, Dublin, Ireland*. Citeseer, 2000.

[28] J. Civera, A.J. Davison, and J.M.M. Montiel. Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945, 2008.

[29] A.J. Davison. Real-time simultaneous localisation and mapping with a single camera, 2003.

[30] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1052–1067, 2007.

[31] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181, 2006.

[32] A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[33] E. Eade and T. Drummond. Scalable Monocular SLAM. *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 1, 2006.

[34] E. Eade and T. Drummond. Monocular slam as a graph of coalesced observations. In *Proc. 11th IEEE International Conference on Computer Vision*, 2007.

[35] P. Elinas, R. Sim, and J.J Little. σSLAM: Stereo vision SLAM using the Rao-Blackwellised particle filter and a novel mixture proposal distribution. pages 1564–1570, 2006.

[36] L. Euler. Formulae generales pro translatione quacunque corporum rigidorum. *Novi Commentarii academiae scientiarum Petropolitanae*, 20:189–207, 1776.

[37] O. Faugeras. *Three-dimensional Computer Vision: a geometric viewpoint*. the MIT Press, 1993.

[38] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.

[39] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[40] D.A. Forsyth and J. Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[41] J. Goldbeck, B. Huertgen, S. Ernst, and L. Kelch. Lane following combining vision and DGPS. *Image and Vision Computing*, 18(5):425–433, 2000.

[42] V.M. Govindu. Lie-algebraic averaging for globally consistent motion estimation. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:684–691, 2004.

[43] R.M. Haralick, C.N. Lee, K. Otternberg, and M. Nolle. Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, 13(3):331–356, 1994.

[44] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, volume 15, page 50. Manchester, UK, 1988.

[45] R. Hartley and P. Sturm. Triangulation. In *Computer Analysis of Images and Patterns*, pages 190–197. Springer.

[46] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[47] D.J. Heeger and A.D. Jepson. Subspace methods for recovering rigid motion i: Algorithm and implementation. *International Journal of Computer Vision*, 7(2):95–117, January,1992.

[48] J. Heel. Dynamical systems and motion vision. Technical Report AIM-1037, 1988.

[49] J. Heel. Direct dynamic motion vision. *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, 2:1142–1147, 1990.

[50] J. Heel. Direct Estimation of Structure and Motion from Multiple Frames. 1990.

[51] H.V. Henderson and S.R. Searle. On deriving the inverse of a sum of matrices. *Siam Review*, pages 53–60, 1981.

[52] A. Heyden. Differential-Algebraic Multiview Constraints. *Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)-Volume 01*, pages 159–162, 2006.

[53] N.J. Higham. *Accuracy and stability of numerical algorithms*. Society for Industrial Mathematics, 2002.

[54] S. Holmes, G. Klein, and D.W. Murray. A square root unscented Kalman filter for visual monoSLAM. In *Proc Int Conf on Robotics and Automation*, pages 3710–3716. Citeseer, 2008.

[55] S.A. Holmes, G. Klein, and D.W. Murray. An O (N^2) Square Root Unscented Kalman Filter for Visual Simultaneous Localization and Mapping. *IEEE transactions on pattern analysis and machine intelligence*, pages 1251–1263, 2008.

[56] Z. Hu and K. Uchimura. Fusion of Vision, GPS and 3D Gyro Data in Solving Camera Registration Problem for Direct Visual Navigation. *International Journal of ITS Research*, 4(1):3–12, 2006.

[57] G. Huang, A. Mourikis, and S. Roumeliotis. A first-estimates Jacobian EKF for improving SLAM consistency. In *Experimental Robotics*, pages 373–382. Springer, 2009.

[58] P.J. Huber and E.M. Ronchetti. *Robust statistics*. John Wiley & Sons Inc, 2009.

[59] Y.S. Hung and H.T. Ho. A kalman filter approach to direct depth estimation incorporating surface structure. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 21(6):571, 1999.

[60] A. Jepson and D. Heeger. Linear subspace methods for recovering rigid motion. *Spatial Vision in Humans and Robots, Cambridge University Press*, 1992.

[61] A. Jepson and D. Heeger. Linear subspace methods for recovering translational direction. U. of Toronto TR RBCV-TR-92-40,1992, p. 19, 1992.

[62] E. Jones, A. Vedaldi, and S. Soatto. Inertial structure from motion with autocalibration. In *Proceedings of the ICCV Workshop on Dynamical Vision*, 2007.

[63] S.J. Julier and J.K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. 3, 1997.

[64] I.K. Jung and S. Lacroix. High resolution terrain mapping using low altitude aerial stereo imagery. In *Proc. Ninth Intl Conf. Computer Vision*, 2003.

[65] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Fast incremental smoothing and mapping with efficient data association. In *2007 IEEE International Conference on Robotics and Automation*, pages 1670–1677, 2007.

[66] K. Kanatani. Renormalization for unbiased estimation. *ICCV*, 93:599–606, 1993.

[67] N. Karlsson, E. di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and ME Munich. The vSLAM Algorithm for Robust Localization and Mapping. pages 24–29, 2005.

[68] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. IEEE Computer Society, 2007.

[69] G. Klein and D. Murray. Improving the agility of keyframe-based slam. *Computer Vision–ECCV 2008*, pages 802–815, 2008.

[70] K. Konolige and M. Agrawal. FrameSLAM: From bundle adjustment to realtime visual mappping. *IEEE Transactions on Robotics*, 24(5):1066–1077, 2008.

[71] K. Konolige, M. Agrawal, and J. Sola. Large scale visual odometry for rough terrain. In *Proc. International Symposium on Robotics Research*, 2007.

[72] N.M. Kwok and G. Dissanayake. Bearing-only slam in indoor environments using a modified particle filter. In *in Australasian Conference on Robotics and Automation*, pages 1–3, 2003.

[73] T. Lefebvre, H. Bruyninckx, and J. Schutter. *Nonlinear Kalman filtering for force-controlled robot tasks*. Springer Verlag, 2005.

[74] J.P. Lewis. Fast normalized cross-correlation. In *Vision Interface*, volume 10, pages 120–123. Citeseer, 1995.

[75] M. Lhuillier and L. Quan. A quasi-dense approach to surface reconstruction from uncalibrated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):418–433, 2005.

[76] H.C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Proc R Soc Lond B Biol Sci*, 208:385–397, 1980.

[77] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[78] B.D. Lucas and T. Kanade. An iterative technique of image registration and its application to stereo. *Proc. 7th Intl Joint Conf. on Artificial Intelligence*, pages 674–679, 1981.

[79] D. Marr and W.H. Vision. Freeman and Company. *New York*, pages 88–89, 1982.

[80] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[81] L. Matthies, T. Kanade, and R. Szeliski. Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3(3):209–238, 1989.

[82] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[83] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[84] M. Montemerlo. *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2003.

[85] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. 18:1151–1156, 2003.

[86] J. More. The Levenberg-Marquardt algorithm: implementation and theory. *Numerical analysis*, pages 105–116, 1977.

[87] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real time localization and 3d reconstruction. In *IEEE Conference of Vision and Pattern Recognition*. Citeseer, 2006.

[88] D. Nister. An efficient solution to the five-point relative pose problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(6):756–770, 2004.

[89] D. Nistér. Preemptive RANSAC for live structure and motion estimation. *Machine Vision and Applications*, 16(5):321–329, 2005.

[90] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2006.

[91] J. Oliensis. A new structure-from-motion ambiguity. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(7):685–700, 2000.

[92] J. Oliensis. Exact two-image structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1618–1633, 2002.

[93] K. Pauwels and M.M. Van Hulle. Optimal instantaneous rigid motion estimation insensitive to local minima. *Comput. Vis. Image Underst.*, 104(1):77–86, 2006.

[94] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual Modeling with a Hand-Held Camera. *International Journal of Computer Vision*, 59(3):207–232, 2004.

[95] J. Ponce and Y. Genc. Epipolar geometry and linear subspace methods: A new approach to weak calibration. *International Journal of Computer Vision*, 28(3):223–243, 1998.

[96] M. Pupilli and A. Calway. Real-time camera tracking using a particle filter. *Proceedings of the British Machine Vision Conference*, pages 519–528, 2005.

[97] M. Pupilli and A. Calway. Real-time visual slam with resilience to erratic motion. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, 2006.

[98] G. Qian and R. Chellappa. Structure from motion using sequential monte carlo methods. *International Journal of Computer Vision*, 59(1):5–31, 2004.

[99] G. Qian, R. Chellappa, and Q. Zheng. Robust structure from motion estimation using inertial data. *Journal of the Optical Society of America A*, 18:2982–2997, 2001.

[100] I. Rekleitis. Visual motion estimation based on motion blur interpretation. Technical report, 1995.

[101] D.I.C. Ressl. *Geometry, constraints and computation of the trifocal tensor*. Citeseer, 2003.

[102] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[103] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. *Lecture Notes in Computer Science*, 3951:430, 2006.

[104] S.I. Roumeliotis, A.E. Johnson, and J.F. Montgomery. Augmenting inertial navigation with image-based motion estimation. In *IEEE International Conference on Robotics and Automation, 2002. Proceedings. ICRA'02*, volume 4, 2002.

[105] P.D. Sampson. Fitting conic sections to. *Computer Graphics and Image Processing*, 18(1):97–108, 1982.

[106] H. Sawhney, A. Zisserman, S. Peleg, R. Szeliski, M. Irani, P. Torr, J. Knight, J. Malik, and P. Anandan. Discussion for direct versus features session. *Lecture notes in computer science*, 1883/2000:219–256.

[107] J. Shi and C. Tomasi. Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600, June 1994.

[108] R. Sim, P. Elinas, and J.J. Little. A study of the rao-blackwellised particle filter for efficient and accurate vision-based slam. *Int. J. Comput. Vision*, 74(3):303–318, 2007.

[109] R. Sim and J.J. Little. Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2082–2089, 2006.

[110] A. Singh. An estimation-theoretic framework for image-flow computation. *Computer Vision, 1990. Proceedings, Third International Conference on*, pages 168–177, 1990.

[111] S.N. Sinha, D. Steedly, and R. Szeliski. A multi-stage linear approach to structure from motion. In *ECCV Workshop on Reconstruction and Modeling of Large Scale 3D Virtual Environments*, 2010.

[112] N. Snavely, S.M. Seitz, and R. Szeliski. Skeletal graphs for efficient structure from motion. In *Proc. of IEEE CVPR*. Citeseer, 2008.

[113] S. Soatto. 3-d structure from visual motion: modeling, representation and observability. *AUTOMATICA-OXFORD-*, 33:1287–1312, 1997.

[114] S. Soatto and R. Brockett. Optimal structure from motion: Local ambiguities and global estimates. *Proc. Conf. Computer Vision and Pattern Recognition*, (3):282–228, 1998.

[115] S. Soatto, R. Frezza, and P. Perona. Recursive motion estimation on the essential manifold. In *Proc. 3 rd Europ. Conf. Comput. Vision, J.-O. Eklundh (Ed.), LNCS-Series*, volume 800, page 801. Citeseer.

[116] S. Soatto, R. Frezza, and P. Perona. Motion estimation via dynamic vision. *IEEE Transactions on Automatic Control*, 41(3):393–413, 1996.

[117] S. Soatto and P. Perona. On the exact linearization of structure from motion. *California Institute of Technology, Technical Report CITCDS*, pages 94–011, 1994.

[118] S. Soatto and P. Perona. Recursive 3-D Visual Motion Estimation Using Subspace Constraints. *International Journal of Computer Vision*, 22(3):235–259, 1997.

[119] H. Strasdat, J. M. M. Montiel, and A. Davison. Scale drift-aware large scale monocular slam. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[120] H. Strasdat, JMM Montiel, and A.J. Davison. Real-time monocular SLAM: Why filter? In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[121] M. Subbarao and AM Waxman. On the uniqueness of image flow solutions for planar surfaces in motion. *NASA STI/Recon Technical Report N*, 86:16520, 1985.

[122] R. Szeliski and S. Kang. Shape ambiguities in structure from motion. *Computer VisionECCV'96*, pages 709–721.

[123] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 23(7-8):693, 2004.

[124] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, International Journal of Computer Vision, 1991.

[125] C. Tomasi and T. Kanade. *Shape and Motion from Image Streams: a Factorization Method, Full Report on the Orthographic Case*. Number CMU-CS-92-104. Cornell University, Dept. of Computer Science, March 1992.

[126] P.H.S Torr and DW Murray. The development and comparison of robust methods for estimating the fundamental matrix. *International Journal of Computer Vision*, 24(3):271–300, 1997.

[127] P.H.S Torr and A. Zisserman. Robust parameterization and computation of the trifocal tensor. *Image and Vision Computing*, 15(8):591–605, 1997.

[128] B. Triggs. Factorization methods for projective structure and motion. In *cvpr*, page 845. Published by the IEEE Computer Society, 1996.

[129] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment–a modern synthesis. *Vision Algorithms: Theory and Practice*, 1883:298–372, 2000.

[130] B. Triggs, P.F. McLauchlan, R.I. Hartley, and A.W Fitzgibbon. Bundle adjustment-a modern synthesis. *Lecture notes in Computer Science*, pages 298–372, 1999.

[131] R. Tsai and T. Huang. Estimating three-dimensional motion parameters of a rigid planar patch, iii: Finite point correspondences and the three-view problem. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 32(2):213 – 220, apr. 1984.

[132] T.K. Xia, M. Yang, and R.Q. Yang. Vision based global localization for intelligent vehicles. In *2006 IEEE Intelligent Vehicles Symposium*, pages 571–576, 2006.

[133] Y. Xiong and S.A. Shafer. Dense structure from a dense optical flow sequence. In *International Symposium on Computer Vision,iscv*, page 1, Los Alamitos, CA, USA, 1995.

[134] J.S. Zelek. Towards real-time bayesian optical flow. *Elsevier Vision & Image Computing Journal*, 22:1051–1069, 2004.

[135] T. Zhang and C. Tomasi. On the consistency of instantaneous rigid motion estimation. *Int. J. Comput. Vision*, 46(1):51–79, 2002.

[136] Z. Zhang and Y. Shan. Incremental motion estimation through modified bundle adjustment. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 2, 2003.