Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

NASA - Jet Propulsion Laboratory
California Institute of Technology

Autonomous Systems Lab
Prof. Roland Siegwart

Master Thesis

Robust Vision-Based Navigation for Micro Air Vehicles

Spring Term 2014

Supervised by:
Dr. Stephan Weiss, JPL
Prof. Dr. Roland Yves Siegwart, ASL

Author:
Timo Hinzmann

Acknowledgement

This work was funded by the Army Research Laboratory under the Micro Autonomous Systems Technology Collaborative Technology Alliance program (MAST-CTA). JPL contributions were carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Robust Vision-Based Navigation for Micro Air Vehicles

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

Timo	Hinzmann

Student supervisor(s)

Stephan	Weiss

Supervising lecturer

Roland	Siegwart

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`https://www.ethz.ch/content/dam/ethz/associates/students/studium/exams/files-en/plagiarism-citationetiquette.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Kieselbronn, 11/26/14
Place and date

Timo Hinzmann
Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Contents

Symbols & Abbreviations

Frequently used symbols

Symbol	Unit	Description
$\mathbf{a}$	m/s^2	linear acceleration
ω	$1/s^2$	angular velocity
${}^{G}\mathbf{p}_I$	m	IMU position w.r.t. global frame {G}
${}^{G}\mathbf{v}_I$	m/s	IMU velocity w.r.t. global frame {G}
${}^{I}_{G}q$	–	quaternion rotation from global frame {G} to IMU frame {I}
$\mathbf{b}_g$	$1/s^2$	bias angular velocity (gyroscope)
$\mathbf{b}_a$	m/s^2	bias acceleration (accelerometer)
${}^{I}\mathbf{p}_C$	m	position of camera in IMU frame {I}
${}^{C}_{I}q$	-	quaternion rotation from IMU {I} to camera frame {C}
${}^{C_i}\mathbf{p}_{f_j} = [{}^{C_i}X_j,\ {}^{C_i}Y_j,\ {}^{C_i}Z_j]^T$	m	position of feature j in camera frame i
${}^{G}\mathbf{p}_{f_j}$	m	position of feature j in global frame {G}
$\mathbf{z}_i^{(j)}$	m	i-th measurement of j-th feature ($z = 1$ plane)
$\hat{\mathbf{z}}_i^{(j)}$	m	predicted observation of i-th measurement and j-th feature
$\mathbf{r}$	–	residual
$\mathbf{R}$	–	measurement covariance matrix
$\mathbf{H}$	–	measurement matrix
$\mathbf{K}$	–	Kalman filter gain matrix
$\mathbf{P}_{k+1\|k}$	–	a priori state covariance matrix at time $k+1$
$\mathbf{P}_{k+1\|k+1}$	–	a posteriori state covariance matrix at time $k+1$
$\mathbf{F}$	–	IMU state propagation matrix
$\mathbf{G}$	–	IMU noise propagation matrix

Abbreviations

Abbreviation	Meaning
EKF	Error State Kalman Filter
ENU	East-North-Up coordinate system
FAST	Feature from Accelerated Segment Test
FIFO	First-in-first-out
FOV	Field-of-View
GD	Gradient descent
GDLS	Gradient descent with line search
GN	Gauss-Newton
HPW	Half patch width
IMU	Inertial measurement unit
JTAG	Joint Test Action Group
KL	Known landmark
LIFO	Last-in-first-out
LM	Levenberg-Marquardt
MSC-EKF	Multi-state constraint EKF
MSF	Modular framework for multi-sensor fusion
N	Newton
N.b.	Nota bene
NED	North-East-Down coordinate system
OF	Opportunistic feature
PF	Persistent feature
PTAM	Parallel Tracking and Mapping
RANSAC	Random Sample Consensus
RBF	Radial Basis Function
SLAM	Simultaneous Localization and Mapping
SSF	Single sensor fusion framework
SVD	Singular value decomposition
SVM	Support vector machine
ZMGN	Zero-mean Gaussian Noise
ZMSSD	Zero-mean Sum of Squared Differences

Chapter 1

Introduction

Motivation Current systems for autonomous vision-based navigation of micro air vehicles are prone to failure in environments where the flat ground assumption is violated, during static motion phases and usually show a lack of autonomy during initialization. A robust vision-based navigation system is crucial for a variety of high-level tasks such as obstacle avoidance and path planning for autonomous reconnaissance and surveillance in GPS denied areas.

Previous work One approach to fuse inertial measurements obtained from an **I**nertial **M**easurement **U**nit (IMU) and visual feature observations is tightly-coupled visual **S**imultaneous **L**ocalization **A**nd **M**apping (SLAM) as illustrated in figure 1.1. SLAM-based algorithms augment the **E**rror state **K**alman **F**ilter (EKF) core states, which usually consist of the pose, scale and sensor biases, with M observed 3d features to jointly estimate the IMU pose and feature positions. The main disadvantage of this method is that the complexity of the filter grows quadratically with the number of observed features as illustrated in figure 1.2.

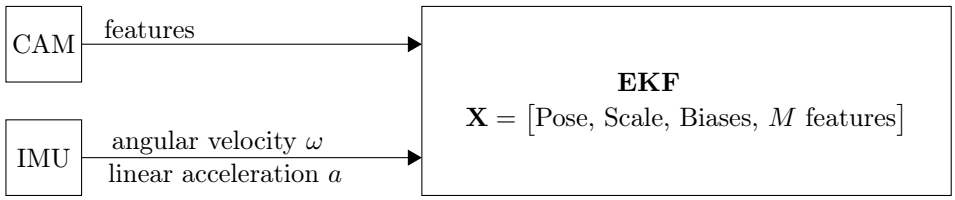


Figure 1.1: Simulatenous Localization and Mapping (SLAM) augments the EKF state vector with M features. The filter complexity increases quadratically with the number of features M.

A different approach is the well known **P**arallel **T**racking **A**nd **M**apping (PTAM) framework which was proposed by Klein et al. [1] and incorporated into a loosely coupled filter approach[1] by Weiss et al. [2, 3, 4]. The modified version is particularly suited for large environments and is more robust in self-similar scenes such as grass. The loosely coupled filter approach is shown in figure 1.4: Since the EKF state vector consists only of the pose, scale and sensor biases the complexity of the filter is constant as illustrated in figure 1.2. Two main disadvantages remain, however: Firstly, the filter is not consistent since the pose comes in unscaled and secondly, if

[1] The modified version uses the **S**ingle **S**ensor **F**usion (SSF) framework (`http://wiki.ros.org/ethzasl_sensor_fusion`) in combination with PTAM and is abbreviated by SSF/PTAM or simply by SSF in the following.

PTAM loses the map, the filter may diverge. This may, for example, happen when the quadrotor is flying over flat ground followed by a table as shown in figure 1.3.

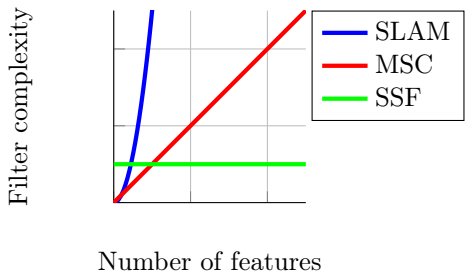


Figure 1.2: The SLAM filter complexity increases quadratically with the number of features M; the MSC filter complexity grows only linear with M and SSF's complexity is constant. Note: this is only an illustration, the constant SSF filter complexity may be higher or lower.

Figure 1.3: SSF/PTAM usually fails when flying over flat ground followed by a table.

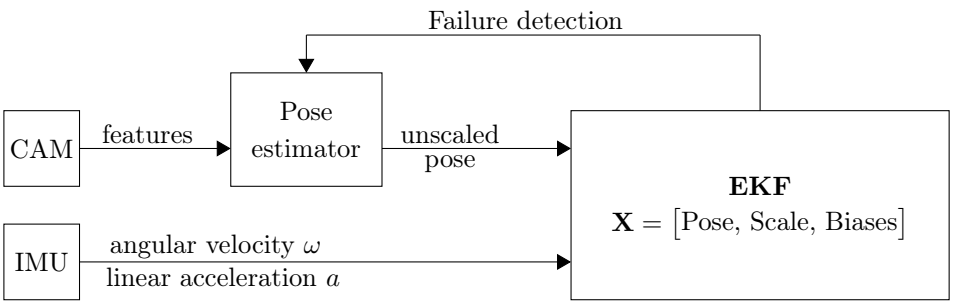


Figure 1.4: Parallel Tracking and Mapping (PTAM) in combination with the Single Sensor Fusion framework (SSF)

MSC-EKF In this master thesis a **M**ulti-**S**tate **C**onstraint (MSC) error state Kalman Filter based on the measurement model proposed by Mourikis et al. [5] is designed. Like the two approaches described above only an IMU as proprioceptive sensor and a single monochromatic camera as exteroceptive sensor are fused (see figure 1.5) for state estimation. The implemented filter is based on the `ethzasl_msf` framework[2] to guarantee modularity. The measurement model expressing the constraints of observing a static feature in multiple camera frames was proposed by Mourikis et al. [5]. The advantage of the formulation is that the triangulated 3d feature positions are not included in the state vector resulting in a computational complexity that is linear in the number of observed features as illustrated in figure 1.2.

N.b. that this a map-free approach and consequently the map cannot "get lost". The MSC-EKF can be categorized as a tightly-loosely coupled approach: Loosely coupled since it is used in a modular filter framework where additional sensors such as GPS can be added. Also, the feature tracker and feature triangulation module can

[2]Modular framework for **M**ulti **S**ensor **F**usion (MSF), see `https://github.com/ethz-asl/ethzasl_msf`

be considered black boxes that can be exchanged. The filter approach is, however, not purely loosely coupled because the triangulation module needs the camera poses from the Kalman filter as input and is thus not completely independent. Therefore, and due to the filter's consistency the approach is categorized as tightly-loosely coupled.

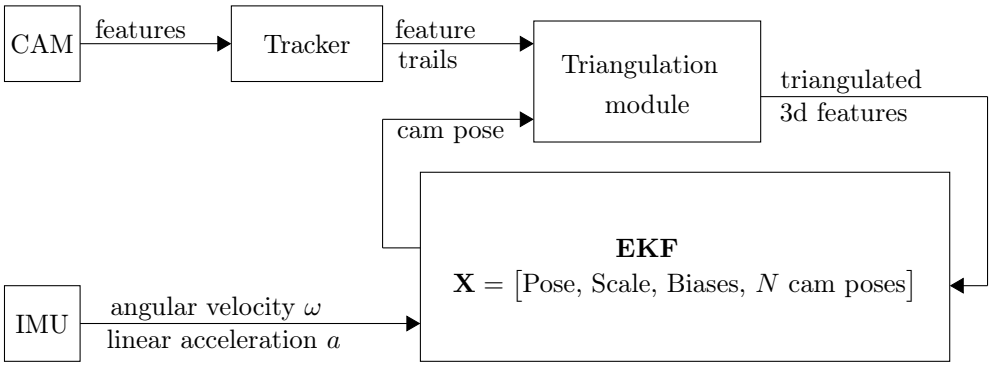


Figure 1.5: Multi-state constraint (MSC) error state Kalman filter (EKF).

The properties of the three approaches described in the previous paragraphs are summarized in table 1.1. The MSF/MSC-EKF's otherwise consistent filter derivation is narrowed by the fact that the Gaussian noise assumption of the EKF is corrupted in the measurement model.[3]

	SLAM	SSF/PTAM	MSF/MSC-EKF
Coupling	Tightly	Loosely	"Tightly/Loosely"
Complexity	$\mathcal{O}(M^2)$	$\mathcal{O}(M^0)$	$\mathcal{O}(M^1)$
Filter consistency	$+++$	$+$	$++$
Modularity	$+$	$+++$	$+++$
Map	✓	✓	✗

Table 1.1: Comparison of three different navigation approaches: SLAM, SSF/PTAM and MSF/MSC-EKF

[3]The measurement model is described in chapter 3.

Chapter 2

System setup and coordinate frames

In the first part of chapter 2, the hardware setup is *shortly* presented and referred to the corresponding datasheets otherwise. The second part introduces the various coordinate frames and the basic measurement equations used in this thesis.

2.1 Quadrocopter

The dimensions and technical details of the utilized AscTec Hummingbird quadrotor are shown in figure 2.1 and 2.2 respectively.

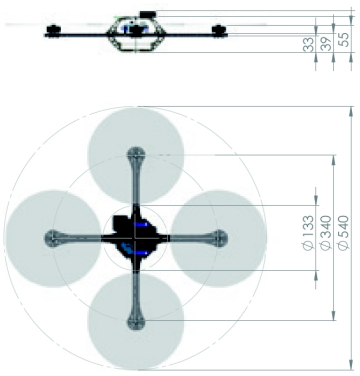


Figure 2.1: AscTec Hummingbird dimensions in millimeter. (image: [6])

Model	**AscTec Hummingbird**
Type	Quadrotor
Size	540 × 540 × 85.5 mm
Engines	4 electrical, brushless motors (80W max.)
Empty weight	≈ 350g
Wireless communication	2.4 GHz XBee link, WiFi
Battery	LiPo, 3 cells, 2100 mAh
Weight	≈ 50g

Figure 2.2: Technical details AscTec Hummingbird [6]

2.2 Processors

Three processors are mounted on the AscTec Hummingbird:

- **AscTec low-level processor** "The low-level processor is the heart of the vehicle running all critical functions as sensing and filtering, attitude estimation and flight control. The code on this processor is protected and cannot be accessed using a JTAG[1] interface nor be read through the serial interface."[2]

[1] Joint Test Action Group (JTAG)
[2] Adopted from the AscTec Hummingbird manual [6].

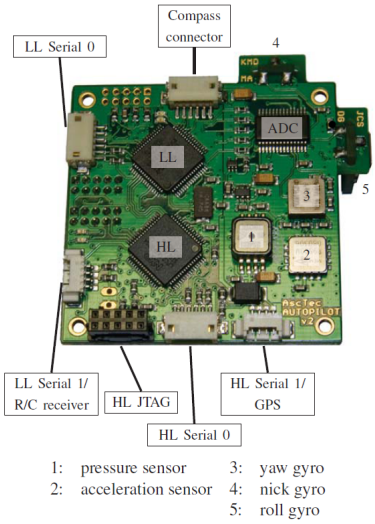


Figure 2.3: AscTec AutoPilot sensor board v2 featuring a low-level and high-level processor as well as three gyroscopes and three accelerometers [6]

Model	**AutoPilot low-level**
Processor	1000Hz
Task	Safety

Model	**AutoPilot high-level**
Processor	1000Hz
Task	User code

Model	**ODROID-U**
Processor	1.4 Ghz
Number of cores	4
RAM	1 GB
Storage	microSD slot
Task	User code

Figure 2.4: Processors: AscTec low-level, AscTec high-level and ODROID-U

- **AscTec high-level processor** The user's algorithm needs to be uploaded to the high-level processor using a JTAG adapter mounted on the high-level JTAG bootloader jumper pads shown in figure 2.3.
- **Odroid-U** A more powerful and convenient way to integrate the user code is the ODROID board running Ubuntu.

2.3 Sensors

In this thesis, an IMU as proprioceptive sensor and a single monochromatic camera as exteroceptive sensor are used.

2.3.1 Inertial measurement unit (IMU)

The AscTec AutoPilot sensor board mounts an Inertial Measurement Unit consisting of three gyroscopes and three accelerometers for measuring angular velocity and linear acceleration respectively.

Accelerometer

A three axis accelerometer in a single package as shown in figure 2.3 (see accelerometer: 2) is mounted on the AscTec AutoPilot board and can measure both dynamic accleration such as vibrations and static accelerations such as gravity [3].

Gyroscope

The yaw, nick and roll gyroscope measure the angular velocity around each individual axis and are located on the AutoPilot board as indicated in figure 2.3. [4].

[3] See accelerometer datasheet for details G
[4] See gyroscope datasheet for details H

2.3.2 Camera

The mvBlueFox MLC200 is a 1/3" CMOS monochrome camera capturing images with 752×480 pixels at up to 90 fps. The 8 bit monochrome camera can distinguish between 2^8 gray levels.

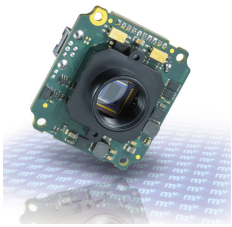


Figure 2.5: mvBlueFox monochrome CMOS camera [7]

Model	**mvBlueFox**
Resolution	752×480
Max. frame rate	90 Hz
Mpixels	0.4
Shutter type	Global
Image output	8 bit monochrome
Sensor size	1/3"
Weight	≈ 50 g

Figure 2.6: Technical details mvBlueFox [7]

2.4 Coordinate systems

The body frame of the AscTec Hummingbird quadrotor is shown in figure 2.7 in North-East-Down (NED) notation. The roll, pitch and yaw moments **L**, **M** and **N** are defined as the moment around the x-, y- and z-axis respectively. The motor speeds of the four rotors are denoted with n_1, n_2, n_3 and n_4.

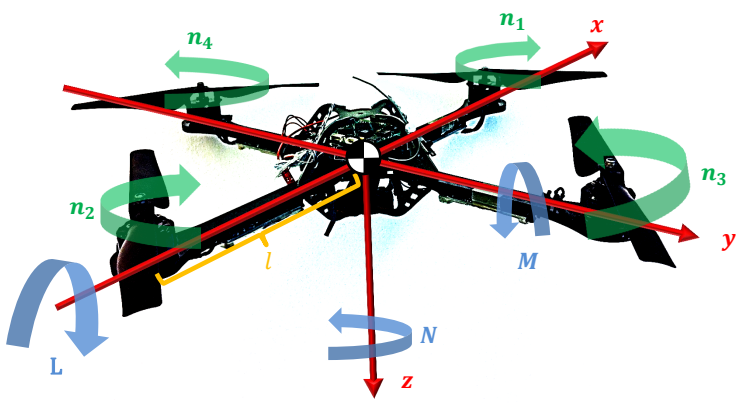


Figure 2.7: Asctec Hummingbird quadrotor with body frame illustration

$$\begin{aligned}
\mathbf{L}: &\ \text{Roll moment around the } x\text{-axis} \\
\mathbf{M}: &\ \text{Pitch moment around the } y\text{-axis} \\
\mathbf{N}: &\ \text{Yaw moment around the } z\text{-axis} \\
l: &\ \text{Quadrotor arm length} \\
n_i: &\ \text{Rotor speed of rotor } i = 1...4
\end{aligned}$$

The global, IMU and camera coordinate frames are shown in figure 2.8 and denoted with $\{\mathbf{G}\}$, $\{\mathbf{I}\}$ and $\{\mathbf{C}\}$; the red, green and blue arrows symbolize the x, y and z axes of the coordinate systems. ${}^{G}\mathbf{p}_{f_j}$ is the position of feature j in global coordinates, ${}^{C}\mathbf{p}_{f_j}$ is the feature position in camera coordinates. The Camera-IMU translation is given by ${}^{I}\mathbf{p}_{C}$. The position of the IMU in global coordinates which is a MSC-EKF filter state is ${}^{G}\mathbf{p}_{I}$.

Based on figure 2.8 the following equations can be derived and will be used in later chapters of this thesis:

Global camera position in terms of global IMU position, rotation and Camera-IMU translation

$$\begin{aligned}
{}^{G}\mathbf{p}_{C_i} &= {}^{G}\mathbf{p}_{I} + \mathbf{C}({}^{G}_{I}\bar{q})\,{}^{I}\mathbf{p}_{C} && (2.1)\\
&= {}^{G}\mathbf{p}_{I} + \mathbf{C}^{T}({}^{I}_{G}\bar{q})\,{}^{I}\mathbf{p}_{C} && (2.2)
\end{aligned}$$

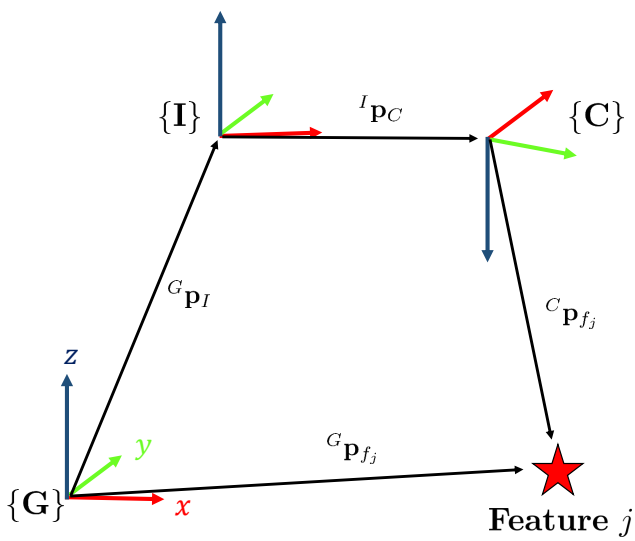


Figure 2.8: Coordinate systems: The red, green and blue arrows symbolize the x, y and z axes of the coordinate systems; $\{\mathbf{G}\}$, $\{\mathbf{I}\}$ and $\{\mathbf{C}\}$ stand for the global, IMU and camera coordinate frame respectively

Global camera rotation in terms of global IMU rotation and Camera-IMU rotation

$$ {}^{C_i}_{G}\bar{q} = {}^{C}_{I}\bar{q} \otimes {}^{I}_{G}\bar{q} \tag{2.3} $$

Feature position in camera coordinates in terms of global camera position, rotation and global feature position

$$ {}^{C_i}\mathbf{p}_{f_j} = \begin{bmatrix} {}^{C_i}X_j \\ {}^{C_i}Y_j \\ {}^{C_i}Z_j \end{bmatrix} = \mathbf{C}({}^{C_i}_{G}\bar{q})({}^{G}\mathbf{p}_{f_j} - {}^{G}\mathbf{p}_{C_i}) \tag{2.4} $$

Feature position in camera coordinates in terms of Camera-IMU transformation, global IMU position and rotation, global feature position The above equation can be rewritten in terms of the Camera-IMU transformation which is necessary for estimating the Camera-IMU translation and rotation (see section 3.6).

$$ \begin{aligned} {}^{C_i}\mathbf{p}_{f_j} &= \mathbf{C}({}^{C_i}_{G}\bar{q})({}^{G}\mathbf{p}_{f_j} - {}^{G}\mathbf{p}_{C_i}) && (2.5) \\ &= \mathbf{C}({}^{C}_{I}\bar{q})[\mathbf{C}({}^{I}_{G}\bar{q})({}^{G}\mathbf{p}_{f_j} - {}^{G}\mathbf{p}_{C_i})] && (2.6) \\ &= \mathbf{C}({}^{C}_{I}\bar{q})[\mathbf{C}({}^{I}_{G}\bar{q})({}^{G}\mathbf{p}_{f_j} - ({}^{G}\mathbf{p}_{I} + \mathbf{C}({}^{G}_{I}\bar{q})^{I}\mathbf{p}_{C})] && (2.7) \\ &= \mathbf{C}({}^{C}_{I}\bar{q})[\mathbf{C}({}^{I}_{G}\bar{q})({}^{G}\mathbf{p}_{f_j} - {}^{G}\mathbf{p}_{I}) + \mathbf{C}({}^{I}_{G}\bar{q})\mathbf{C}({}^{G}_{I}\bar{q})^{I}\mathbf{p}_{C}] && (2.8) \\ &= \mathbf{C}({}^{C}_{I}\bar{q})[\mathbf{C}({}^{I}_{G}\bar{q})({}^{G}\mathbf{p}_{f_j} - {}^{G}\mathbf{p}_{I}) + {}^{I}\mathbf{p}_{C}] && (2.9) \end{aligned} $$

Chapter 3

Multi-state constraint error state Kalman filter (MSC-EKF)

Figure 3.1 shows the system overview including sensors, feature tracker, feature triangulation and the elements of the multi-state constraint error state Kalman filter. The filter works as follows: As soon as an IMU measurement is available it is processed in the `ethzasl_msf` framework which propagates the EKF state and EKF state covariance (section 3.1). This is usually done with a rate of 100 Hz. Whenever an image is recorded (around 15 times per second) the corresponding camera position and attitude is appended to the EKF state vector until the maximum number of N camera poses is reached. Since the state vector is augmented, also the state covariance matrix needs to be updated accordingly (section 3.2).

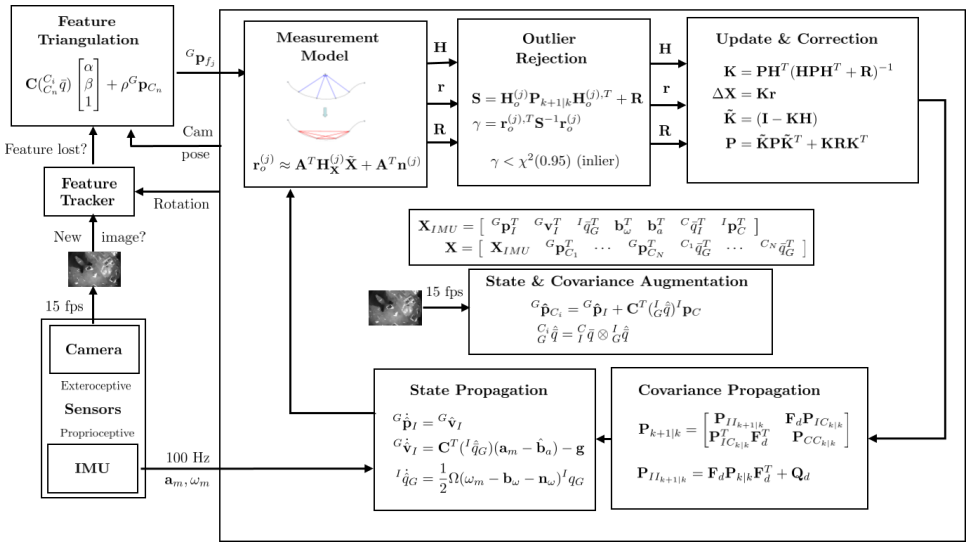


Figure 3.1: System overview: Sensors, feature tracker, feature triangulation and MSC-EKF

The feature tracker module (chapter 5) identifies salient features and assigns to each feature the camera frames in which it was observed as well as the correspond-

ing camera poses. Whenever a feature is lost (or the maximum number of camera poses in the EKF state vector is reached) the feature tracker module sends a list of pixel coordinates and corresponding camera poses in which the feature was observed to the triangulation module (chapter 6). Using this information the triangulation module calculates the most likely 3d feature position in global coordinates.

The measurement model (section 3.3) is based on the work of Mourikis et al. [5]. The residual between the predicted observation and the measurement is stacked up and, after rejecting possible outliers (section 3.3.4), used in the standard Kalman filter update equations (section 3.3.6). The state and covariance are corrected accordingly (section 3.3.7).

3.1 State propagation

The state and covariance propagation is mainly adopted from Weiss et al. [4] and Trawny et al. [8].

3.1.1 Physical motion model

The accelerometers and gyroscopes measure the linear acceleration and angular rate respectively. The measurements are disturbed by a bias $\mathbf{b}$ and white Gaussian noise $\mathbf{n}$:

$$\omega_m = {}^I\omega + \mathbf{b}_\omega + \mathbf{n}_\omega \tag{3.1}$$

$$\mathbf{a}_m = \mathbf{C}({}^I\bar{q}_G)({}^G\mathbf{a} + \mathbf{g}) + \mathbf{b}_a + \mathbf{n}_a \tag{3.2}$$

where $\mathbf{g}$ is the gravity expressed in the global frame G. Rewriting above equations in terms of angular velocity ${}^I\omega$ in the inertial frame and the linear acceleration ${}^G\mathbf{a}$ in the global frame yields

$${}^I\omega = \omega_m - \mathbf{b}_\omega - \mathbf{n}_\omega \tag{3.3}$$

$${}^G\mathbf{a} = \mathbf{C}^T({}^I\bar{q}_G)(\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a) - \mathbf{g} \tag{3.4}$$

With this the physical motion model results in

$$\boxed{\begin{aligned}
{}^G\dot{\mathbf{p}}_I &= {}^G\mathbf{v}_I && (3.5)\\
{}^G\dot{\mathbf{v}}_I &= {}^G\mathbf{a} = \mathbf{C}^T({}^I\bar{q}_G)(\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a) - \mathbf{g} && (3.6)\\
{}^I\dot{q}_G &\overset{1}{=} \frac{1}{2}\Omega(\omega_m - \mathbf{b}_\omega - \mathbf{n}_\omega){}^Iq_G && (3.7)\\
\dot{\mathbf{b}}_\omega &= \mathbf{n}_{b_\omega} && (3.8)\\
\dot{\mathbf{b}}_a &= \mathbf{n}_{b_a} && (3.9)\\
{}^C\dot{q}_I &= \mathbf{0} && (3.10)\\
{}^I\dot{\mathbf{p}}_C &= \mathbf{0} && (3.11)\\
& && (3.12)
\end{aligned}}$$

with

$$\mathbf{n}_\omega = \text{noise gyroscope} \tag{3.13}$$

$$\mathbf{n}_a = \text{noise accelerometer} \tag{3.14}$$

$$\mathbf{n}_{b_\omega} = \text{noise of bias of gyroscope} \tag{3.15}$$

$$\mathbf{n}_{b_a} = \text{noise of bias of accelerometer} \tag{3.16}$$

3.1.2 Motion model in state representation

Taking the expectation of the above equations by replacing the true values by state estimates and furthermore assuming ZMGN[1], i.e. $\mathbf{E}[\mathbf{n}_a] = \mathbf{E}[\mathbf{n}_\omega] = \mathbf{E}[\mathbf{n}_{b_\omega}] = \mathbf{E}[\mathbf{n}_{b_s}]$, equations 3.5 and ff. become

$$\boxed{\begin{aligned}
{}^G\dot{\hat{\mathbf{p}}}_I &= {}^G\hat{\mathbf{v}}_I && (3.17)\\
{}^G\dot{\hat{\mathbf{v}}}_I &= \mathbf{C}^T({}^I\hat{\bar{q}}_G)(\mathbf{a}_m - \hat{\mathbf{b}}_a) - \mathbf{g} && (3.18)\\
{}^I\dot{\hat{q}}_G &= \frac{1}{2}\Omega(\omega_m - \mathbf{b}_\omega - \mathbf{n}_\omega){}^Iq_G && (3.19)\\
\dot{\hat{\mathbf{b}}}_\omega &= \mathbf{0} && (3.20)\\
\dot{\hat{\mathbf{b}}}_a &= \mathbf{0} && (3.21)\\
{}^C\dot{\hat{q}}_I &= \mathbf{0} && (3.22)\\
{}^I\dot{\hat{\mathbf{p}}}_C &= \mathbf{0} && (3.23)
\end{aligned}}$$

The state vector is defined as

$$\mathbf{X} = \begin{bmatrix} {}^G\mathbf{p}_I^T & {}^G\mathbf{v}_I^T & {}^I\bar{q}_G^T & \mathbf{b}_\omega^T & \mathbf{b}_a^T & {}^C\bar{q}_I^T & {}^I\mathbf{p}_C^T & {}^G\mathbf{p}_{C_1}^T & \cdots & {}^G\mathbf{p}_{C_N}^T & {}^{C_1}\bar{q}_G^T & \cdots & {}^{C_N}\bar{q}_G^T \end{bmatrix} \tag{3.24}$$

with

$$\begin{aligned}
{}^G\mathbf{p}_I^T &= \text{IMU position w.r.t. global frame \{G\}} && (3.25)\\
{}^G\mathbf{v}_I^T &= \text{IMU velocity w.r.t. global frame \{G\}} && (3.26)\\
{}^I\bar{q}_G^T &= \text{quaternion rotation from global frame \{G\} to IMU frame \{I\}} && (3.27)\\
\mathbf{b}_\omega^T &= \text{bias angular velocity (gyroscope)} && (3.28)\\
\mathbf{b}_a^T &= \text{bias acceleration (accelerometer)} && (3.29)\\
{}^C\bar{q}_I^T &= \text{quaternion rotation from IMU \{I\} to camera frame \{C\}} && (3.30)\\
{}^I\mathbf{p}_C^T &= \text{position of the camera in IMU frame \{I\}} && (3.31)\\
{}^G\mathbf{p}_{C_i}^T &= \text{position of the } i\text{-th camera in the global frame \{G\}} && (3.32)\\
{}^{C_i}\bar{q}_G^T &= \text{quaternion rotation from } i\text{-th camera frame to global frame \{G\}} && (3.33)
\end{aligned}$$

3.1.3 Motion model in error state representation

Instead of the full state representation, the state vector is usually defined in terms of errors. The advantage of this notion is that a 4-element quaternion can be replaced by a 3-element error quaternion which reduces the state vector. The quaternion is then in its minimal representation [8][4].

$$\Delta x = x - \hat{x} \tag{3.34}$$

and

$$\Delta\dot{x} = \dot{x} - \dot{\hat{x}} \tag{3.35}$$

This results in the following differential error equations:

[1] **Z**ero **M**ean **G**aussian **N**oise (ZMGN)

[1] Quaternion integration: ${}^L_G\dot{\bar{q}} = \frac{1}{2}\Omega(\omega){}^L_G\bar{q}(t)$, see appendix A or [8], equation (101)

Position

$$\Delta^G\dot{\mathbf{p}}_I = {}^G\dot{\mathbf{p}}_I - {}^G\dot{\hat{\mathbf{p}}}_I \tag{3.36}$$
$$= {}^G\mathbf{v}_I - {}^G\hat{\mathbf{v}}_I \tag{3.37}$$
$$= \Delta^G\mathbf{v}_I \tag{3.38}$$

Velocity

$$\Delta^G\dot{\mathbf{v}}_I = {}^G\dot{\mathbf{v}}_I - {}^G\dot{\hat{\mathbf{v}}}_I \tag{3.39}$$
$$= \mathbf{C}^T({}^I\bar{q}_G)(\hat{\mathbf{a}} - \mathbf{b}_a - \mathbf{n}_a) - \mathbf{g} - (\mathbf{C}^T({}^I\hat{\bar{q}}_G)(\mathbf{a}_m - \hat{\mathbf{b}}_a) - \mathbf{g}) \tag{3.40}$$
$$= -\mathbf{C}^T({}^I\hat{\bar{q}}_G))\left\lfloor \mathbf{a}_m - \hat{\mathbf{b}}_a\times \right\rfloor \delta\theta - \mathbf{C}^T({}^I\hat{\bar{q}}_G)\mathbf{n}_a - \mathbf{C}^T({}^I\hat{\bar{q}}_G)\Delta\mathbf{b}_a \tag{3.41}$$

Rotation

$$\delta^I\dot{\theta}_G = -\left\lfloor \omega_m - \hat{\mathbf{b}}_\omega\times \right\rfloor \delta\theta - \mathbf{n}_\omega - \Delta b_\omega \tag{3.42}$$

The remaining differential equations for the error state representation are

$$\Delta\dot{\mathbf{b}}_\omega = \mathbf{n}_{b_\omega} \tag{3.43}$$
$$\Delta\dot{\mathbf{b}}_a = \mathbf{n}_{b_a} \tag{3.44}$$
$$\Delta^C\dot{\theta}_I = \mathbf{0} \tag{3.45}$$
$$\Delta^I\mathbf{p}_C = \mathbf{0} \tag{3.46}$$

The error state vector for the core states is defined as

$$\tilde{\mathbf{X}}_{core} = [\ \overbrace{\Delta^G p_I^T \quad \Delta^G v_I^T \quad \delta^I\theta_G^T \quad b_\omega^T \quad \Delta b_a^T \quad \delta^C\theta_I^T \quad \Delta^I p_C^T}^{\text{Core states}}\ ] \tag{3.47}$$

and

$$\boxed{\begin{aligned}
\Delta^G\dot{\mathbf{p}}_I &= \Delta^G\mathbf{v}_I && (3.48)\\
\Delta^G\dot{\mathbf{v}}_I &= -\mathbf{C}^T({}^I\hat{\bar{q}}_G))\left\lfloor \mathbf{a}_m - \hat{\mathbf{b}}_a\times \right\rfloor \delta\theta - \mathbf{C}^T({}^I\hat{\bar{q}}_G)\mathbf{n}_a - \mathbf{C}^T({}^I\hat{\bar{q}}_G)\Delta\mathbf{b}_a && (3.49)\\
\delta^I\dot{\theta}_G &= -\left\lfloor \omega_m - \hat{\mathbf{b}}_\omega\times \right\rfloor \delta\theta - \mathbf{n}_\omega - \Delta b_\omega && (3.50)\\
\Delta\dot{\mathbf{b}}_\omega &= \mathbf{n}_{b_\omega} && (3.51)\\
\Delta\dot{\mathbf{b}}_a &= \mathbf{n}_{b_a} && (3.52)\\
\Delta^C\dot{\theta}_I &= \mathbf{0} && (3.53)\\
\Delta^I\mathbf{p}_C &= \mathbf{0} && (3.54)
\end{aligned}}$$

The continuous differential equations in error state representation can now be summarized to [4]:

$$\dot{\tilde{\mathbf{X}}} = \mathbf{F}_c\tilde{\mathbf{X}} + \mathbf{G}_c\mathbf{n} \tag{3.55}$$

where the noise vector is defined as $\mathbf{n} = [\mathbf{n}_a^T, \mathbf{n}_{b_a}^T, \mathbf{n}_\omega^T, \mathbf{n}_{b_\omega}^T]^T$. Assuming constant matrices $\mathbf{F}_c$ and $\mathbf{G}_c$ during the integration period $\Delta t = t_{k+1} - t_k$ the continuous system equations can be discretized using a Taylor series approximation [4, 8]:

$$\mathbf{F}_d = \exp(\mathbf{F}_c\Delta t) = \mathbf{I} + \mathbf{F_c}\Delta t + \frac{1}{2!}\mathbf{F}_c^2\Delta t^2 + H.O.T. \tag{3.56}$$

The discrete state propagation matrix $\mathbf{F}_d$ is found to be sparse with repetitive structure [4]:

$$\mathbf{F}_d = \begin{bmatrix} \mathbf{I}_3 & \Delta t & A & B & -\mathbf{C}^T({}^I\hat{\bar{q}}_G)\frac{\Delta t^2}{2} & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{C} & \mathbf{D} & -\mathbf{C}^T({}^I\hat{\bar{q}}_G)\Delta t & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{E} & \mathbf{F} & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \end{bmatrix} \tag{3.57}$$

Using the small angle approximation for which $|\omega| \to 0$ and applying l'Hopital's rule [4]

$$\mathbf{A} = -\mathbf{C}^T({}^I\hat{\bar{q}}_G)\left\lfloor \mathbf{a}_m - \hat{\mathbf{b}}_a \times \right\rfloor \left(\frac{\Delta t^2}{2!} - \frac{\Delta t^3}{3!}\lfloor \omega \times \rfloor + \frac{\Delta t^4}{4!}\lfloor \omega \times \rfloor^2 \right) \tag{3.58}$$

$$\mathbf{B} = -\mathbf{C}^T({}^I\hat{\bar{q}}_G)\left\lfloor \mathbf{a}_m - \hat{\mathbf{b}}_a \times \right\rfloor \left(-\frac{\Delta t^3}{3!} + \frac{\Delta t^4}{4!}\lfloor \omega \times \rfloor - \frac{\Delta t^5}{5!}\lfloor \omega \times \rfloor^2 \right) \tag{3.59}$$

$$\mathbf{C} = -\mathbf{C}^T({}^I\hat{\bar{q}}_G)\left\lfloor \mathbf{a}_m - \hat{\mathbf{b}}_a \times \right\rfloor \left(\Delta t - \frac{\Delta t^2}{2!} + \frac{\Delta t^3}{3!}\lfloor \omega \times \rfloor^2 \right) \tag{3.60}$$

$$\mathbf{D} = -\mathbf{A} \tag{3.61}$$

$$\mathbf{E} = \mathbf{I} - \Delta t \lfloor \omega \times \rfloor + \frac{\Delta t^2}{2!}\lfloor \omega \times \rfloor^2 \tag{3.62}$$

$$\mathbf{F} = -\Delta t + \frac{\Delta t^2}{2!}\lfloor \omega \times \rfloor - \frac{\Delta t^3}{3!}\lfloor \omega \times \rfloor^2 \tag{3.63}$$

The *continuous* time noise propagation matrix is

$$\mathbf{G}_c = \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ -\mathbf{C}^T({}^I\hat{\bar{q}}_G) & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & -\mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix} \tag{3.64}$$

The discrete time covariance matrix $\mathbf{Q}_d$ is given by:

$$\mathbf{Q}_d = \int_{\Delta t} \mathbf{F}_d(\tau)\mathbf{G}_c\mathbf{Q}_c\mathbf{G}_c^T\mathbf{F}_d(\tau)^T d\tau \tag{3.65}$$

with the continuous time system noise covariance matrix $\mathbf{Q}_c$

$$\mathbf{Q}_c = \begin{bmatrix} \sigma^2_{n_a} & 0 & 0 & 0 \\ 0 & \sigma^2_{n_{b_a}} & 0 & 0 \\ 0 & 0 & \sigma^2_{n_\omega} & 0 \\ 0 & 0 & 0 & \sigma^2_{n_{b_\omega}} \end{bmatrix} \tag{3.66}$$

Besides, the IMU-IMU correlation also the IMU-Camera cross-correlation and Camera-Camera auto-correlation need to be considered. The complete covariance matrix is given by

$$\mathbf{P}_{k+1|k} = \begin{bmatrix} \mathbf{P}_{II_{k+1|k}} & \mathbf{F}_d\mathbf{P}_{IC_{k|k}} \\ \mathbf{P}^T_{IC_{k|k}}\mathbf{F}_d^T & \mathbf{P}_{CC_{k|k}} \end{bmatrix} \tag{3.67}$$

where

$$\mathbf{P}_{II} : \text{IMU-IMU auto-correlation} \tag{3.68}$$

$$\mathbf{P}_{IC} : \text{IMU-Camera cross-correlation} \tag{3.69}$$

$$\mathbf{P}_{CC} : \text{Camera-Camera auto-correlation} \tag{3.70}$$

The propagation steps can be summarized as follows:

Propagation steps

1. Propagate the states
2. Calculate $\mathbf{F}_d$ and $\mathbf{Q}_d$
3. Propagate the state covariance matrix for the IMU-IMU auto-correlation:

$$\mathbf{P}_{II_{k+1|k}} = \mathbf{F}_d \mathbf{P}_{II_{k|k}} \mathbf{F}_d^T + \mathbf{Q}_d \tag{3.71}$$

4. Propagate the state covariance matrix for the IMU-Camera and Camera-IMU cross-correlation:

$$\mathbf{P}_{IC_{k+1|k}} = \mathbf{F}_d \mathbf{P}_{IC_{k|k}} \tag{3.72}$$

5. Simply clone the state covariance matrix for the Camera-Camera auto-correlation.

3.2 New image: State vector and state covariance update

Every time a new image is recorded, the current camera position as well as the camera attitude are inserted at the correct position of the state vector.[2]

3.2.1 Image update: state

The camera position in the global frame is computed from the camera-IMU transformation and the current position estimate:

$$\textbf{Position: } {}^G\hat{\mathbf{p}}_{C_i} = {}^G\hat{\mathbf{p}}_I + \mathbf{C}({}^G_I\hat{\bar{q}})^I\mathbf{p}_C \tag{3.73}$$

$$= {}^G\hat{\mathbf{p}}_I + \mathbf{C}^T({}^I_G\hat{\bar{q}})^I\mathbf{p}_C \tag{3.74}$$

The camera rotation from the global to the i-th camera frame is given by the camera-IMU rotation and the current attitude estimate:

$$\textbf{Attitude: } {}^{C_i}_G\hat{\bar{q}} = {}^C_I\bar{q} \otimes {}^I_G\hat{\bar{q}} \tag{3.75}$$

3.2.2 Image update: covariance

The state covariance matrix $\mathbf{P}_{k|k}$ needs to be updated based on the *error propagation law*. The error propagation law states how the uncertainties of the variables are propagated to a function constructed by these variables.

That is, a *linear* function f made up by a linear combination of $\mathbf{x}$

$$\mathbf{f} = \mathbf{A}\mathbf{x} \tag{3.76}$$

has the covariance matrix

$$\Sigma_f = A\Sigma_x A^T \tag{3.77}$$

If the variable combinations are *nonlinear*, the covariance matrix can be linearized by a first-order Taylor expansion:

$$\Sigma_f = J\Sigma_x J^T \tag{3.78}$$

[2] This is the basic strategy when exploring new environments; in hovering mode not every image is necessarily added to the EKF vector.

Based on this the state covariance matrix can be derived: The core states and earlier camera poses remain unchanged: they are only copied. For the other entries of the covariance matrix the corresponding jacobians need to be evaluated according to the error propagation law:

$$\mathbf{P}_{k|k} = \begin{bmatrix} \mathbf{I}_{21} \\ \mathbf{J} \end{bmatrix} \mathbf{P}_{k|k} \begin{bmatrix} \mathbf{I}_{21} \\ \mathbf{J} \end{bmatrix}^T \tag{3.79}$$

$$:= \tilde{\mathbf{J}} \mathbf{P}_{k|k} \tilde{\mathbf{J}}^T \tag{3.80}$$

where $\tilde{\mathbf{J}}$ is [3]

$$\tilde{\mathbf{J}} = \left[\begin{array}{cccccccc|ccc|ccc}
\overbrace{\mathbf{I}_3}^{{}^G p_I} & \overbrace{\mathbf{0}_3}^{v} & \overbrace{\mathbf{0}_3}^{{}^G q_I} & \overbrace{\mathbf{0}_3}^{{}^C q_I} & \overbrace{\mathbf{0}_3}^{{}^C p_I} & \overbrace{\mathbf{0}_3}^{b_\omega} & \overbrace{\mathbf{0}_3}^{b_a} & \overbrace{\mathbf{0}_3}^{{}^G p_{C_1}} & \cdots & \overbrace{\mathbf{0}_3}^{{}^G p_{C_N}} & \overbrace{\mathbf{0}_3}^{{}^{C_1} q_G} & \cdots & \overbrace{\mathbf{0}_3}^{{}^{C_N} q_G} \\
\mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 \\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 \\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 \\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 \\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 \\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 \\
\hline
\nabla_{{}^G p_I}({}^G\hat{\mathbf{p}}_{C_1}) & \mathbf{0}_3 & \nabla_{\delta\theta_I}({}^G\hat{\mathbf{p}}_{C_1}) & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 \\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\
\hline
\mathbf{0}_3 & \mathbf{0}_3 & \nabla_{\delta\theta_I}({}^{C_1}_{G}\hat{q}) & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{I}_3 & \cdots & \mathbf{0}_3 \\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3
\end{array}\right] \tag{3.81}$$

for the first camera pose in the state vector.

Camera position jacobians

Recall the equation for the camera position

$${}^G\mathbf{p}_C = {}^G\mathbf{p}_I + \mathbf{C}({}^G_I\bar{q})\,{}^I\mathbf{p}_C \tag{3.82}$$

which can be expressed in error term notation as

$$\Delta^G\mathbf{p}_C = {}^G\hat{\mathbf{p}}_I + \Delta^G\mathbf{p}_I + \hat{\mathbf{R}}(\mathbf{I}_3 + \lfloor\delta\theta\times\rfloor)({}^I\hat{\mathbf{p}}_C + \Delta^I\mathbf{p}_C) \tag{3.83}$$

The jacobian with respect to the position estimate is then given by

$$\frac{\partial \Delta^G\mathbf{p}_C}{\partial \Delta^G\mathbf{p}_I} = \frac{\partial}{\partial \Delta^G\mathbf{p}_I}({}^G\hat{\mathbf{p}}_I + \Delta^G\mathbf{p}_I + \hat{\mathbf{R}}(\mathbf{I}_3 + \lfloor\delta\theta\times\rfloor)({}^I\hat{\mathbf{p}}_C + \Delta^I\mathbf{p}_C)) \tag{3.84}$$

$$= \frac{\partial}{\partial \Delta^G\mathbf{p}_I}(\Delta^G\mathbf{p}_I) \tag{3.85}$$

$$= \mathbf{I}_3 \tag{3.86}$$

[3] $\delta\theta_I$ is the small angle error notation for ${}^Gq_I$

The jacobian with respect to the attitude estimate $\delta\theta$ can be derived as

$$\begin{aligned}
\frac{\partial \Delta^G \mathbf{p}_C}{\partial \delta\theta} =& \frac{\partial}{\partial \delta\theta}({}^G\hat{\mathbf{p}}_I + \Delta^G \mathbf{p}_I + \hat{\mathbf{R}}(\mathbf{I}_3 + \lfloor \delta\theta \times \rfloor)({}^I\hat{\mathbf{p}}_C + \Delta^I \mathbf{p}_C)) && (3.87)\\
=& \frac{\partial}{\partial \delta\theta}({}^G\hat{\mathbf{p}}_I + \Delta^G \mathbf{p}_I + \hat{\mathbf{R}}{}^I\hat{\mathbf{p}}_C + \hat{\mathbf{R}}\Delta^I \mathbf{p}_C && (3.88)\\
&+ \hat{\mathbf{R}} \lfloor \delta\theta \times \rfloor {}^I\hat{\mathbf{p}}_C + \hat{\mathbf{R}} \lfloor \delta\theta \times \rfloor \Delta^I \mathbf{p}_C) && (3.89)\\
=& \frac{\partial}{\partial \delta\theta}(\hat{\mathbf{R}} \lfloor \delta\theta \times \rfloor {}^I\hat{\mathbf{p}}_C) && (3.90)\\
=& -\frac{\partial}{\partial \delta\theta}(\hat{\mathbf{R}} \left\lfloor {}^I\hat{\mathbf{p}}_C \times \right\rfloor \delta\theta) && (3.91)\\
=& -\hat{\mathbf{R}} \left\lfloor {}^I\hat{\mathbf{p}}_C \times \right\rfloor && (3.92)\\
=& -\hat{\mathbf{C}}({}^G_I\bar{q}) \left\lfloor {}^I\hat{\mathbf{p}}_C \times \right\rfloor && (3.93)\\
=& -\hat{\mathbf{C}}_q \left\lfloor {}^I\hat{\mathbf{p}}_C \times \right\rfloor && (3.94)
\end{aligned}$$

Camera attitude jacobians

Recall the equation for the camera attitude

$$ {}^C_I\bar{q} \otimes {}^I_G\hat{\bar{q}} \tag{3.95}$$

To derive the jacobian with respect to the the camera attitude $\delta\theta$ the error quaternion notation is used:

$$\begin{aligned}
\delta\theta \overset{1}{=}& {}^C\bar{q}_G \otimes ({}^C\hat{\bar{q}}_G)^{-1} && (3.96)\\
=& [{}^C\bar{q}_I \otimes {}^I\bar{q}_G] \otimes [{}^C\hat{\bar{q}}_I \otimes {}^I\hat{\bar{q}}_G]^{-1} && (3.97)\\
=& {}^C\bar{q}_I \otimes \delta\theta \otimes {}^I\hat{\bar{q}}_G \otimes ({}^I\hat{\bar{q}}_G)^{-1} \otimes ({}^C\hat{\bar{q}}_I)^{-1} && (3.98)\\
\overset{2}{=}& {}^C\bar{q}_I \otimes \delta\theta \otimes ({}^C\hat{\bar{q}}_I)^{-1} && (3.99)\\
=& \mathbf{C}({}^C_I\bar{q}) && (3.100)
\end{aligned}$$

For the first camera pose in the state vector this results in the following augmentation matrix $\tilde{\mathbf{J}}$:

$$\tilde{\mathbf{J}} = \left[\begin{array}{cccccccc|ccc|ccc}
\overbrace{\mathbf{I}_3}^{{}^G p_I} & \overbrace{\mathbf{0}_3}^{{}^G v_I} & \overbrace{\mathbf{0}_3}^{{}^G q_I} & \overbrace{\mathbf{0}_3}^{b_\omega} & \overbrace{\mathbf{0}_3}^{b_a} & \overbrace{\mathbf{0}_3}^{{}^C q_I} & \overbrace{\mathbf{0}_3}^{{}^C p_I} & \overbrace{\mathbf{0}_3}^{{}^G p_{C_1}} & \cdots & \overbrace{\mathbf{0}_3}^{{}^G p_{C_N}} & \overbrace{\mathbf{0}_3}^{{}^{C_1} q_G} & \cdots & \overbrace{\mathbf{0}_3}^{{}^{C_N} q_G}\\
\mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3\\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3\\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3\\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3\\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3\\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3\\
\hline
\mathbf{I}_3 & \mathbf{0}_3 & -\hat{\mathbf{C}}({}^G_I\bar{q}) \left\lfloor {}^I\mathbf{p}_C \times \right\rfloor & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3\\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots\\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\\
\hline
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{C}({}^{C_1}_I\bar{q}) & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{I}_3 & \cdots & \mathbf{0}_3\\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3 & \mathbf{0}_3 & \cdots & \mathbf{0}_3\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots\\
\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3
\end{array}\right] \tag{3.101}$$

[1] $\tilde{z} = z - \hat{z}$ in error state representation: $\delta\theta = \bar{q} \otimes \hat{\bar{q}}^{-1}$

[2] See for example the work of Trawny et al. [8].

[6] Analogue to derivation of jacobian J_3, see equation 3.125 and following.

3.3 Measurement model for opportunistic features

The observation of feature j in camera i is modeled by

$$\mathbf{z}_i^{(j)} = \frac{1}{{}^{C_i}Z_j} \begin{bmatrix} {}^{C_i}X_j \\ {}^{C_i}Y_j \end{bmatrix} + n_i^{(j)} \tag{3.102}$$

The position of feature j seen from the i-th camera frame is

$${}^{C_i}\mathbf{p}_{f_j} = \begin{bmatrix} {}^{C_i}X_j \\ {}^{C_i}Y_j \\ {}^{C_i}Z_j \end{bmatrix} = \mathbf{C}({}^{C_i}_{G}\bar{q})({}^{G}\mathbf{p}_{f_j} - {}^{G}\mathbf{p}_{C_i}) \tag{3.103}$$

Thus, the measurement $\mathbf{z}_i^{(j)}$ is a function of the i-th camera position expressed in the global frame, the quaternion rotation from the global to the i-th camera frame and the j-th feature position expressed in the global frame. This can be expressed in terms of the measurement function h

$$\mathbf{z}_i^{(j)} = h({}^{G}\mathbf{p}_{C_i}, {}^{C_i}_{G}\bar{q}, {}^{G}\mathbf{p}_{f_j}) \tag{3.104}$$

The measurement residual for feature j in camera i is defined as the difference between the measurement and the expected observation

$$\mathbf{r}_i^{(j)} = \mathbf{z}_i^{(j)} - \hat{\mathbf{z}}_i^{(j)} \tag{3.105}$$

Based on equation 3.159 the residual can be approximated by linearising about the camera pose estimate and feature position estimate:

$$\mathbf{r}_i^{(j)} \approx \mathbf{H}_{\mathbf{X}_i}^{(j)}\tilde{\mathbf{X}} + \mathbf{H}_{f_i}^{(j)}{}^{G}\tilde{\mathbf{p}}_{f_j} + \mathbf{n}_i^{(j)} \tag{3.106}$$

The jacobian of the measurement $\mathbf{z}$ with respect to the state is given by:

$$\mathbf{H}_{\mathbf{X}_i}^{(j)} = \left[\mathbf{H}_{core,i}^{(j)} \;\middle|\; \mathbf{H}_{camPose,i}^{(j)}\right]_{2\times(15+6N)} \tag{3.107}$$

In the following sections the entries of the measurement matrix $\mathbf{H}_{\mathbf{X}_i}^{(j)}$ are derived:

3.3.1 Measurement matrix: core states

For simplicity, in this section the Camera-IMU translation and rotation are not included in the state vector. For estimating the Camera-IMU transformation see section 3.6.

The measurement matrix for the core states results in

$$\mathbf{H}_{core,i}^{(j)} = \left[\frac{\partial}{\partial \Delta^{G}\mathbf{p}_I} \quad \frac{\partial}{\partial \Delta^{G}\mathbf{v}_I} \quad \frac{\partial}{\partial \delta\theta_I} \quad \frac{\partial}{\partial \Delta b_\omega} \quad \frac{\partial}{\partial \Delta b_a}\right]_{2\times 15} \Delta\mathbf{z}_i^{(j)} \tag{3.108}$$

$$= \left[\mathbf{0}_{2\times 15}\right] \tag{3.109}$$

3.3.2 Measurement matrix: camera pose states

The measurement matrix with respect to the i-th camera pose is defined as

$$\mathbf{H}_{camPose,i}^{(j)} = \left[\underbrace{\mathbf{0}_2 \quad \cdots \quad \frac{\partial \Delta\mathbf{z}_i^{(j)}}{\partial \Delta^{G}p_C} \quad \cdots \quad \mathbf{0}_2}_{\text{camera positions}} \;\middle|\; \underbrace{\mathbf{0}_2 \quad \cdots \quad \frac{\partial \Delta\mathbf{z}_i^{(j)}}{\partial {}^{G}\theta_I} \quad \cdots \quad \mathbf{0}_2}_{\text{camera attitudes}}\right]_{2\times 6N} \tag{3.110}$$

Jacobian of measurement equation wrt. global camera position

The derivative with respect to the camera pose can be calculated using the chain rule:

$$\left.\frac{\partial \mathbf{z}_i^{(j)}}{\partial {}^G p_{C_i}}\right|_{{}^G p_{C_i} = {}^G \hat{p}_{C_i}} = \underbrace{\left.\frac{\partial \mathbf{z}_i^{(j)}}{\partial {}^{C_i} p_{f_j}}\right|_{{}^{C_i} p_{f_j} = {}^{C_i} \hat{p}_{f_j}}}_{J_1} \cdot \underbrace{\left.\frac{\partial {}^{C_i} p_{f_j}}{\partial {}^G p_{C_i}}\right|_{{}^G p_{C_i} = {}^G \hat{p}_{C_i}}}_{J_2} \tag{3.111}$$

Jacobian J_1 It is derived with respect to ${}^G p_{C_i} = \begin{bmatrix} {}^{C_i}X_j & {}^{C_i}Y_j & {}^{C_i}Z_j \end{bmatrix}^T$:

$$\begin{aligned}
J_1 &:= \left.\frac{\partial \mathbf{z}_i^{(j)}}{\partial {}^G p_{C_i}}\right|_{{}^G p_{C_i} = {}^G \hat{p}_{C_i}} && (3.112)\\
&= \frac{\partial}{\partial {}^G p_{C_i}} \left(\frac{1}{{}^{C_i}Z_j} \begin{bmatrix} {}^{C_i}X_j \\ {}^{C_i}Y_j \end{bmatrix} + n_i^{(j)} \right) \Bigg|_{{}^G p_{C_i} = {}^G \hat{p}_{C_i}} && (3.113)\\
&= \begin{bmatrix} \frac{\partial\left(\frac{{}^{C_i}X_j}{{}^{C_i}Z_j}\right)}{\partial\left({}^{C_i}X_j\right)} & \frac{\partial\left(\frac{{}^{C_i}X_j}{{}^{C_i}Z_j}\right)}{\partial\left({}^{C_i}Y_j\right)} & \frac{\partial\left(\frac{{}^{C_i}X_j}{{}^{C_i}Z_j}\right)}{\partial\left({}^{C_i}Z_j\right)} \\ \frac{\partial\left(\frac{{}^{C_i}Y_j}{{}^{C_i}Z_j}\right)}{\partial\left({}^{C_i}X_j\right)} & \frac{\partial\left(\frac{{}^{C_i}Y_j}{{}^{C_i}Z_j}\right)}{\partial\left({}^{C_i}Y_j\right)} & \frac{\partial\left(\frac{{}^{C_i}Y_j}{{}^{C_i}Z_j}\right)}{\partial\left({}^{C_i}Z_j\right)} \end{bmatrix} \Bigg|_{{}^G p_{C_i} = {}^G \hat{p}_{C_i}} && (3.114)\\
&= \begin{bmatrix} \frac{1}{{}^{C_i}Z_j} & 0 & \frac{{}^{C_i}X_j}{({}^{C_i}Z_j)^2} \\ 0 & \frac{1}{{}^{C_i}Z_j} & \frac{{}^{C_i}Y_j}{({}^{C_i}Z_j)^2} \end{bmatrix} \Bigg|_{{}^G p_{C_i} = {}^G \hat{p}_{C_i}} && (3.115)\\
&= \frac{1}{{}^{C_i}\hat{Z}_j} \begin{bmatrix} \mathbf{I_2} & \hat{\mathbf{z}}_i^{(j)} \end{bmatrix} && (3.116)
\end{aligned}$$

Jacobian J_2 Recall equation 3.103 which expresses the feature position in camera coordinates in terms of the feature and camera position in global coordinates:

$${}^{C_i}\mathbf{p}_{f_j} = \begin{bmatrix} {}^{C_i}X_j \\ {}^{C_i}Y_j \\ {}^{C_i}Z_j \end{bmatrix} = \mathbf{C}({}^{C_i}_{G}\bar{q})({}^G\mathbf{p}_{f_j} - {}^G\mathbf{p}_{C_i}) \tag{3.117}$$

Expressing this equation in error terms yields

$$\Delta {}^{C_i}\mathbf{p}_{f_j} = \hat{\mathbf{R}}(\mathbf{I}_3 + \lfloor \delta\theta \times \rfloor)({}^G\hat{\mathbf{p}}_{f_j} + \Delta {}^G\mathbf{p}_{f_j} - ({}^G\hat{\mathbf{p}}_{C_i} + \Delta {}^G\mathbf{p}_{C_i})) \tag{3.118}$$

The jacobian with respect to the camera position in global coordinates is defined as J_2:

$$\begin{aligned}
J_2 &:= \frac{\partial \Delta {}^{C_i}\mathbf{p}_{f_j}}{\partial \Delta {}^G\mathbf{p}_{C_i}} && (3.119)\\
&= \frac{\partial}{\partial \Delta {}^G\mathbf{p}_{C_i}} (\hat{\mathbf{R}}(\mathbf{I}_3 + \lfloor \delta\theta \times \rfloor)({}^G\hat{\mathbf{p}}_{f_j} + \Delta {}^G\mathbf{p}_{f_j} - ({}^G\hat{\mathbf{p}}_{C_i} + \Delta {}^G\mathbf{p}_{C_i}))) && (3.120)\\
&= -\hat{\mathbf{R}}(\mathbf{I}_3 + \lfloor \delta\theta \times \rfloor) && (3.121)\\
&= -\hat{\mathbf{R}} && (3.122)\\
&= -\hat{\mathbf{C}}({}^{C_i}_{G}\bar{q}) && (3.123)
\end{aligned}$$

Jacobian of measurement equation wrt. global camera rotation Next the derivative with respect to the quaternion rotation from the camera to the global frame is calculated. The derivative is taken with respect to the error angle vector $\delta\theta$ due to the small angle representation used in the EKF state:

$$\left.\frac{\partial \mathbf{z}_i^{(j)}}{\partial \delta\theta}\right|_{{}^G p_{C_i} = {}^G\hat{p}_{C_i}} = \underbrace{\left.\frac{\partial \mathbf{z}_i^{(j)}}{\partial {}^{C_i}p_{f_j}}\right|_{{}^{C_i}p_{f_j} = {}^{C_i}\hat{p}_{f_j}}}_{J_1} \cdot \underbrace{\frac{\partial {}^{C_i}p_{f_j}}{\partial \delta\theta}}_{J_3} \tag{3.124}$$

The jacobian J_1 has been calculated above. Recall the identities for error quaternions and small angle approximations from appendix A.
With these results, the jacobian J_3 with respect to the error angle vector can be calculated:

$$\begin{aligned}
J_3 :=& \frac{\partial \Delta^{C_i}\mathbf{p}_{f_j}}{\partial \delta\theta} && (3.125)\\
=& \frac{\partial}{\partial \delta\theta}(\hat{\mathbf{R}}(\mathbf{I}_3 + \lfloor \delta\theta\times \rfloor)({}^G\hat{\mathbf{p}}_{f_j} + \Delta^G\mathbf{p}_{f_j} - ({}^G\hat{\mathbf{p}}_{C_i} + \Delta^G\mathbf{p}_{C_i}))) && (3.126)\\
=& \frac{\partial}{\partial \delta\theta}(\hat{\mathbf{R}}\mathbf{I}_3{}^G\hat{\mathbf{p}}_{f_j} + \hat{\mathbf{R}}\mathbf{I}_3\Delta^G\mathbf{p}_{f_j} + \hat{\mathbf{R}}\mathbf{I}_3{}^G\hat{\mathbf{p}}_{C_i} + \hat{\mathbf{R}}\mathbf{I}_3\Delta^G\mathbf{p}_{C_i} + && (3.127)\\
& \hat{\mathbf{R}}\lfloor \delta\theta\times \rfloor {}^G\hat{\mathbf{p}}_{f_j} + \hat{\mathbf{R}}\lfloor \delta\theta\times \rfloor \Delta^G\mathbf{p}_{f_j} - \hat{\mathbf{R}}\lfloor \delta\theta\times \rfloor {}^G\hat{\mathbf{p}}_{C_i} - && (3.128)\\
& \hat{\mathbf{R}}\lfloor \delta\theta\times \rfloor \Delta^G\mathbf{p}_{C_i}) && (3.129)\\
=& \frac{\partial}{\partial \delta\theta}(\hat{\mathbf{R}}\lfloor \delta\theta\times \rfloor ({}^G\hat{\mathbf{p}}_{f_j} - {}^G\hat{\mathbf{p}}_{C_i}) && (3.130)\\
=& -\frac{\partial}{\partial \delta\theta}(\hat{\mathbf{R}}\left\lfloor {}^G\hat{\mathbf{p}}_{f_j} - {}^G\hat{\mathbf{p}}_{C_i}\times \right\rfloor \delta\theta) && (3.131)\\
=& -\hat{\mathbf{R}}\left\lfloor {}^G\hat{\mathbf{p}}_{f_j} - {}^G\hat{\mathbf{p}}_{C_i}\times \right\rfloor && (3.132)\\
=& -\hat{\mathbf{C}}({}^{C_i}_{G}\bar{q})\left\lfloor {}^G\hat{\mathbf{p}}_{f_j} - {}^G\hat{\mathbf{p}}_{C_i}\times \right\rfloor && (3.133)
\end{aligned}$$

Summarizing, the measurement matrix expressing the observation of feature j in camera frame i is given by:

$$\mathbf{H}_{X_i}^{(j)} = \Big[\overbrace{\mathbf{0}_{2\times 21}}^{\mathbf{H}_{core,i}^{(j)}} \;\Big|\; \overbrace{\underbrace{\mathbf{0}_{2\times 3}}_{1} \;\; \underbrace{\cdots}_{2\ldots i-1} \;\; \underbrace{J_{1,i}^{(j)} J_{2,i}^{(j)}}_{i} \;\; \underbrace{\cdots}_{i+1\ldots N}}^{\text{Camera positions}} \;\Big|\; \overbrace{\underbrace{\mathbf{0}_{2\times 3}}_{1} \;\; \underbrace{\cdots}_{2\ldots i-1} \;\; \underbrace{J_{1,i}^{(j)} J_{3,i}^{(j)}}_{i} \;\; \underbrace{\cdots}_{i+1\ldots N}}^{\text{Camera attitudes}} \Big]$$

Note that the measurement matrices of *all* the observations of feature j are simply stacked to obtain $\mathbf{H}_{\mathbf{X}}^{(j)}$.

3.3.3 Feature constraints

Reconsider equation 3.106 of the residual for feature j:

$$\mathbf{r}^{(j)} \approx \mathbf{H}_{\mathbf{X}}^{(j)}\tilde{\mathbf{X}} + \mathbf{H}_f^{(j)}{}^G\tilde{\mathbf{p}}_{f_j} + \mathbf{n}^{(j)} \tag{3.134}$$

The error in the position estimate of feature j is given by

$${}^G\tilde{\mathbf{p}}_{f_j} = {}^G\mathbf{p}_{f_j} - {}^G\hat{\mathbf{p}}_{f_j} \tag{3.135}$$

The feature position estimate ${}^G\hat{\mathbf{p}}_{f_j}$ is triangulated using the camera poses of the state vector. Consequently, ${}^G\tilde{\mathbf{p}}_{f_j}$ is correlated with $\tilde{\mathbf{X}}$ which violates the Kalman filter assumption that the residual vector depends *linearly* on the state error $\tilde{\mathbf{X}}$.

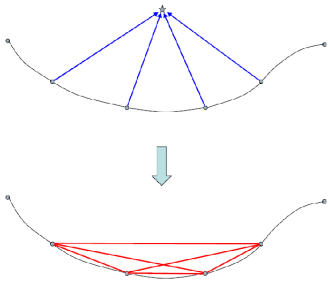


Figure 3.2: Constraint of one static feature being observed by multiple camera frames (top) is transformed to the constraint of the camera poses only (bottom). (Image: [9])

Anastasios Mourikis' idea to overcome this problem is visualized in image 3.2.
The constraint of a static feature being observed from multiple camera poses is transformed into a constraint involving only the camera poses.

Mathematically, this can be achieved by left-multiplying the residual equation by the left nullspace of $\mathbf{H}_f$ which is denoted as $\mathbf{A}^T$.
Multiplying equation 3.106 by $\mathbf{A}^T$ results in:

$$\mathbf{r}_o^{(j)} = \mathbf{A}^T(\mathbf{z}^{(j)} - \hat{\mathbf{z}}^{(j)}) \tag{3.136}$$

$$\approx \mathbf{A}^T\mathbf{H}_{\mathbf{X}}^{(j)}\tilde{\mathbf{X}} + \mathbf{A}^T\mathbf{H}_f^{(j)G}\tilde{\mathbf{p}}_{f_j} + \mathbf{A}^T\mathbf{n}^{(j)} \tag{3.137}$$

$$= \mathbf{A}^T\mathbf{H}_{\mathbf{X}}^{(j)}\tilde{\mathbf{X}} + \mathbf{A}^T\mathbf{n}^{(j)} \tag{3.138}$$

$$= \mathbf{H}_o^{(j)}\tilde{\mathbf{X}}^{(j)} + \mathbf{n}_o^{(j)} \tag{3.139}$$

where $\mathbf{H}_o^{(j)}$ is the projected measurement matrix and $\mathbf{n}_o^{(j)}$ is the projected measurement noise for feature j.

The measurement covariance matrix of the noise vector $\mathbf{n}_o^{(j)}$ is given by

$$\mathbf{R}_o := \mathbf{E}[\mathbf{n}_o^{(j)}\mathbf{n}_o^{(j)T}] \tag{3.140}$$

$$= \sigma_{image}^2\mathbf{A}^T\mathbf{A} \tag{3.141}$$

$$= \sigma_{image}^2\mathbf{I}_{2M_j-3} \tag{3.142}$$

where the last equal sign is due to $\mathbf{A}$ being unitary. [4]

Multiple feature constraints

Assume that at a certain time step, L features need to be processed by the update model:

- Calculate the projected residual $\mathbf{r}_o^{(j)}$, with $j = 1, ...L$, that is for each feature
- Calculate the projected measurement matrix $\mathbf{H}_o^{(j)}$, with $j = 1, ...L$, that is for each feature
- $\mathbf{r}_o$ is obtained by simply stacking all feature residuals $\mathbf{r}_o^{(j)}$, with $j = 1, ...L$
- $\mathbf{H}_X$ is obtained by simply stacking all feature residuals $\mathbf{H}_o^{(j)}$, with $j = 1, ...L$

[4] Refer to Mourikis et al. [5]

- Assuming that the feature measurements are statistically independent[5] implies that also the noise vectors $\mathbf{n}_o^{(j)}$ are not mutually correlated. The off-diagonal entries of the projected noise covariance matrix are therefore zero:

$$\mathbf{R}_o = \sigma_{image}^2 \mathbf{I}_d \tag{3.143}$$

  where $d = \sum_{j=1}^{L}(2M_j - 3)$.

Theoretically, the standard Kalman filter equations could be applied now:

$$\mathbf{K} = \mathbf{P}_{k+1|k}\mathbf{H}_X^T(\mathbf{H}_X\mathbf{P}_{k+1|k}\mathbf{H}_X^T + \mathbf{R}_o)^{-1} \tag{3.144}$$

$$\Delta\mathbf{X} = \mathbf{K}\mathbf{r}_n \tag{3.145}$$

$$\mathbf{P}_{k+1|k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H}_X)\mathbf{P}_{k+1|k}(\mathbf{I} - \mathbf{K}\mathbf{H}_X)^T + \mathbf{K}\mathbf{R}_o\mathbf{K}^T \tag{3.146}$$

However, the dimension $d \times 1$ of the residual vector introduces a high computational complexity for equations 3.144 to 3.146. Section 3.3.5 describes how the computational effort can be decreased.

3.3.4 Outlier rejection

Before the correction vector $\Delta\mathbf{X}$ is calculated using the Kalman filter equations (see section 3.3.6) each feature is checked individually if it is within the validity gap or not. To detect outliers, the Mahalanobis distance γ

$$\mathbf{S} = \mathbf{H}_o^{(j)}\mathbf{P}_{k+1|k}(\mathbf{H}_o^{(j)})^T + \mathbf{R}_o \tag{3.147}$$

$$= \mathbf{H}_o^{(j)}\mathbf{P}_{k+1|k}(\mathbf{H}_o^{(j)})^T + \sigma_{image}^2\mathbf{I}_{2M_j-3} \tag{3.148}$$

$$\gamma = (\mathbf{r}_o^{(j)})^T\mathbf{S}^{-1}\mathbf{r}_o^{(j)} \tag{3.149}$$

is calculated and compared to the 95-th percentile of the χ^2-distribution. The number of degrees of freedoms equals the dimension of the residual $\mathbf{r}_o^{(j)}$ which is $2M^{(j)} - 3$ where $M^{(j)}$ is the number of observations for feature j. If γ is below this "dynamic threshold" it is an inlier otherwise an outlier.

3.3.5 Implementation remarks

In order to reduce the computational complexity of the filter update, A. Mourikis et al. propose to apply the QR decomposition on $\mathbf{H}_X$:

$$\mathbf{H}_X = \begin{bmatrix}\mathbf{Q}_1 & \mathbf{Q}_2\end{bmatrix}\begin{bmatrix}\mathbf{T}_H \\ \mathbf{0}\end{bmatrix} \tag{3.150}$$

Following the derivation in [5], the final residual vector and measurement covariance matrix used in the Kalman filter update are as follows:

$$\mathbf{r}_n = \mathbf{Q}_1^T\mathbf{r}_o \tag{3.151}$$

$$\mathbf{R}_n = \mathbf{Q}_1^T\mathbf{R}_o\mathbf{Q}_1^T = \sigma_{image}^2\mathbf{I}_r \tag{3.152}$$

The residual vector $\mathbf{r}_n$ as well as the measurement matrix $\mathbf{T}_H$ can be computed using Givens rotations.

[5] This assumption is violated if only features from e.g. a line are extracted.

3.3.6 Kalman Filter update equations

The standard Kalman Filter update equations are used to calculate the Kalman gain $\mathbf{K}$, the correction vector $\Delta\mathbf{X}$ and the a-posteriori state covariance $\mathbf{P}_{k+1|k+1}$. The Kalman gain $\mathbf{K}$ is given by:

$$\mathbf{K} = \mathbf{P}_{k+1|k}\mathbf{T}_H^T(\mathbf{T}_H\mathbf{P}_{k+1|k}\mathbf{T}_H^T + \mathbf{R}_n)^{-1} \tag{3.153}$$

The multiplication of the Kalman gain $\mathbf{K}$ and the residuals $\mathbf{r}_n$ yields the correction vector $\Delta\mathbf{X}$. The direction of the correction is controlled by the residual - the Kalman gain influences only the magnitude.

$$\Delta\mathbf{X} = \mathbf{K}\mathbf{r}_n \tag{3.154}$$

Finally, the a-posteriori state covariance needs to be updated:

$$\mathbf{P}_{k+1|k+1} = (\mathbf{I}_{21+6N} - \mathbf{K}\mathbf{T}_H)\mathbf{P}_{k+1|k}(\mathbf{I}_{21+6N} - \mathbf{K}\mathbf{T}_H)^T + \mathbf{K}\mathbf{R}_n\mathbf{K}^T \tag{3.155}$$

3.3.7 State correction

The IMU position, velocity, sensor biases, IMU-Camera translation and the N camera positions are corrected according to the standard additive error-state notation:

$$\tilde{x} = x - \hat{x} \tag{3.156}$$

The IMU rotation, IMU-Camera rotation and the N camera rotations are updated according to the multiplicative error-state notation: [6]

$$\bar{q} = \hat{\bar{q}} \otimes \delta\bar{q} \tag{3.157}$$

3.4 Measurement model for known landmarks

Recall equation 3.103 which is the feature position of the j-th feature in the i-th camera frame:

$$^{C_i}\mathbf{p}_{f_j} = \begin{bmatrix} ^{C_i}X_j \\ ^{C_i}Y_j \\ ^{C_i}Z_j \end{bmatrix} = \mathbf{C}(^{C_i}_{G}\bar{q})(^{G}\mathbf{p}_{f_j} - {}^{G}\mathbf{p}_{C_i}) \tag{3.158}$$

When using opportunistic features, that is features whose positions are not known apriori, then the position of the j-th feature in the global frame $^{G}\mathbf{p}_{f_j}$ is estimated by multi-view triangulation using the camera poses and pixel values as inputs.

In this section, it is assumed that the feature positions $^{G}\mathbf{p}_{f_j}$ are known positions of markers, e.g. APRIL tags that are tracked by a tag detection algorithm.
Since the position of the landmark is known, the measurement $\mathbf{z}_i^{(j)}$ is only a function of the i-th camera position expressed in the global frame and the quaternion rotation from the global to the i-th camera frame:[7,8]

$$\mathbf{z}_i^{(j)} = h(^{G}\mathbf{p}_{C_i}, ^{C_i}_{G}\bar{q}) \tag{3.159}$$

[6] N.b. the different notation used in contrast to the one defined in A. Mourikis' work [5]
[7] Compare to equation 3.159
[8] Adapted from the work of Mourikis et al. [10]

The residual can then be approximated by linearising about the camera pose estimate only [9,10]

$$\mathbf{r}_i^{(j)} = \mathbf{z}_i^{(j)} - \hat{\mathbf{z}}_i^{(j)} \approx \mathbf{H}_{\mathbf{X}_i}^{(j)} \tilde{\mathbf{X}} + \mathbf{n}_i^{(j)} \tag{3.160}$$

where $\mathbf{H}_{X_i}^{(j)}$ is given by

$$\mathbf{H}_{X_i}^{(j)} = \left[\begin{array}{c|ccccc|ccccc} \mathbf{0}_{2\times 15} & \mathbf{0}_{2\times 3} & \cdots & J_{1,i}^{(j)} J_{2,i}^{(j)} & \cdots & & \mathbf{0}_{2\times 3} & \cdots & J_{1,i}^{(j)} J_{3,i}^{(j)} & \cdots & \end{array} \right]$$

This is the equivalent measurement matrix as defined for the opportunistic update model; the main difference is that the residual equation is *not* multiplied by the left nullspace of $\mathbf{H}_f^{(j)}$ and thus the measurement matrix can directly be used for the Kalman filter update.
The measurement noise covariance matrix $\mathbf{R}$ is given by

$$\mathbf{R} = \sigma^2 \mathbf{I}_{2M_j} \tag{3.161}$$

analogue to the opportunistic update model.

Multiple known landmark constraints

Assume that at a certain time step, L known landmarks need to be processed by the update model:

- Stack the residuals $\mathbf{r}_i^{(j)}$ for all observations and all landmarks to obtain $\mathbf{r}$.
- Stack the measurement matrices $\mathbf{H}_{X_i}^{(j)}$ for all observations and all landmarks to obtain $\mathbf{H}_X$.
- Calculate $\mathbf{R} = \sigma_{image}^2 \mathbf{I}_\zeta$ with $\zeta = \sum_{j=1}^{L} (2M_j - 3)$
- Use the residual $\mathbf{r}$, the measurement matrix $\mathbf{H}_X$ and the noise covariance matrix $\mathbf{R}$ as input for the Kalman filter update equations.

3.5 Measurement model for opportunistic features and known landmarks

The measurement model for opportunistic features and the measurement model for known landmarks can be combined to use both kinds of features in parallel[11]. The procedure is outlined as follows:

- Stack the residuals of the opportunistic features $\mathbf{r}_{OF}$ and of the known landmarks $\mathbf{r}_{KL}$.
- Stack the residuals of the opportunistic features $\mathbf{H}_{X_{OF}}$ and of the known landmarks $\mathbf{H}_{X_{KL}}$.
- Calculate the measurement noise covariance matrix $\mathbf{R} = \sigma_{image}^2 \mathbf{I}_\eta$ where η is the dimension of the stacked residual.
- Reduce the complexity of the update by employing the QR decomposition as explained in section 3.3.5.
- Use the reduced measurement matrix $\mathbf{T}_H$, reduced residual $\mathbf{r}_n$ and reduced measurement noise covariance matrix $\mathbf{R}_n$ as input for the standard Kalman filter update equations.

[9] See Mourkis' work [10]
[10] Compare to equation 3.106
[11] As proposed by Mourikis et al. [10]

3.6 Measurement model including CAM-IMU transformation

In this section, the EKF state vector is augmented with the camera-IMU transformation. The corresponding measurement matrix is then given by:

$$\mathbf{H}^{(j)}_{core,i} = \begin{bmatrix} \frac{\partial}{\partial \Delta^G \mathbf{p}_I} & \frac{\partial}{\partial \Delta^G \mathbf{v}_I} & \frac{\partial}{\partial \delta\theta_I} & \frac{\partial}{\partial \Delta b_\omega} & \frac{\partial}{\partial \Delta b_a} & \frac{\partial}{\partial \delta^C \theta_I} & \frac{\partial}{\partial \Delta^I \mathbf{p}_C} \end{bmatrix}_{2\times 21} \Delta \mathbf{z}^{(j)}_i \tag{3.162}$$

$$= \begin{bmatrix} \mathbf{0}_{2\times 15} & \frac{\partial}{\partial \delta^C \theta_I} & \frac{\partial}{\partial \Delta^I \mathbf{p}_C} \end{bmatrix}_{2\times 21} \tag{3.163}$$

General approach: Derivative chain law The derivative of the measurement equation with respect to the camera-IMU translation is given by

$$\left.\frac{\partial \mathbf{z}^{(j)}_i}{\partial {}^I p_C}\right|_{{}^I p_C = {}^I \hat{p}_C} = \underbrace{\left.\frac{\partial \mathbf{z}^{(j)}_i}{\partial {}^I p_{f_j}}\right|_{{}^C p_{f_j} = {}^C \hat{p}_{f_j}}}_{J_1} \cdot \left.\frac{\partial {}^C p_{f_j}}{\partial {}^I p_C}\right|_{{}^I p_C = {}^I \hat{p}_C} \tag{3.164}$$

The derivative with respect to the camera-IMU rotation is derived analogue using the derivative chain law.

Equation 3.103 which expresses the feature position of the j-th feature in the i-th camera frame needs to be rewritten to calculate the jacobian with respect to the camera-IMU translation ${}^I\mathbf{p}_C$ and camera-IMU rotation $\delta\theta^C_I$:[12]

$$\begin{aligned}
{}^{C_i}\mathbf{p}_{f_j} &= \mathbf{C}({}^{C_i}_G\bar{q})({}^G\mathbf{p}_{f_j} - {}^G\mathbf{p}_{C_i}) && (3.165)\\
&= \mathbf{C}({}^C_I\bar{q})[\mathbf{C}({}^I_G\bar{q})({}^G\mathbf{p}_{f_j} - {}^G\mathbf{p}_{C_i})] && (3.166)\\
&= \mathbf{C}({}^C_I\bar{q})[\mathbf{C}({}^I_G\bar{q})({}^G\mathbf{p}_{f_j} - ({}^G\mathbf{p}_I + \mathbf{C}({}^G_I\bar{q})^I\mathbf{p}_C)] && (3.167)\\
&= \mathbf{C}({}^C_I\bar{q})[\mathbf{C}({}^I_G\bar{q})({}^G\mathbf{p}_{f_j} - {}^G\mathbf{p}_I) + \mathbf{C}({}^I_G\bar{q})\mathbf{C}({}^G_I\bar{q})^I\mathbf{p}_C] && (3.168)\\
&= \mathbf{C}({}^C_I\bar{q})[\mathbf{C}({}^I_G\bar{q})({}^G\mathbf{p}_{f_j} - {}^G\mathbf{p}_I) + {}^I\mathbf{p}_C] && (3.169)\\
& && (3.170)
\end{aligned}$$

Expressing the last equation in error term notation yields:

$$\begin{aligned}
\Delta^C\mathbf{p}_{f_j} =& \hat{\mathbf{R}}^C_I(\mathbf{I}_3 + \lfloor \delta\theta^C_I \rfloor)[\hat{\mathbf{R}}^I_G(\mathbf{I}_3 + \lfloor \delta\theta^I_G \rfloor)(\Delta^G\mathbf{p}_{f_j} + {}^G\hat{\mathbf{p}}_{f_j} && (3.171)\\
& -\Delta^G\mathbf{p}_I + {}^G\hat{\mathbf{p}}_I) - (\Delta^I\mathbf{p}_C + {}^I\hat{\mathbf{p}}_C)] && (3.172)
\end{aligned}$$

Jacobian of measurement equation wrt. IMU-camera translation

$$\begin{aligned}
\frac{\partial \Delta^C\mathbf{p}_{f_j}}{\partial \Delta^I\mathbf{p}_C} &= -\hat{\mathbf{R}}^C_I\mathbf{I}_3 && (3.173)\\
&= -\hat{\mathbf{C}}({}^C_I\bar{q}) && (3.174)
\end{aligned}$$

[12] See the work of Trawny et al. [11] as well as Li et al. [12].

Jacobian of measurement equation wrt. IMU-camera rotation

$$\frac{\partial \Delta^C \mathbf{p}_{f_j}}{\partial \delta\theta_I^C} = \frac{\partial}{\partial \delta\theta_I^C} \left(\hat{\mathbf{R}}_I^C \left\lfloor \delta\theta_I^C \right\rfloor (\hat{\mathbf{R}}_G^I ({}^G\hat{\mathbf{p}}_{f_j} - {}^G\hat{\mathbf{p}}_I) - {}^I\hat{\mathbf{p}}_C) \right) \tag{3.175}$$

$$= - \frac{\partial}{\partial \delta\theta_I^C} \left(\hat{\mathbf{R}}_I^C \left\lfloor \hat{\mathbf{R}}_G^I ({}^G\hat{\mathbf{p}}_{f_j} - {}^G\hat{\mathbf{p}}_I) - {}^I\hat{\mathbf{p}}_C \times \right\rfloor \delta\theta_I^C \right) \tag{3.176}$$

$$= - \hat{\mathbf{R}}_I^C \left\lfloor \hat{\mathbf{R}}_G^I ({}^G\hat{\mathbf{p}}_{f_j} - {}^G\hat{\mathbf{p}}_I) - {}^I\hat{\mathbf{p}}_C \times \right\rfloor \tag{3.177}$$

$$= - \hat{\mathbf{C}}({}^C_I\bar{q}) \left\lfloor \hat{\mathbf{C}}({}^I_G\bar{q})({}^G\hat{\mathbf{p}}_{f_j} - {}^G\hat{\mathbf{p}}_I) - {}^I\hat{\mathbf{p}}_C \times \right\rfloor \tag{3.178}$$

Chapter 4

Camera models

In this chapter, the equations for the three camera models used in the thesis are derived [1]:

1. **ATAN model**
   - uses the field-of-view[2] fish-eye model as presented by Devernay and Faugeras in "Straight lines have to be straight" [13]
   - the model is used by PTAM and can be also be calibrated using PTAM[3]
   - the tangential distortion can often be neglected yielding a faster projection and un-projection model
2. **Pinhole model**
   - uses three radial and two tangential distortion parameters.
   - used in `OpenCV` and `ROS`.
   - can be calibrated via the `ROS` camera calibration tool[4]
3. **Ocam model**
   - high field of view or omnidirectional
   - OCamCalib toolbox to calibrate camera [5]

4.1 ATAN camera model

In this section, the equations of the field-of-view (FoV) fish-eye model as proposed by Devernay and Faugeras [13] and the projection model presented by Klein et al. [1] are derived.

The camera calibration parameters are

- Normalized focal length in x-direction: f_x
- Normalized focal length in y-direction: f_y
- Normalized x-offset: c_x

[1] Information adopted from `https://github.com/uzh-rpg/rpg_svo/wiki/Camera-Calibration`
[2] **F**ield **O**f **V**iew (FOV)
[3] `https://github.com/ethz-asl/ethzasl_ptam`
[4] `http://wiki.ros.org/camera_calibration`
[5] Based on the work of Scaramuzza `http://rpg.ifi.uzh.ch/software_datasets.html`

- Normalized y-offset: c_y
- Distortion parameter to keep straight lines straight: ω
  ($\omega = 0$: No distortion)

The focal length and image center are transformed to pixel coordinates and defined as

$$f_u = n_x f_x \tag{4.1}$$
$$f_v = n_y f_y \tag{4.2}$$
$$c_u = n_x c_x \tag{4.3}$$
$$c_v = n_x c_y \tag{4.4}$$

Field-of-View model (FOV) incorporating barrel radial distortion

The radial distortion transformation factor is defined as the ratio of distorted radius r_d to undistorted radius r_u:

$$r_{factor} = \frac{r_d}{r_u} \tag{4.5}$$

where the distorted and undistorted radius are given by

$$r_d = f_{undist \to dist}(r_u) = \frac{1}{\omega} \arctan\left(2 r_u \tan\frac{\omega}{2}\right) \tag{4.6}$$

and

$$r_u = f_{dist \to undist}(r_d) = \frac{\tan(r_d \omega)}{2 \tan\frac{\omega}{2}} \tag{4.7}$$

respectively [13]. The distortion model is illustrated in figure 4.1 for parameter $\omega = 0.95365$. Note that the field-of-view ω of the corresponding ideal fish-eye lens is the only parameter of the FOV-model.

Project point $[X_C, Y_C, Z_C]^T$ from camera frame to pixel coordinates $[u, v]^T$
Assuming known parameters, the points are projected from camera frame into the image:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} c_u \\ c_v \end{bmatrix} + \begin{bmatrix} f_u & 0 \\ 0 & f_v \end{bmatrix} \frac{r_d}{r} \begin{bmatrix} \frac{X_C}{Z_C} \\ \frac{Y_C}{Z_C} \end{bmatrix} \tag{4.8}$$

with

$$r = \sqrt{\frac{X_C^2 + Y_C^2}{Z_C^2}} \tag{4.9}$$

and

$$r_d = f_{undist \to dist}\left(\sqrt{\frac{X_C^2 + Y_C^2}{Z_C^2}}\right) = \frac{1}{\omega} \arctan\left(2\sqrt{\frac{X_C^2 + Y_C^2}{Z_C^2}} \tan\frac{\omega}{2}\right) \tag{4.10}$$

When setting $Z_C = 1$ the last equations simplify to:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} c_u \\ c_v \end{bmatrix} + \begin{bmatrix} f_u & 0 \\ 0 & f_v \end{bmatrix} \frac{r_d}{r} \begin{bmatrix} X_C \\ Y_C \end{bmatrix} \tag{4.11}$$

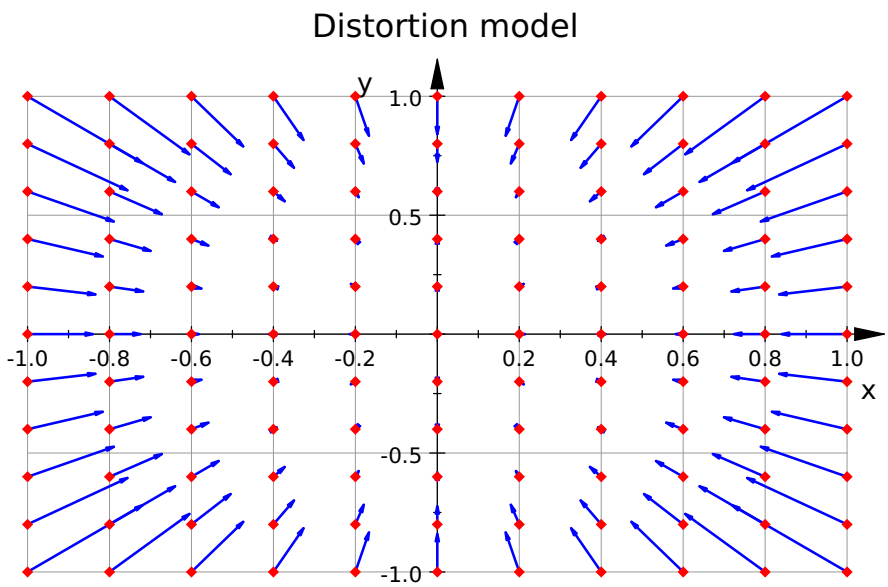


Figure 4.1: Distortion model for $\omega = 0.95365$

with

$$r = \sqrt{X_C^2 + Y_C^2} \tag{4.12}$$

and

$$r_d = f_{undist \to dist}\left(\sqrt{X_C^2 + Y_C^2}\right) = \frac{1}{\omega}\arctan\left(2\sqrt{X_C^2 + Y_C^2}\tan\frac{\omega}{2}\right) \tag{4.13}$$

Unproject from image pixel coordinates to camera $z = 1$-plane Starting from equation 4.11, the unprojection is given by:

$$\begin{bmatrix} X_C \\ Y_C \end{bmatrix} = \frac{r_u}{r}\begin{bmatrix} \frac{1}{f_u} & 0 \\ 0 & \frac{1}{f_v} \end{bmatrix}\begin{bmatrix} u - c_u \\ v - c_v \end{bmatrix} \tag{4.14}$$

with

$$r = \sqrt{X_C^2 - Y_C^2} = \sqrt{\left(\frac{u - c_u}{f_u}\right)^2 + \left(\frac{v - c_v}{f_v}\right)^2} \tag{4.15}$$

and

$$r_u = f_{dist \to undist}\left(\left(\frac{u - c_u}{f_u}\right)^2 + \left(\frac{v - c_v}{f_v}\right)^2\right) = \frac{\tan\left(\left(\frac{u-c_u}{f_u}\right)^2\omega + \left(\frac{v-c_v}{f_v}\right)^2\omega\right)}{2\tan\frac{\omega}{2}} \tag{4.16}$$

4.2 Perspective camera model

In this section, the transformation chain of the perspective camera model from world to pixel coordinates is derived:

1. **World coordinates to camera coordinates**

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \tag{4.17}$$

$$= \begin{bmatrix} \mathbf{R} & | & \mathbf{T} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{4.18}$$

$\mathbf{R}$: Rotation from camera to world coordinates
$\mathbf{T}$: Translation from camera to world coordinates

2. **Camera coordinates to image plane coordinates** Similar triangle relation yields

$$\frac{x}{f} = \frac{X_c}{Z_c} \rightarrow x = \frac{fX_c}{Z_c} \tag{4.19}$$

$$\frac{y}{f} = \frac{Y_c}{Z_c} \rightarrow y = \frac{fY_c}{Z_c} \tag{4.20}$$

f : focal length [m]

3. **Image plane coordinates to pixel coordinates**

$$u = u_0 + kx = u_0 + k\frac{fX_c}{Z_c} \tag{4.21}$$

$$v = v_0 + ky = v_0 + k\frac{fY_c}{Z_c} \tag{4.22}$$

which can be expressed in matrix form using homogeneous coordinates:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} kf & 0 & u_0 \\ 0 & kf & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{K} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{4.23}$$

u_0, v_0 : offset to optical center
λ : homogeneous scale factor
$\mathbf{K}$: camera matrix
k : pixel scale factor
f : focal length [m]
kf : focal length [pixel]

The final **transformation from world to pixel coordinates** is thus given by iterative inserting

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} kf & 0 & u_0 \\ 0 & kf & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{4.24}$$

$$= \begin{bmatrix} kf & 0 & u_0 \\ 0 & kf & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{4.25}$$

$$= \mathbf{K} \begin{bmatrix} \mathbf{R} & | & \mathbf{T} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{4.26}$$

Radial distortion model The radial distortion can be modelled as

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = (1 + k_1 r^2) \begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \tag{4.27}$$

with

$$r^2 = (u - u_0)^2 + (v - v_0)^2 \tag{4.28}$$

u_d, v_d : distorted, real coordinates [pixels]
u, v : undistorted, ideal coordinates [pixels]

4.3 Omnidirectional camera model

In this section, the equations of the omnidirectional camera model proposed by Scaramuzza [14] are presented: Let $\mathbf{P}$ be the 3d-vector from the viewpoint to the pixel position and x, y, z be the pixel position in camera coordinates. Assuming that the mirror and camera are perfectly aligned and the mirror is rotationally symmetric, vector $\mathbf{P}$ can be written as

$$\mathbf{P} = \begin{bmatrix} u \\ v \\ f(\rho) \end{bmatrix} \tag{4.29}$$

where u, v are the pixel coordinates with respect to the image center and $\rho = \sqrt{u^2 + v^2}$ is the distance of an arbitrary point from this image center. Scaramuzza proposes a polynomial function to model $f(\rho)$:

$$f(\rho) = \alpha_0 + \alpha_1 \rho^1 + \alpha_2 \rho^2 + \alpha_3 \rho^3 + ... \tag{4.30}$$

The affine distortion model to handle non-alignment of the mirror and camera is proposed as follows:

$$\begin{bmatrix} u_{dist} \\ v_{dist} \end{bmatrix} = \begin{bmatrix} c & d \\ e & 1 \end{bmatrix} \begin{bmatrix} u_{undist} \\ v_{undist} \end{bmatrix} + \begin{bmatrix} x_{c,dist} \\ y_{c,dist} \end{bmatrix} \tag{4.31}$$

Chapter 5

Feature tracking

The implemented feature tracker algorithm described in this chapter is similar to the one used in PTAM [1] but tracks features over multiple images and creates a feature trail for every observed feature.

5.1 Feature tracker algorithm

The pseudo code of the top level feature tracking algorithm is shown in algorithm 1 and illustrated in figure 5.1. The procedure is as follows:

- **Half sampling:** Every new image is half-sampled to extract features from different pyramidal levels (section 5.1.1).
- **Feature detection:** For every level in the pyramid, the features are extracted using the 9-point FAST feature detector (section 5.1.2).
- **Trail creation:** If this is the first image recorded, then for every valid patch a new trail is created (section 5.1.3).
- **Matching:** If there have been 1+ images, the patches around the FAST features are matched between consecutive images and the new match is inserted at the end of the corresponding feature trail (section 5.1.4).
- **Trail update:** The function `GetPastTrails` outputs the feature trail if the feature is lost [1] or the predefined maximum trail length is reached. The trail is moved to the lost trail list and removed from the active trail list (section 5.1.5).

Algorithm 1 Feature tracker

```
function GetPastTrails(image)
    HalfSampling()                          ▷ Sampling according to 5.1.1
    FeatureDetection()                      ▷ Extract FAST 9-point features [15]
    if Not First image then
        Match()                             ▷ Match patches around FAST features
    end if
    CollectFinishedTrails()
    AddTrails()                             ▷ Start new feature trails
end function
```

[1] That is, the feature could not be re-detected in the current image.

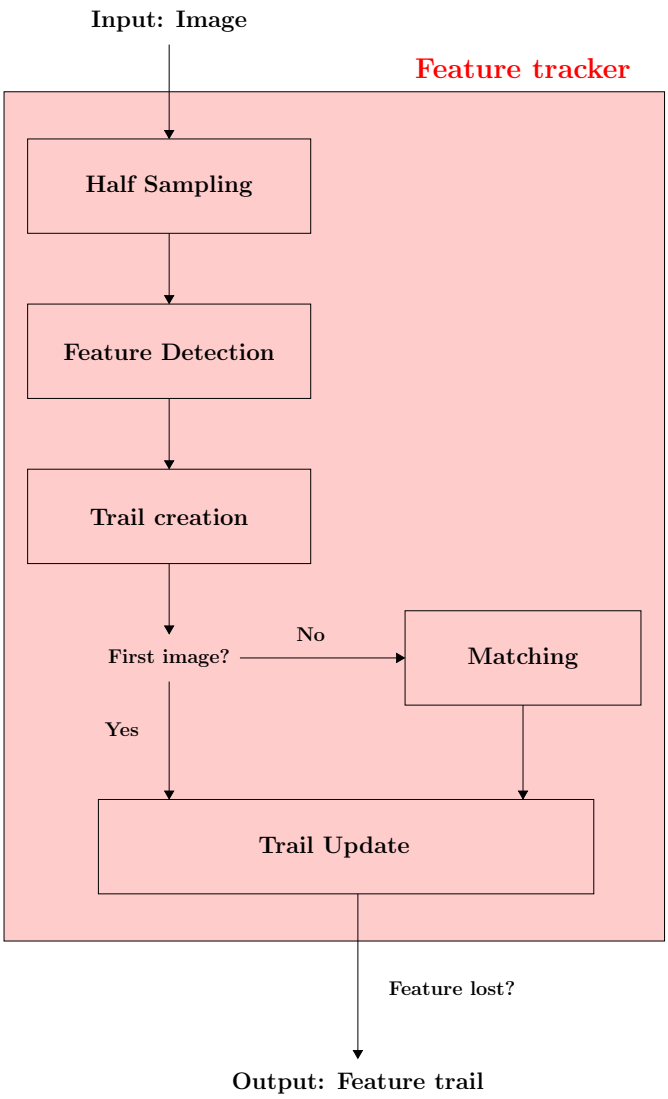


Figure 5.1: Basic feature tracking routines

5.1.1 Half-Sampling

The down-sampling procedure simulates images taken from further distance making it possible to handle larger displacements of points between consecutive camera frames. The incoming image is half/down sampled by a factor of two in width and height which results in e.g. four pyramidal layers: The level zero corresponds to the original image, level one stores the image sampled by $2^1 = 2$, level two stores the image sampled by $2^2 = 4$, and level three by $2^3 = 8$ as shown in figure 5.2 and 5.3: The higher the pyramid level, the lower the resolution. Note that the images stored for the individual pyramidal levels are not smoothed unlike other tracking approaches.

The pseudo code of the half sampling algorithm at the top level is shown in algorithm 2: The depth corresponds to the number of pyramidal levels.

Algorithm 2 Half-Sampling

```
function HalfSampling(image)
    for i=1:Size(Depth) do
        Downsampling()                    ▷ libCVD implementation [15]
    end for
end function
```

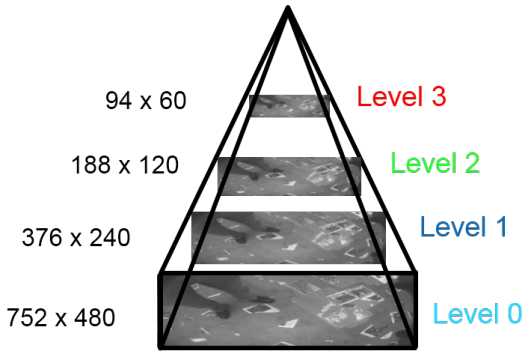


Figure 5.2: Pyramidal levels. Left side shows the image width × height for the individual levels assuming an original image of 752 × 480 pixels.

level	sampled by
0	$2^0 = 1$
1	$2^1 = 2$
2	$2^2 = 4$
3	$2^3 = 8$
...	...

Figure 5.3: Pyramidal levels and corresponding down sampling factor.

5.1.2 Feature detection

For every image level the features are extracted using the **F**eatures from **A**ccelerated **S**egment **T**est (FAST) 9 point feature detector[2] as shown in figure 5.4 and 5.5. The focus of the feature detector is on speed: It is a very fast detector in the order of 100 mega-pixel/second [15]. The corner detection threshold is a function parameter input: A low threshold corresponds to many low-quality features, a high threshold returns few high-quality features.

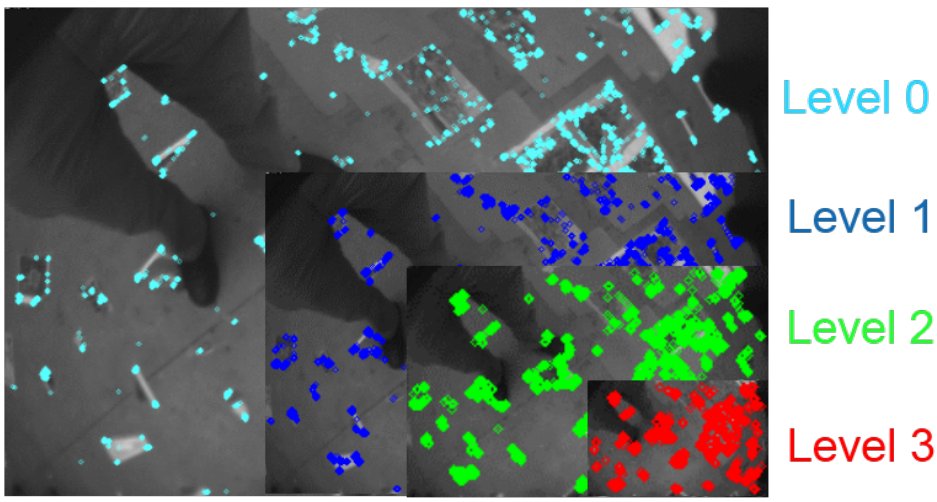


Figure 5.4: Features detected by FAST on level zero, one, two and three before scoring and invalidation of the border features. The corner detection threshold is set to 10.

The the feature detection pseudo code is given in algorithm 3: In line 4, features

[2]Proposed by Rosten et al. in [15] and [16].

that are too close to the border and whose patch is outside the image are invalidated.

Algorithm 3 Feature detection
T: corner detection threshold ($T = 0$: all features)
x, y: pixel value [pixel]
w: half patch width [pixel]
m: border margin [pixel]
s: image size depending on pyramid level (e.g. 752, 376, 188,...)

```
function GrabFeatures
    for i=1:Size(Depth) do
        FAST(T)                ▷ Extract FAST 9-point features, libCVD [15]
        CheckFAST()            ▷ Invalidate FAST features that are on border
    end for
end function

function CheckFAST
    patch ← invalid
    if (x − w ≥ m) and (x + w < s − m) and (y − w ≥ m) and (y + w < s − m)
then
        patch ← valid
    end if
end function
```

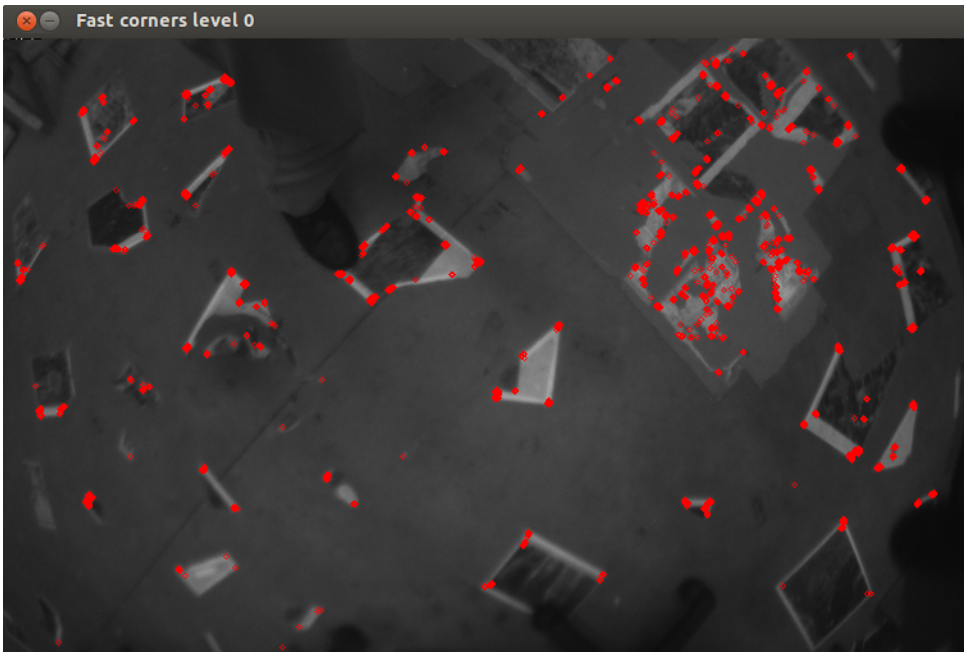


Figure 5.5: Features detected by FAST on level zero before scoring and invalidation of the border features. The corner detection threshold is set to 10.

5.1.3 Feature trail creation

The FAST feature detector returns a list of n points [3]. This feature list is then sorted according to each feature's corner detection FAST Score. Starting with the most

[3] In general a different amount of features for each pyramidal level.

promising feature, the patch around the feature is initialized and an area around the feature is blocked as shown in figure 5.6. That is, every candidate feature that is inside the blocked range and has not been initialized yet will become invalid. This makes sure that that two features are not too close to each other as explained in algorithm 4 and 5.

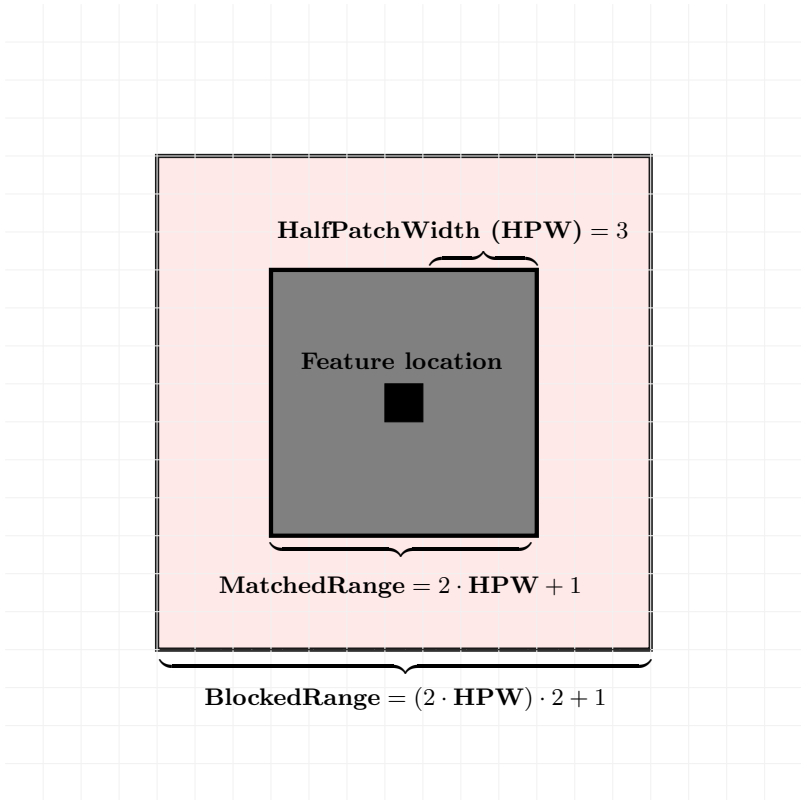


Figure 5.6: The feature location extracted by the FAST detector is represented by the black square. The gray 7×7 square defines the patch associated with the feature assuming a predefined **H**alf **P**atch **W**idth (HPW) of three pixels. Based on this patch the ZMSSD is calculated. The blocked range is the 13×13 red square.

Algorithm 4 Trail creation using FAST feature score

Create new trails from list of unmatched patches. The patches used for creation of new trails are selected according to to their score.

```
function AddTrails
    trailsNeeded = maxNumberTrails - activeNumberTrails
    for i = 3, 2, 1, 0 do                          ▷ Start with highest level
        if trailsNeeded = 0 then
            Invalidate all patches that are not yet associated
        end if
        if (patch == valid) then
            GetNBestPatches
        end if
    end for
end function
```

Algorithm 5 Extract the best patches
n : requested number of patches

```
function GetNBestPatches(n)
    for (Size(patchList)) do
        if (patch != associated and patch == valid) then
            Sort patchList according to score                ▷ libCVD [15]
        end if
    end for
    counter ← 0
    for (Size(patchListSorted)) do
        if ( counter ≥ n) then
            Invalidate patch in next frame
        end if
        if (patch == valid) then
            Block area around patch in all levels
            counter ← counter+1
        end if
    end for
end function
```

5.1.4 Matching

The pseudo code for the matching process is given in algorithm 6 and is as follows

- Loop over all active feature trails and all pyramidal levels in the order given by table 5.1. For example, if the template feature patch was found in level two, it is matched to the probe patches found in level two, then in level three, level one and level zero.

0	0	1	2	3
1	1	2	0	3
2	2	3	1	0
3	3	2	1	0

Table 5.1: Matching order. For example if the template feature patch was found in level two, it is matched to the probe patches found in level two, then in level three, level one and level zero.

- The location of the *last* feature of trail k is projected into the new image using the homography computed based on the two consecutive images. This is called homography aided feature tracking and the theory is explained in section 5.1.6. The homography may be calculated based on gyroscope measurements, the rotation estimates of the filter or based on feature correspondences & RANSAC[4].

- A search region is defined around the *projected* feature location. Figure 5.7 shows the search region (red square) without projecting the feature location and the search region when using a homography (turqoise square). In this example, the feature matching would have failed if no homography was used.

[4] **Ran**dom **Sa**mple **C**onsensus (RANSAC)

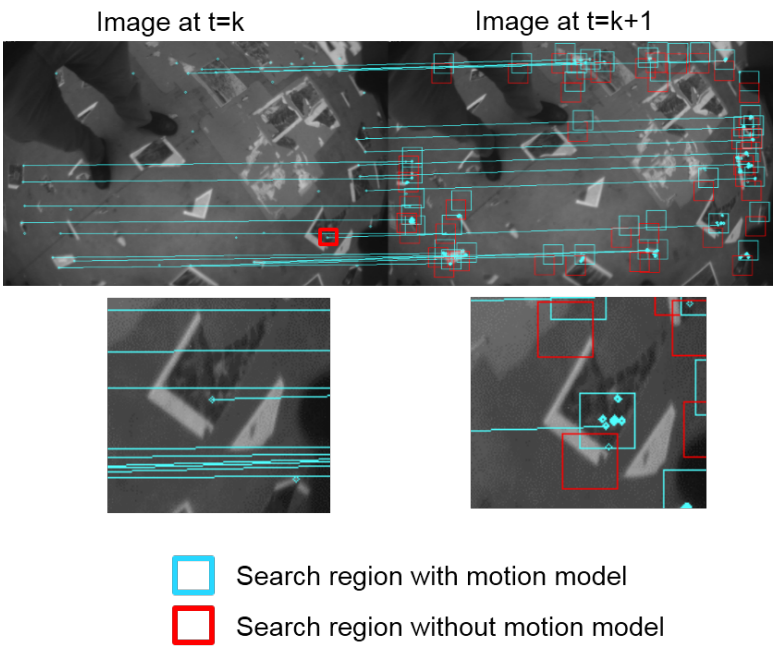


Figure 5.7: Homography aided feature matching: In the top two consecutive images are shown. The thick red square at $t = k$ illustrates the search region around the feature in the old image. In the new image, the search region is represented by a thin red square. In this example, the feature which is to be matched is not inside the search region. In contrast, the search region projected by the use of a homography, shown as a turquoise square, contains the desired feature. The turquoise line represents the matches.

- For every FAST feature (illustrated in figure 5.7 as turquoise points) detected in the search region in the new image the **Z**ero **M**ean **S**um of **S**quared **D**ifferences (ZMSSD) is calculated with respect to the template patch of feature k.

  The Zero Mean Sum of Squared Differences (ZMSSD) is defined as

$$\text{ZMSSD} = \sum_{(i,j)} \left(I_1(i,j) - \bar{I}_1(i,j) - (I_2(x+i, y+j) - \bar{I}_2(x+i, y+j)\right)^2 \tag{5.1}$$

  where $I_1(i,j)$ denotes the image intensity of the first image (the template image) at pixel i, j and $\bar{I}_1$ is the mean value at this pixel. $I_1(i,j)$ and $\bar{I}_1$ are the analogue pixel and mean values respectively of the second image (the probe image).

  It can be approximated by

$$\text{ZMSSD} \approx \frac{1}{N}(2S_A S_B - S_A^2 - S_B^2) - (2S_A S_B - S_A^2 - S_B^2) \tag{5.2}$$

  with

$$S_A := \sum_{(i,j)} I_1, \;\; S_B := \sum_{(i,j)} I_2 \tag{5.3}$$

and N is the patch width times the patch height.

- The feature patch combination yielding the lowest ZMSSD which is below a predefined threshold is declared a match (illustrated in figure 5.7 as turquoise line) and inserted at the end of the feature trail.

Algorithm 6 Matching

```
function MATCH
    for i=1:Size(trailList) do
        MatchAllLevels()
    end for
end function

function MATCHALLLEVELS
    for j=1:Size(Depth) do
        level ← index(j)
        MatchOneLevel()
        if (match found) then
            Insert patch at end of trailList
        end if
    end for
end function

function MATCHONELEVEL
    Project feature from old to current image using a homography
    Define a search region around projected feature in current image
    ZMSSD_lowest ← ZMSSD_max + 1
    for all FAST features inside projected search region do
        ZMSSD_current ← ZMSSD with respect to template patch
        if ZMSSD_current < ZMSSD_lowest then
            ZMSSD_lowest ← ZMSSD_current
        end if
    end for
    if ZMSSD_lowest < ZMSSD_max then
        match found ← true
    else
        match found ← false
    end if
end function
```

To prevent false feature matching a maximum ZMSSD threshold is defined. The maximum ZMSSD threshold is determined by looking at several sample patches and their ZMSSD as shown in figure 5.8 and 5.9.
A feature may be lost due to several reasons:

- the camera has moved and the tracked feature is now outside the camera field of view
- the feature is occluded in the new image
- the (possibly low contrast) feature is not detected any more by the FAST feature detector.

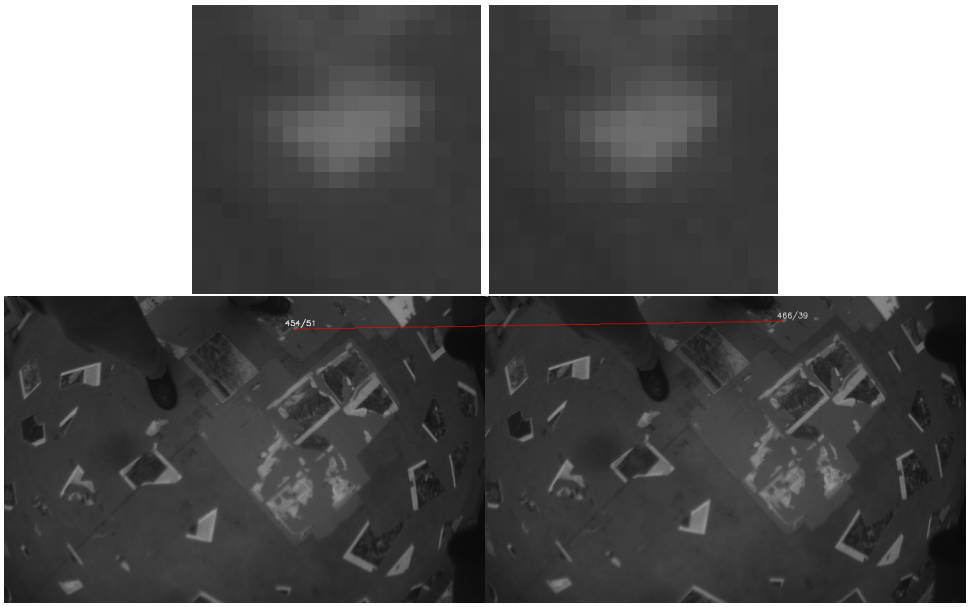


Figure 5.8: Correct matching: ZMSSD = 1644. The maximum ZMSSD threshold should be well above this value.

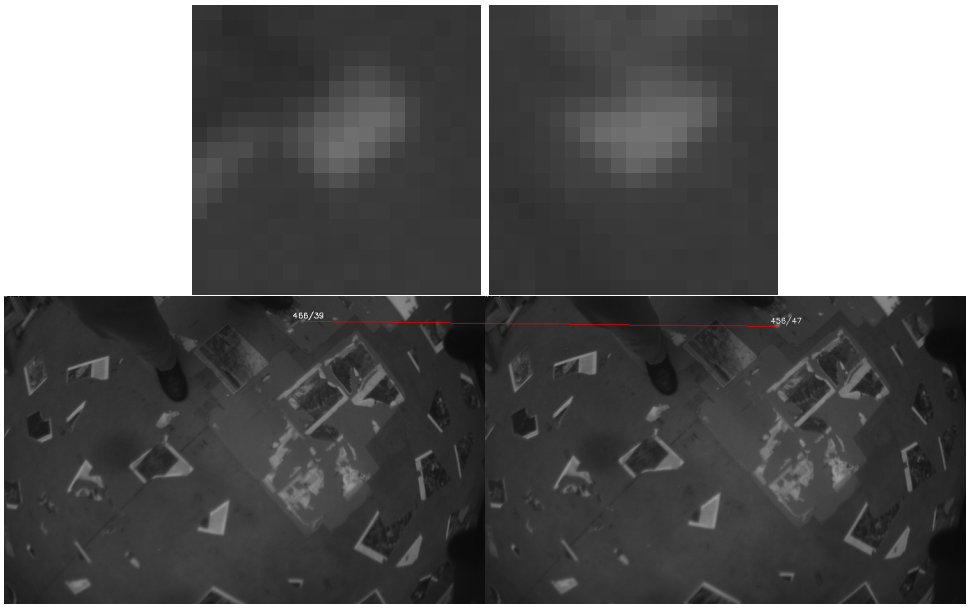


Figure 5.9: Wrong matching: ZMSSD = 34354. The maximum ZMSSD threshold should be well below this value.

5.1.5 Feature trail update

The matching procedure of feature j can lead to two possible outcomes:

1. **Match found:** The last patch of the trail for feature j from the *last* image could be matched to a feature patch in the *current* image. Then the new patch is simply added at the end of the feature trail and the feature trail remains in the *active* trail list.

2. **No match found:** For all the FAST features inside the search region gen-

erated around the feature j the ZMSSD is calculated. If none of these probe patches have a ZMSSD below the predefined threshold then the feature trail is considered to be 'lost' and moved to the *lost* trail list.

The lost trails are shown in figure 5.10: The four patches on the top illustrate one feature trail. The image shows the observations of a feature as red dots and the connection of the first and last observation with a black line.

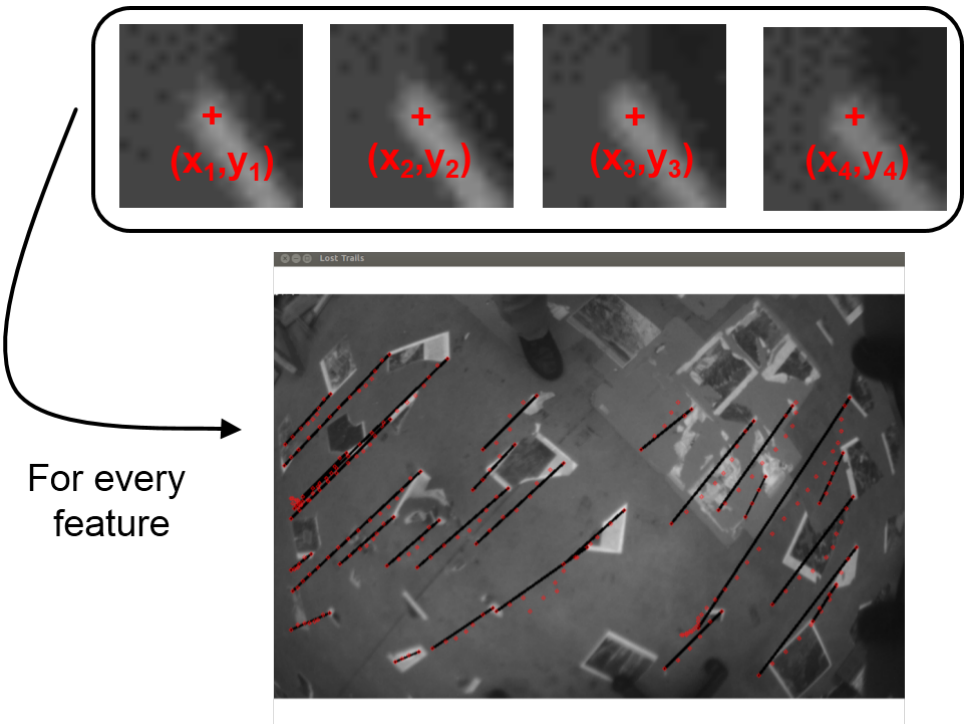


Figure 5.10: Lost trails: The four patches on the top illustrate one feature trail. The image shows the observations of a feature as red dots and the connection of the first and last observation with a black line.

5.1.6 Homography aided feature tracking

For *calibrated* cameras, the coordinate transformation from camera frame 1 to camera frame 2 is given by [5]

$$\mathbf{X}_2 = \mathbf{R}\mathbf{X}_1 + \mathbf{T} \tag{5.4}$$

$$= \mathbf{R}\mathbf{X}_1 + \mathbf{T}\frac{1}{d}\mathbf{N}^T \tag{5.5}$$

$$= \underbrace{\left(\mathbf{R} + \frac{1}{d}\mathbf{T}\mathbf{N}^T\right)}_{\mathbf{H}\in\mathbb{R}^{3\times 3}} \mathbf{X}_1 \tag{5.6}$$

[5] Adopted from Soatto et al. [17]

with

$\mathbf{R}$: Rotation from camera frame 1 to camera frame 2
$\mathbf{T}$: Translation from camera frame 1 to camera frame 2
P : Image plane
d : Distance of image plane P to optical center of first camera
$\mathbf{N}$: Unit normal vector of plane P w.r.t. optical center of first camera
$\mathbf{H}$: (Planar) homography matrix

The homography mapping

$$x_2 \sim \mathbf{H} x_1 \tag{5.7}$$

relates image points from camera view 1 to image points in camera view 2 as illustrated in figure 5.11.

For *uncalibrated* cameras, the coordinate transformation is given by [6]

$$\bar{\mathbf{X}} = \mathbf{K}\mathbf{R}\mathbf{K}^{-1}\bar{\mathbf{X}} + \bar{\mathbf{T}} \tag{5.8}$$

with

$$\bar{\mathbf{X}} = \mathbf{K}\mathbf{X} \tag{5.9}$$

$$\bar{\mathbf{T}} = \mathbf{K}\mathbf{T} \tag{5.10}$$

and the camera calibration matrix $\mathbf{K}$. Assuming zero (or neglectable) translation the homography can be approximated by

$$\mathbf{H} \approx \mathbf{K}\mathbf{R}\mathbf{K}^{-1} \tag{5.11}$$

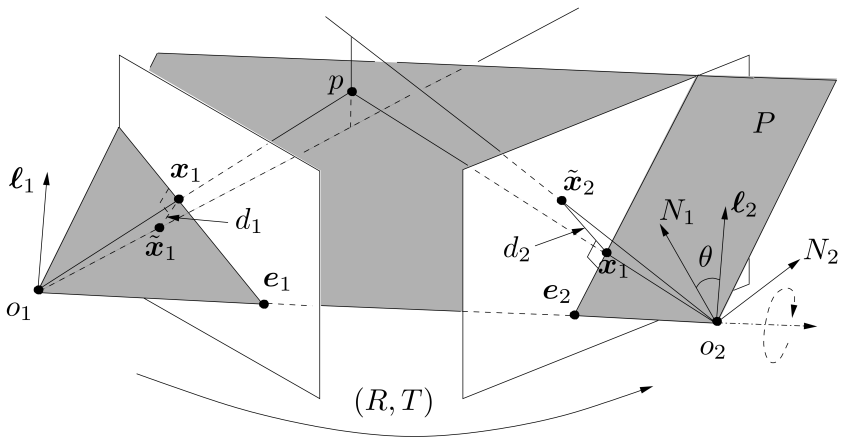


Figure 5.11: Homography between two camera views (image: [17])

5.2 Experiments

As described in chapter 3, N camera poses are inserted in the EKF state vector. The question is now how large should N be to collect all the information. Obviously, the number of observations per feature depends to a large extent on the camera trajectory. Figure 5.12 shows the trajectory of the camera translating back and forth. The euclidean distance from start to end point is about 1 meter. The

[6] Adopted from Soatto et al. [17]

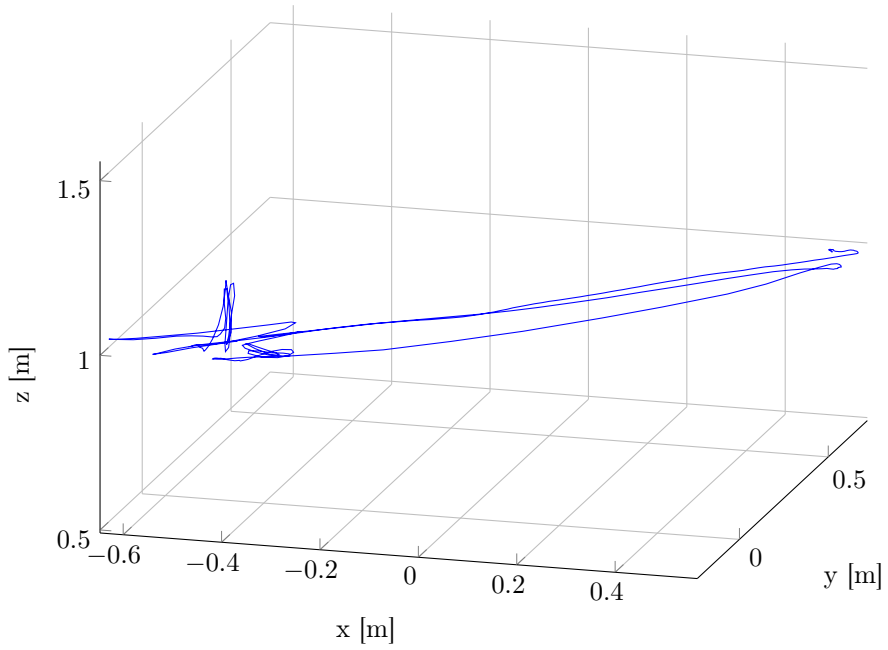


Figure 5.12: Trajectory 1 (Translation): The figure shows the trajectory of the camera translating back and forth. The euclidean distance from start to end point is about 1 meter.

properties of the generated feature trails are shown in figure 5.14 and in the second column of table 5.2. Since this is a relative static trajectory, the maximal feature trail length is 155 and the average trail length is about 10 observations per feature.

In contrast, figure 5.13 shows a dynamic trajectory of a circular motion with diameter of about 3 m. The properties of the generated feature trails for the circular trajectory are shown in figure 5.15 and in the third column of table 5.2. The maximum trail length is only 58 and the features were observed in six camera frames on average.

Trajectory	**Translation**	**Circle**
Duration [s]	32.66	55.62
Num. lost trails (total)	1911	6673
Num. lost trails / image (avg.)	3.92	8.01
Trail length (avg.)	10.38	6.31
Trail length (max.)	155	58
Trail length (min.)	2	2
Trail length (std.)	13.39	5.01

Table 5.2: Comparing feature trails for different kinds of trajectories

As a comparison, figure 5.16 shows the feature track length distribution for two outdoor datasets presented by Li and Mourikis [18]. They used a Harris corner detection in combination with normalized cross-correlation and obtained a similar distribution. Summarizing, the probability that a feature is tracked over more

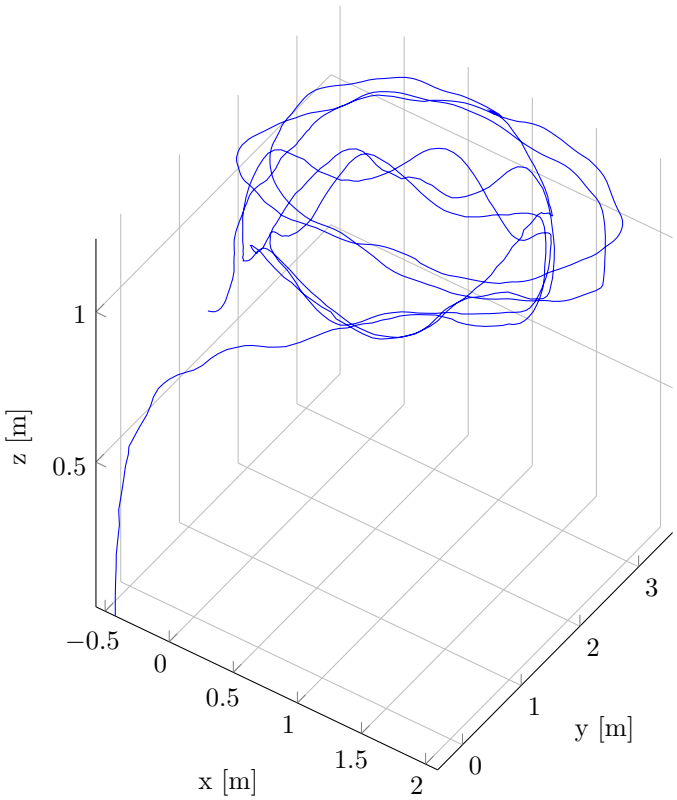


Figure 5.13: Trajectory 2 (Circle): The figure shows a circular camera trajectory. The radius of the circle is about 1.5 meter.

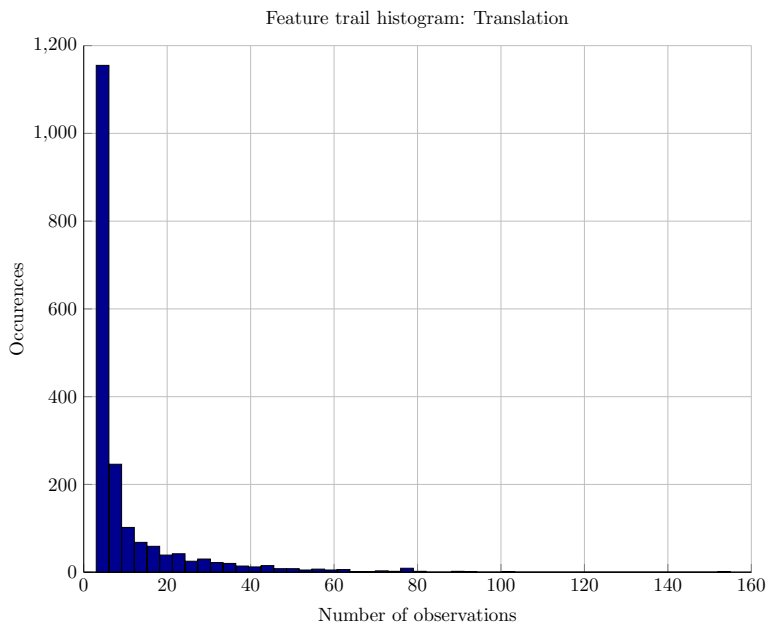


Figure 5.14: Number of observations per feature before losing the feature for trajectory 1 in figure 5.12. The longest trail contains 155 observations. N.b. the different x and y scale for fig. 5.14 and fig. 5.15

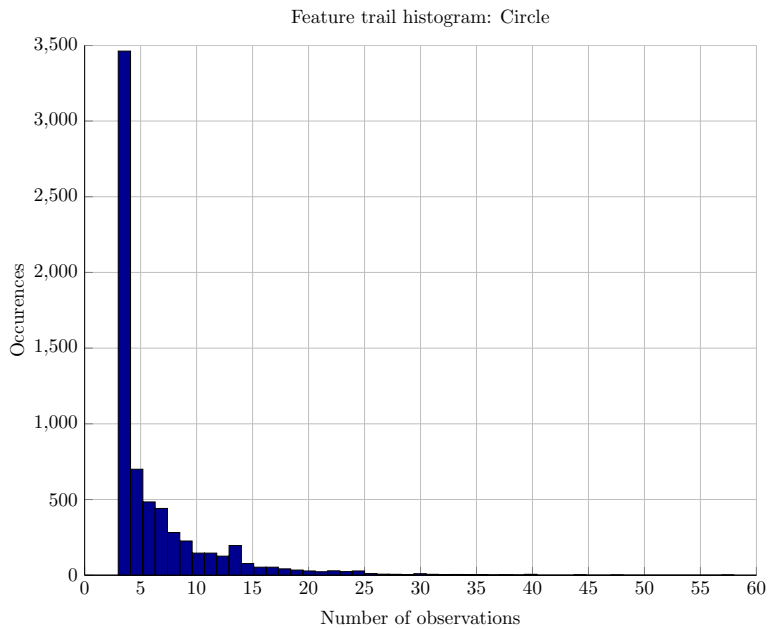


Figure 5.15: Number of observations per feature before losing the feature for trajectory 2 in figure 5.13. The longest trail contains 58 observations. N.b. the different x and y scale for fig. 5.14 and fig. 5.15

than 20 camera frames is relatively low. As a reference, Mourikis et al. [5] set the maximum number of camera poses in the EKF state to 30.

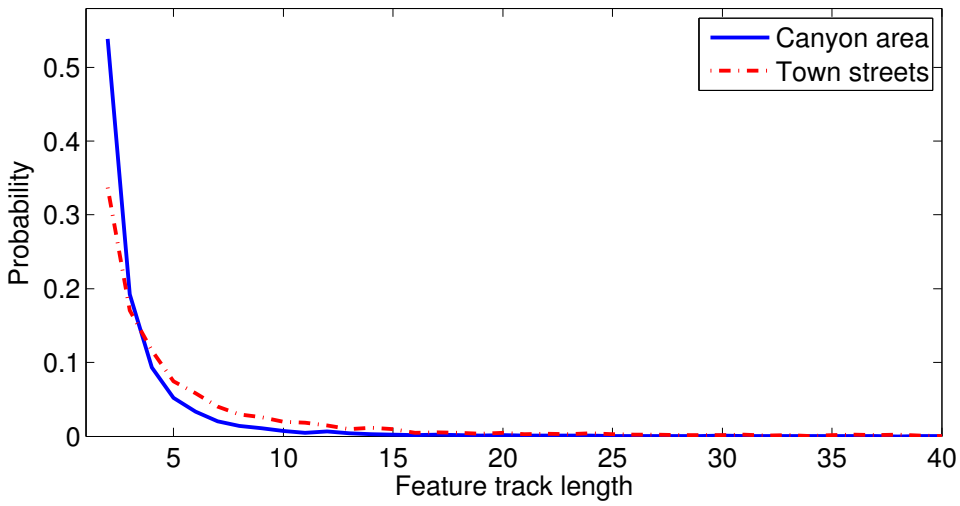


Figure 5.16: As a comparison: Feature track length distribution for two outdoor datasets presented by Li and Mourikis [18]. They used a Harris corner detection in combination with normalized cross-correlation. (image: [18])

Chapter 6

Multi-view feature triangulation

This chapter presents several multi-view triangulation methods. They all have in common that they use the n camera poses and feature observations to calculate the most likely 3d position. They mainly differ in the applied camera model, the definition of the measurement residual (e.g. in pixel or camera coordinates) and in the way the solution is found (direct or iterative).

6.1 Linear least-squares multi-view triangulation [pixel coordinates]

The linear least-squares triangulation approach minimizes the residual in pixel coordinates directly, that is without any iterations. Recall the equation for the perspective projection:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}[\mathbf{R}|\mathbf{T}] \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{6.1}$$

where $\mathbf{P}$

$$\mathbf{P} := \mathbf{K}[\mathbf{R}|\mathbf{T}] \tag{6.2}$$

is the projection matrix and contains the intrinsic and extrinsic camera parameters.

Equation 6.1 results in the following equation for every feature observation:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{6.3}$$

The homogeneous scale factor λ can be eliminated by applying the cross-product which results in three equations for each feature observation: [19]

$$\mathbf{x} \times \mathbf{P}\mathbf{X}_W = \mathbf{0} \tag{6.4}$$

or

$$x\mathbf{P}_{3,:}\mathbf{X} - \mathbf{P}_{1,:}\mathbf{X} = 0 \tag{6.5}$$
$$y\mathbf{P}_{3,:}\mathbf{X} - \mathbf{P}_{2,:}\mathbf{X} = 0 \tag{6.6}$$
$$x\mathbf{P}_{2,:}\mathbf{X} - y\mathbf{P}_{1,:}\mathbf{X} = 0 \tag{6.7}$$

which has the linear form

$$\mathbf{A}\mathbf{X} = \mathbf{0} \tag{6.8}$$

with

$$\mathbf{A} := \begin{bmatrix} x\mathbf{P}_{3,:} - \mathbf{P}_{1,:} \\ y\mathbf{P}_{3,:} - \mathbf{P}_{2,:} \end{bmatrix} \tag{6.9}$$

Assuming feature j was observed in n cameras, then $\mathbf{A}$ is given by simply stacking the equation of each measurement:

$$\mathbf{A} := \begin{bmatrix} x^1\mathbf{P}^1_{3,:} - \mathbf{P}^1_{1,:} \\ y^1\mathbf{P}^1_{3,:} - \mathbf{P}^1_{2,:} \\ x^2\mathbf{P}^2_{3,:} - \mathbf{P}^2_{1,:} \\ y^2\mathbf{P}^2_{3,:} - \mathbf{P}^2_{2,:} \\ \dots \\ x^n\mathbf{P}^n_{3,:} - \mathbf{P}^n_{1,:} \\ y^n\mathbf{P}^n_{3,:} - \mathbf{P}^n_{2,:} \end{bmatrix} \tag{6.10}$$

Equation 6.8 can be solved by the **S**ingular **V**alue **D**ecomposition (SVD) as described in the pseudo code in algorithm 7. Figure 6.2 and 6.1 show the triangulation results using the linear least-squares triangulation benchmarked on the Oxford multi-view corridor dataset. The results obtained by the linear method are not satisfying; the following sections are dedicated to the more promising iterative triangulation approaches.

Figure 6.1: Oxford 11-view corridor dataset

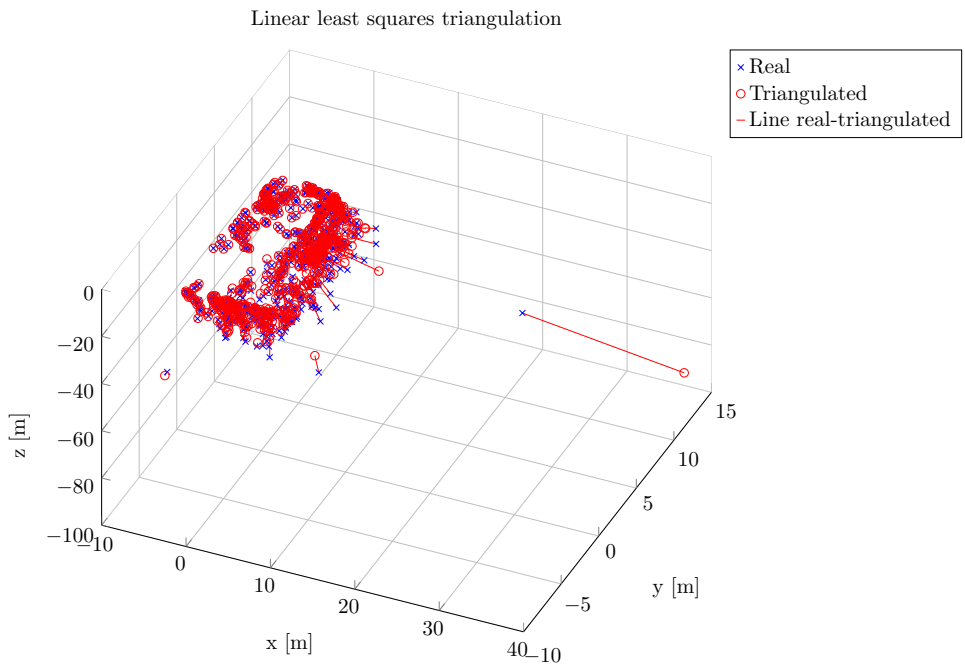


Figure 6.2: Multi-view linear least squares triangulation benchmarked on Oxford corridor dataset

Algorithm 7 Linear multi-view triangulation,
x, y : feature observation [pixel]

function LINEAR($P^{1\ldots n}$, $x^{1\ldots n}$, $y^{1\ldots n}$)
  $A \leftarrow \{\}$
  for $i = 1 : n$ **do**
    $A_{tmp} \leftarrow \begin{bmatrix} xP^i_{3,:} - P^i_{1,:} \\ yP^i_{3,:} - P^i_{2,:} \end{bmatrix}$
    $A \leftarrow \begin{bmatrix} A \\ A_{tmp} \end{bmatrix}$ ▷ Stacking the feature observations
  end for
  for $k = 1 : rows(A)$ **do**
    $A(k,:) \leftarrow A(k,:)./\text{norm}(A(k,:))$ ▷ Normalizing A
  end for
  $\begin{bmatrix} U & S & V \end{bmatrix} \leftarrow \text{svd}(A)$
  $X \leftarrow V(:, end)$ ▷ Last column of V
  $X_w \leftarrow \begin{bmatrix} X(1)/X(4) & X(2)/X(4) & X(3)/X(4) \end{bmatrix}^T$ ▷ 4d homogeneous → 3d
end function

6.2 Iterative least-squares multi-view triangulation [camera coordinates]

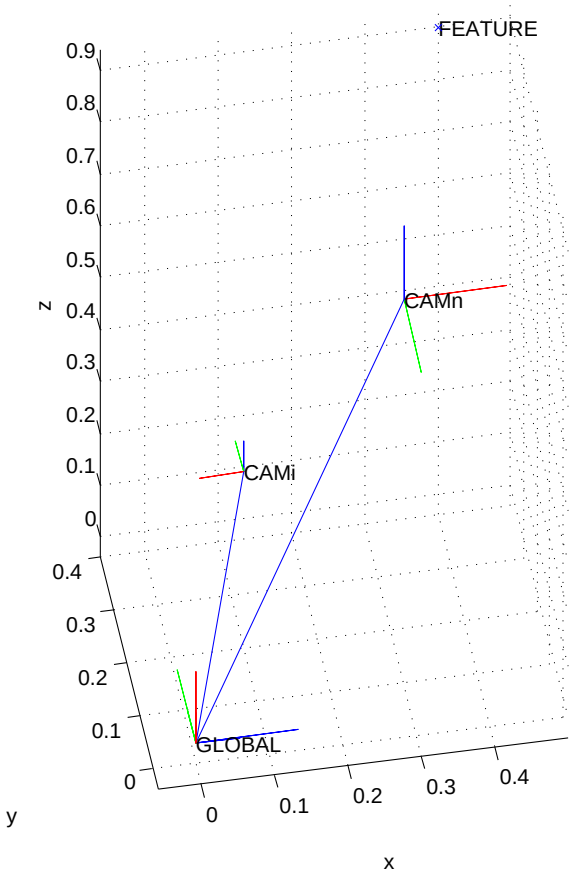


Figure 6.3: Feature observed in n-th and i-th camera frame. n stands for the camera frame in which feature j was observed for the first time, i stands for the i-th observation of that same feature

The algorithms described in this section are primarily adopted from the work of Mourikis et al. [5] and Hartley et al. [20]: The position of the j-th feature observed in the i-th camera frame (as illustrated in figure 6.3) can be expressed in terms of the n-th camera frame:

$$
\begin{aligned}
{}^{C_i}\mathbf{p}_{f_j} &= \mathbf{C}({}^{C_i}_{C_n}\bar{q})\,{}^{C_n}\mathbf{p}_{f_j} + {}^{C_i}\mathbf{p}_{C_n} && (6.11)\\
&= {}^{C_n}Z_j \left(\mathbf{C}({}^{C_i}_{C_n}\bar{q}) \begin{bmatrix} \alpha_j \\ \beta_j \\ 1 \end{bmatrix} + \rho_j\, {}^{C_i}\mathbf{p}_{C_n} \right) && (6.12)\\
&= {}^{C_n}Z_j \begin{bmatrix} h_{i1}(\alpha_j, \beta_j, \rho_j) \\ h_{i2}(\alpha_j.\beta_j, \rho_j) \\ h_{i3}(\alpha_j, \beta_j, \rho_j) \end{bmatrix} && (6.13)
\end{aligned}
$$

with

$$\alpha_j = \frac{{}^{C_n}X_j}{{}^{C_n}Z_j} \tag{6.14}$$

$$\beta_j = \frac{{}^{C_n}Y_j}{{}^{C_n}Z_j} \tag{6.15}$$

$$\rho_j = \frac{1}{{}^{C_n}Z_j} \tag{6.16}$$

where equation 6.13 is in inverse depth parametrization and α, β and ρ are the minimization variables.

The expected observation is

$$\hat{\mathbf{z}}_i^{(j)} = \begin{bmatrix} \hat{z}_{i,1}^{(j)} \\ \hat{z}_{i,2}^{(j)} \end{bmatrix} \tag{6.17}$$

$$= \frac{1}{h_{i3}(\hat{\alpha}_j, \hat{\beta}_j, \hat{\rho}_j)} \begin{bmatrix} h_{i1}(\hat{\alpha}_j, \hat{\beta}_j, \hat{\rho}_j) \\ h_{i2}(\hat{\alpha}_j, \hat{\beta}_j, \hat{\rho}_j) \end{bmatrix} \tag{6.18}$$

$$= \frac{1}{C_{33} + C_{31}\hat{\alpha}_j + C_{32}\hat{\beta}_j + p_z\hat{\rho}_j} \begin{bmatrix} C_{13} + C_{11}\hat{\alpha}_j + C_{12}\hat{\beta}_j + p_x\hat{\rho}_j \\ C_{23} + C_{21}\hat{\alpha}_j + C_{22}\hat{\beta}_j + p_y\hat{\rho}_j \end{bmatrix} \tag{6.19}$$

where C_{ij} and p_x are short forms of $C_{ij}({}^{C_i}_{C_n}q)$ and ${}^{C_i}p_{x,C_n}$ respectively.

The measurement is given by

$$\mathbf{z}_i^{(j)} = \begin{bmatrix} z_{i,1}^{(j)} \\ z_{i,2}^{(j)} \end{bmatrix} \tag{6.20}$$

$$= \frac{1}{h_{i3}(\alpha_j, \beta_j, \rho_j)} \begin{bmatrix} h_{i1}(\alpha_j, \beta_j, \rho_j) \\ h_{i1}(\alpha_j, \beta_j, \rho_j) \end{bmatrix} + \mathbf{n}_i^{(j)} \tag{6.21}$$

The goal of the optimization is to minimize the residual $r_i^{(j)}$ defined as

$$\mathbf{r}_i^{(j)} = \begin{bmatrix} r_{i,1}^{(j)} \\ r_{i,2}^{(j)} \end{bmatrix} = \mathbf{z}_i^{(j)} - \hat{\mathbf{z}}_i^{(j)} \tag{6.22}$$

The minimization algorithm terminates after a predefined number of iterations is reached or when the residual is small enough. The last estimate of the inverse depth parametrization is used to calculate the feature position in the global frame:

$${}^{G}\hat{\mathbf{p}}_{f_j} = \frac{1}{\hat{\rho}_j}\mathbf{C}^T({}^{C_n}_{G}\hat{q}) \begin{bmatrix} \hat{\alpha}_j \\ \hat{\beta}_j \\ 1 \end{bmatrix} + {}^{G}\hat{\mathbf{p}}_{C_n} \tag{6.23}$$

6.2.1 Newton (N)

The Newton update equation is

$$\begin{bmatrix} \hat{\alpha}_{k+1} \\ \hat{\beta}_{k+1} \\ \hat{\rho}_{k+1} \end{bmatrix} = \begin{bmatrix} \hat{\alpha}_k \\ \hat{\beta}_k \\ \hat{\rho}_k \end{bmatrix} - (\mathbf{H})^{-1}\mathbf{J}^T\mathbf{r} = \begin{bmatrix} \hat{\alpha}_k \\ \hat{\beta}_k \\ \hat{\rho}_k \end{bmatrix} - (\mathbf{J}^T\mathbf{J} + \mathbf{F}\mathbf{r})^{-1}\mathbf{J}^T\mathbf{r} \tag{6.24}$$

where $\mathbf{J}$ is the Jacobian and $\mathbf{H}$ is the Hessian of the residual $\mathbf{r}$. The Jacobian is given by

$$\mathbf{J}_i^{(j)} = \begin{bmatrix} \frac{\partial r_{i,1}^{(j)}}{\partial \hat{\alpha}_j} & \frac{\partial r_{i,1}^{(j)}}{\partial \hat{\beta}_j} & \frac{\partial r_{i,1}^{(j)}}{\partial \hat{\rho}_j} \\ \frac{\partial r_{i,2}^{(j)}}{\partial \hat{\alpha}_j} & \frac{\partial r_{i,2}^{(j)}}{\partial \hat{\beta}_j} & \frac{\partial r_{i,2}^{(j)}}{\partial \hat{\rho}_j} \end{bmatrix} \tag{6.25}$$

$$= - \begin{bmatrix} \frac{\partial \hat{z}_{i,1}^{(j)}}{\partial \hat{\alpha}_j} & \frac{\partial \hat{z}_{i,1}^{(j)}}{\partial \hat{\beta}_j} & \frac{\partial \hat{z}_{i,1}^{(j)}}{\partial \hat{\rho}_j} \\ \frac{\partial \hat{z}_{i,2}^{(j)}}{\partial \hat{\alpha}_j} & \frac{\partial \hat{z}_{i,2}^{(j)}}{\partial \hat{\beta}_j} & \frac{\partial \hat{z}_{i,2}^{(j)}}{\partial \hat{\rho}_j} \end{bmatrix} \tag{6.26}$$

To be specific, the Jacobian $\mathbf{J}_i^{(j)}$ is given by

$$\mathbf{J}_i^{(j)} = -\frac{\partial \hat{\mathbf{z}}_i^{(j)}}{\partial [\hat{\alpha}_j, \hat{\beta}_j, \hat{\rho}_j]^T} \tag{6.27}$$

$$= -\frac{\partial \hat{\mathbf{z}}_i^{(j)}}{\partial [\hat{h}_{i1}, \hat{h}_{i2}, \hat{h}_{i3}]^T} \frac{\partial [\hat{h}_{i1}, \hat{h}_{i2}, \hat{h}_{i3}]^T}{\partial [\hat{\alpha}_j, \hat{\beta}_j, \hat{\rho}_j]^T} \tag{6.28}$$

expanding into two factors using the chain law. The first factor is given by

$$\frac{\partial \hat{\mathbf{z}}_i^{(j)}}{\partial [\hat{h}_{i1}, \hat{h}_{i2}, \hat{h}_{i3}]^T} = \frac{\partial}{\partial [\hat{h}_{i1}, \hat{h}_{i2}, \hat{h}_{i3}]^T} \left(\frac{1}{\hat{h}_{i3}} \begin{bmatrix} \hat{h}_{i1} \\ \hat{h}_{i2} \end{bmatrix} \right) \tag{6.29}$$

$$= \begin{bmatrix} \frac{1}{\hat{h}_{i3}} & 0 & -\frac{\hat{h}_{i1}}{\hat{h}_{i3}^2} \\ 0 & \frac{1}{\hat{h}_{i3}} & -\frac{\hat{h}_{i2}}{\hat{h}_{i3}^2} \end{bmatrix} \tag{6.30}$$

$$= \frac{1}{\hat{h}_{i3}} \begin{bmatrix} \mathbf{I}_2 & -\hat{\mathbf{z}}_i \end{bmatrix}_{2\times 3} \tag{6.31}$$

and the right factor is calculated as

$$\frac{\partial [\hat{h}_{i1}, \hat{h}_{i2}, \hat{h}_{i3}]^T}{\partial [\hat{\alpha}_j, \hat{\beta}_j, \hat{\rho}_j]^T} = \frac{\partial}{\partial [\hat{\alpha}_j, \hat{\beta}_j, \hat{\rho}_j]^T} \left(\mathbf{C}({}^{C_i}_{C_n}\hat{\bar{q}}) \begin{bmatrix} \hat{\alpha}_j \\ \hat{\beta}_j \\ 1 \end{bmatrix} + \hat{\rho}_j {}^{C_i}\hat{\mathbf{p}}_{C_n} \right) \tag{6.32}$$

$$= \begin{bmatrix} \mathbf{C}({}^{C_i}_{C_n}\hat{\bar{q}}) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & \mathbf{C}({}^{C_i}_{C_n}\hat{\bar{q}}) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} & {}^{C_i}\hat{\mathbf{p}}_{C_n} \end{bmatrix}_{3\times 3} \tag{6.33}$$

$$=: \begin{bmatrix} \mathbf{J}_\alpha & \mathbf{J}_\beta & \mathbf{J}_\rho \end{bmatrix} \tag{6.34}$$

The Jacobian $\mathbf{J}_i^{(j)}$ has dimensions 2×3 and is given by

$$\mathbf{J}_i^{(j)} = -\frac{1}{\hat{h}_{i3}} \begin{bmatrix} \mathbf{I}_2 & -\hat{\mathbf{z}}_i \end{bmatrix} \begin{bmatrix} \mathbf{J}_\alpha & \mathbf{J}_\beta & \mathbf{J}_\rho \end{bmatrix} \tag{6.35}$$

In the Newton update equation 6.24, $\mathbf{F}$ is a 3-dimensional array and $\mathbf{Fr}$ can be rewritten as [1,2]

$$\mathbf{Fr} = \sum_i (r_i)_{\mathbf{pp}} r_i \tag{6.36}$$

that is, since $\mathbf{r}$ is a 2-dimensional vector:

$$\mathbf{Fr} = \begin{bmatrix} \frac{\partial^2 r_1}{\partial\hat{\alpha}\partial\hat{\alpha}} r_1 + \frac{\partial^2 r_2}{\partial\hat{\alpha}\partial\hat{\alpha}} r_2 & \frac{\partial^2 r_1}{\partial\hat{\alpha}\partial\hat{\beta}} r_1 + \frac{\partial^2 r_2}{\partial\hat{\alpha}\partial\hat{\beta}} r_2 & \frac{\partial^2 r_1}{\partial\hat{\alpha}\partial\hat{\rho}} r_1 + \frac{\partial^2 r_2}{\partial\hat{\alpha}\partial\hat{\rho}} r_2 \\ \frac{\partial^2 r_1}{\partial\hat{\beta}\partial\hat{\alpha}} r_1 + \frac{\partial^2 r_2}{\partial\hat{\beta}\partial\hat{\alpha}} r_2 & \frac{\partial^2 r_1}{\partial\hat{\beta}\partial\hat{\beta}} r_1 + \frac{\partial^2 r_2}{\partial\hat{\beta}\partial\hat{\beta}} r_2 & \frac{\partial^2 r_1}{\partial\hat{\beta}\partial\hat{\rho}} r_1 + \frac{\partial^2 r_2}{\partial\hat{\beta}\partial\hat{\rho}} r_2 \\ \frac{\partial^2 r_1}{\partial\hat{\rho}\partial\hat{\alpha}} r_1 + \frac{\partial^2 r_2}{\partial\hat{\rho}\partial\hat{\alpha}} r_2 & \frac{\partial^2 r_1}{\partial\hat{\rho}\partial\hat{\beta}} r_1 + \frac{\partial^2 r_2}{\partial\hat{\rho}\partial\hat{\beta}} r_2 & \frac{\partial^2 r_1}{\partial\hat{\rho}\partial\hat{\rho}} r_1 + \frac{\partial^2 r_2}{\partial\hat{\rho}\partial\hat{\rho}} r_2 \end{bmatrix}_{3\times 3} \tag{6.37}$$

or in vector notation

$$\mathbf{Fr} = \begin{bmatrix} \mathbf{r}_{\alpha\alpha}^T \mathbf{r} & \mathbf{r}_{\alpha\beta}^T \mathbf{r} & \mathbf{r}_{\alpha\rho}^T \mathbf{r} \\ \mathbf{r}_{\beta\alpha}^T \mathbf{r} & \mathbf{r}_{\beta\beta}^T \mathbf{r} & \mathbf{r}_{\beta\rho}^T \mathbf{r} \\ \mathbf{r}_{\rho\alpha}^T \mathbf{r} & \mathbf{r}_{\rho\beta}^T \mathbf{r} & \mathbf{r}_{\rho\rho}^T \mathbf{r} \end{bmatrix}_{3\times 3} \tag{6.38}$$

The entries are given by

$$\mathbf{r}_{\mathbf{pp}} := \frac{\partial \mathbf{J}}{\partial \mathbf{p}^T} = \frac{\partial \mathbf{J}}{\partial \mathbf{h}} \cdot \frac{\partial \mathbf{h}}{\partial \mathbf{p}} \tag{6.39}$$

$$= \frac{\partial}{\partial \mathbf{h}} \left(-\frac{1}{h_3} \begin{bmatrix} \mathbf{I}_2 & -\mathbf{z} \end{bmatrix} \begin{bmatrix} \mathbf{J}_\alpha & \mathbf{J}_\beta & \mathbf{J}_\rho \end{bmatrix} \right) \cdot \begin{bmatrix} \mathbf{J}_\alpha & \mathbf{J}_\beta & \mathbf{J}_\rho \end{bmatrix} \tag{6.40}$$

$$= -\frac{\partial}{\partial \mathbf{h}} \underbrace{\left(\frac{1}{h_3} \begin{bmatrix} \mathbf{I}_2 & -\mathbf{z} \end{bmatrix} \right)}_{=:\mathbf{f}(h1,h2,h3)} \begin{bmatrix} \mathbf{J}_\alpha & \mathbf{J}_\beta & \mathbf{J}_\rho \end{bmatrix} \begin{bmatrix} \mathbf{J}_\alpha & \mathbf{J}_\beta & \mathbf{J}_\rho \end{bmatrix} \tag{6.41}$$

where $\mathbf{f_h}$ is a $(3 \times 2 \times 3)$-dimensional matrix with

$$\frac{\partial \mathbf{f}(h1,h2,h3)}{\partial h1} = \frac{1}{h_3^2} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{6.42}$$

$$\frac{\partial \mathbf{f}(h1,h2,h3)}{\partial h2} = \frac{1}{h_3^2} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.43}$$

$$\frac{\partial \mathbf{f}(h1,h2,h3)}{\partial h3} = \frac{1}{h_3^2} \begin{bmatrix} 1 & 0 & -2h_1 \\ 0 & 1 & -2h_2 \end{bmatrix} \tag{6.44}$$

The multiplication scheme with the multi-dimensional matrix is shown in figure 6.4.

The entries of the final matrix $\mathbf{Fr}$ are given by

First row:

$$\mathbf{r}_{\alpha\alpha}^T = \begin{bmatrix} 2\frac{J_\alpha(1)J_\alpha(3)}{h_3^2} - 2\frac{J_\alpha(3)^2 h_1}{h_3^3} & 2\frac{J_\alpha(2)J_\alpha(3)}{h_3^2} - 2\frac{J_\alpha(3)^2 h_2}{h_3^3} \end{bmatrix}_{1\times 2}$$

$$\mathbf{r}_{\alpha\beta}^T = \begin{bmatrix} \frac{J_\alpha(1)J_\beta(3)+J_\beta(1)J_\alpha(3)}{h_3^2} - 2\frac{J_\alpha(3)J_\beta(3)h_1}{h_3^3} & \frac{J_\alpha(2)J_\beta(3)+J_\beta(2)J_\alpha(3)}{h_3^2} - 2\frac{J_\alpha(3)J_\beta(3)h_2}{h_3^3} \end{bmatrix}_{1\times 2}$$

$$\mathbf{r}_{\alpha\rho}^T = \begin{bmatrix} \frac{J_\alpha(3)J_\rho(1)+J_\alpha(1)J_\rho(3)}{h_3^2} - 2\frac{J_\alpha(3)J_\rho(3)h_1}{h_3^3} & \frac{J_\alpha(3)J_\rho(2)+J_\alpha(2)J_\rho(3)}{h_3^2} - 2\frac{J_\alpha(3)J_\rho(3)h_2}{h_3^3} \end{bmatrix}_{1\times 2}$$

[1] See footnote in [20] on page 599.

[2] The parameter vector is defined as $\mathbf{p} = [\alpha, \beta, \rho]^T$

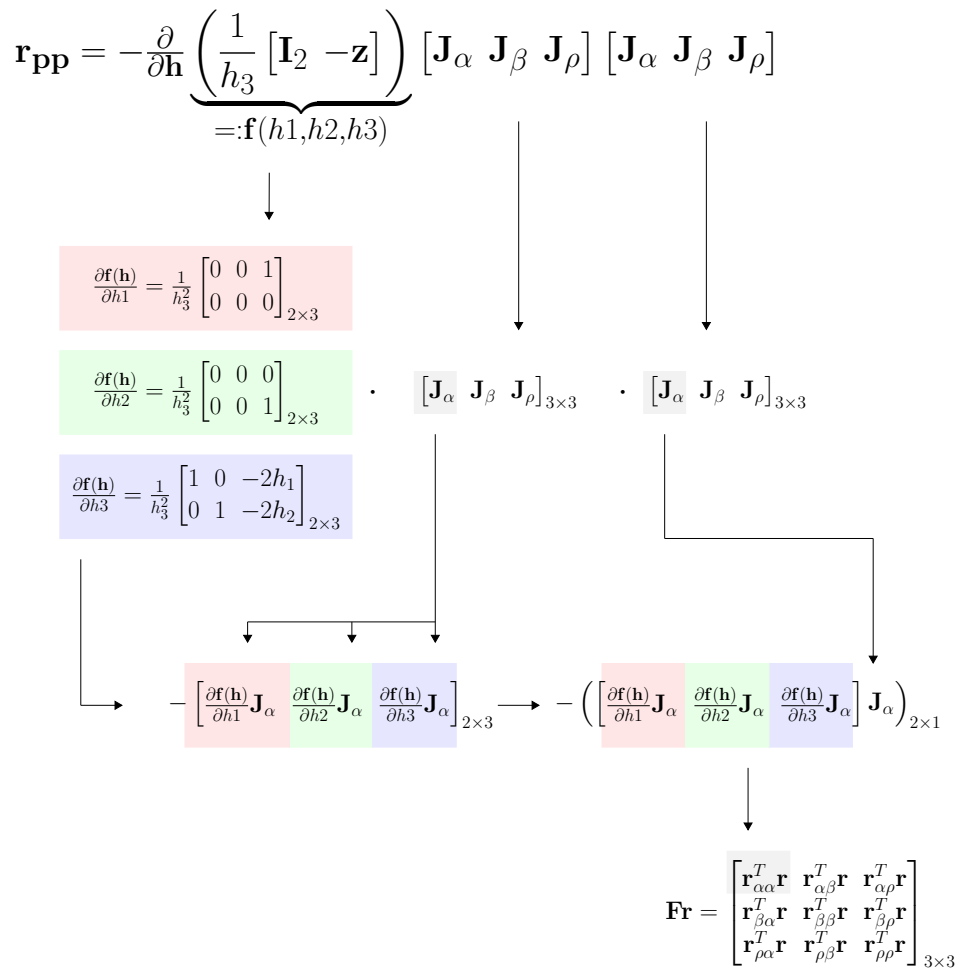


Figure 6.4: Detailed multiplication scheme for calculating the multi-dimensional matrix $\mathbf{F}$ and (3×3)-dimensional matrix $\mathbf{Fr}$.

Second row:

$$\mathbf{r}_{\beta\alpha}^T = \left[\frac{J_\alpha(1)J_\beta(3)+J_\beta(1)J_\alpha(3)}{h_3^2} - 2\frac{J_\alpha(3)J_\beta(3)h_1}{h_3^2} \quad \frac{J_\alpha(2)J_\beta(3)+J_\beta(2)J_\alpha(3)}{h_3^2} - 2\frac{J_\alpha(3)J_\beta(3)h_2}{h_3^2}\right]_{1\times2}$$

$$\mathbf{r}_{\beta\beta}^T = \left[2\frac{J_\beta(1)J_\beta(3)}{h_3^2} - 2\frac{J_\beta(3)^2h_1}{h_3^3} \quad 2\frac{J_\beta(2)J_\beta(3)}{h_3^2} - 2\frac{J_\beta(3)^2h_2}{h_3^3}\right]_{1\times2}$$

$$\mathbf{r}_{\beta\rho}^T = \left[\frac{J_\beta(3)J_\rho(1)+J_\beta(1)J_\rho(3)}{h_3^2} - 2\frac{J_\beta(3)J_\rho(3)h_1}{h_3^2} \quad \frac{J_\beta(3)J_\rho(2)+J_\beta(2)J_\rho(3)}{h_3^2} - 2\frac{J_\beta(3)J_\rho(3)h_2}{h_3^2}\right]_{1\times2}$$

Third row:

$$\mathbf{r}_{\rho\alpha}^T = \left[\frac{J_\alpha(3)J_\rho(1)+J_\alpha(1)J_\rho(3)}{h_3^2} - 2\frac{J_\alpha(3)J_\rho(3)h_1}{h_3^3} \quad \frac{J_\alpha(3)J_\rho(2)+J_\alpha(2)J_\rho(3)}{h_3^2} - 2\frac{J_\alpha(3)J_\rho(3)h_2}{h_3^3}\right]_{1\times2}$$

$$\mathbf{r}_{\rho\beta}^T = \left[\frac{J_\beta(3)J_\rho(1)+J_\beta(1)J_\rho(3)}{h_3^2} - 2\frac{J_\beta(3)J_\rho(3)h_1}{h_3^3} \quad \frac{J_\beta(3)J_\rho(2)+J_\beta(2)J_\rho(3)}{h_3^2} - 2\frac{J_\beta(3)J_\rho(3)h_2}{h_3^3}\right]_{1\times2}$$

$$\mathbf{r}_{\rho\rho}^T = \left[2\frac{J_\rho(1)J_\rho(3)}{h_3^2} - 2\frac{J_\rho(3)^2h_1}{h_3^3} \quad 2\frac{J_\rho(2)J_\rho(3)}{h_3^2} - 2\frac{J_\rho(3)^2h_2}{h_3^3}\right]_{1\times2}$$

Note that $\mathbf{r}_{\alpha\beta} = \mathbf{r}_{\beta\alpha}$, $\mathbf{r}_{\alpha\rho} = \mathbf{r}_{\rho\alpha}$ and $\mathbf{r}_{\beta\rho} = \mathbf{r}_{\rho\beta}$. This property of matrix $\mathbf{Fr}$ is

exploited in the implementation to speed up the calculation of the update equation. See algorithm 10 for the Newton pseudo-code.

6.2.2 Gauss-Newton (GN)

The Gauss-Newton approximates the Hessian $\mathbf{H}$ by $\mathbf{J}^T\mathbf{J}$ to reduce the computational complexity. The update equation simplifies to

$$\begin{bmatrix} \hat{\alpha}_{k+1} \\ \hat{\beta}_{k+1} \\ \hat{\rho}_{k+1} \end{bmatrix} = \begin{bmatrix} \hat{\alpha}_k \\ \hat{\beta}_k \\ \hat{\rho}_k \end{bmatrix} - (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{r} \tag{6.45}$$

The approximation is legitimate close to the global minimum, but may result in convergence to a local minimum otherwise. See algorithm 8 for the Gauss-Newton pseudo-code.

6.2.3 Gradient descent (GD)

The Hessian matrix is approximated by a scalar γ times the identity matrix. The update equation is given by

$$\begin{bmatrix} \hat{\alpha}_{k+1} \\ \hat{\beta}_{k+1} \\ \hat{\rho}_{k+1} \end{bmatrix} = \begin{bmatrix} \hat{\alpha}_k \\ \hat{\beta}_k \\ \hat{\rho}_k \end{bmatrix} - \gamma\mathbf{J}^T\mathbf{r} \tag{6.46}$$

Gradient descent is fast but usually shows poor convergence. See algorithm 9 for the gradient descent pseudo-code.

6.2.4 Levenberg-Marquardt (LM)

The Levenberg-Marquardt minimization combines the good convergence properties of the Gauss-Newton algorithm with the speed of gradient descent. The update equation is equivalent to Gauss-Newton's except for the regularization factor λ times the identity matrix:

$$\begin{bmatrix} \hat{\alpha}_{k+1} \\ \hat{\beta}_{k+1} \\ \hat{\rho}_{k+1} \end{bmatrix} = \begin{bmatrix} \hat{\alpha}_k \\ \hat{\beta}_k \\ \hat{\rho}_k \end{bmatrix} - (\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^T\mathbf{r} \tag{6.47}$$

If the estimates for α, β, ρ

- *reduce* the residual $\mathbf{r}$ then α, β, ρ is accepted and λ is decreased (e.g. $\lambda = \lambda \cdot 0.1$). The algorithm continues with the next iteration. Note that for $\lambda\mathbf{I} \ll \mathbf{J}^T\mathbf{J}$ Levenberg-Marquardt approximates Gauss-Newton.
- *increase* the residual $\mathbf{r}$ then α, β, ρ is rejected and λ is increased (e.g. $\lambda = \lambda \cdot 10$). The update equation is solved again with the increased λ until the residual is decreased. Note that for $\lambda\mathbf{I} \gg \mathbf{J}^T\mathbf{J}$ Levenberg-Marquardt approximates gradient descent.

See algorithm 11 for the Levenberg-Marquardt pseudo-code.

Algorithm 8 Gauss-Newton multi-view triangulation,
T : precision threshold
γ : step size (set to 1)
r : measurement residual
J : measurement jacobian
$h_{m,i}$: observation of feature j in camera i
$h_{p,i}$: predicted observation of feature j in camera i
$M^{(j)}$: number of observations for feature j

function GaussNewton(α_{init}, β_{init}, ρ_{init}, $iter_{max}$, T)
  $\alpha \leftarrow \alpha_{init}$
  $\beta \leftarrow \beta_{init}$
  $\rho \leftarrow \rho_{init}$
  while $\|r_{last} - r_{current}\|_2 > T$ and $iter < iter_{max}$ **do**
    for $i = 1$: all camera views $M^{(j)}$ **do**
      ${}^{C_n}\mathbf{C}_{C_i} \leftarrow {}^{C_n}\mathbf{C}_G \cdot {}^{G}\mathbf{C}_{C_i}$
      ${}^{C_i}\mathbf{p}_{C_n} \leftarrow {}^{C_i}\mathbf{C}_G {}^{G}\mathbf{p}_{C_n} - {}^{C_i}\mathbf{C}_G {}^{G}\mathbf{p}_{C_n}$
      $h_{p,i} \leftarrow {}^{C_n}\mathbf{C}_{C_i} \cdot \begin{bmatrix}\alpha & \beta & 1\end{bmatrix}^T + \rho \cdot {}^{C_i}\mathbf{p}_{C_n}$
      $h_{m,i} = \frac{1}{Z_{C,m}} \begin{bmatrix}X_{C,m} & Y_{C,m}\end{bmatrix}^T$
      $r_i \leftarrow h_{p,i} - h_{m,i}$ ▷ Residuals in camera coordinates
      $r \leftarrow \begin{bmatrix}r & r_i\end{bmatrix}^T$ ▷ Stack residuals
      $J_p \leftarrow \frac{1}{h_{p,i}(3)} \begin{bmatrix}1 & 0 & -\frac{h_{p,i}(1)}{h_{p,i}(3)} \\ 0 & 1 & -\frac{h_{p,i}(2)}{h_{p,i}(3)}\end{bmatrix}$ ▷ Jacobian perspective camera model
      $J_\alpha \leftarrow {}^{C_n}\mathbf{C}_{C_i} \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$
      $J_\beta \leftarrow {}^{C_n}\mathbf{C}_{C_i} \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$
      $J_\rho \leftarrow {}^{C_i}\mathbf{p}_{C_n}$
      $J_i \leftarrow J_p \begin{bmatrix}J_\alpha & J_\beta & J_\rho\end{bmatrix}$ ▷ $J_i \equiv \begin{bmatrix}r_\alpha & r_\beta & r_\rho\end{bmatrix}$
      $J \leftarrow \begin{bmatrix}J & J_i\end{bmatrix}^T$ ▷ Stack Jacobians
    end for
    $\Delta \leftarrow (J^T J)^{-1} J^T r$ ▷ Dim.: $J = 2M^{(j)} \times 3$, $r = 2M^{(j)} \times 1$
    $\begin{bmatrix}\alpha & \beta & \rho\end{bmatrix}^T \leftarrow \begin{bmatrix}\alpha & \beta & \rho\end{bmatrix}^T - \gamma\Delta$
  end while
  ${}^{G}\mathbf{p}_{fj} \leftarrow \frac{1}{\rho} {}^{G}\mathbf{C}_{C_n} \begin{bmatrix}\alpha & \beta & 1\end{bmatrix}^T + {}^{G}\mathbf{p}_{C_n}$
end function

6.3 Simulation

Questions:

- Which and how many camera views should be included in the EKF state?
- How many iterations are needed/sufficient?
- Which algorithm yields the best trade-off between speed and performance?

6.3.1 Simulation A: Constant distance, varying number of camera poses

The first simulation setup is shown in figure 6.5 and 6.6: Figure 6.5 shows the straight camera trajectory with 25 camera frames. The camera x-, y- and z-axis are illustrated by the blue, green and red lines respectively. The camera moves with a pure translation motion for 0.7 m in a height of 1 m. The 500 features are distributed randomly in a predefined volume near the ground ($z = 0$). All features are seen from all 25 camera frames.

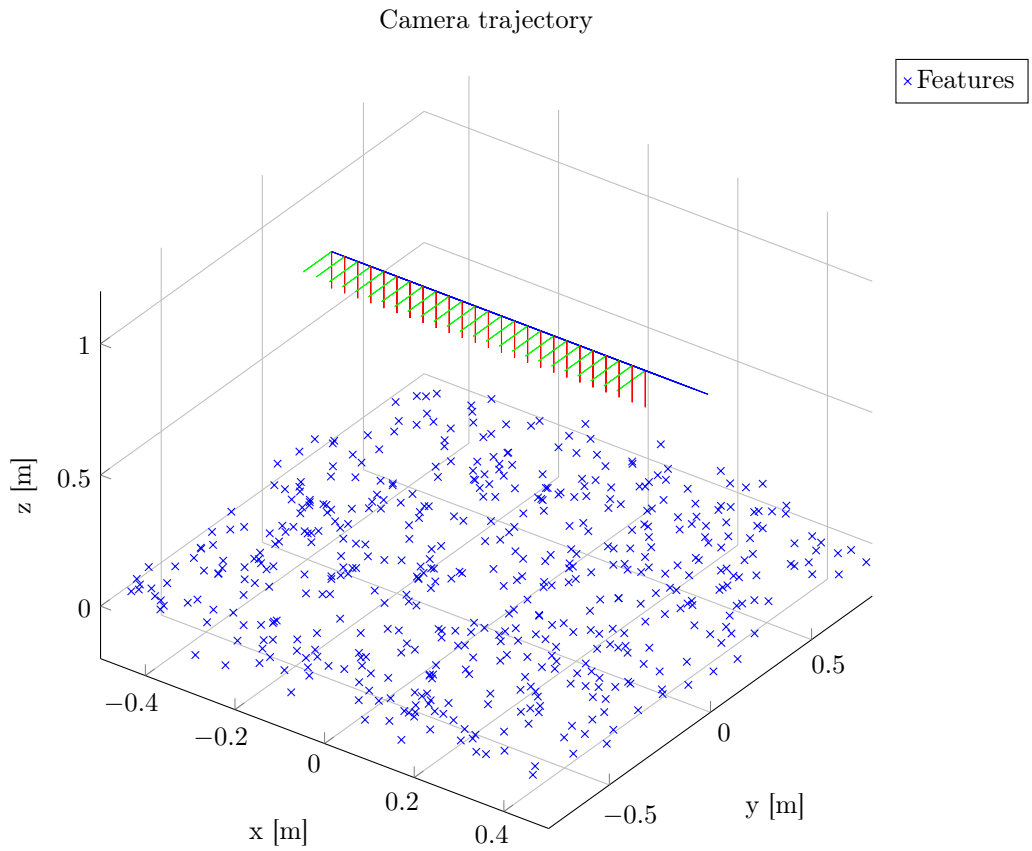


Figure 6.5: Straight camera trajectory with 25 camera frames which are 1 meter above ground simulated with 500 features. The camera x-, y- and z-axis are illustrated by the blue, green and red lines respectively.

Figure 6.6 shows the placement of the 25, 9, 5, 3 and 2 camera frames used for triangulating the features: The first and 25th camera frame are always used, that is

the baseline of 0.7 m stays constant and only the number of camera views in between change.

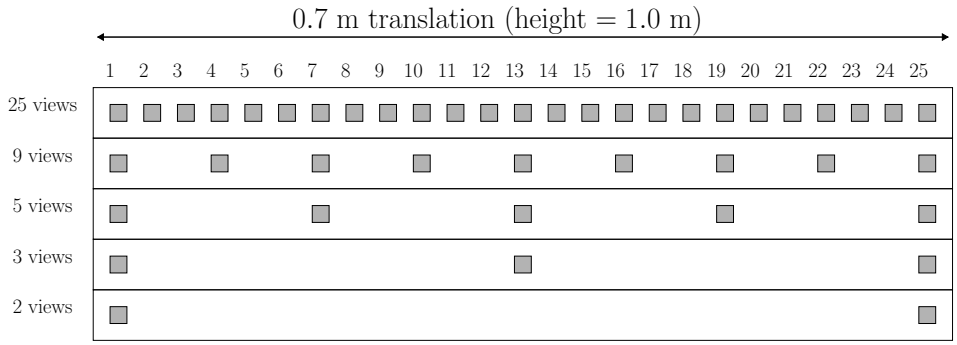


Figure 6.6: Simulation A. Triangulation setup for section 6.3.1: The distance stays constant, only the number of views in between (equally distributed) change.

The following approaches were analzyed

- Newton (N, red)
- Gauss-Newton (GN, blue)
- Levenberg-Marquardt (LM, black)
- Gradient descent (GD, green)
- Gradient descent with line search (GDLS, dark green)

based on

- number of iterations
- 3D-position error of the 500 features
- number of camera views and
- runtime.

The dependence of the number of camera views on the 3D position error is plotted in figure 6.7. Gradient descent with a fixed step size and with line search performs poorly: The triangulation error is several magnitudes higher than for Newton, Gauss-Newton and Levenberg-Marquardt[3] but gradient descent is only around double as fast as the Gauss-Newton approach (see figure 6.8). Therefore, in order to obtain the same accuracy, the gradient descent based approaches would need more total computation time. Gradient descent with fixed step size and with line search will therefore be neglected in the further analysis.[4]

While Newton, Gauss-Newton and Levenberg-Marquardt have similar error plots, their runtime is distinct: In figure 6.8 one can see the average runtime per feature in microseconds versus the number of iterations on the x-axis. For 20 iterations and 9 camera views, the Newton, Gauss-Newton and Levenberg-Marquardt approach

[3]Newton, Gauss-Newton and Levenberg-Marquardt are shown as the black line in figure 6.7 (plotted above each other)

[4]Please refer to appendix C for more simulation plots including gradient descent based approaches.

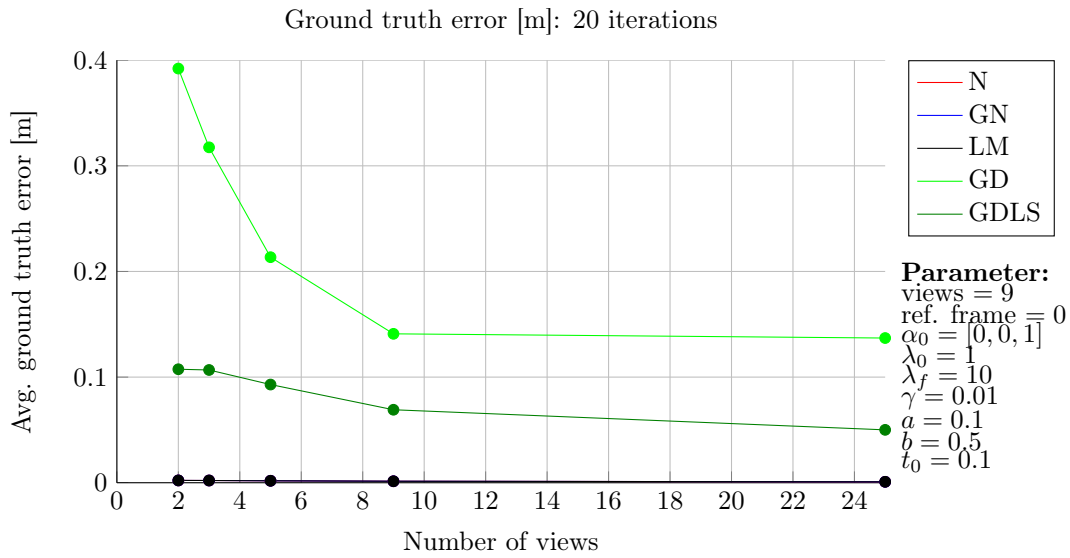


Figure 6.7: Simulation A. Average 3D-position error depending on the number of views for 20 iterations.

need around 20, 30 and 40 μs per feature respectively. N.b. that the runtime depends on the number of views as well.

In figure 6.9, the triangulation error of the Newton, Gauss-Newton and Levenberg-Marquardt approach is plotted for various number of camera frames and for one, three, seven and ten iterations. N, GN and LM have comparable error curves; in fact, after around 5 iterations their accuracy is identical as shown in figure 6.9 and table 6.1.

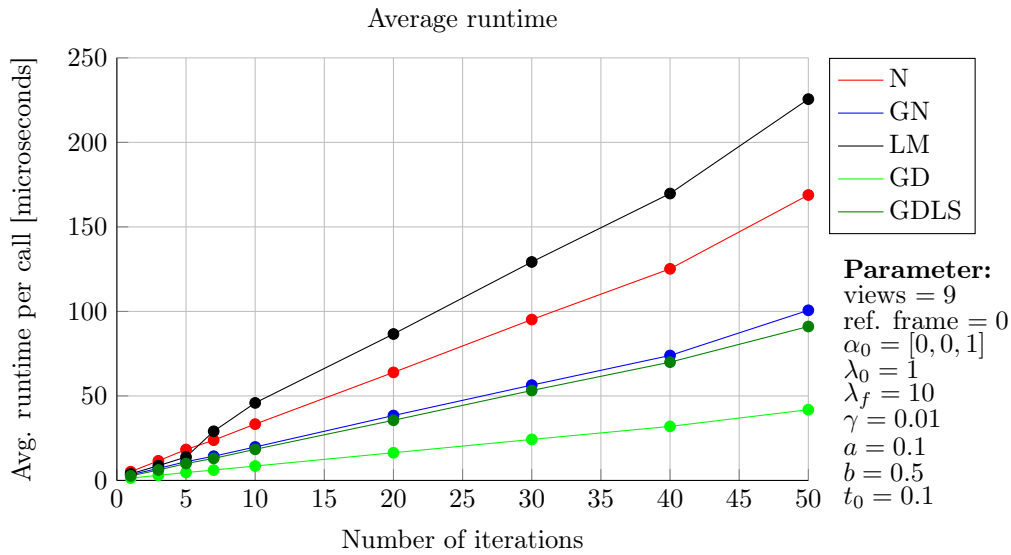


Figure 6.8: Simulation A. Average runtime per feature plotted versus the number of iterations. For 10 iterations and 9 camera views, the Gauss-Newton approach needs around 20 μs per feature. N.b. that the runtime depends on the number of views as well (see for-loop in pseudo code and appendix).

Levenberg-Marquardt needs some iterations to attain the performance of Gauss-Newton and Newton. This is because it is a mixture between Gradient Descent and Gauss-Newton. The pure Gauss-Newton approach obtains therefore the best trade-off in terms of speed and performance and is used as the triangulation method in later chapters.

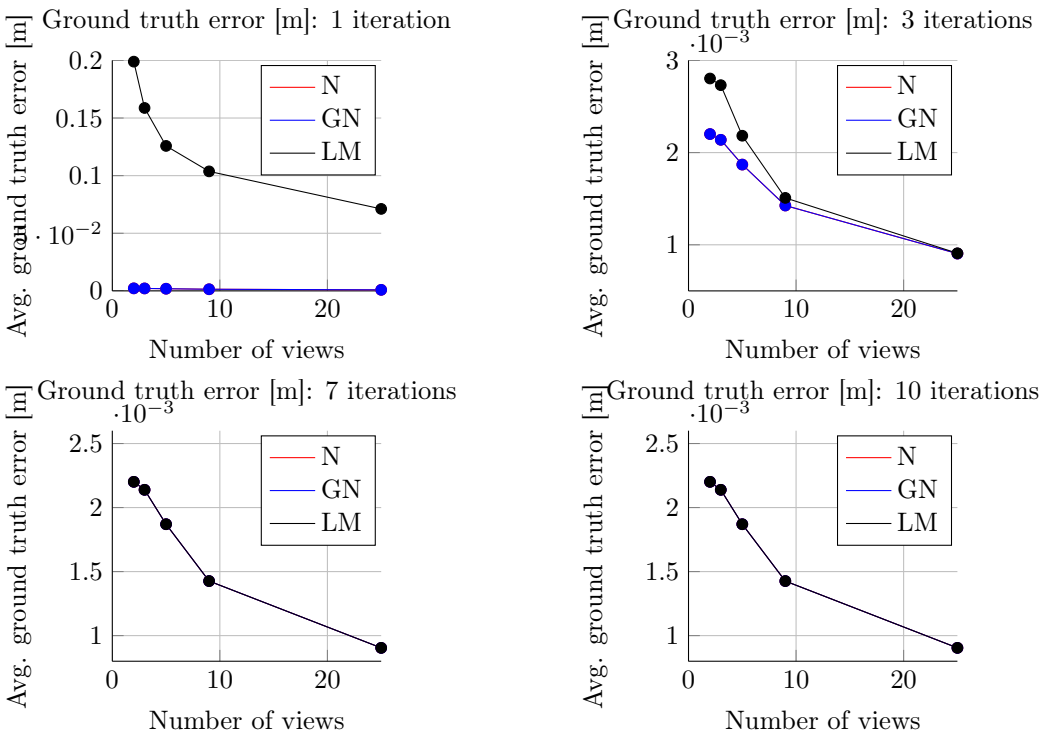


Figure 6.9: Simulation A. Average 3D-position error depending on the number of views and number of iterations. Note that the y-scale is different in the four plots above.

6.3.2 Simulation B: Varying distance, constant number of camera poses

Simulation B investigates the influence of the baseline length on the error curve while the number of camera poses is held constant[5] trying to answer the question which camera poses should be included in the EKF state vector. The triangulation setup is visualized in figure 6.10: The baseline changes from 0.175 m, to 0.35 m up to 0.7 m.

Figure 6.11 shows the 3d triangulation error for increasing distance between the first and last camera frame for three and five iterations of N, GN and LM. The plots are based on the data given in table 6.2. The Newton and Gauss-Newton approaches converge after only one iteration. Doubling the baseline from 0.175 m to 0.35 m corresponds to an error decrease of around 47 %; doubling the baseline from 0.35 m to 0.7 m corresponds to an error drop of around 42 %. Levenberg-Marquardt achieves this error after around five iterations as observed in section 6.3.2.

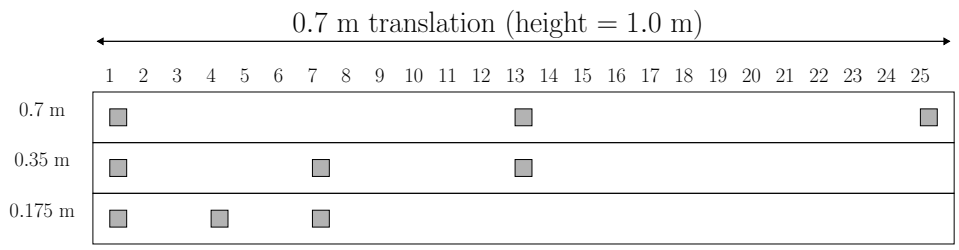


Figure 6.10: Simulation B. Triangulation setup for section 6.3.2: The number of camera frames stays constant, but the distance between two consecutive camera frames are halved.

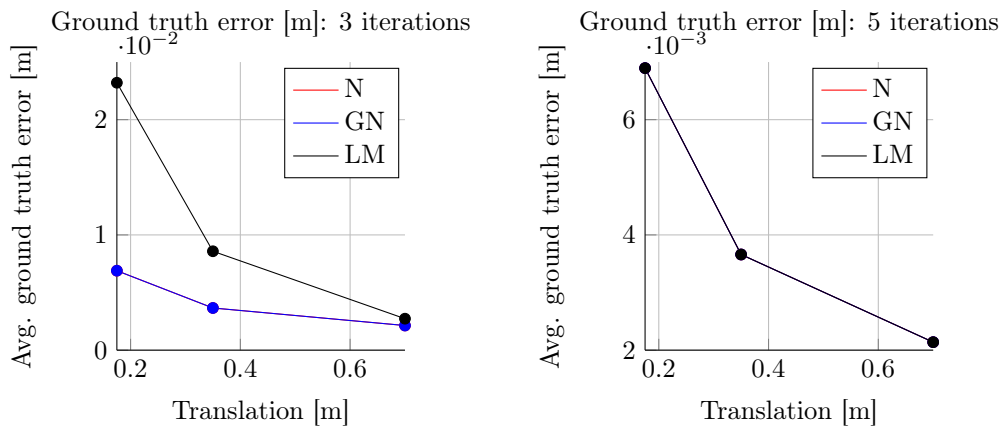


Figure 6.11: Simulation B. Average 3D-position error depending on the number of views and number of iterations. Note that the y-scale is different in the four plots above.

[5] In this section, the number of camera poses is three while the location of the poses changes.

6.3.3 Simulation C: Varying distance, varying number of camera poses

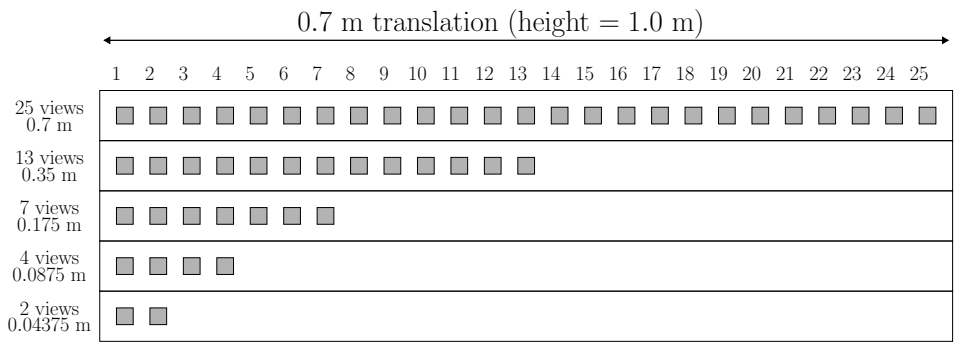


Figure 6.12: Simulation C. Triangulation setup for section 6.3.3: The distance between is halved and every available camera frame for the translation is used.

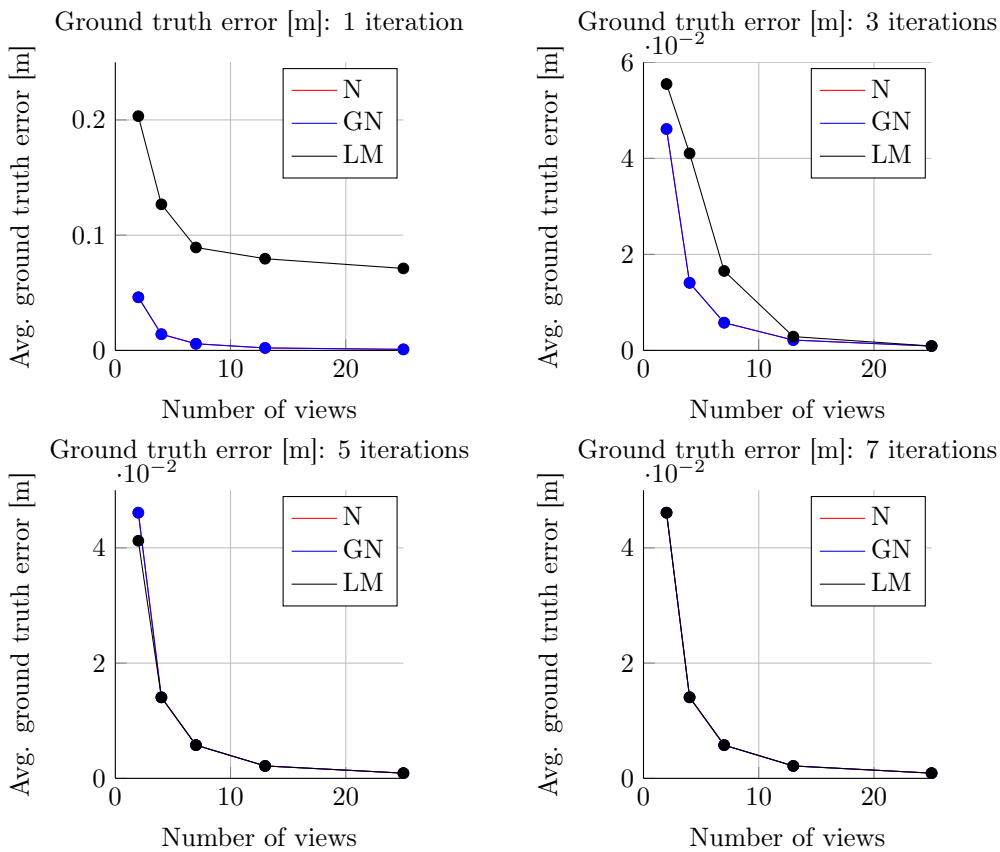


Figure 6.13: Simulation C. Average 3D-position error depending on the number of views and number of iterations. Note that the y-scale is different in the four plots above.

The triangulation setup for simulation C is shown in 6.10: Starting from a baseline of 4.375 cm the baseline is doubled until 0.7 m. In contrast to simulation B, the number of camera views is not kept constant but every available camera frame in between the baseline is used for feature triangulation. Thus for every setup the

baseline doubles and the number of camera views increases.

As shown in figure 6.13 and table 6.3, doubling the baseline corresponds to an error decrease of 70 %, 59 %, 62 % and 58 % respectively.

N	**niters=1**	**niters=3**	**niters=5**	**niters=7**	**niters=10**	**niters=50**
nviews=1	0.00220	0.00220	0.00220	0.00220	0.00220	0.00220
nviews=2	0.00214	0.00214	0.00214	0.00214	0.00214	0.00214
nviews=5	0.00187	0.00187	0.00187	0.00187	0.00187	0.00187
nviews=9	0.00143	0.00143	0.00143	0.00143	0.00143	0.00143
nviews=25	0.00090	0.00090	0.00090	0.00090	0.00090	0.00090
GN	**niters=1**	**niters=3**	**niters=5**	**niters=7**	**niters=10**	**niters=50**
nviews=1	0.00220	0.00220	0.00220	0.00220	0.00220	0.00220
nviews=2	0.00214	0.00214	0.00214	0.00214	0.00214	0.00214
nviews=5	0.00187	0.00187	0.00187	0.00187	0.00187	0.00187
nviews=9	0.00143	0.00143	0.00143	0.00143	0.00143	0.00143
nviews=25	0.00090	0.00090	0.00090	0.00090	0.00090	0.00090
LM	**niters=1**	**niters=3**	**niters=5**	**niters=7**	**niters=10**	**niters=50**
nviews=1	0.19892	0.00280	0.00220	0.00220	0.00220	0.00220
nviews=2	0.15873	0.00273	0.00214	0.00214	0.00214	0.00214
nviews=5	0.12579	0.00218	0.00187	0.00187	0.00187	0.00187
nviews=9	0.10370	0.00151	0.00143	0.00143	0.00143	0.00143
nviews=25	0.07114	0.00091	0.00090	0.00090	0.00090	0.00090

Table 6.1: Simulation A. Triangulation error depending on the number of camera views and iterations for Newton, Gauss-Newton and Levenberg-Marquardt.

N	**niters=1**	**niters=3**	**niters=5**	**niters=7**	**niters=10**	**niters=50**
$\Delta x = 0.175$	0.00689	0.00689	0.00689	0.00689	0.00689	0.00689
$\Delta x = 0.35$	0.00366	0.00366	0.00366	0.00366	0.00366	0.00366
$\Delta x = 0.7$	0.00214	0.00214	0.00214	0.00214	0.00214	0.00214
GN	**niters=1**	**niters=3**	**niters=5**	**niters=7**	**niters=10**	**niters=50**
$\Delta x = 0.175$	0.00689	0.00689	0.00689	0.00689	0.00689	0.00689
$\Delta x = 0.35$	0.00366	0.00366	0.00366	0.00366	0.00366	0.00366
$\Delta x = 0.7$	0.00214	0.00214	0.00214	0.00214	0.00214	0.00214
LM	**niters=1**	**niters=3**	**niters=5**	**niters=7**	**niters=10**	**niters=50**
$\Delta x = 0.175$	0.15072	0.02320	0.00689	0.00689	0.00689	0.00689
$\Delta x = 0.35$	0.14950	0.00858	0.00366	0.00366	0.00366	0.00366
$\Delta x = 0.7$	0.15873	0.00273	0.00214	0.00214	0.00214	0.00214

Table 6.2: Simulation B. Triangulation error depending on the number of camera views and iterations for Newton, Gauss-Newton and Levenberg-Marquardt.

N	**niters=1**	**niters=3**	**niters=5**	**niters=7**	**niters=10**	**niters=50**
nviews=2	0.04610	0.04610	0.04610	0.04610	0.04610	0.04610
nviews=4	0.01407	0.01407	0.01407	0.01407	0.01407	0.01407
nviews=7	0.00577	0.00577	0.00577	0.00577	0.00577	0.00577
nviews=13	0.00215	0.00215	0.00215	0.00215	0.00215	0.00215
nviews=25	0.00090	0.00090	0.00090	0.00090	0.00090	0.00090
GN	**niters=1**	**niters=3**	**niters=5**	**niters=7**	**niters=10**	**niters=50**
nviews=2	0.04610	0.04610	0.04610	0.04610	0.04610	0.04610
nviews=4	0.01407	0.01407	0.01407	0.01407	0.01407	0.01407
nviews=7	0.00577	0.00577	0.00577	0.00577	0.00577	0.00577
nviews=13	0.00215	0.00215	0.00215	0.00215	0.00215	0.00215
nviews=25	0.00090	0.00090	0.00090	0.00090	0.00090	0.00090
LM	**niters=1**	**niters=3**	**niters=5**	**niters=7**	**niters=10**	**niters=50**
nviews=2	0.20326	0.05548	0.04123	0.04610	0.04610	0.04610
nviews=4	0.12677	0.04103	0.01405	0.01407	0.01407	0.01407
nviews=7	0.08932	0.01654	0.00577	0.00577	0.00577	0.00577
nviews=13	0.07956	0.00288	0.00215	0.00215	0.00215	0.00215
nviews=25	0.07114	0.00091	0.00090	0.00090	0.00090	0.00090

Table 6.3: Simulation C. Triangulation error depending on the number of camera views and iterations for Newton, Gauss-Newton and Levenberg-Marquardt.

6.4 Experiments

The two experiments in this section show the process from simulation to real world measurements: In experiment A the camera poses as well as the feature positions are measured manually. In experiment B the features are extracted automatically and the camera poses are estimated using the `ethzasl_ssf` & Parallel Tracking And Mapping[6] filter framework.

6.4.1 Experiment A: Camera poses measured, features tracked manually

In this experiment, the simulated and real triangulation is compared. The triangulation setup is shown in figure 6.14: The camera mounted on the quadrotor is placed on a table in around 0.7 m height and an image of the checkerboard pattern parallel to the table is recorded. Then the camera is translated on the table in x-direction to increase the baseline and the next image is taken. The four corner features of each black square were tracked manually (see images in figure 6.15) and the static camera poses were measured by hand.

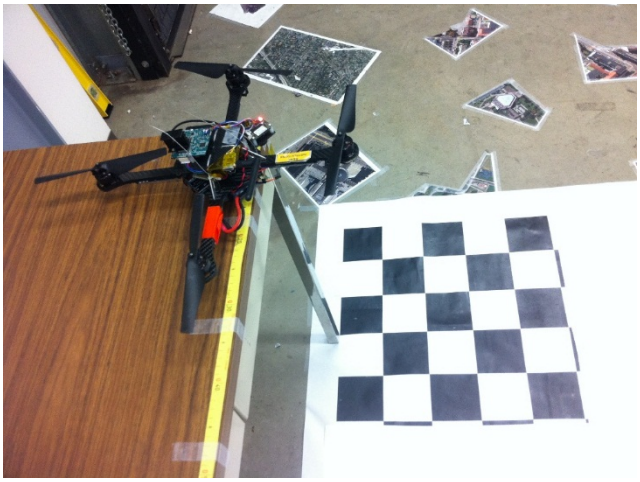


Figure 6.14: Experiment A. Multi-view triangulation setup

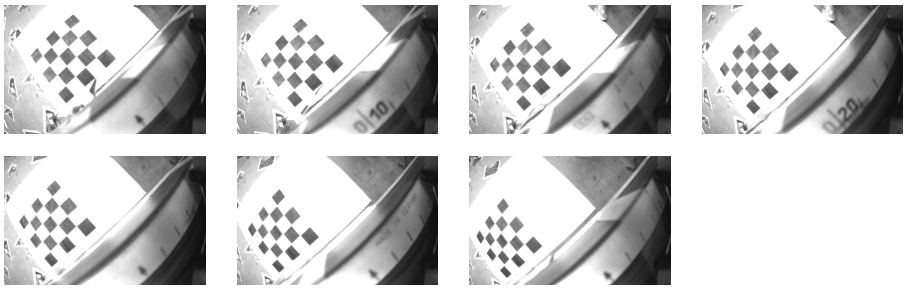


Figure 6.15: Experiment A. Seven images taken from static camera poses. In each image the four corners of each black square are tracked manually.

These real observations using static camera poses are then compared to the results obtained from simulation in figure 6.16 The figure shows the increasing baseline

[6] In the following denoted as SSF/PTAM.

from top to bottom (5 cm, 15 cm, 30 cm); on the left the top view and on the right the side view of the checkerboard pattern. The red points are triangulated points obtained from simulation, the blue points obtained from real camera measurements taken from static camera poses which were measured by hand.

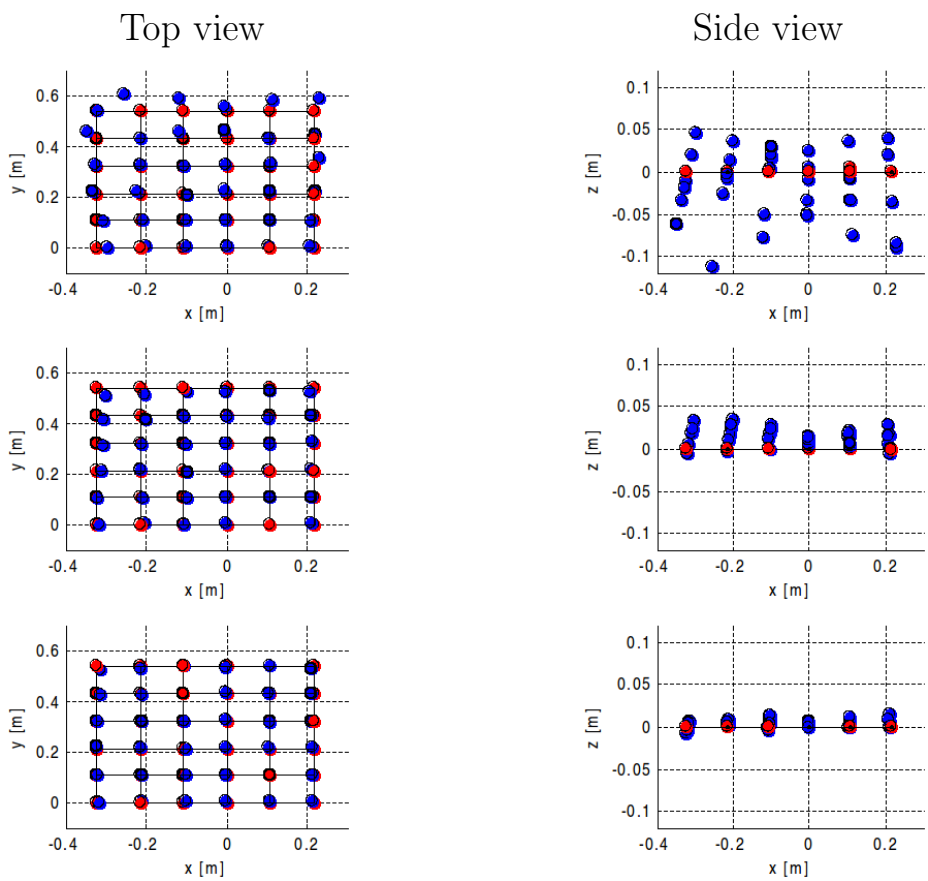


Figure 6.16: Experiment A. Increasing baseline from top to bottom (5cm, 15 cm, 30 cm), left: top view, right: side view. The red points are triangulated points obtained from simulation, the blue points obtained from real camera measurements and by measuring the static camera poses by hand

Figure 6.17 confirms the real world measurements have the error curve as expected from simulation: Both the simulation error as well as the real world error drop for increasing baseline. Note, however, that tripling the baseline from 5 cm to 15 cm leads to an error drop of around 74 % in simulation but only of around 53 % in the real world. The discrepancy can be explained by an accumulation of error sources:

- manual feature selection (approximately one pixel accuracy)
- manual camera pose measurements (deviations of up to 5 cm are likely)
- systematic model errors of the camera

6.4.2 Experiment B: Camera poses estimated, features tracked automatically / Outlier rejection

In this experiment, the camera poses are estimated by the SSF/PTAM filter framework and then used in the MSC-EKF framework for triangulation. The MSC filter

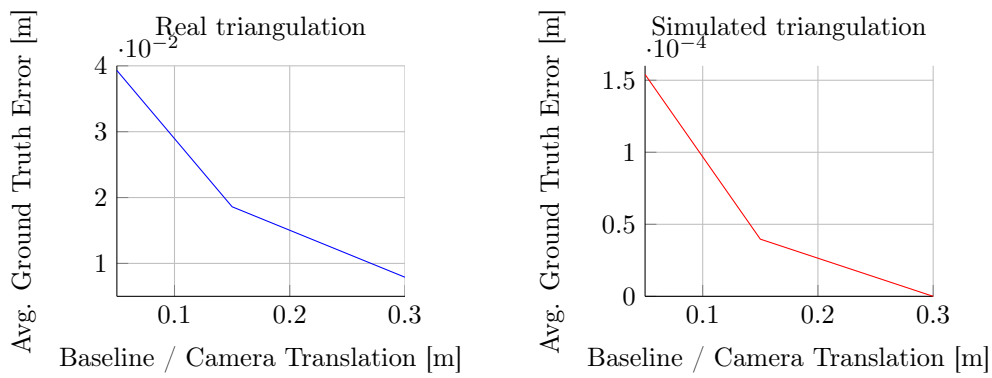


Figure 6.17: Experiment A. Ground truth error for increasing baseline. Left: Triangulated features obtained from real camera measurement and by measuring the static camera poses by hand. Right: Triangulated features obtained from simulation. Simulation and real world error drop for increasing baseline. Note the different y-scale for the left and right image.

utilized in this section is in sliding mode augmenting the EKF state vector with 17 camera poses. The features are automatically extracted by the regular opportunistic feature tracker as described in chapter 5.

Figure 6.18 shows the trajectory of the IMU: The camera mounted on the quadrotor is translated back and forth above the flat ground; the euclidean distance from the start to the end point is about 1 m in approximately 1 m height. Figure 6.19 represents the triangulated 3d features. The outliers shown correspond to camera poses with close-to-hover-phases at the start and end point.

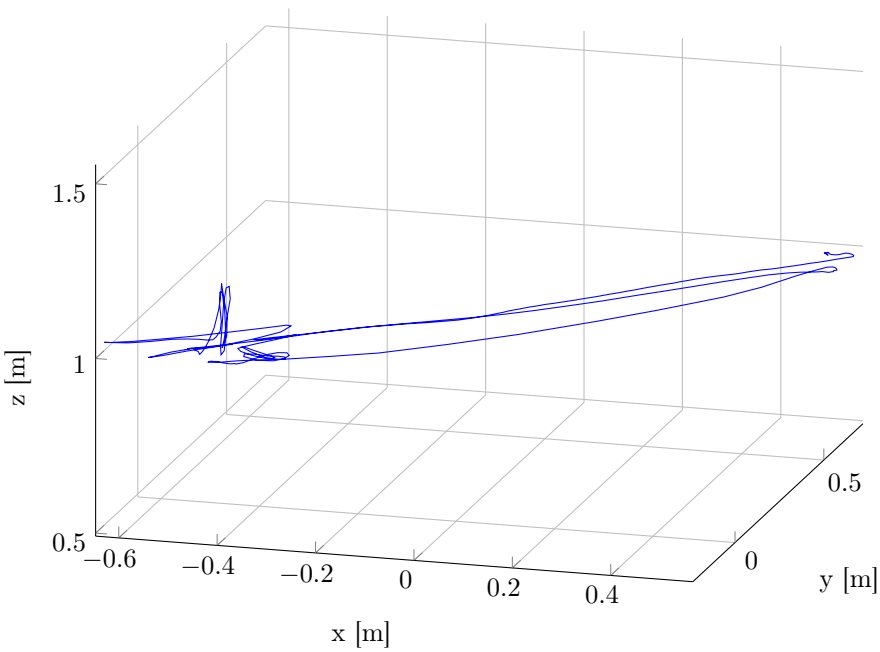


Figure 6.18: Experiment B: Trajectory of the IMU: The camera mounted on the quadrotor is translated back and forth above the flat ground; the euclidean distance from the start to the end point is about 1 m in approximately 1 m height.

The idea described in the next paragraph is to identify camera pose configurations that are likely to lead to poor triangulation results. Outliers have similar characteristics in terms of number of camera poses, translation and rotation between the camera poses. These properties can be exploited to remove feature tracks before they are even used inside the MSC-EKF filter. The outlier rejection proposed in this section can be used in parallel to the Mahalanobis gating test described in section 3.3.4. The disadvantage of the statistic gating test is that it that the feature needs to be evaluated in the MSC-EKF framework, triangulated and then the Mahalanobis distance needs to be calculated before the track is eventually removed. Consequently, the Mahalanobis distance outlier rejection 'wastes' computation time.

The classification of an outlier is performed by using a **S**upport **V**ector **M**achine (SVM) with a RBF kernel[7]. The theory of SVM classification is *shortly* outlined in the following paragraph and else referred to [21] for detailed information.
In the learning phase the kernelized SVM solves the following Quadratic Problem (QP):

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \text{ s.t. } 0 \leq \alpha_i \leq C \tag{6.48}$$

The kernel is a function $k : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}$ that satisfies symmetry and positive semi-definiteness. The RBF kernel, also known as Gaussian or squared exponential kernel, is selected for classification:

$$k(x, x') = \exp(-\|x - x'\|_2^2 / h^2) \tag{6.49}$$

where the parameter h defines the bandwidth of the Gaussian bell curve.
The SVM predicts if the feature trail is an outlier or an inlier based on

$$y = sgn\left(\sum_{i=1}^{n} \alpha_i y_i k(x_i, x)\right) \tag{6.50}$$

The SVM was trained on a different dataset and the SVM parameters were selected using 10-fold cross-validation. Assuming L observations for feature j the SVM data vector $\mathbf{X}^{(j)}$ for feature j is constructed using translational and rotational elements of the camera poses from which the static feature was observed. The translational and rotational elements are defined as

$$t^{(j)} := \sum_{i=1}^{L-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2 + (z_i - z_{i+1})^2} \tag{6.51}$$

and

$$r^{(j)} := \sum_{i=1}^{L-1} |\varphi_i^{i+1}| + |\theta_i^{i+1}| + |\psi_i^{i+1}| \tag{6.52}$$

respectively.

For this experiment the data vector $\mathbf{X}^{(j)}$ contains the following entries:[8]

$$\mathbf{X}^{(j)} = \begin{bmatrix} t^{(j)} & (t^{(j)})^2 & 1/t^{(j)} & r^{(j)} & (r^{(j)})^2 & 1/r^{(j)} & t^{(j)}/r^{(j)} \end{bmatrix}^T \tag{6.53}$$

The triangulated features classified into inlier (blue) and outlier (red) are shown in figure 6.19.

[7] **R**adial **B**asis **F**unction (RBF)

[8] The definitions introduced here led to a high outlier rejection rate but are rather arbitrary. Many other (reasonable) definitions may lead to similar results.

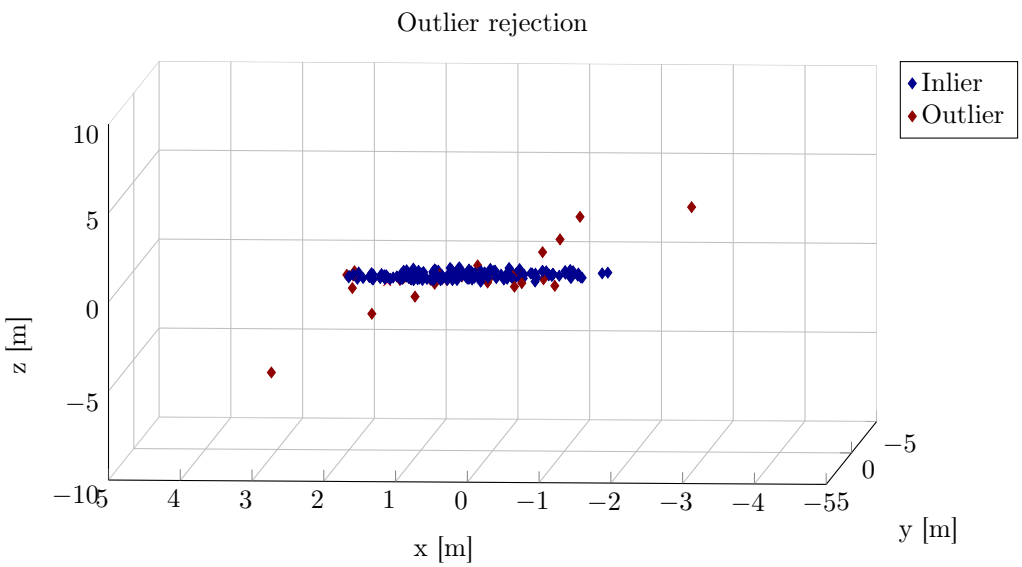


Figure 6.19: Experiment B. Triangulated features classified into inlier (blue) and outlier (red) by the Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel.

Remark: The outlier rejection described in this section only becomes necessary because the camera poses are close together during close-to-hover phases. A better approach than completely rejecting these features would be to detect hover phases and to intelligently manage the camera poses in the EKF vector in order to maximize the camera baseline and minimize the triangulation error. This is described in chapter 7.

6.5 Summary

Table 6.4 summarizes the main results found in the previous sections: The Gauss-Newton multi-view triangulation approach achieves the best trade-off in terms of speed and performance and is selected as the default triangulation method.

	Performance	**Speed**
Newton	$+++$	$-$
Gauss-Newton	$+++$	$+$
Levenberg-Marquardt	$++$	$--$
Gradient descent with fixed step size	$-$	$+++$
Gradient descent with line search	$+$	$++$

Table 6.4: Triangulation analysis summary: The Gauss-Newton triangulation approach achieves the best trade-off in terms of speed and performance.

Chapter 7

Robust navigation

The first part of chapter 7 is dedicated to the pose selection during dynamic & static motion phases. In the second part, the MSC-EKF vector is augmented with persistent features to improve the performance.

7.1 Camera pose selection during exploration phase

This section analyses the EKF in **F**irst-**I**n- **F**irst-**O**ut (FIFO) and in n-discard mode during exploration phase.

7.1.1 First-in-first-out (FIFO) mode

The first-in-first-out scheme is shown in figure 7.1. The grey squares correspond to camera poses and the black box symbolizes the poses which are stored in the EKF state vector for triangulation and Kalman filter updates. In the following it is assumed that the maximal number of camera poses that can be added to the state vector is $N_{max} = 6$ for simplicity. For $k = k_0, ...k_0 + 5$ the filter builds up the query. At $k = k_0 + 5$ the maximal number of poses is reached. The FIFO scheme removes the oldest camera pose from the state vector and adds the newest one which is shown in figure 7.1 at $k = k_0 + 6$. The FIFO scheme slides over the camera poses which is the reason why it is also known as sliding window approach.
The FIFO scheme is best suited for exploring phases with *enough* translation in between the camera poses. However, for static motion, i.e. during hovering, the FIFO scheme may fail as described in section 7.2.

Remark: If the computational effort for augmenting the state with every incoming image is too high, one can either select every n-th camera pose or simply lower the frame rate of the camera.

7.1.2 n-discard mode

The n-discard mode was proposed in [5] and aims at increasing the baseline for triangulation and Kalman filter updates. The idea is shown in figure 7.2. For $k = k_0, ...k_0 + 5$ the filter builds up the query by using every image analogue to the FIFO approach described above. At timestep $k = k_0 + 5$ the maximal number of camera poses inside the EKF state vector is reached. To make place for new camera poses every $N_{max}/3$ camera pose is removed starting with the *second* oldest pose, that is the oldest camera pose is always kept. In figure 7.2 at timestep $k = k_0 + 6$ every second camera pose is removed, starting with the second oldest one and the current camera pose is added at the end of the state vector. The n-discard approach

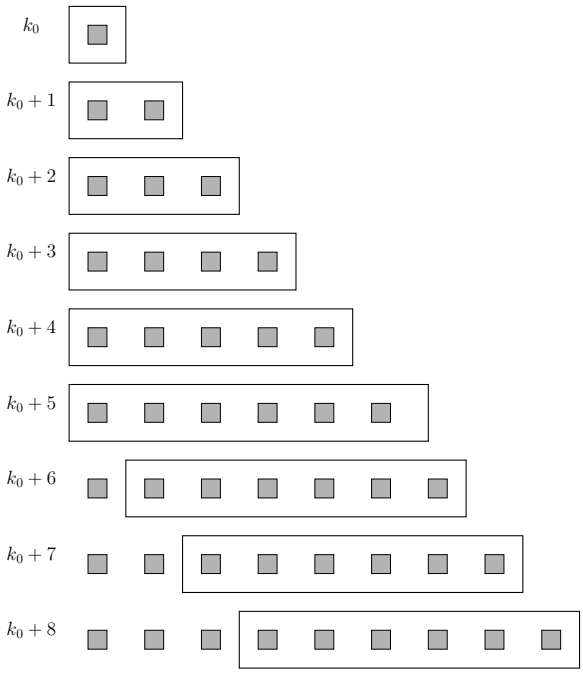


Figure 7.1: First-in-first-out (FIFO) mode also known as sliding window. The FIFO approach slides over the camera poses in time by removing the oldest camera pose and adding the current one.

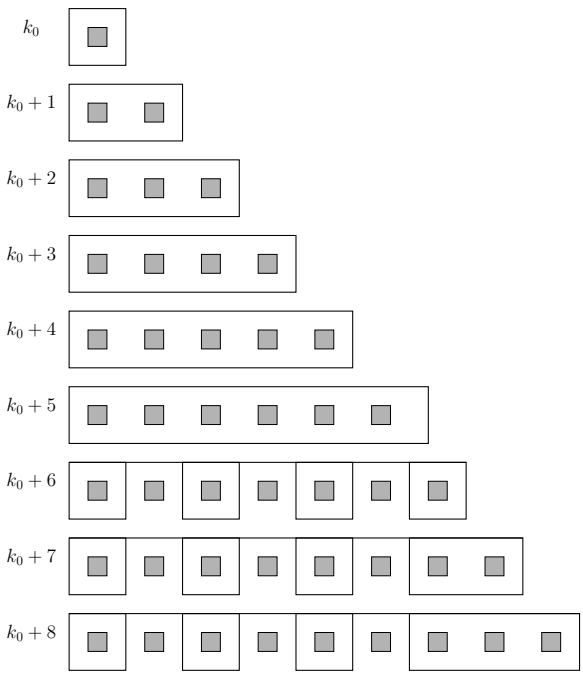


Figure 7.2: The n-discard approach tries to increase the baseline between camera poses by removing every $N_{max}/3$ pose and keeping the oldest one.

increases the baseline between the camera poses during exploration phase. However, the approach assumes that the feature tracker tracks feature over a long period

which may not be the case for all trajectories. After a longer flight period and in certain scenarios no feature trail may be associated with old, especially the first camera pose.

7.2 Camera pose selection during hover phase

In this section it is analysed how the camera pose query of the FIFO and n-discard approach is effected when coming from motion with enough baseline to static motion, i.e. hovering.

7.2.1 First-in-first-out (FIFO) mode

In figure 7.3 the FIFO approach is illustrated starting from timestep $k = k_0 + 6$. The grey squares symbolize normal motion with enough baseline; the red squares stand for static motion. The FIFO approach slides over the camera poses forward in time and eventually at timestep $k = k_0 + 6$ the state vector is completely filled with static camera poses. Triangulation with camera poses without enough baseline is prone to errors and the filter may diverge.

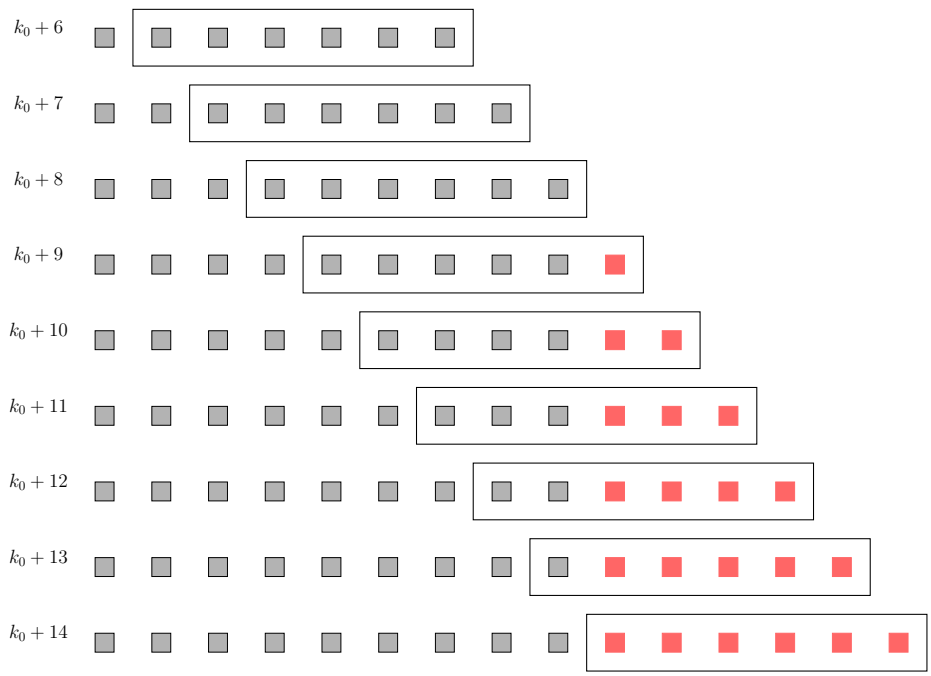


Figure 7.3: FIFO approach during hovering (symbolized by the red squares). The camera pose vector of the sliding window approach is eventually filled with hovering poses only. The triangulation with very small baseline is prone to errors.

7.2.2 n-discard mode

The n-discard mode undergoing a normal-flight to hover motion is shown in figure 7.4 - starting at timestep $k = k_0 + 6$. In contrast to the FIFO-approach, at $k = k_0 + 13$ the n-discard scheme stores two camera poses with generic motion inside the state vector. However, at step $k = k_0 + 15$, only the oldest camera pose is associated with dynamic motion. If the oldest pose is not associated with any feature trails any longer, the n-discard approach is as problematic as the FIFO scheme.

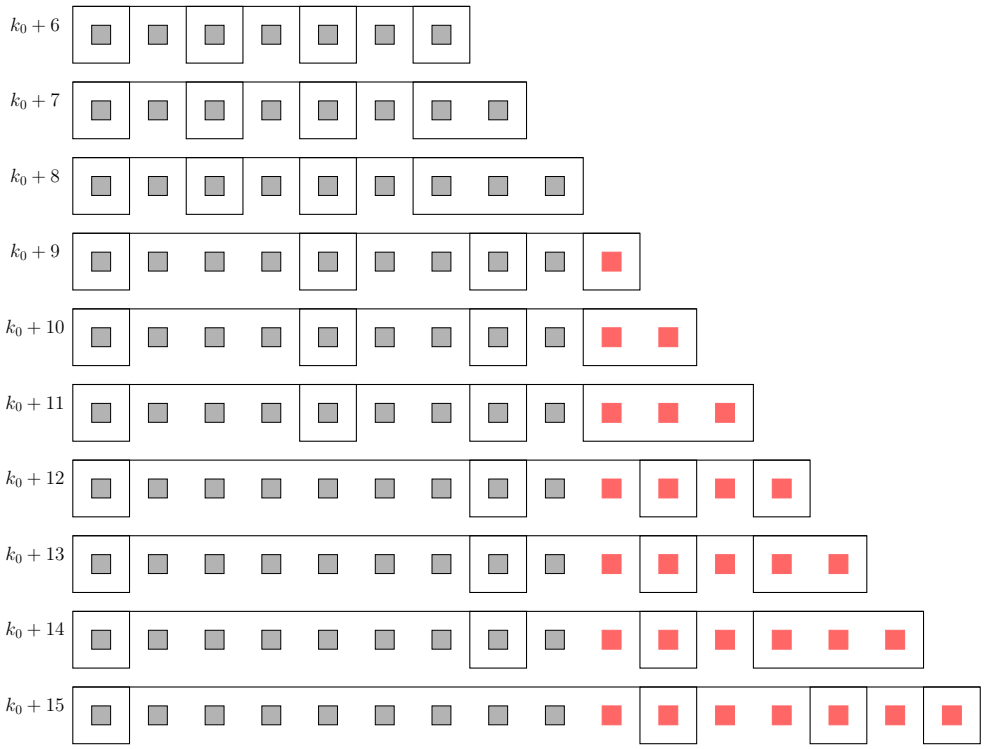


Figure 7.4: n-discard approach during hovering (symbolized by the red squares). At $k = k_0 + 15$, the camera pose vector is completely filled with static poses - except the very oldest pose.

7.2.3 FIFO-LIFO mode

Another approach which was proposed by Kottas et al. in [22] uses the FIFO mode when enough baseline is available and switches to last-in-first-out (LIFO) mode when detecting static motion. The basic idea is illustrated in figure 7.5: Up to timestep $k = k_0 + 10$ it is in FIFO mode. After detecting, for example, two consecutive camera poses associated with static motion, the proposed approach switches to LIFO mode. Only the second-newest camera pose is removed to make place for the newest one as shown in figure 7.5 for $k = k_0 + 11, ..., k_0 + 14$. At timestep $k = k_0 + 14$ still four of the six camera poses are associated with generic motion.

Static motion detection

In [22], Kottas et al. propose the following static motion detection criterion

- **Sufficient translation:** Use 2-point-RANSAC to estimate the unit vector of translation based on epipolar constraint:

$$\mathbf{b}_{k+1}^{i^T} \lfloor {}^{I_{k+1}}\mathbf{p}_{I_k} \times \rfloor \, \mathbf{C}({}^{I_{k+1}}\hat{\bar{q}}_{I_k}) \mathbf{b}_k^i = 0 \tag{7.1}$$

with

$$\begin{aligned} \mathbf{b}_k^i = \frac{\mathbf{z}_k^i}{\|\mathbf{z}_k\|^2} &: \text{1-norm bearing measurement to a feature} \\ {}^{I_{k+1}}\mathbf{p}_{I_k} &: \text{Unit vector of translation from } k \text{ to } k+1 \\ \mathbf{C}({}^{I_{k+1}}\hat{\bar{q}}_{I_k}) &: \text{Rotation between } k \text{ and } k+1 \end{aligned}$$

- **Zero translation:** Compute

$$d_k = \frac{1}{M} \sum_{i=1}^{M} \left\| \mathbf{b}_{k+1}^i - \mathbf{C}({}^{I_{k+1}}\hat{\bar{q}}_{I_k}) \mathbf{b}_k^i \right\|_2 \tag{7.2}$$

$$M: \text{ Number of features at timestep } k$$

to categorize the motion into static or dynamic motion.

Adapted MSC-EKF framework

- FIFO mode: Update the state and covariance of the MSC-EKF as described in chapter 3.
- LIFO mode: When the filter is in LIFO mode, that is during static motion, only update the state and not the covariance.[1]

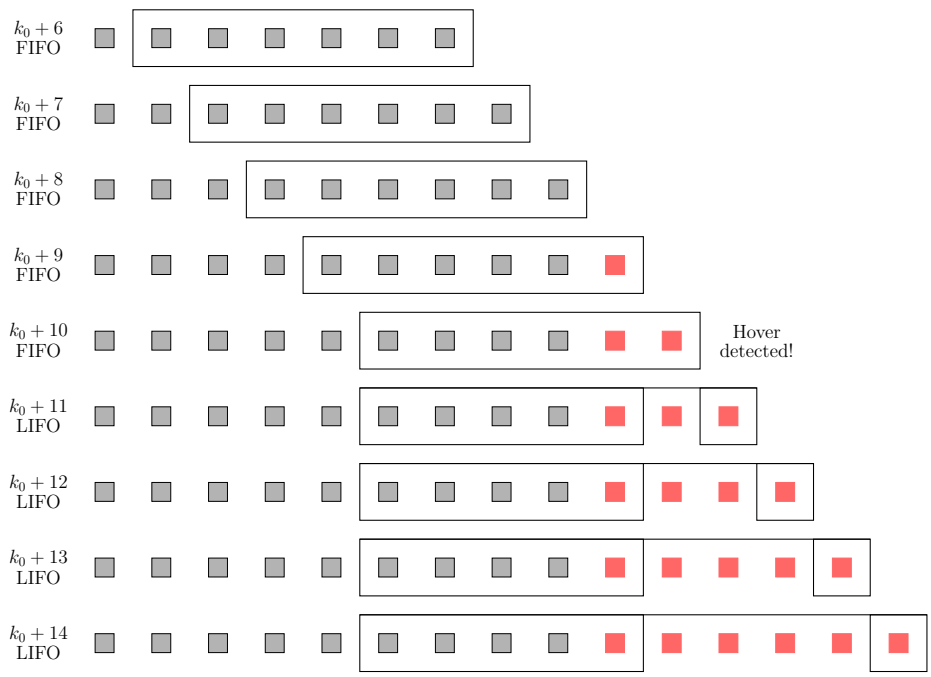


Figure 7.5: FIFO-LIFO approach during hovering. After timestep $k = k_0 + 10$ the filter switches to LIFO mode after having detected static motion for e.g. two consecutive camera poses. At $k = k_0 + 14$ four out of the six camera poses were recorded during dynamic motion.

7.3 Persistent features

The performance of the MSC-EKF can be improved by not only using opportunistic features for visual-inertial odometry but also to combine it with a SLAM approach. The hybrid MSC-EKF/SLAM approach described in this section is mainly based on the ideas proposed in [18] and [23].

[1] The covariance should not change (decrease) in the update step since the same camera poses are used for every update while hovering.

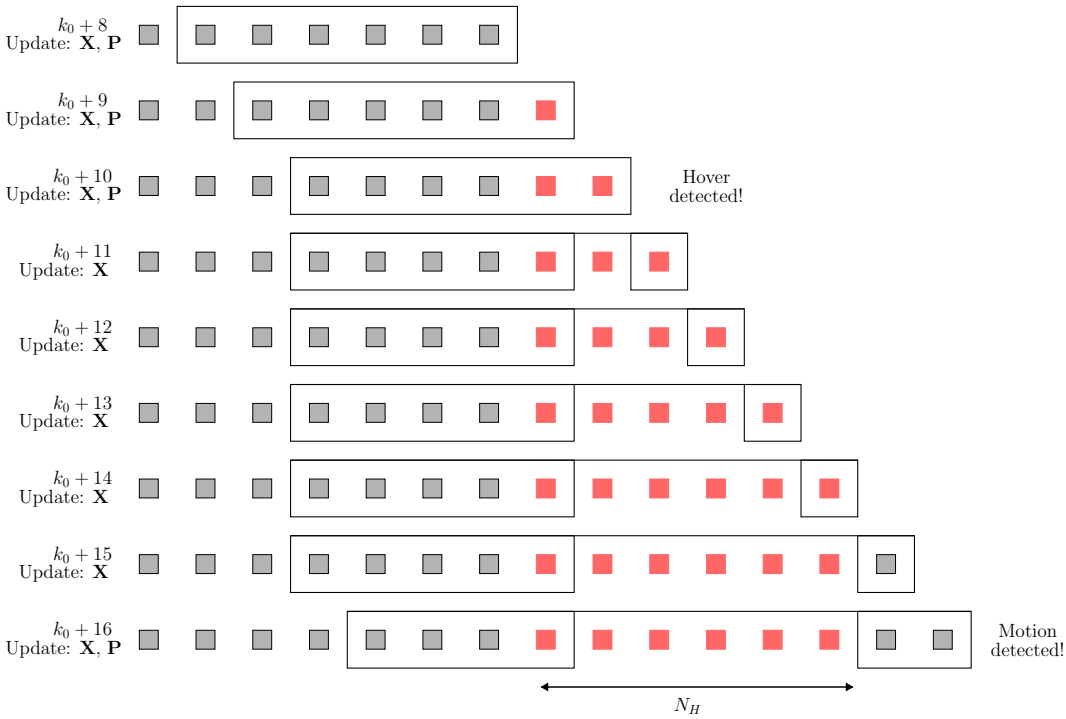


Figure 7.6: Adaption to the MSC-EKF when using the FIFO-LIFO approach. During hover mode only the filter states are updated.

7.3.1 MSC-EKF/SLAM state and feature parametrization

$$\mathbf{X_{IMU}} = \begin{bmatrix} {}^G p_I^T & {}^G v_I^T & {}^I \bar{q}_G^T & b_\omega^T & b_a^T & {}^C \bar{q}_I^T & {}^I p_C^T \end{bmatrix} \tag{7.3}$$

$$\mathbf{X_{hybrid}} = \begin{bmatrix} X_{IMU} & {}^G p_{C_1}^T & \dots & {}^G p_{C_N}^T & {}^{C_1} \bar{q}_G^T & \dots & {}^{C_N} \bar{q}_G^T & f_1^T & \dots & f_n^T \end{bmatrix} \tag{7.4}$$

where the feature $\mathbf{f}_i = \begin{bmatrix} \alpha_i & \beta_i & \rho_i \end{bmatrix}^T$ is stored in inverse-depth parametrization: [2]

$${}^G\mathbf{p}_{f_i} = {}^G\mathbf{p}_{C_i} + \frac{1}{\rho_i} \mathbf{R}_i \begin{bmatrix} \alpha_i \\ \beta_i \\ \rho_i \end{bmatrix} \tag{7.5}$$

$${}^G\mathbf{p}_{C_i} : \text{latest camera position in global coordinates used as } \textit{anchor} \text{ state}$$

$$\mathbf{R}_i = {}^G_{C_i}\hat{\mathbf{R}} : \text{camera rotation associated with anchor state}$$

Whenever the anchor state is removed from the EKF state a re-parametrization becomes necessary as shown in figure 7.7. The feature re-parametrization is given

[2] As proposed in [23]

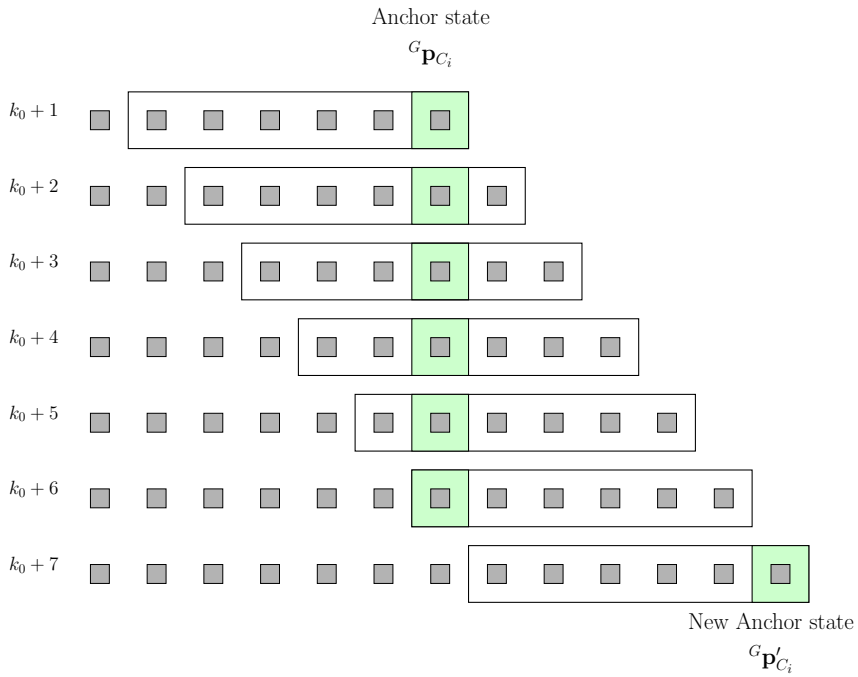


Figure 7.7: Anchor state reparametrization is necessary when for example using the sliding window (FIFO) approach.

by [3]

$$ {}^{G}\mathbf{p}_{f_i} = {}^{G}\mathbf{p}_{C_i} + \frac{1}{\rho_i}\mathbf{R}_i \begin{bmatrix} \alpha_i \\ \beta_i \\ \rho_i \end{bmatrix} \overset{!}{=} {}^{G}\mathbf{p}'_{C_i} + \frac{1}{\rho'_i}\mathbf{R}'_i \begin{bmatrix} \alpha'_i \\ \beta'_i \\ \rho'_i \end{bmatrix} \tag{7.6} $$

$$ \text{which results in: } \frac{1}{\rho'_i}\begin{bmatrix} \alpha'_i \\ \beta'_i \\ 1 \end{bmatrix} = (\mathbf{R}'_i)^T \left(-{}^{G}\mathbf{p}'_{C_i} + {}^{G}\mathbf{p}_{C_i} + \frac{1}{\rho_i}\mathbf{R}_i \begin{bmatrix} \alpha_i \\ \beta_i \\ \rho_i \end{bmatrix} \right) \tag{7.7} $$

7.3.2 EKF state and covariance augmentation

After the standard MSC-EKF state and covariance augmentation, proceed as follows:

Method a: Simple augmentation

- **State:** Augment the state $\mathbf{X}_{hybrid}$ with $\mathbf{f}_i$:

$$ \hat{\mathbf{X}}^{+}_{\mathbf{hybrid,k}} \leftarrow \begin{bmatrix} \hat{\mathbf{X}}^{-}_{\mathbf{hybrid,k}} \\ \hat{\mathbf{f}}_i \end{bmatrix} \tag{7.8} $$

[3] As proposed in [23]

- **Covariance:** Augment the covariance matrix with $\mu\mathbf{I}$:

$$\mathbf{P}^+ \leftarrow \begin{bmatrix} \mathbf{P}^- & 0 \\ 0 & \mu\mathbf{I} \end{bmatrix} \tag{7.9}$$

  where $\mu \to \infty$.

In principle, any initialization for $\hat{\mathbf{f}}_i$ is valid since the covariance is infinite [4]. However, this augmentation procedure may lead to numerical instabilities.

Method b: Augmentation as proposed in [23]

The more numerical stable procedure to augment the filter state and covariance as proposed in [23] is as follows:

1. Triangulate the feature using all camera poses in which the feature has been detected.
2. Calculate the **covariance**:[5]

$$\mathbf{P}^+_{aug} = \begin{bmatrix} \mathbf{P}^+ & -(\mathbf{H}_2^{-1}\mathbf{H}_1\mathbf{P}^+)^T \\ -\mathbf{H}_2^{-1}\mathbf{H}_1\mathbf{P}^+ & \mathbf{P}^+_{22} \end{bmatrix} \tag{7.10}$$

   with

$$\mathbf{P}^+ = \mathbf{P}_{k|k+1} - \mathbf{P}_{k|k+1}\mathbf{H}_o^T(\mathbf{H}_o\mathbf{P}_{k|k+1}\mathbf{H}_o^T + \sigma^2\mathbf{I})^{-1}\mathbf{H}_o\mathbf{P}_{k|k+1} \tag{7.11}$$

$$\mathbf{P}^+_{22} = (\mathbf{H}_2^{-1}\mathbf{H}_1)\mathbf{P}^+_{k|k+1}(\mathbf{H}_2^{-1}\mathbf{H}_1)^T + \sigma^2\mathbf{H}_2^{-1}\mathbf{H}_1 \tag{7.12}$$

$$\mathbf{P}^+ := \mathbf{P}_{k+1|k+1} \tag{7.13}$$

   where

$$\mathbf{H}_o = \mathbf{V}^T\mathbf{H}_i \tag{7.14}$$

$$\mathbf{H}_1 = \mathbf{U}^T\mathbf{H}_i \tag{7.15}$$

$$\mathbf{H}_2 = \mathbf{U}^T\mathbf{H}_{f_i} \tag{7.16}$$

$$\tag{7.17}$$

   and $\mathbf{V}$ and $\mathbf{U}$ form the bases and left nullspace of the column space of $\mathbf{H}_{f_i}$ respectively[6].

3. Calculate the **state**:[7]

$$\Delta x = \begin{bmatrix} \Delta x_0 \\ \mathbf{H}_2^{-1}\mathbf{H}_1\Delta x_0 + \mathbf{H}_2^{-1}\tilde{\mathbf{z}}_1 \end{bmatrix} \tag{7.18}$$

   with

$$\Delta x_0 = \mathbf{P}_{k|k+1}\mathbf{H}_o^T(\mathbf{H}_o\mathbf{P}_{k|k+1}\mathbf{H}_o^T + \sigma^2\mathbf{I})^{-1}\tilde{\mathbf{z}}_o \tag{7.19}$$

$$\tilde{\mathbf{z}}_o = \mathbf{V}^T\tilde{\mathbf{z}}_i \tag{7.20}$$

$$\tilde{\mathbf{z}}_1 = \mathbf{U}^T\tilde{\mathbf{z}}_i \tag{7.21}$$

[4] [23]
[5] Only the results are presented here. Please find the derivation and details in [23].
[6] Matrix $\mathbf{H}_{f_i}$ is defined in chapter 3. For details see [23]
[7] Only the results are presented here. Please find the derivation and details in [23].

7.3.3 EKF update

The Kalman filter update is performed by stacking the MSC-EKF's opportunistic features and the SLAM's persistent features.

$$\tilde{\mathbf{z}}_k = \mathbf{H}_k \tilde{\mathbf{x}}_k + \mathbf{n}_k \tag{7.22}$$

In detail, the residual vector, measurement matrix and noise covariance matrix are calculated as follows

- The **measurement matrix** for performing the Kalman filter update at timestep k is defined as

$$\mathbf{H}_k = \begin{bmatrix} \mathbf{H}_{MSC} \\ \mathbf{H}_{SLAM} \end{bmatrix} \tag{7.23}$$

  where

$$\mathbf{H}_{MSC} := \mathbf{A}^T \mathbf{H}_o \tag{7.24}$$

  is obtained by stacking the measurement matrices for all observations and all features and multiplying by the left nullspace of $\mathbf{H}_f$ as defined in chapter 3. To reduce the complexity of $\mathbf{H}_{MSC}$, employ QR decomposition (see section 3.3.5).
  The measurement matrix for the SLAM features is defined as

$$\mathbf{H}_{SLAM} = \begin{bmatrix} \mathbf{H}_{f_m}^{(1)} \\ \vdots \\ \mathbf{H}_{f_m}^{(s_k)} \end{bmatrix} \tag{7.25}$$

  where $H_{f_i}^{(j)}$ with features $j = 1, ..., s_k$ is defined in chapter 3 and $i = m$ refers to the *latest* camera pose in the EKF state vector.

- The **noise covariance matrix** at timestep k is given by

$$\mathbf{R}_k = \sigma_{noise}^2 \mathbf{I}_\zeta \tag{7.26}$$

  where $\zeta = 21 + 6N + 3s_k$.[8]

- The **residual vector** is defined as [9]

$$\tilde{\mathbf{z}}_k := \begin{bmatrix} \tilde{\mathbf{z}}_{MSC} \\ \tilde{\mathbf{z}}_{SLAM} \end{bmatrix} \tag{7.27}$$

  where

$$\tilde{\mathbf{z}}_{MSC} = \mathbf{A}^T \tilde{\mathbf{z}} \tag{7.28}$$

  is obtained by stacking the residuals for all observations and all features and multiplying by the left nullspace of $\mathbf{H}_f$ as defined in chapter 3. To reduce the

[8] Assuming N camera poses s_k SLAM features at timestep k and the EKF state vector includes the Camera-IMU transformation.

[9] Using the notation proposed in [18].

complexity of $\tilde{\mathbf{z}}_{MSC}$, employ QR decomposition (see section 3.3.5).
The residual vector for the SLAM features is defined as

$$\tilde{\mathbf{z}}_{SLAM} = \begin{bmatrix} \tilde{\mathbf{z}}_{f_m}^{(1)} \\ \vdots \\ \tilde{\mathbf{z}}_{f_m}^{(s_k)} \end{bmatrix} \tag{7.29}$$

where $\tilde{\mathbf{z}}_{f_m}^{(j)}$ with features $j = 1, ..., s_k$ are the observation residuals of the s_k persistent features [18].

Chapter 8

Implementation remarks

8.1 MSF framework: State and covariance buffer management

The buffer states need to be carefully managed to avoid overwriting buffer values and/or to avoid redundant filter steps. The critical steps include the state & covariance augmentation as well as the filter update.

8.1.1 State and covariance augmentation

For every incoming image, the state vector and covariance matrix need to be augmented with the camera pose and uncertainty of the camera pose due to the uncertainty in the IMU state estimates respectively. The management of the buffer states is best explained with an example as shown in figure 8.1. The black box symbolizes the initial setup, the blue boxes the IMU state and covariance propagation step and the red box illustrates the state augmentation step triggered by an image callback.

1. Image 8.1-1 shows the initial setup. The small rectangles symbolize the buffer states which are identified by an individual timestamp. The P_{prop} flag points to the buffer state to which the *covariance* was last propagated. X_{last} points to the buffer state to which the *state* was last propagated, X_{curr} points to the state that was currently inserted due to a new IMU measurement. Note that the IMU rate is set to 100 Hz.

2. For the incoming IMU measurement reading a new state is created at the end of the buffer list ($ts = 02369$) and registered by the flag X_{curr}. The pointer X_{curr} from figure 8.1-1 is now X_{last}. The state is propagated one step from X_{last} to X_{curr} illustrated by $PropX$ 8.1-2. Similarly, the covariance is propagated one step starting from the last propagated state to the next state in the buffer list. After the covariance propagation the flag P_{prop} pointer is updated ($ts = 01369$).

3. Figure 8.1-3 shows another state and covariance propagation step evoked by an IMU measurement.

4. In figure 8.1-4 a new image measurement is registered with the timestamp $ts = 03064$ (red box) which is in between two buffer states which were previously inserted due to IMU readings (black boxes); note that the image framerate is set to 15 fps. For the new buffer state at $ts = 03064$, the state vectors and covarince matrix needs to be predicted. The process covariance of this new state is predicted starting from the state to which P_{prop} was *last* pointing

to ($ts = 02369$). Similarly, the state is propagated from the state to which X_{last} was *last* pointing to (also $ts = 02369$).

Covariance augmentation: The previously computed covariance matrix for $ts = 03064$ is then augmented with the IMU-Camera cross-correlation. This is the buffer state from which the covariance matrix will be propagated from in the next IMU propagation step.

State augmentation: The camera poses for the current IMU state ($ts = 03369$) are calculated according to the CAM-IMU transformation equations

$$^{G}\hat{\mathbf{p}}_{C_i} = {}^{G}\hat{\mathbf{p}}_I + \mathbf{C}({}^{G}_{I}\hat{q})^{I}\mathbf{p}_C \tag{8.1}$$

$$^{C_i}_{G}\hat{q} = {}^{C}_{I}\bar{q} \otimes {}^{I}_{G}\hat{q} \tag{8.2}$$

where the state estimates are taken from buffer at timestamp $ts = 03369$. This is the buffer state from which the state matrix will be propagated from in the next IMU propagation step. N.b. that it is not necessary to write the camera pose to the buffer state at $ts = 03064$.

5. Figure 8.1-5 illustrates the next IMU propagation step for an IMU reading at $ts = 04369$: The covariance is propagated from the buffer state to which the covariance was last propagated to ($ts = 03064$), which is the buffer state generated by an image callback and whose covariance matrix was augmented by the Camera-IMU cross-correlation. The state is propagated from X_{last} to X_{curr} where X_{last} (pointing to $ts = 03369$) contains the camera poses. In fact, the filter routine $PropX$ simply copies the camera poses from buffer state at $ts = 03369$ to buffer state at $ts = 04369$.

8.1.2 Kalman filter update

The state buffers for the Kalman filter update step are managed by the MSF framework:

1. Figure 8.2-1 and 8.2-2 show the standard IMU propagation steps as described above

2. In step 3 in figure 8.2-3, an incoming image at timestamp $ts = 36409$ triggers a Kalman filter update due to lost feature trails. The covariance and state estimates are initialized by propagating from the closest state in the past ($ts = 36372$) - similar to figure 8.1-4 above. The predicted process covariance is used as a-priori covariance for the outlier rejection using the Mahalanobis distance and the Kalman filter equations. After the Kalman filter update, the core and augmented states are corrected and the the a-posteriori covariance is calculated based on the a-priori covariance.

3. Figure 8.2-4 shows the next IMU propagation step: The covariance is propagated starting from buffer state at $ts = 36409$; the state is propagates from state at $ts = 37372$.

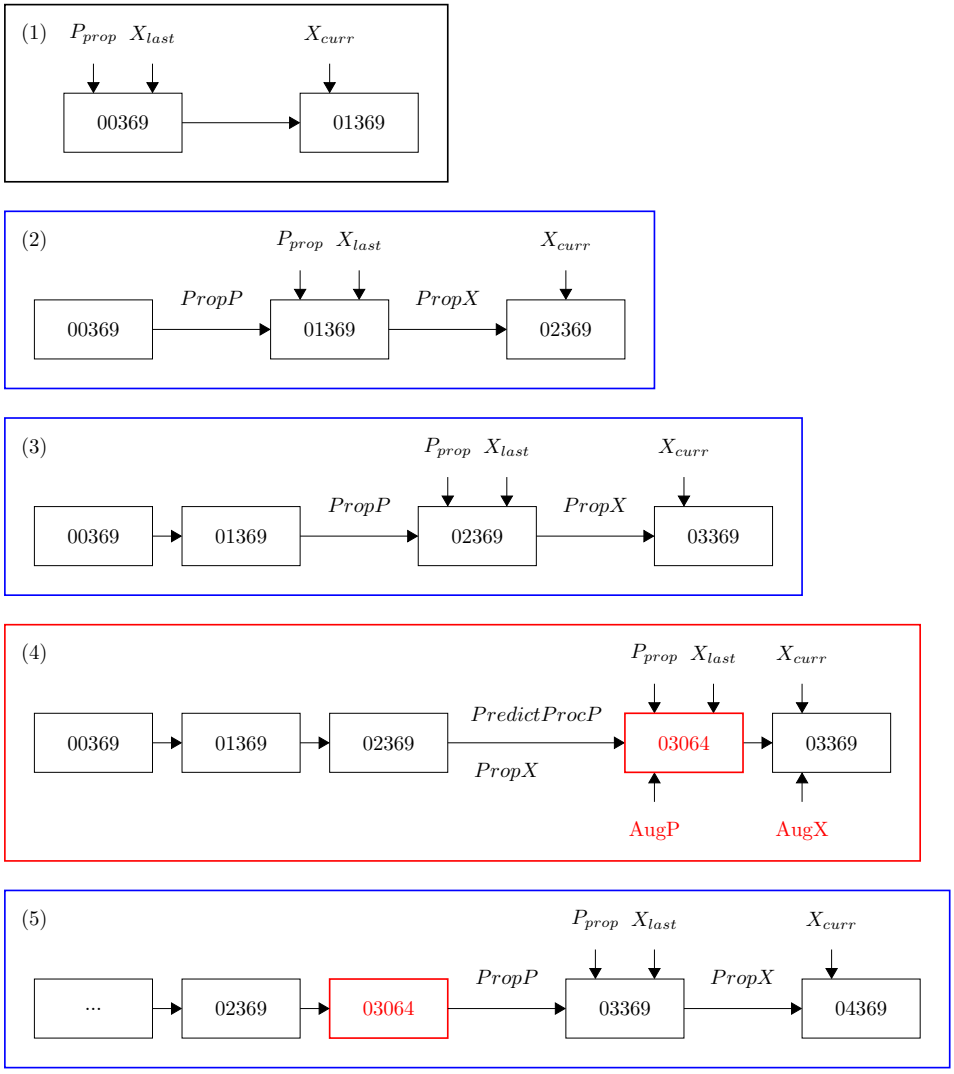


Figure 8.1: State and covariance buffer management for the state and covariance augmentation step explained with an example. The black box shows the initial setup, the blue boxes illustrate IMU state and covariance propagation steps and the red box is the image augmentation triggered by an image measurement.

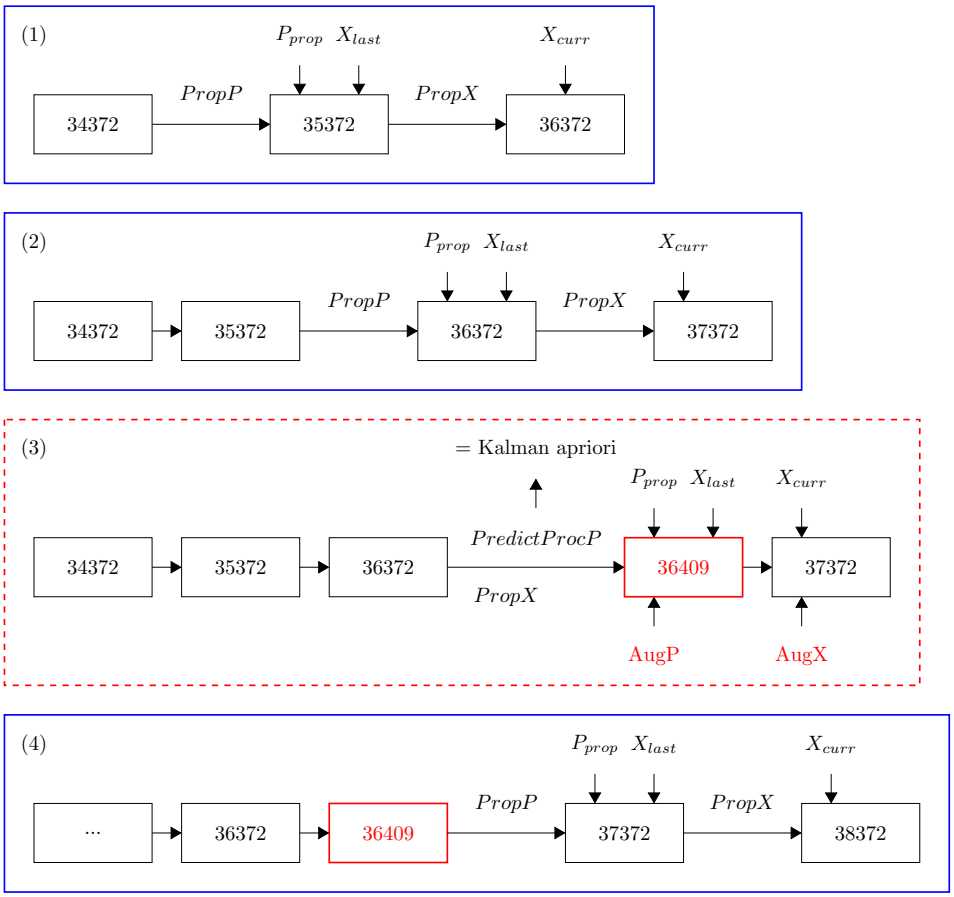


Figure 8.2: State and covariance buffer management for the Kalman filter update step explained with an example. The black box shows the initial setup, the blue boxes illustrate IMU state and covariance propagation steps and the dashed red box is the combined image augmentation and Kalman filter step

8.2 Timing

Figure 8.3 shows the timing of the feature tracker, triangulation and filter module for an IMU rate of 100 Hz, an image frame rate of 15 fps and 17 camera poses in the EKF state vector. It turns out that the update step including the measurement

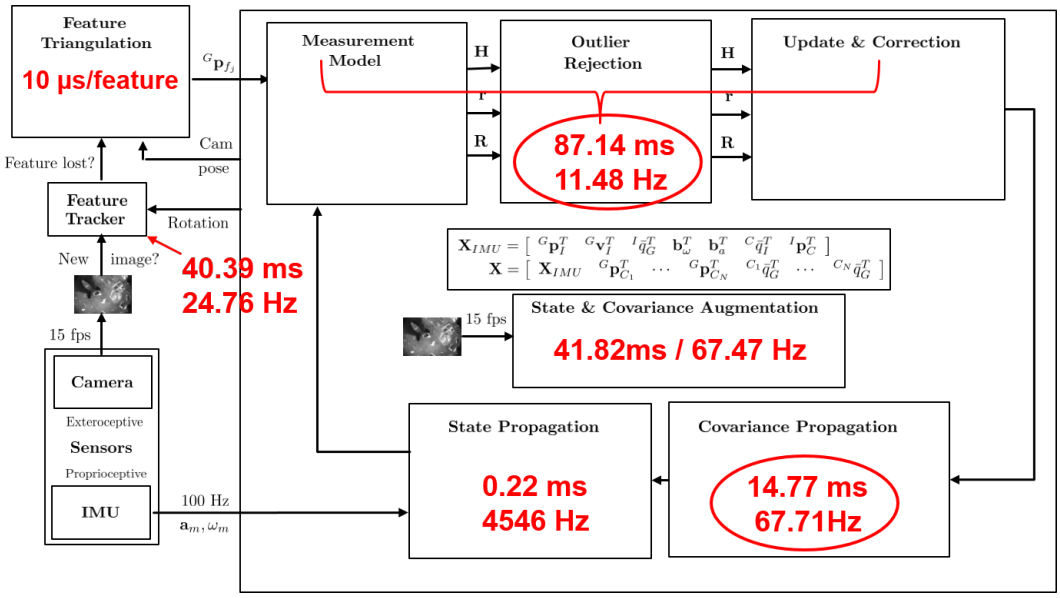


Figure 8.3: Timing of the feature tracker, triangulation and filter module

model, outlier rejection and Kalman filter update as well as the covariance propagation become bottlenecks. For example, assuming 17 camera poses, these modules need to handle 123×123-dimensional covariance matrices.

Besides further optimizing the code, the number of camera poses, the IMU rate or the image frame rate could be turned down.

Chapter 9

Results

In this chapter, the performance of the MSF/MSC-EKF framework is evaluated:

- **Circular Trajectory:** Section 9.1 compares the MSF/MSC-EKF and SSF/PTAM estimates for a circular trajectory.
- **Flat ground → table trajectory:** Section 9.2 compares the performance of the MSF/MSC-EKF and SSF/PTAM framework when flying a fast trajectory with various height changes of the surrounding 3d features.
- **Initialization procedure:** Section 9.3 describes the shortcomings of the SSF/PTAM initialization and presents two different initialization procedures for the MSF/MSC-EKF framework.

9.1 MSF/MSC-EKF vs. SSF/PTAM

To evaluate the performance of the MSF/MSC-EKF framework it is compared to the SSF/PTAM filter estimates since no real ground truth was available.
The initial covariance is set to heuristic values from the Google TANGO project:

```
P_core.setZero();

// position (I wrt G): sigma = 0.0 m
P_core(0,0) = 0.0; // px
P_core(1,1) = 0.0; // py
P_core(2,2) = 0.0; // pz

// velocity (I wrt G): sigma = 1.0 m/s
P_core(3,3) = 1.0; // vx
P_core(4,4) = 1.0; // vy
P_core(5,5) = 1.0; // vz

// attitude (I to G): sigma = 5deg
P_core(6,6) = std::pow(5.0*M_PI/180.0,2); // thetax
P_core(7,7) = std::pow(5.0*M_PI/180.0,2); // thetay
P_core(8,8) = std::pow(5.0*M_PI/180.0,2); // thetaz

// bias gyroscope: sigma = 1deg
P_core(9,9) = std::pow(1.0*M_PI/180.0,2); // bwx
P_core(10,10) = std::pow(1.0*M_PI/180.0,2); // bwy
```

```
P_core(11,11) = std::pow(1.0*M_PI/180.0,2); // bwz

// bias acceleration: sigma = 0.166m/s^2
P_core(12,12) = std::pow(0.166,2); // bax
P_core(13,13) = std::pow(0.166,2); // bay
P_core(14,14) = std::pow(0.166,2); // baz

// cam-imu attitude:
P_core(15,15) = 0; // std::pow(0.5*M_PI/180.0,2); // thetax
P_core(16,16) = 0; // std::pow(0.5*M_PI/180.0,2); // thetay
P_core(17,17) = 0; // std::pow(0.5*M_PI/180.0,2); // thetaz

// cam-imu translation: sigma = 0.003
P_core(18,18) = 0; // std::pow(0.003,2); // tx
P_core(19,19) = 0; // std::pow(0.003,2); // ty
P_core(20,20) = 0; // std::pow(0.003,2); // tz

P_core = P_core;
P_core = 0.5 * (P_core + P_core.transpose());
```

The covariance for the camera-IMU rotation and translation is initialized with zero since the state is kept constant and initialized with a good estimate from the CAD, SSF-PTAM & Kalibr[1] frameworks. The remaining core states were initialized with the SSF/PTAM estimates. Figure 9.2 and figure 9.1 show the circular trajectory and experiment setup respectively.

Figure 9.1: Experiment setup for evaluating the performance of the MSF/MSC-EKF framework

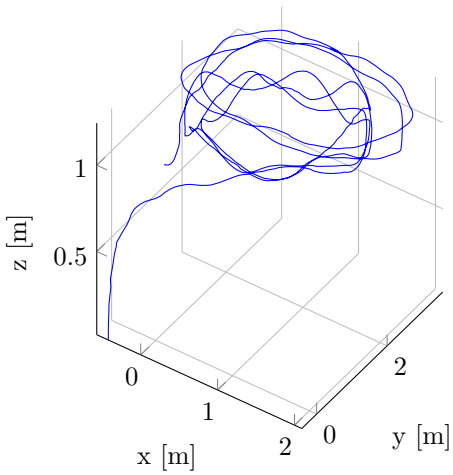


Figure 9.2: The figure shows a circular camera trajectory. The radius of the circle is about 1.5 meter.

The position ${}^{G}\mathbf{p}_I$, velocity ${}^{G}\mathbf{v}_I$ and rotation ${}^{I}\bar{q}_G$ estimates of the IMU are shown in figure 9.3[2]. The pose estimates of the SSF/PTAM and the MSF/MSC-EKF framework are comparable, however, a Vicon camera setup as described in chapter 10 would be needed to truly benchmark the performance of the MSC-EKF.

[1] The camera-IMU calibration using Kalibr is shown in appendix F.

[2] The SSF/PTAM and MSF/MSC-EKF filter framework is abbreviated with SSF and MSF in the following.

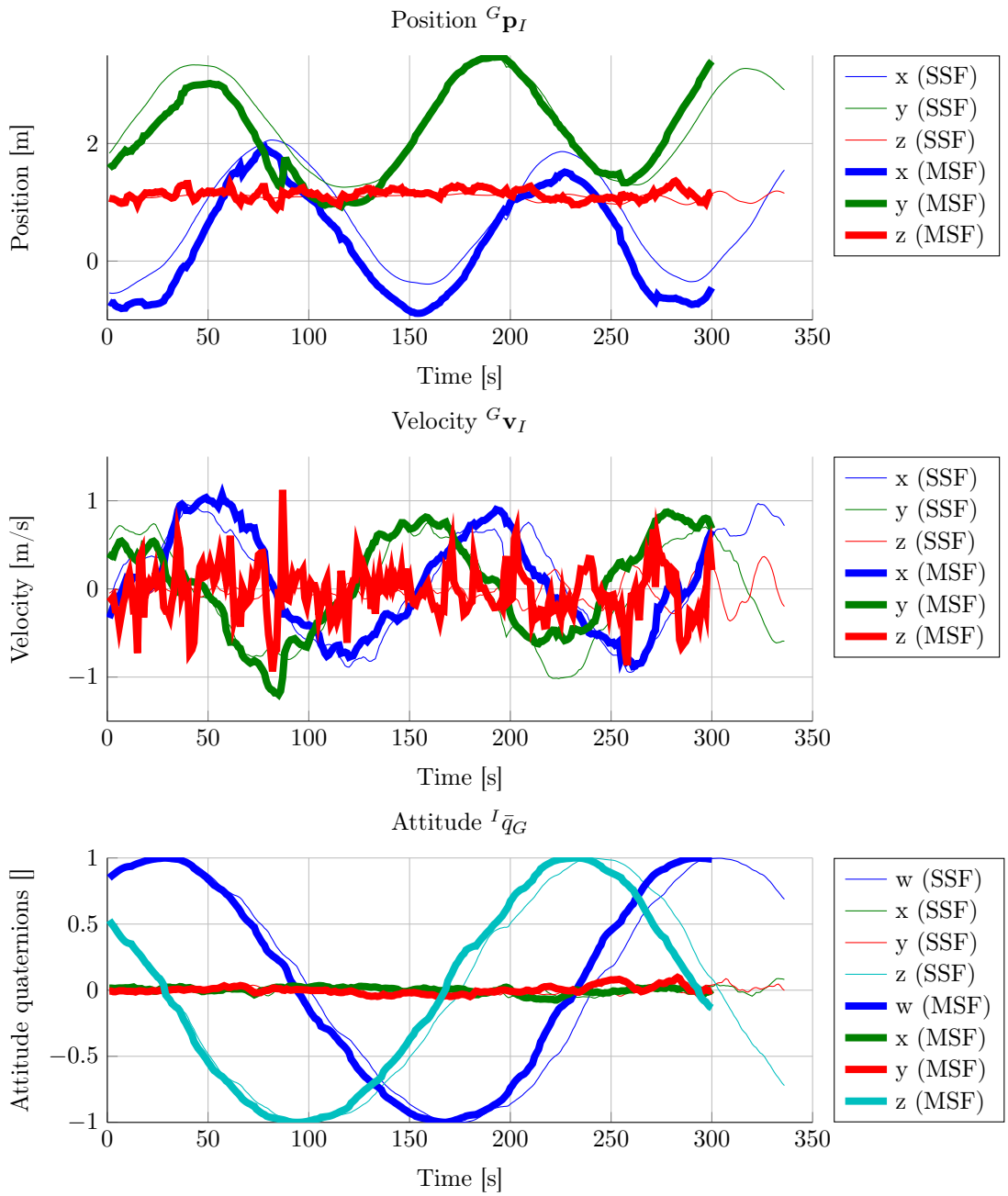


Figure 9.3: Position ${}^{G}\mathbf{p}_I$, velocity ${}^{G}\mathbf{v}_I$ and rotation ${}^{I}\bar{q}_G$ of the IMU estimated by the MSF/MSC-EKF framework and compared to the SSF/PTAM estimates.

9.2 SSF/PTAM failure mode: Flat ground → table trajectory

In this section, a typical failure mode of the SSF/PTAM filter framework is analyzed: The quadrotor is first flown over flat ground and the SSF/PTAM filter is initialized by the standard stereo initialization procedure. After around 20 s the camera reaches the border of a table and other 3d objects with different height as can be seen in figure 9.4. At that moment, PTAM fails to map both the features

Figure 9.4: SSF/PTAM fails when flying over flat ground followed by a table. The PTAM map (green grid) is inclined and PTAM outputs "Attempting recovery".

from the flat ground and from the 3d structure due to its flat ground assumption. This failure on the other hand impacts the state estimates as shown in figure 9.6: After 20 s to 25 s the states estimated by the SSF filter framework diverge which can be seen in the trajectory in figure 9.5.

In constrast, the MSF/MSC-EKF framework, which is initialized as described in section 9.1, does not show any difficulties and ends with a circular trajectory after around 35 s.

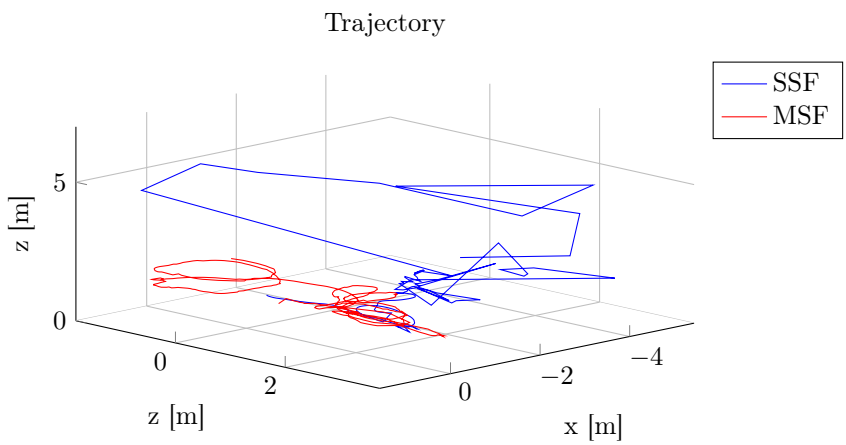


Figure 9.5: PTAM looses the map when flying over flat ground followed by a table leading to divergence of the SSF filter framework (red).

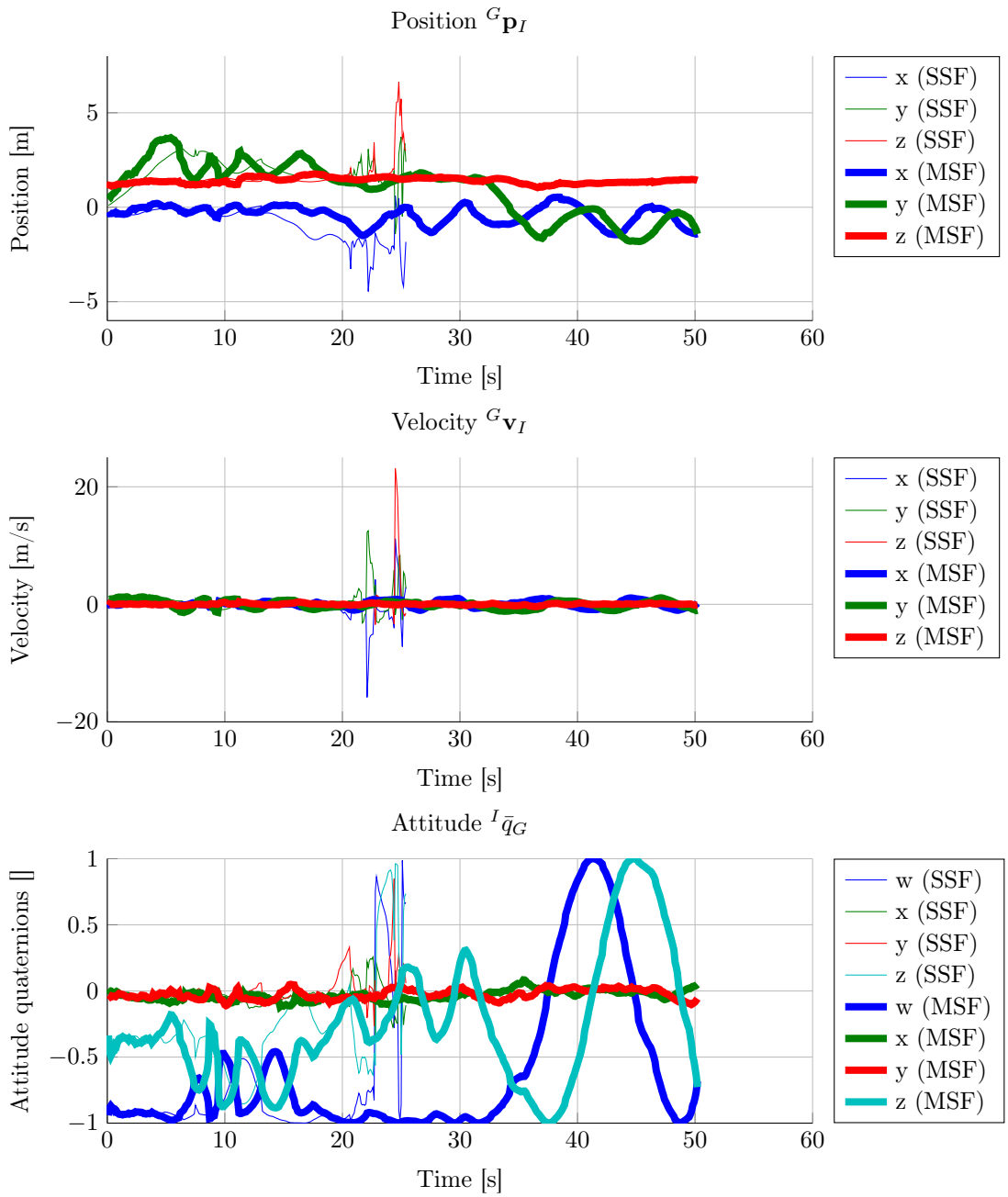


Figure 9.6: Position ${}^{G}\mathbf{p}_{I}$, velocity ${}^{G}\mathbf{v}_{I}$ and rotation ${}^{I}\bar{q}_{G}$ of the IMU estimated by the MSF/MSC-EKF framework and compared to the SSF/PTAM estimates. SSF/PTAM diverges after 25.4 s - no more state estimates are published after that timestamp.

9.3 SSF/PTAM failure mode: Initialization from ground / without translation

The main limitations of the SSF/PTAM's initialization procedure are the two following:

1. PTAM is **initialized manually** by pressing the spacebar twice to define the start and end point of the stereo initialization. The filter is best initialized with pure translation motion.

2. Due to this essential initial translation phase it is **not possible to start** the filter **from the ground**.

In the following paragraph, two different initialization procedures for the MSF/MSC-EKF filter framework are proposed to overcome the limitations of the SSF/PTAM framework described above.

9.3.1 Initialization using known landmarks (KL's)

In this section, the idea is to start the filter from the ground and move the camera above APRIL markers. The APRIL tag detector identifies each marker with a unique ID and tracks the four corners of each marker. Since the position of each corner is known apriori no triangulation is needed and the filter can be initialized very robustly. In this section, the measurement model for known landmarks as described in section 3.4 is used.

The reference frame ($p = [0, 0, 0]^T$) is set to the lower left corner of the APRIL tag number 0 which is *not* identical to the reference frame printed on the APRIL tag sheet shown in figure 9.7 (see also small offset in position initialization below).

The initialization procedure is as follows:

1. State initialization: If the filter starts from the ground and the quadrocopter is located above the reference frame printed on the APRIL tag sheet, the filter should be initialized as follows:

```
p << -0.02, 0.02, 0.06;
v << 0.0, 0.0, 0.0;
q.w() = -1.0;
q.x() = 0.0;
q.y() = 0.0;
q.z() = 0.0;
q.normalize();
g << 0.0, 0.0, 9.81;
meas->Geta_m() = g;

b_w << 0.0, 0.0, 0.0;
b_a << 0.0, 0.0, 0.0;
p_ic << -0.043767751073328504,
         0.020883659054156595,
        -0.059969859838216094;

q_ic.x() = 0.923921268262626;
q_ic.y() = -0.3825817981382697;
q_ic.z() = 0.00018584003881617656;
```

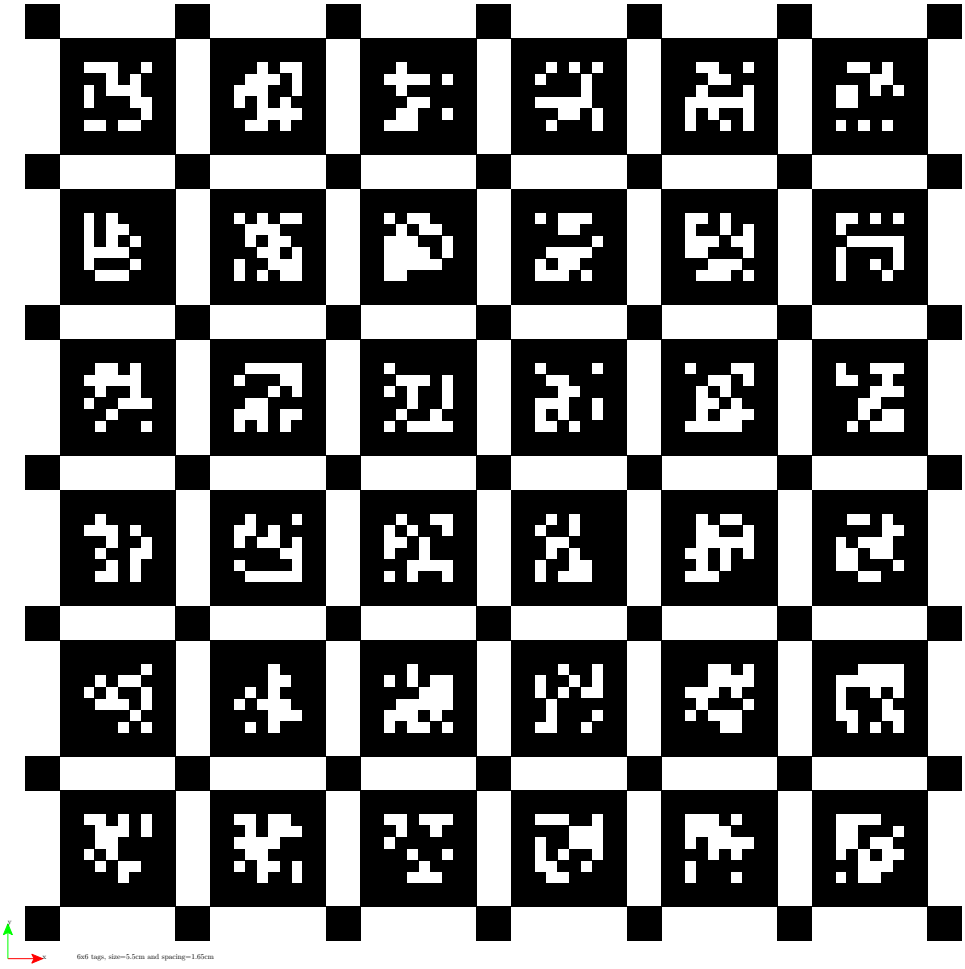


Figure 9.7: 6×6 APRIL tag printout used to initialize the filter with apriori known landmarks (image: [24])

```
q_ic.w() = 0.0007894610496075483;

q_ic.normalize();
q.normalize();
```

1. Covariance initialization: As described in section 9.1.

2. After the initialization procedure, the standard feature tracker is used to track apriori unknown features (opportunistic features).

3. The filter switches from known feature tracking to opportunistic feature tracking after a certain condition is met, such as e.g.

   - Image timestamp
   - After a certain number of APRIL tags have been detected and in the current image no more APRIL tags are detected
   - The covariance of the filter states drops below a certain threshold

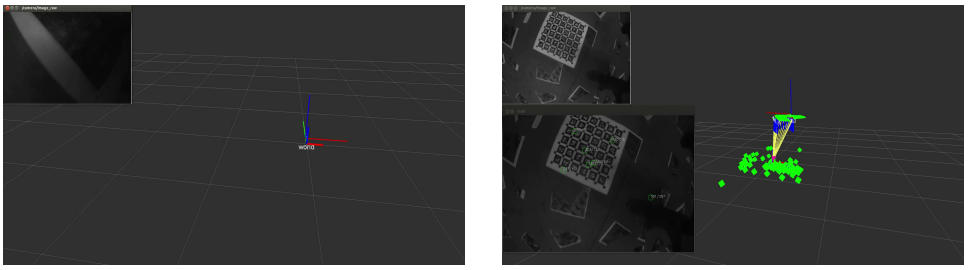


Figure 9.8: Initialization procedure: The figure on the left shows the take-off from ground and part of the APRIL tag printout. The corners of each APRIL tags are used as known landmarks to initialize the filter. The right figure shows the opportunistic feature tracker taking over after the initialization procedure.

9.3.2 Initialization using opportunistic features (OF's) only

A more convenient way to initialize the filter is to again take off from ground and to simply use the opportunistic feature tracker right from the start.

The initialization procedure is as follows:

1. State initialization:

```
p << 0.0, 0.0, 0.0;
v << 0.0, 0.0, 0.0;
q.w() = 1.0;
q.x() = 0.0;
q.y() = 0.0;
q.z() = 0.0;
q.normalize();
g << 0.0, 0.0, 9.81;
meas->Geta_m() = g;

b_w << 0.0, 0.0, 0.0;
b_a << 0.0, 0.0, 0.0;
p_ic << -0.043767751073328504,
        0.020883659054156595,
        -0.059969859838216094;

q_ic.x() = 0.923921268262626;
q_ic.y() = -0.3825817981382697;
q_ic.z() = 0.00018584003881617656;
q_ic.w() = 0.0007894610496075483;

q_ic.normalize();
q.normalize();
```

   In contrast to the known feature tracking described above, the initial position is **arbitrary**.

2. Covariance initialization: As described in section 9.1.

Figure 9.9 shows the estimated height during the initialization phase for the two proposed initialization methods:

- Initialization using known landmarks (KL) in blue.

- Initialization using opportunistic features (OF) in red.

The quadrocopter is vertically picked up from ground to a height of about 1.1 m. The plot shows that the height estimate of both approaches is almost identical. Note that the first filter update for both approaches is at 0.1 s. Before that time, the estimates of both approaches are *identical* since they are only governed by the IMU propagation[3]

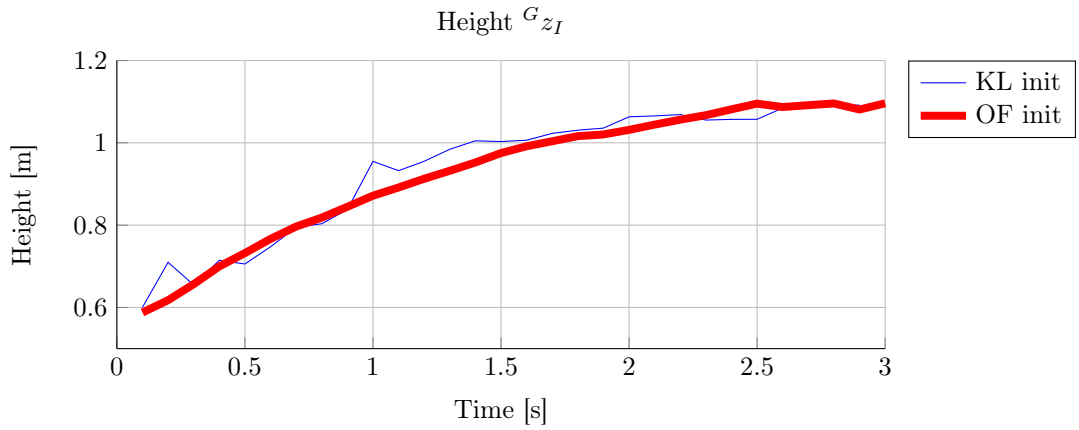


Figure 9.9: Ground initialization: Comparing the height during initialization using known landmarks (red) and opportunistic features (blue)

[3]The plots shows the topics published by the ROS framework; the first data is published after the first Kalman filter update which is at around 0.1 s.

Chapter 10

Discussion and Outlook

In this thesis, the multi-state constraint measurement model proposed by Mourikis et al. [5] was incorporated into ASL's modular sensor fusion framework to guarantee modularity of the filter framework: In a next step, a new sensor, e.g. a GPS receiver, could be added to the existing IMU and camera module. Likewise, the implemented feature tracker and triangulation module can be considered as black boxes with clearly defined inputs and outputs and can be exchanged easily.

Several modification of the original filter version allow a wider range of application: The original EKF filter state is augmented to with camera-IMU rotation as well as translation to automatically estimate the camera-IMU transformation.
Besides the measurement model for opportunistic features which are tracked by the regular feature tracker, an APRIL feature tracker and the corresponding measurement model for known landmarks as well as the measurement model for persistent features are derived.

The results presented in chapter 9 show that the IMU poses estimated by the SSF/PTAM and MSF/MSCEKF framework are comparable - a Vicon camera setup as shown in figure 10.1 and 10.2 would be needed to exactly benchmark the performance and is proposed as a next step.

Figure 10.1: Vicon camera (image: Institute for Dynamic Systems and Control (IDSC), ETH)

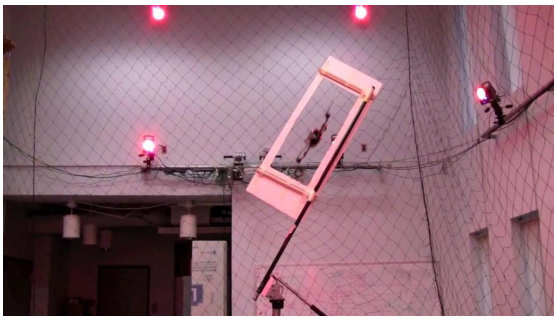


Figure 10.2: Quadrocopter pose tracked by a Vicon camera rig (image: General Robotics, Automation, Sensing and Perception (GRASP) Lab, University of Pennsylvania)

The great advantage of the MSC-EKF approach becomes apparent in difficult manoeuvres such as first flying over flat ground followed by a table as shown in chapter 9. The SSF/PTAM framework has difficulties to map the features of both the flat

ground and the 3d structure leading eventually to complete divergence of the filter. Another shortcoming of the SSF/PTAM could be avoided: The MSC-EKF can autonomously start from ground - with or without known landmarks.

Furthermore, the MSC-EKF framework runs close-to real-time on an external computer with 17 camera poses and at a IMU and camera rate of 100 Hz and 15 fps respectively.

Appendix A

Quaternions

Refer e.g. to [8] for more information.

$$\bar{q} = \delta\bar{q} \otimes \hat{\bar{q}} \tag{A.1}$$

$$\delta\bar{q} = \begin{bmatrix} \delta\mathbf{q} \\ \delta q_4 \end{bmatrix} \approx \begin{bmatrix} \frac{1}{2}\delta\theta \\ 1 \end{bmatrix} \tag{A.2}$$

$$\mathbf{C}(\delta\bar{q}) \approx \mathbf{I}_3 - \lfloor \delta\theta \times \rfloor \tag{A.3}$$

$${}^{L}_{G}\mathbf{C}(\bar{q}) = {}^{L}_{G}\mathbf{C}(\delta\bar{q} \otimes \hat{\bar{q}}) = {}^{L}_{\hat{L}}\mathbf{C}(\delta\bar{q}) \cdot {}^{\hat{L}}_{G}\mathbf{C}(\hat{\bar{q}}) \tag{A.4}$$

$$\lfloor \boldsymbol{a} \times \rfloor \, \boldsymbol{b} = - \lfloor \boldsymbol{b} \times \rfloor \, \boldsymbol{a} \tag{A.5}$$

Quaternion integration

$${}^{L}_{G}\dot{\bar{q}} = \frac{1}{2}\Omega(\omega){}^{L}_{G}\bar{q}(t) \tag{A.6}$$

Appendix B

Multi-view triangulation

Algorithm 9 Gradient Descent multi-view triangulation,
T : precision threshold
γ : step size
r : measurement residual
J : measurement jacobian
$h_{m,i}$: observation of feature j in camera i
$h_{p,i}$: predicted observation of feature j in camera i
$M^{(j)}$: number of observations for feature j

function GradientDescent(α_{init}, β_{init}, ρ_{init}, $iter_{max}$, T)
  $\alpha \leftarrow \alpha_{init}$
  $\beta \leftarrow \beta_{init}$
  $\rho \leftarrow \rho_{init}$
  while $\|r_{last} - r_{current}\|_2 > T$ and $iter < iter_{max}$ **do**
    for $i = 1$: all camera views $M^{(j)}$ **do**
      ${}^{C_n}\mathbf{C}_{C_i} \leftarrow {}^{C_n}\mathbf{C}_G \cdot {}^{G}\mathbf{C}_{C_i}$
      ${}^{C_i}\mathbf{p}_{C_n} \leftarrow {}^{C_i}\mathbf{C}_G {}^{G}\mathbf{p}_{C_n} - {}^{C_i}\mathbf{C}_G {}^{G}\mathbf{p}_{C_n}$
      $h_{p,i} \leftarrow {}^{C_n}\mathbf{C}_{C_i} \cdot \begin{bmatrix}\alpha & \beta & 1\end{bmatrix}^T + \rho \cdot {}^{C_i}\mathbf{p}_{C_n}$
      $h_{m,i} = \frac{1}{Z_{C,m}} \begin{bmatrix}X_{C,m} & Y_{C,m}\end{bmatrix}^T$
      $r_i \leftarrow h_{p,i} - h_{m,i}$ ▷ Residuals in camera coordinates
      $r \leftarrow \begin{bmatrix}r & r_i\end{bmatrix}^T$ ▷ Stack residuals
      $J_p \leftarrow \frac{1}{h_{p,i}(3)} \begin{bmatrix}1 & 0 & -\frac{h_{p,i}(1)}{h_{p,i}(3)} \\ 0 & 1 & -\frac{h_{p,i}(2)}{h_{p,i}(3)}\end{bmatrix}$ ▷ Jacobian perspective camera model
      $J_\alpha \leftarrow {}^{C_n}\mathbf{C}_{C_i} \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$
      $J_\beta \leftarrow {}^{C_n}\mathbf{C}_{C_i} \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$
      $J_i \leftarrow J_p \begin{bmatrix}J_\alpha & J_\beta & J_\rho\end{bmatrix}$ ▷ $J_i \equiv \begin{bmatrix}r_\alpha & r_\beta & r_\rho\end{bmatrix}$
      $J \leftarrow \begin{bmatrix}J & J_i\end{bmatrix}^T$ ▷ Stack Jacobians
    end for
    $\Delta \leftarrow \gamma J^T r$ ▷ Dim.: $J = 2M^{(j)} \times 3$, $r = 2M^{(j)} \times 1$
    $\begin{bmatrix}\alpha & \beta & \rho\end{bmatrix}^T \leftarrow \begin{bmatrix}\alpha & \beta & \rho\end{bmatrix}^T - \gamma\Delta$
  end while
  ${}^{G}\mathbf{p}_{fj} \leftarrow \frac{1}{\rho} {}^{G}\mathbf{C}_{C_n} \begin{bmatrix}\alpha & \beta & 1\end{bmatrix}^T + {}^{G}\mathbf{p}_{C_n}$
end function

Algorithm 10 Newton multi-view triangulation,

T : precision threshold
γ : step size (set to 1)
r : measurement residual
J : measurement jacobian
$h_{m,i}$: observation of feature j in camera i
$h_{p,i}$: predicted observation of feature j in camera i

function NEWTON(α_{init}, β_{init}, ρ_{init}, $iter_{max}$, T)
  $\alpha \leftarrow \alpha_{init}$
  $\beta \leftarrow \beta_{init}$
  $\rho \leftarrow \rho_{init}$
  while $\|r_{last} - r_{current}\|_2 > T$ and $iter < iter_{max}$ **do**
    for $i = 1$: all camera views $M^{(j)}$ **do**
      ${}^{C_n}\mathbf{C}_{C_i} \leftarrow {}^{C_n}\mathbf{C}_G \cdot {}^{G}\mathbf{C}_{C_i}$
      ${}^{C_i}\mathbf{p}_{C_n} \leftarrow {}^{C_i}\mathbf{C}_G {}^{G}\mathbf{p}_{C_n} - {}^{C_i}\mathbf{C}_G {}^{G}\mathbf{p}_{C_n}$
      $h_{p,i} \leftarrow {}^{C_n}\mathbf{C}_{C_i} \cdot \begin{bmatrix}\alpha & \beta & 1\end{bmatrix}^T + \rho \cdot {}^{C_i}\mathbf{p}_{C_n}$
      $h_{m,i} = \frac{1}{Z_{C,m}} \begin{bmatrix}X_{C,m} & Y_{C,m}\end{bmatrix}^T$
      $r_i \leftarrow h_{p,i} - h_{m,i}$ ▷ Residuals in camera coordinates
      $r \leftarrow \begin{bmatrix}r & r_i\end{bmatrix}^T$ ▷ Stack residuals
      $J_p \leftarrow \frac{1}{h_{p,i}(3)} \begin{bmatrix}1 & 0 & -\frac{h_{p,i}(1)}{h_{p,i}(3)} \\ 0 & 1 & -\frac{h_{p,i}(2)}{h_{p,i}(3)}\end{bmatrix}$ ▷ Jacobian perspective camera model
      $J_\alpha \leftarrow {}^{C_n}\mathbf{C}_{C_i} \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$
      $J_\beta \leftarrow {}^{C_n}\mathbf{C}_{C_i} \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$
      $J_i \leftarrow J_p \begin{bmatrix}J_\alpha & J_\beta & J_\rho\end{bmatrix}$
      $J \leftarrow \begin{bmatrix}J & J_i\end{bmatrix}^T$ ▷ Stack Jacobians
      $r_{\alpha\alpha} \leftarrow \begin{bmatrix}2\left(\frac{J_\alpha(1)J_\alpha(3)}{(h_{p,i}(3))^2} - \frac{J_\alpha(3)^2 h_{p,i}(1)}{(h_{p,i}(3))^3}\right) & 2\left(\frac{J_\alpha(2)J_\alpha(3)}{(h_{p,i}(3))^2} - \frac{J_\alpha(3)^2 h_{p,i}(2)}{(h_{p,i}(3))^3}\right)\end{bmatrix}$
      $r_{\alpha\beta} \leftarrow \begin{bmatrix}\frac{J_\alpha(1)J_\beta(3)+J_\beta(1)J_\alpha(3)}{(h_{p,i}(3))^2} - 2\frac{J_\alpha(3)J_\beta(3)h_{p,i}(1)}{(h_{p,i}(3))^3} & \frac{J_\alpha(2)J_\beta(3)+J_\beta(2)J_\alpha(3)}{(h_{p,i}(3))^2} - 2\frac{J_\alpha(3)J_\beta(3)h_{p,i}(2)}{(h_{p,i}(3))^3}\end{bmatrix}$
      $r_{\alpha\rho} \leftarrow \begin{bmatrix}\frac{J_\alpha(3)J_\rho(1)+J_\alpha(1)J_\rho(3)}{(h_{p,i}(3))^2} - 2\frac{J_\alpha(3)J_\rho(3)h_{p,i}(1)}{(h_{p,i}(3))^3} & \frac{J_\alpha(3)J_\rho(2)+J_\alpha(2)J_\rho(3)}{(h_{p,i}(3))^2} - 2\frac{J_\alpha(3)J_\rho(3)h_{p,i}(2)}{(h_{p,i}(3))^3}\end{bmatrix}$
      $r_{\beta\beta} \leftarrow \begin{bmatrix}2\left(\frac{J_\beta(1)J_\beta(3)}{(h_{p,i}(3))^2} - \frac{J_\beta(3)^2 h_{p,i}(1)}{(h_{p,i}(3))^3}\right) & 2\left(\frac{J_\beta(2)J_\beta(3)}{(h_{p,i}(3))^2} - \frac{J_\beta(3)^2 h_{p,i}(2)}{(h_{p,i}(3))^3}\right)\end{bmatrix}$
      $r_{\beta\rho} \leftarrow \begin{bmatrix}\frac{J_\beta(3)J_\rho(1)+J_\beta(1)J_\rho(3)}{(h_{p,i}(3))^2} - 2\frac{J_\beta(3)J_\rho(3)h_{p,i}(1)}{(h_{p,i}(3))^2} & \frac{J_\beta(3)J_\rho(2)+J_\beta(2)J_\rho(3)}{(h_{p,i}(3))^2} - 2\frac{J_\beta(3)J_\rho(3)h_{p,i}(2)}{(h_{p,i}(3))^2}\end{bmatrix}$
      $r_{\rho\rho} \leftarrow \begin{bmatrix}2\left(\frac{J_\rho(1)J_\rho(3)}{(h_{p,i}(3))^2} - \frac{J_\rho(3)^2 h_{p,i}(1)}{(h_{p,i}(3))^3}\right) & 2\left(\frac{J_\rho(2)J_\rho(3)}{(h_{p,i}(3))^2} - 2\frac{J_\rho(3)^2 h_{p,i}(2)}{(h_{p,i}(3))^3}\right)\end{bmatrix}$
      $Fr_i \leftarrow \begin{bmatrix}r_{\alpha\alpha}r_i & r_{\alpha\beta}r_i & r_{\alpha\rho}r_i \\ r_{\alpha\beta}r_i & r_{\beta\beta}r_i & r_{\beta\rho}r_i \\ r_{\alpha\rho}r_i & r_{\beta\rho}r_i & r_{\rho\rho}r_i\end{bmatrix}$
      $Fr \leftarrow Fr + Fr_i$
    end for
    $\Delta \leftarrow (J^T J + Fr)^{-1} J^T r$ ▷ Dim.: $J = 2M^{(j)} \times 3$, $Fr = 3 \times 3$, $r = 2M^{(j)} \times 1$
    $\begin{bmatrix}\alpha & \beta & \rho\end{bmatrix}^T \leftarrow \begin{bmatrix}\alpha & \beta & \rho\end{bmatrix}^T - \gamma\Delta$
  end while
  ${}^{G}\mathbf{p}_{fj} \leftarrow \frac{1}{\rho} {}^{G}\mathbf{C}_{C_n} \begin{bmatrix}\alpha & \beta & 1\end{bmatrix}^T + {}^{G}\mathbf{p}_{C_n}$
end function

Algorithm 11 Levenberg-Marquardt multi-view triangulation,

T : precision threshold
γ : step size (set to 1)
r : measurement residual
J : measurement jacobian
$h_{m,i}$: observation of feature j in camera i
$h_{p,i}$: predicted observation of feature j in camera i
$M^{(j)}$: number of observations for feature j

function LevenbergMarquardt(α_{init}, β_{init}, ρ_{init}, $iter_{max}$, T)
  $\alpha \leftarrow \alpha_{init}$, $\beta \leftarrow \beta_{init}$, $\rho \leftarrow \rho_{init}$
  for $i = 1 : iter_{max}$ **do** ▷ Iteration loop
    while $r_{new} > r_{old}$ **do**
      $\lambda = \lambda \cdot \lambda_{factor}$ ▷ Reject Δ: Increase λ
      for $i = 1$: all camera views $M^{(j)}$ **do** ▷ Measurement loop
        ${}^{C_n}\mathbf{C}_{C_i} \leftarrow {}^{C_n}\mathbf{C}_G \cdot {}^{G}\mathbf{C}_{C_i}$
        ${}^{C_i}\mathbf{p}_{C_n} \leftarrow {}^{C_i}\mathbf{C}_G {}^{G}\mathbf{p}_{C_n} - {}^{C_i}\mathbf{C}_G {}^{G}\mathbf{p}_{C_n}$
        $h_{p,i} \leftarrow {}^{C_n}\mathbf{C}_{C_i} \cdot \begin{bmatrix}\alpha & \beta & 1\end{bmatrix}^T + \rho \cdot {}^{C_i}\mathbf{p}_{C_n}$
        $h_{m,i} = \frac{1}{Z_{C,m}} \begin{bmatrix}X_{C,m} & Y_{C,m}\end{bmatrix}^T$
        $r_i \leftarrow h_{p,i} - h_{m,i}$ ▷ Residuals in camera coordinates
        $r \leftarrow \begin{bmatrix}r & r_i\end{bmatrix}^T$ ▷ Stack residuals
        $J_p \leftarrow \frac{1}{h_{p,i}(3)} \begin{bmatrix}1 & 0 & -\frac{h_{p,i}(1)}{h_{p,i}(3)} \\ 0 & 1 & -\frac{h_{p,i}(2)}{h_{p,i}(3)}\end{bmatrix}$ ▷ Jacobian perspective camera model
        $J_\alpha \leftarrow {}^{C_n}\mathbf{C}_{C_i} \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$
        $J_\beta \leftarrow {}^{C_n}\mathbf{C}_{C_i} \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$
        $J_i \leftarrow J_p \begin{bmatrix}J_\alpha & J_\beta & J_\rho\end{bmatrix}$ ▷ $J_i \equiv \begin{bmatrix}r_\alpha & r_\beta & r_\rho\end{bmatrix}$
        $J \leftarrow \begin{bmatrix}J & J_i\end{bmatrix}^T$ ▷ Stack Jacobians
      end for
      $\Delta \leftarrow \gamma(J^T J + \lambda \mathbf{I}_3)^{-1} J^T r$ ▷ Dim.: $J = 2M^{(j)} \times 3$, $r = 2M^{(j)} \times 1$
      $\begin{bmatrix}\alpha' & \beta' & \rho'\end{bmatrix}^T \leftarrow \begin{bmatrix}\alpha & \beta & \rho\end{bmatrix}^T - \gamma\Delta$
      $r_{old} \leftarrow \sqrt{r^T r}$
      $r \leftarrow [\,], r_i \leftarrow [\,]$ ▷ Clear variables
      for $i = 1$: all camera views $M^{(j)}$ **do** ▷ Re-evaluate measurement loop
        ${}^{C_n}\mathbf{C}_{C_i} \leftarrow {}^{C_n}\mathbf{C}_G \cdot {}^{G}\mathbf{C}_{C_i}$
        ${}^{C_i}\mathbf{p}_{C_n} \leftarrow {}^{C_i}\mathbf{C}_G {}^{G}\mathbf{p}_{C_n} - {}^{C_i}\mathbf{C}_G {}^{G}\mathbf{p}_{C_n}$
        $h_{p,i} \leftarrow {}^{C_n}\mathbf{C}_{C_i} \cdot \begin{bmatrix}\alpha' & \beta' & 1\end{bmatrix}^T + \rho' \cdot {}^{C_i}\mathbf{p}_{C_n}$
        $h_{m,i} = \frac{1}{Z_{C,m}} \begin{bmatrix}X_{C,m} & Y_{C,m}\end{bmatrix}^T$
        $r_i \leftarrow h_{p,i} - h_{m,i}$ ▷ Residuals in camera coordinates
        $r \leftarrow \begin{bmatrix}r & r_i\end{bmatrix}^T$ ▷ Stack residuals
      end for
      $r_{new} \leftarrow \sqrt{r^T r}$
    end while
    $\begin{bmatrix}\alpha & \beta & \rho\end{bmatrix}^T \leftarrow \begin{bmatrix}\alpha' & \beta' & \rho'\end{bmatrix}^T$ ▷ Accept Δ
    $\lambda = \lambda / \lambda_{factor}$ ▷ Decrease λ
  end for
  ${}^{G}\mathbf{p}_{fj} \leftarrow \frac{1}{\rho} {}^{G}\mathbf{C}_{C_n} \begin{bmatrix}\alpha & \beta & 1\end{bmatrix}^T + {}^{G}\mathbf{p}_{C_n}$
end function

Algorithm 12 PXMETHOD multi-view triangulation,

T : precision threshold
γ : step size
r : measurement residual
J : measurement jacobian
u, v : feature observation [pixel]
P_i : projection matrix for camera i

function PXMETHOD(u, v, P_i, $iter_{max}$, T)
  while $\|r_{last} - r_{current}\|_2 > T$ and $iter < iter_{max}$ **do**
    for $i = 1$: all camera views **do**
      $\Delta_{12,1} \leftarrow P_i(1,1)P_i(3,2) - P_i(1,2)P_i(3,1)$
      $\Delta_{13,1} \leftarrow P_i(1,1)P_i(3,3) - P_i(1,3)P_i(3,1)$
      $\Delta_{14,1} \leftarrow P_i(1,1)P_i(3,4) - P_i(1,4)P_i(3,1)$
      $\Delta_{21,1} \leftarrow P_i(1,2)P_i(3,1) - P_i(1,1)P_i(3,2)$
      $\Delta_{23,1} \leftarrow P_i(1,2)P_i(3,3) - P_i(1,3)P_i(3,2)$
      $\Delta_{24,1} \leftarrow P_i(1,2)P_i(3,4) - P_i(1,4)P_i(3,2)$
      $\Delta_{31,1} \leftarrow P_i(1,3)P_i(3,1) - P_i(1,1)P_i(3,3)$
      $\Delta_{32,1} \leftarrow P_i(1,3)P_i(3,2) - P_i(1,2)P_i(3,3)$
      $\Delta_{34,1} \leftarrow P_i(1,3)P_i(3,4) - P_i(1,4)P_i(3,3)$
      $\Delta_{12,2} \leftarrow P_i(2,1)P_i(3,2) - P_i(2,2)P_i(3,1)$
      $\Delta_{13,2} \leftarrow P_i(2,1)P_i(3,3) - P_i(2,3)P_i(3,1)$
      $\Delta_{14,2} \leftarrow P_i(2,1)P_i(3,4) - P_i(2,4)P_i(3,1)$
      $\Delta_{21,2} \leftarrow P_i(2,2)P_i(3,1) - P_i(2,1)P_i(3,2)$
      $\Delta_{23,2} \leftarrow P_i(2,2)P_i(3,3) - P_i(2,3)P_i(3,2)$
      $\Delta_{24,2} \leftarrow P_i(2,2)P_i(3,4) - P_i(2,4)P_i(3,2)$
      $\Delta_{31,2} \leftarrow P_i(2,3)P_i(3,1) - P_i(2,1)P_i(3,3)$
      $\Delta_{32,2} \leftarrow P_i(2,3)P_i(3,2) - P_i(2,2)P_i(3,3)$
      $\Delta_{34,2} \leftarrow P_i(2,3)P_i(3,4) - P_i(2,4)P_i(3,3)$
      $num \leftarrow P_i(3,1)X + P_i(3,2)Y + P_i(3,3)Z + P_i(3,4)$
      $r_1 \leftarrow \frac{P_i(1,1)X + P_i(1,2)Y + P_i(1,3)Z + P_i(1,4)}{num} - u$
      $r_2 \leftarrow \frac{P_i(2,1)X + P_i(2,2)Y + P_i(2,3)Z + P_i(2,4)}{num} - v$
      $r_i \leftarrow \begin{bmatrix} r_1 & r_2 \end{bmatrix}^T$ ▷ Residual in pixel coordinates
      $r \leftarrow \begin{bmatrix} r & r_i \end{bmatrix}^T$ ▷ Stack residuals
      $J_n \leftarrow \begin{bmatrix} 0 & \Delta_{12,1} & \Delta_{13,1} & \Delta_{14,1} & \Delta_{21,1} & 0 & \Delta_{23,1} & \Delta_{24,1} & \Delta_{31,1} & \Delta_{32,1} & 0 & \Delta_{34,1} \\ 0 & \Delta_{12,2} & \Delta_{13,2} & \Delta_{14,2} & \Delta_{21,2} & 0 & \Delta_{23,2} & \Delta_{24,2} & \Delta_{31,2} & \Delta_{32,2} & 0 & \Delta_{34,2} \end{bmatrix}$
      $X_h \leftarrow \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$
      $\delta \leftarrow \begin{bmatrix} X_h & \mathbf{0}_{4,1} & \mathbf{0}_{4,1} \\ \mathbf{0}_{4,1} & X_h & \mathbf{0}_{4,1} \\ \mathbf{0}_{4,1} & \mathbf{0}_{4,1} & X_h \end{bmatrix}$ ▷ Kron
      $J_i \leftarrow J_n \cdot \delta \cdot num^{-2}$
      $J \leftarrow \begin{bmatrix} J & J_i \end{bmatrix}^T$ ▷ Stack jacobians
    end for
    $\Delta \leftarrow (J^T J)^{-1} J^T r$
    $\begin{bmatrix} \alpha & \beta & \rho \end{bmatrix}^T \leftarrow \begin{bmatrix} \alpha & \beta & \rho \end{bmatrix}^T - \gamma\Delta$
  end while
  ${}^G\mathbf{p}_{fj} \leftarrow \frac{1}{\rho}\, {}^G\mathbf{C}_{C_n} \begin{bmatrix} \alpha & \beta & 1 \end{bmatrix}^T + {}^G\mathbf{p}_{C_n}$
end function

Appendix C

Triangulation analysis

C.1 40 views

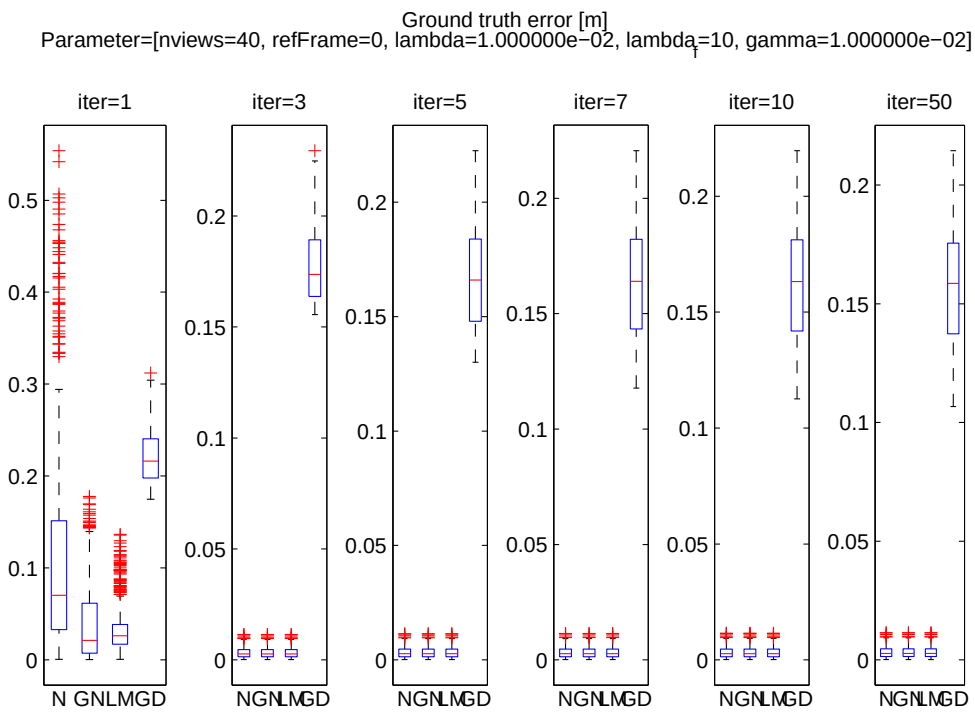


Figure C.1: Ground truth error

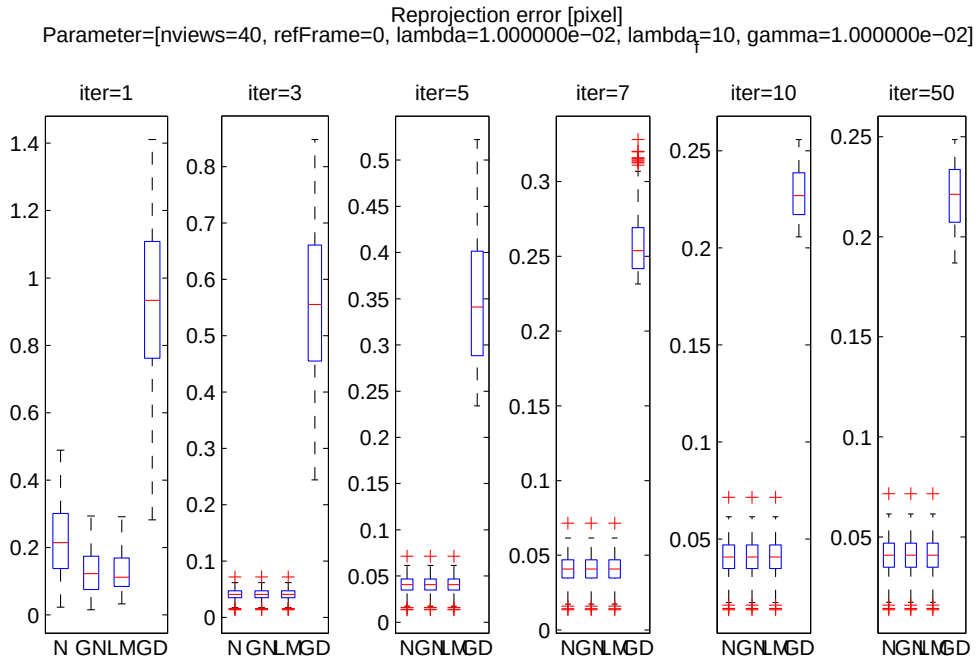


Figure C.2: Reprojection error

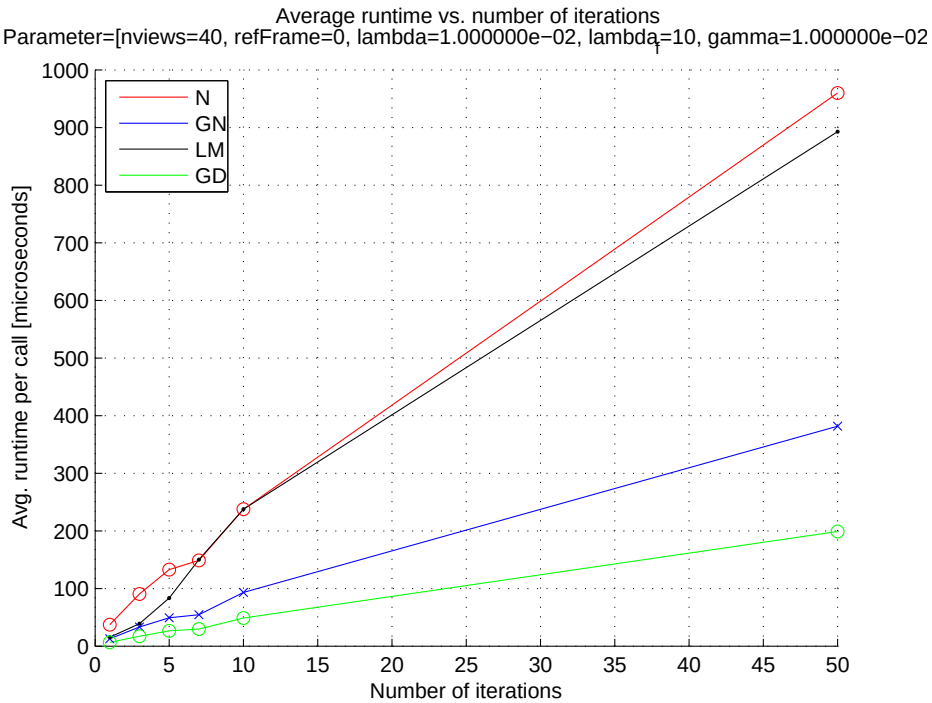


Figure C.3: Runtime

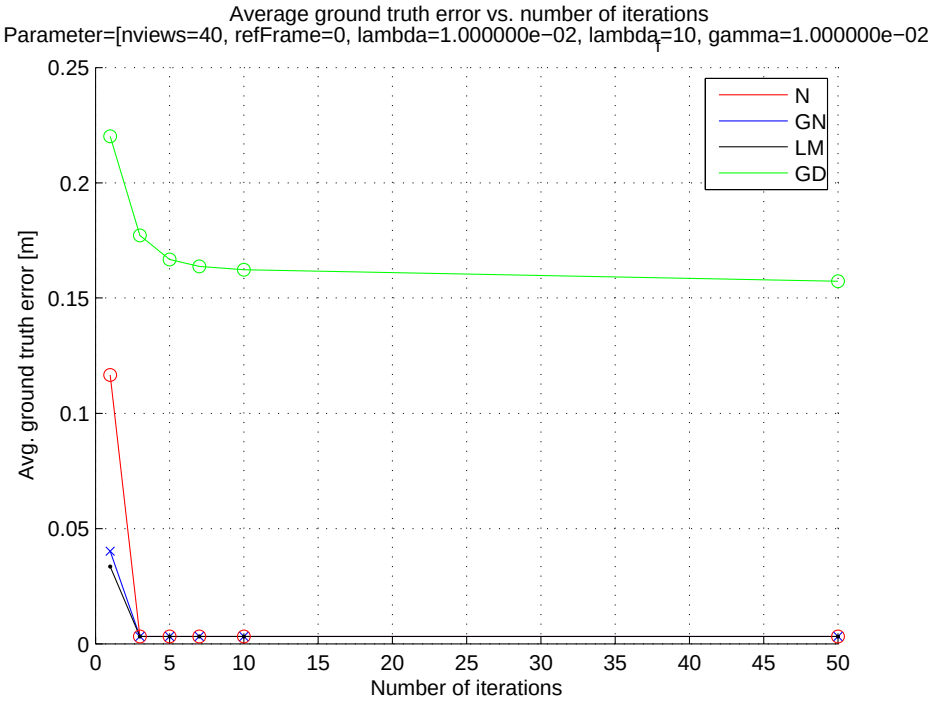


Figure C.4: Ground truth error

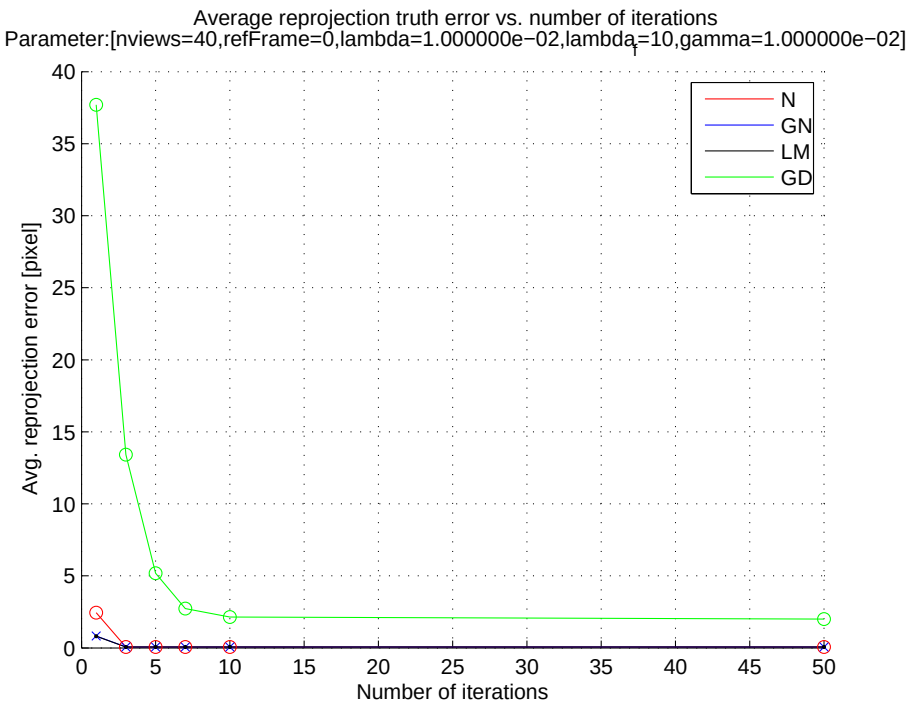


Figure C.5: Reprojection error

C.2 10 views

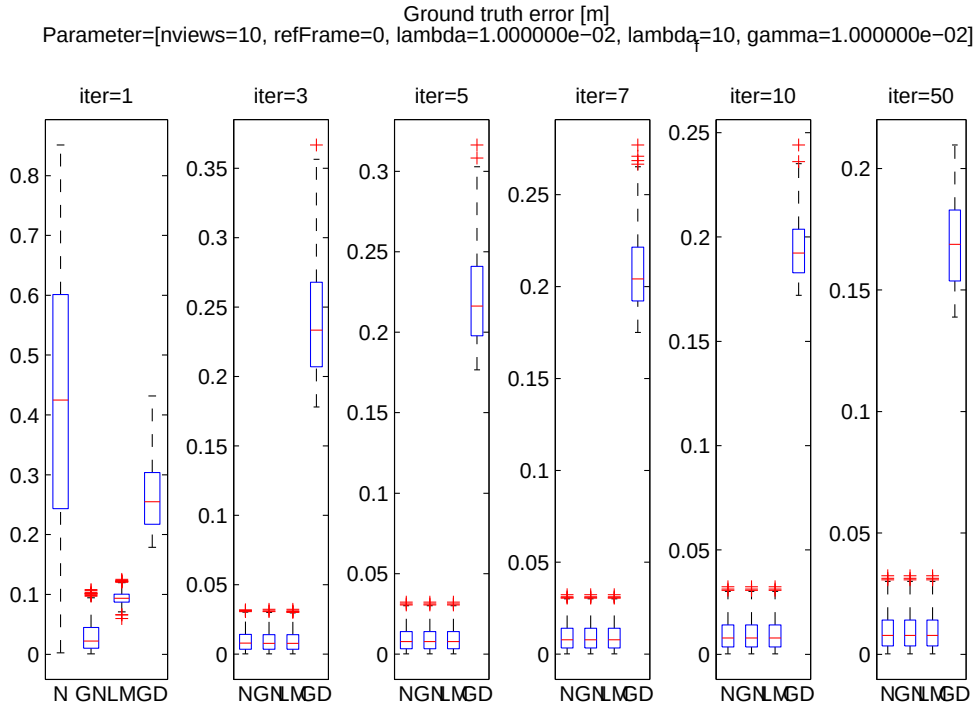


Figure C.6: Ground truth error

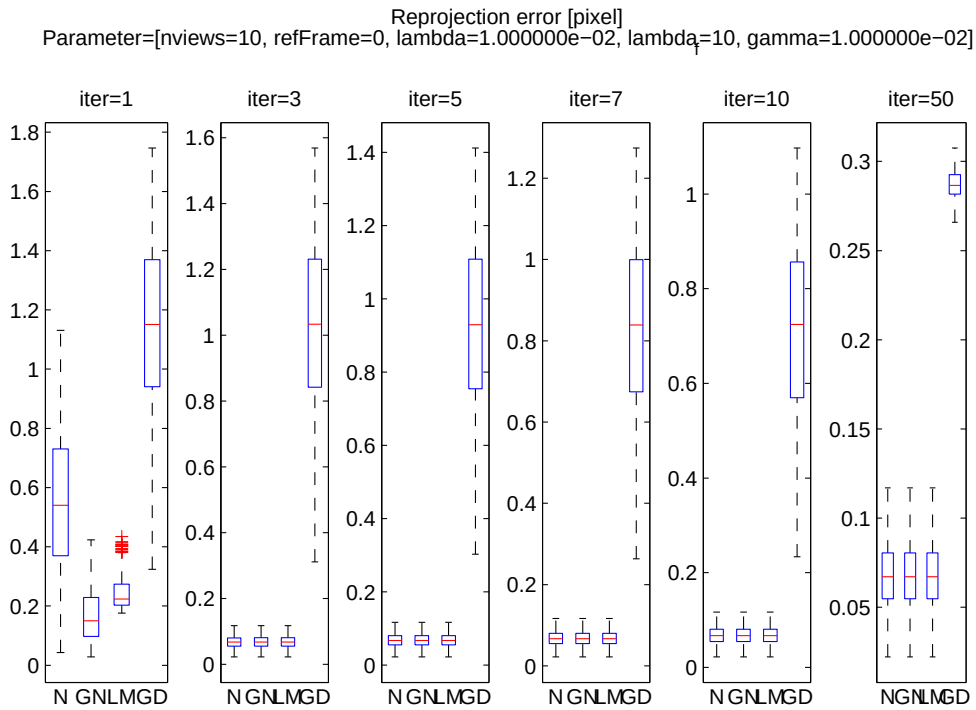


Figure C.7: Reprojection error

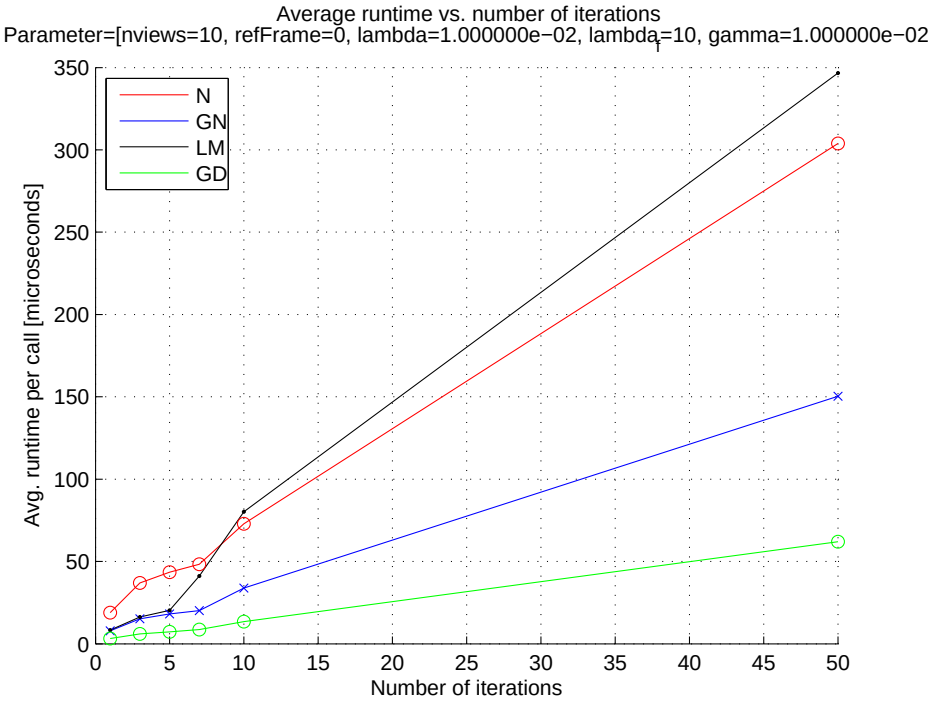


Figure C.8: Runtime

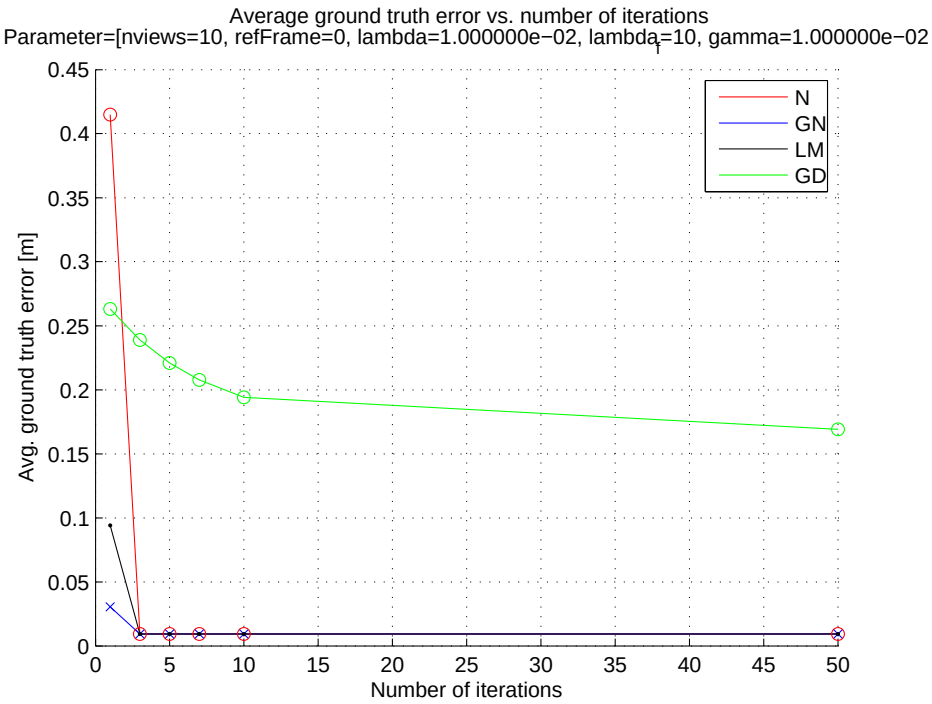


Figure C.9: Ground truth error

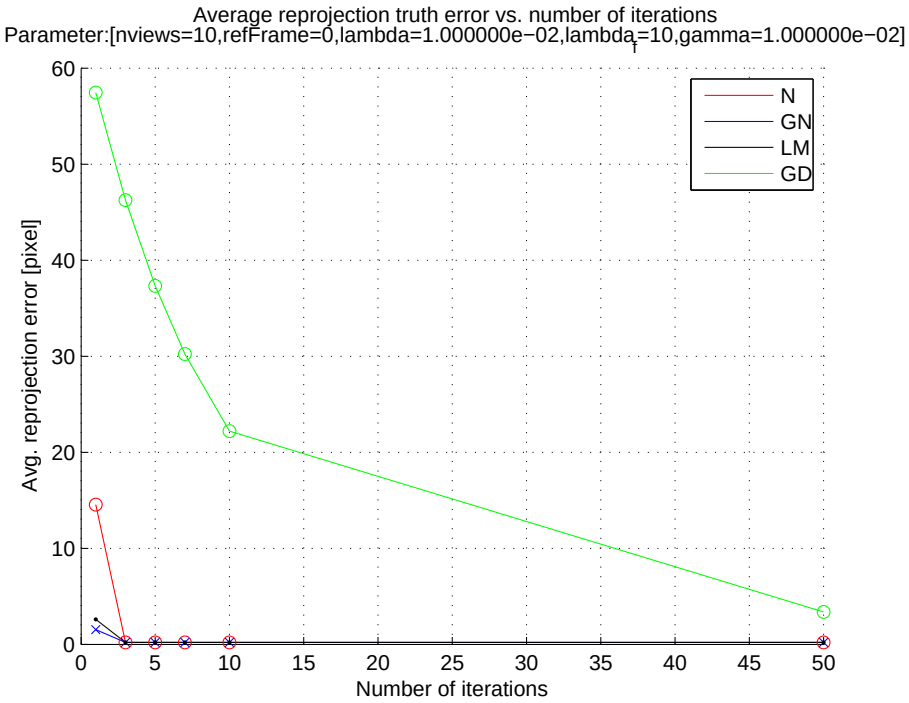


Figure C.10: Reprojection error

C.3 7 views

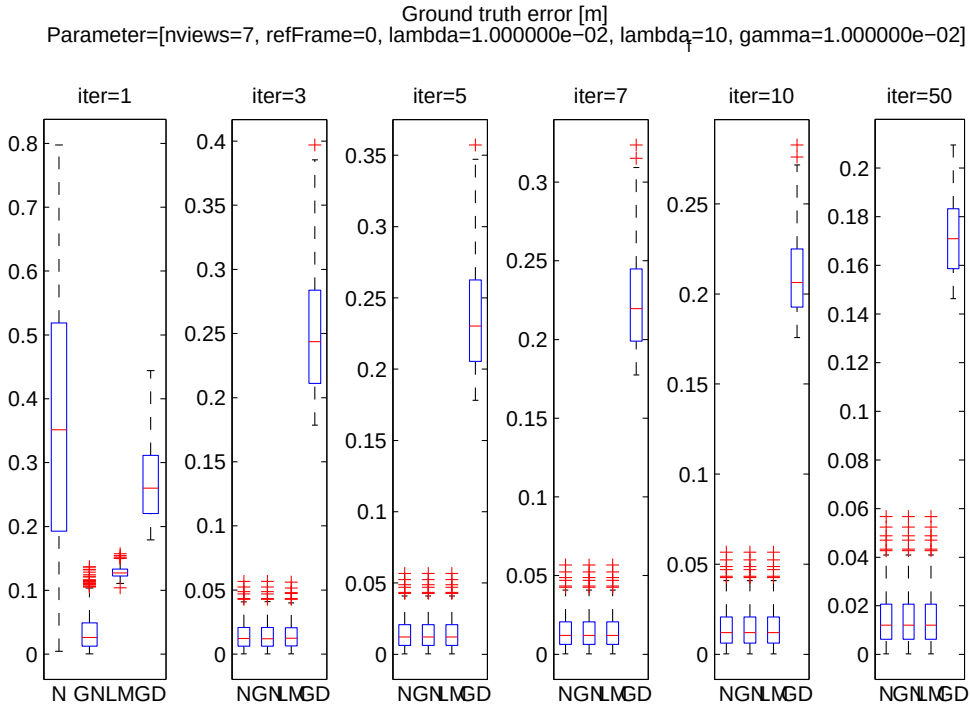


Figure C.11: Ground truth error

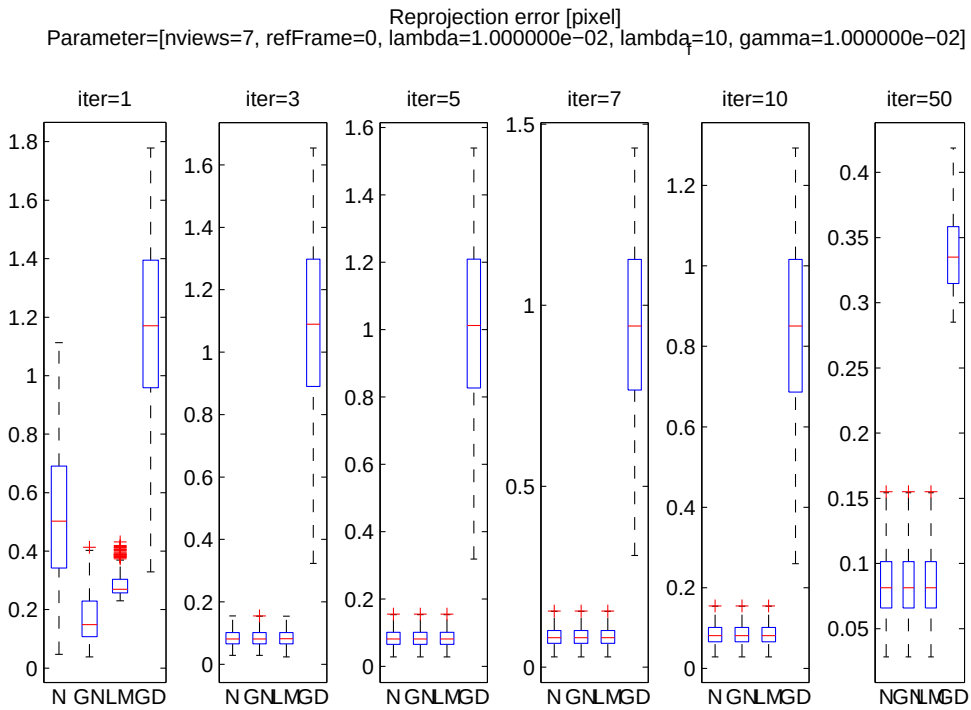


Figure C.12: Reprojection error

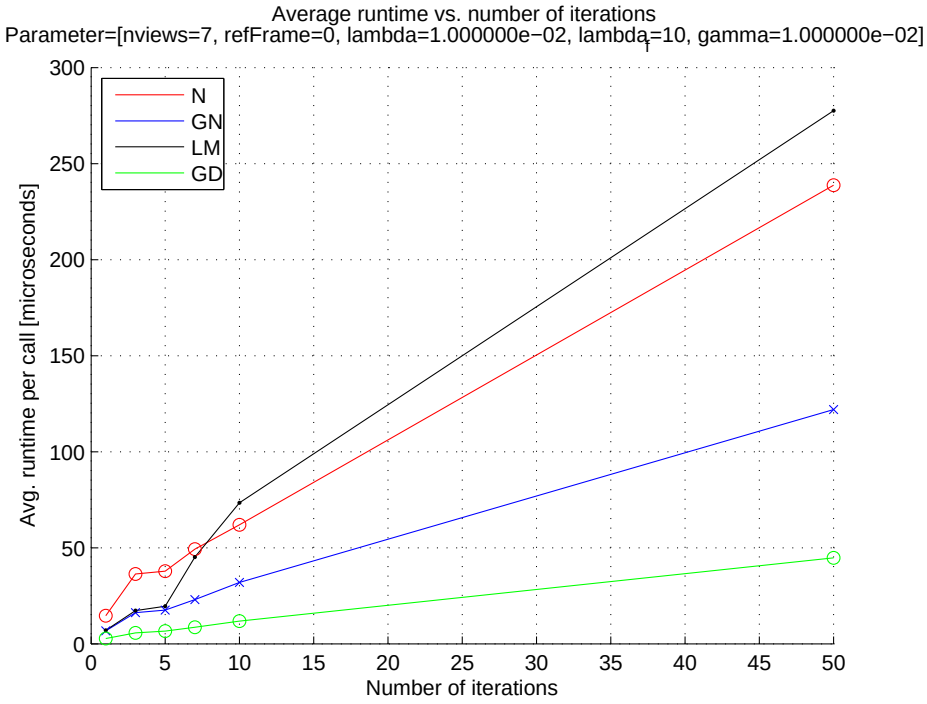


Figure C.13: Runtime

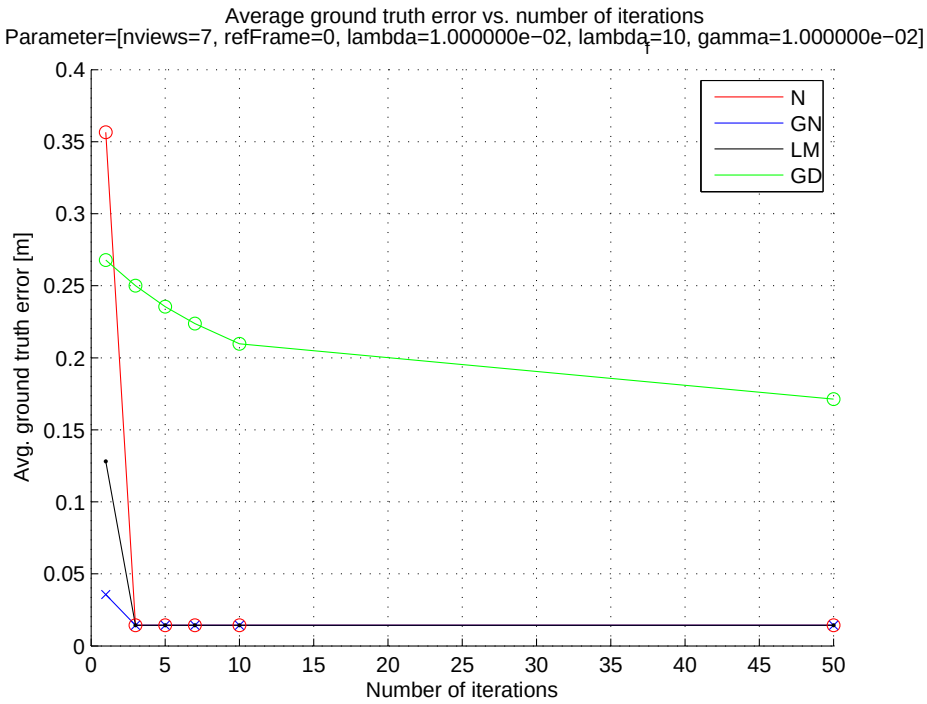


Figure C.14: Ground truth error

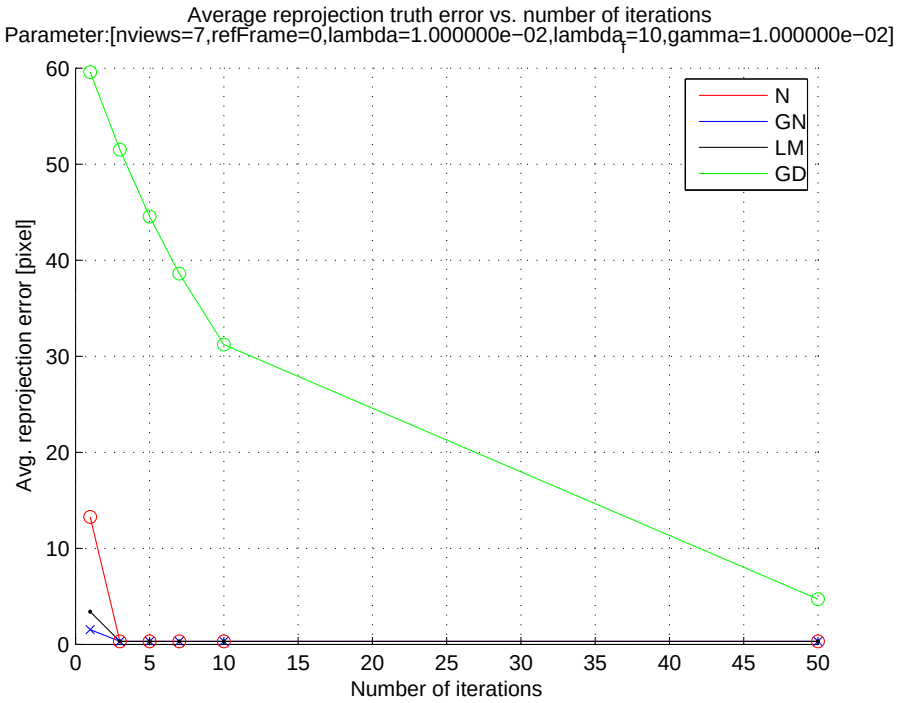


Figure C.15: Reprojection error

C.4 5 views

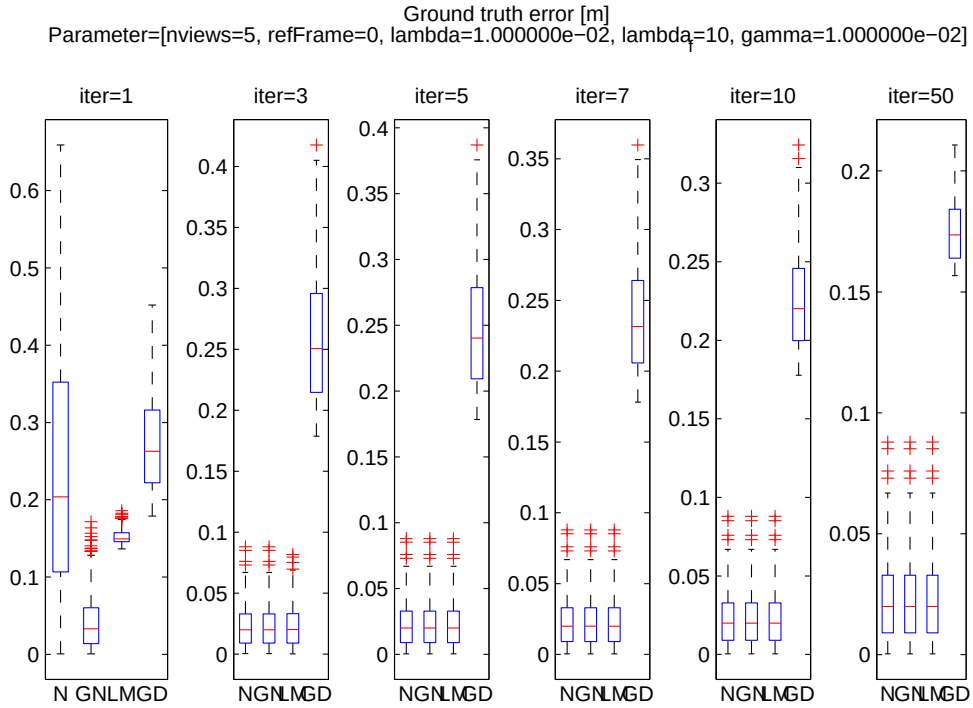


Figure C.16: Ground truth error

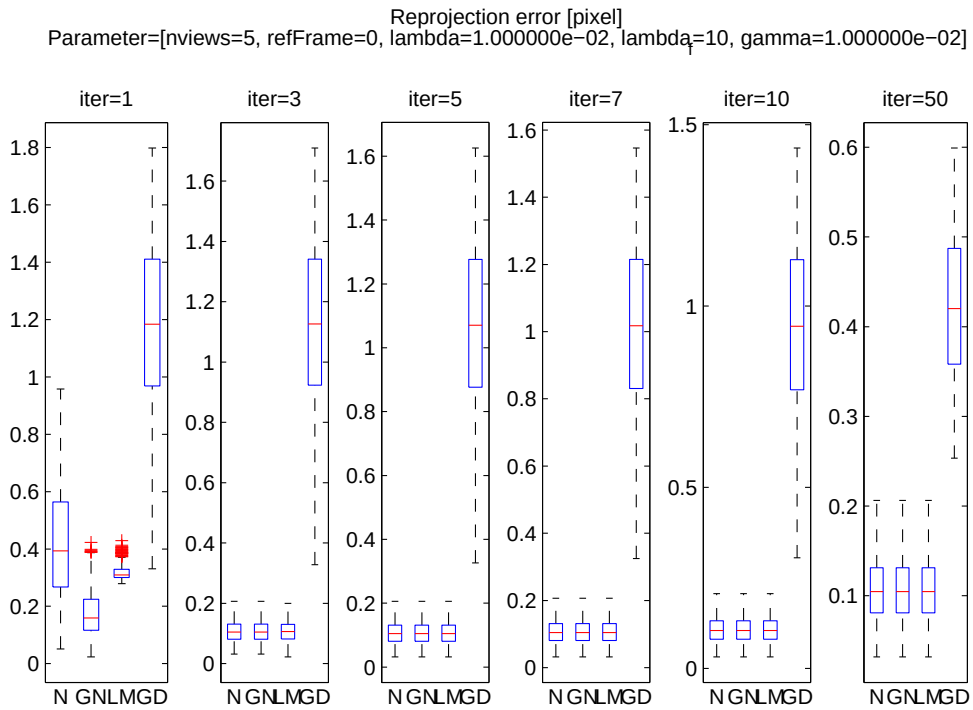


Figure C.17: Reprojection error

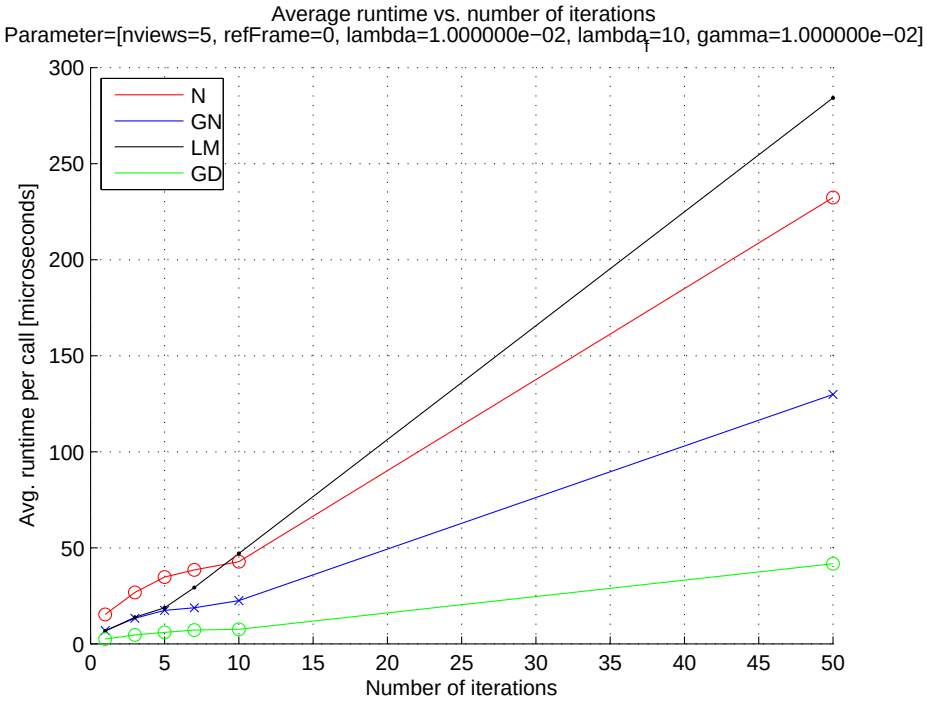


Figure C.18: Runtime

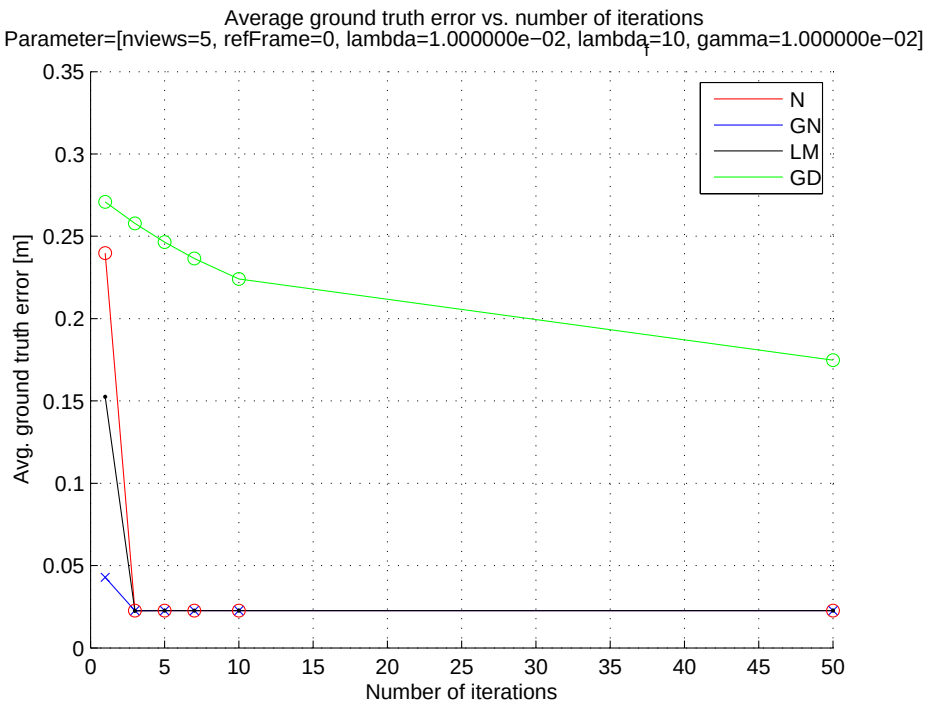


Figure C.19: Ground truth error

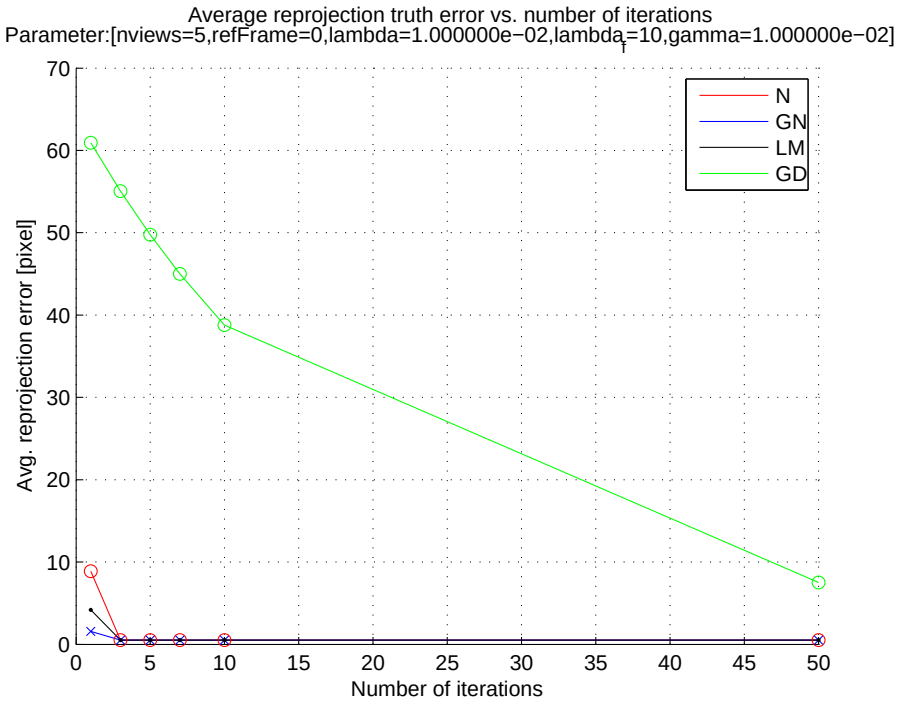


Figure C.20: Reprojection error

C.5 3 views

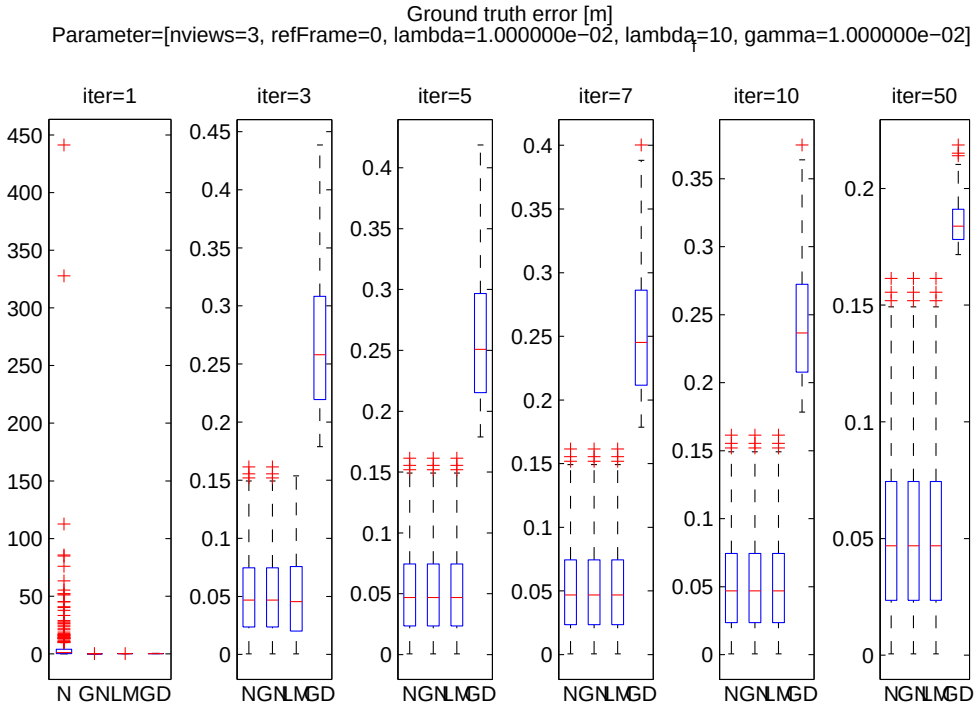


Figure C.21: Ground truth error

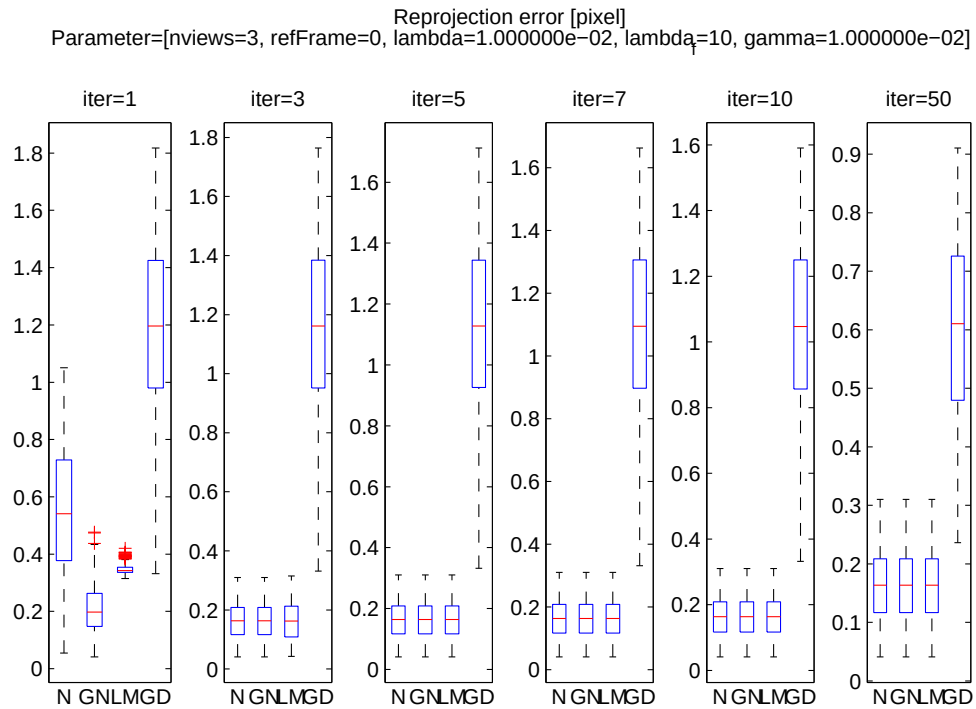


Figure C.22: Reprojection error

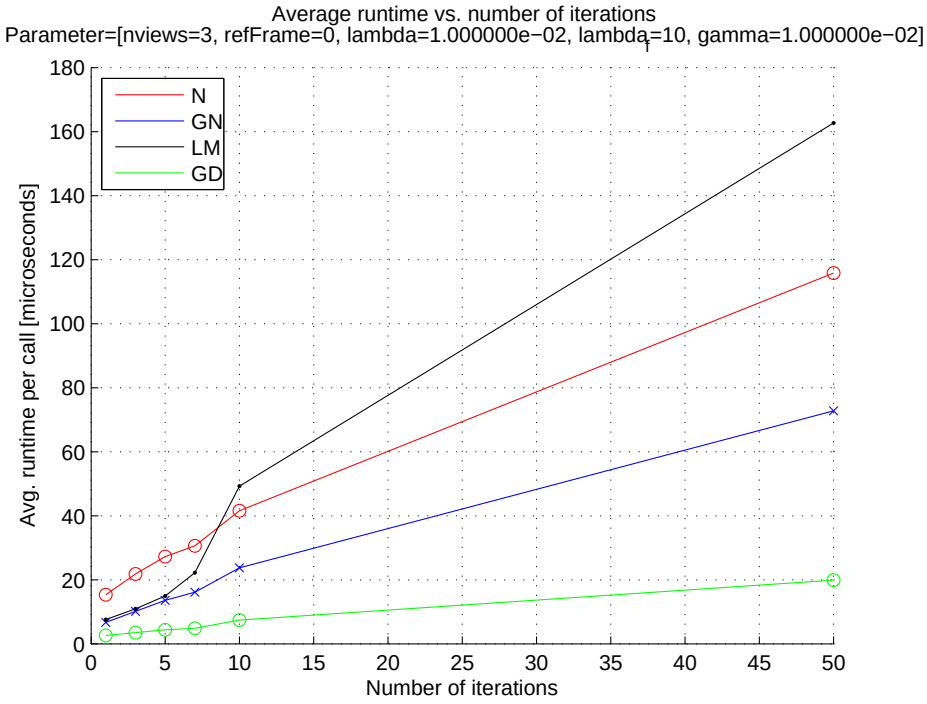


Figure C.23: Runtime

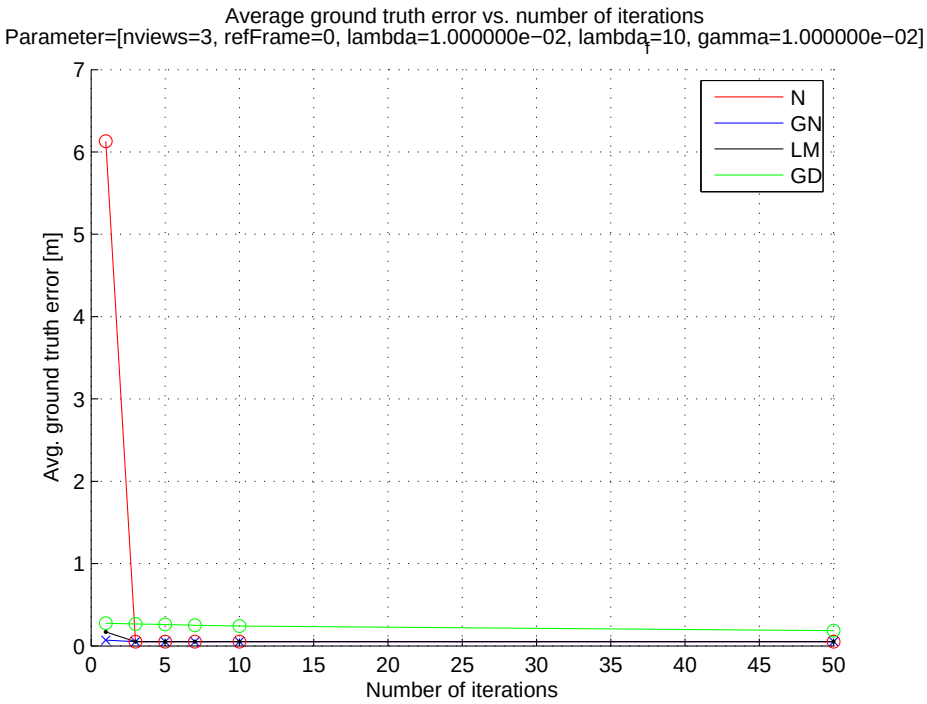


Figure C.24: Ground truth error

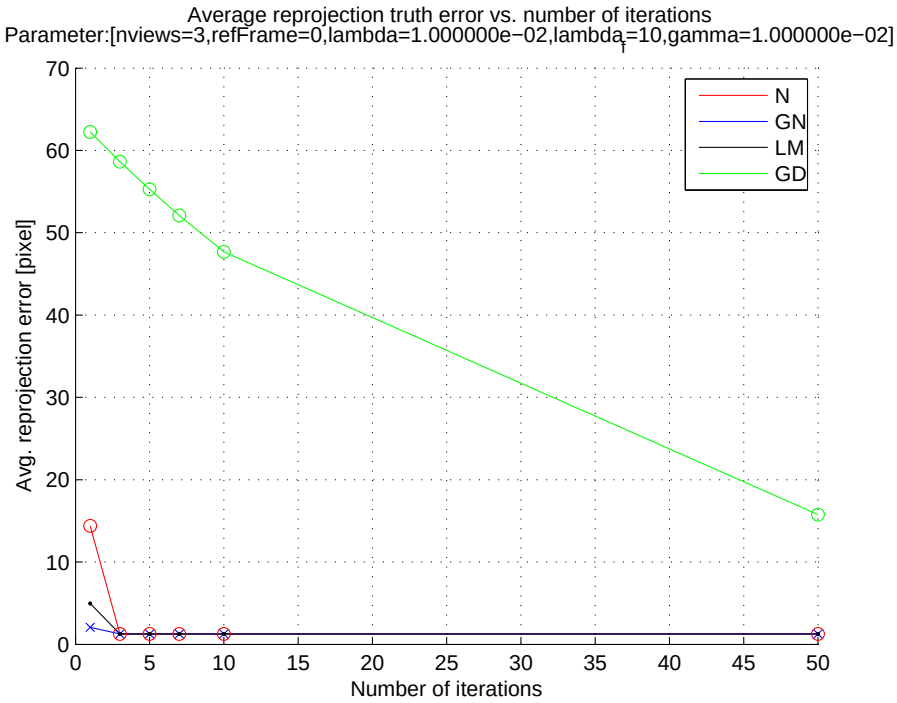


Figure C.25: Reprojection error

C.6 Ground truth error vs. number of views

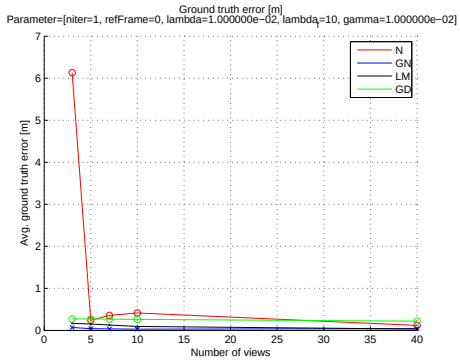


Figure C.26: 1 iteration

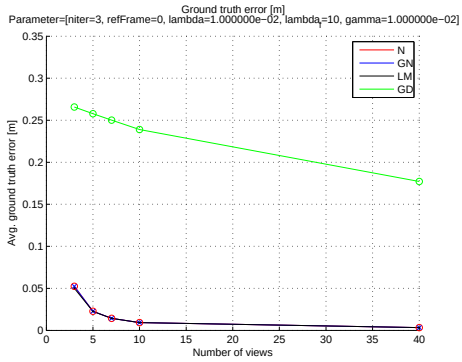


Figure C.27: 3 iterations

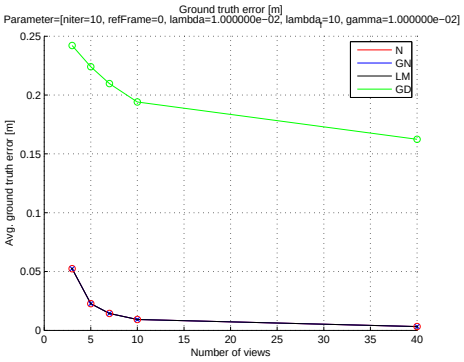


Figure C.28: 10 iterations

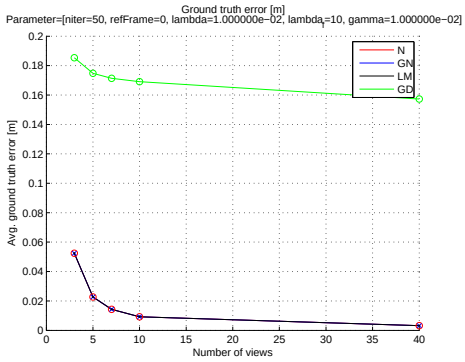
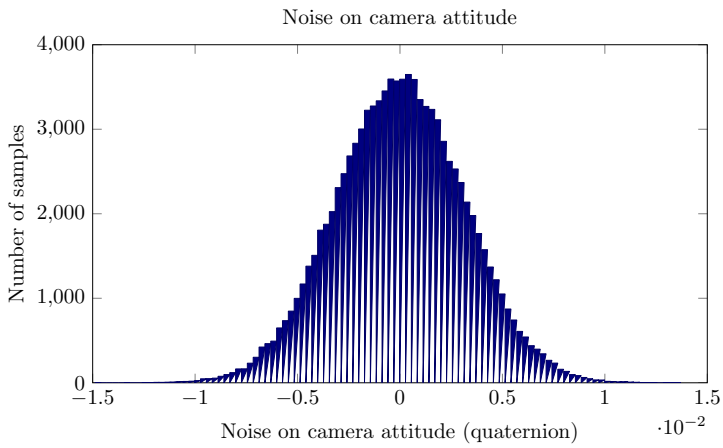


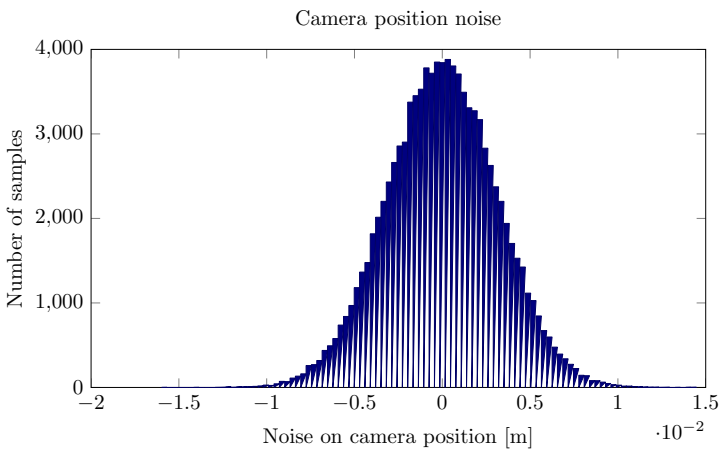
Figure C.29: 50 iterations

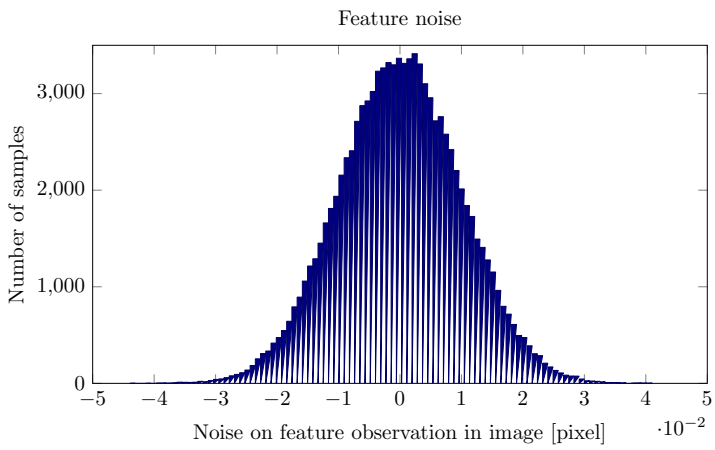
C.7 Simulation with pixel and pose noise



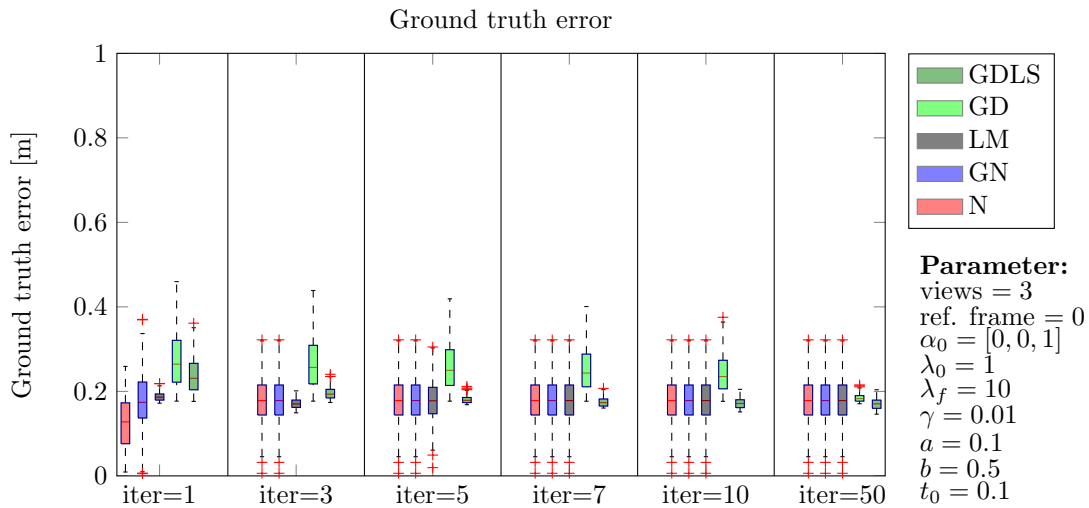
N	**niters=1**	**niters=3**	**niters=5**	**niters=7**	**niters=10**	**niters=50**
nviews=1	6.13	0.0524	0.0524	0.0524	0.0524	0.0524
nviews=3	0.24	0.0227	0.0227	0.0227	0.0227	0.0227
nviews=5	0.357	0.0143	0.0143	0.0143	0.0143	0.0143
nviews=7	0.415	0.00924	0.00925	0.00925	0.00925	0.00925
nviews=40	0.117	0.00322	0.00322	0.00322	0.00322	0.00322
GN	**niters=1**	**niters=3**	**niters=5**	**niters=7**	**niters=10**	**niters=50**
nviews=1	0.0686	0.0524	0.0524	0.0524	0.0524	0.0524
nviews=3	0.0428	0.0227	0.0227	0.0227	0.0227	0.0227
nviews=5	0.0356	0.0143	0.0143	0.0143	0.0143	0.0143
nviews=7	0.0304	0.00925	0.00925	0.00925	0.00925	0.00925
nviews=40	0.0401	0.00322	0.00322	0.00322	0.00322	0.00322
LM	**niters=1**	**niters=3**	**niters=5**	**niters=7**	**niters=10**	**niters=50**
nviews=1	0.168	0.0502	0.0524	0.0524	0.0524	0.0524
nviews=3	0.152	0.0225	0.0227	0.0227	0.0227	0.0227
nviews=5	0.128	0.0143	0.0143	0.0143	0.0143	0.0143
nviews=7	0.0942	0.00925	0.00925	0.00925	0.00925	0.00925
nviews=40	0.0335	0.00322	0.00322	0.00322	0.00322	0.00322
GD	**niters=1**	**niters=3**	**niters=5**	**niters=7**	**niters=10**	**niters=50**
nviews=1	0.274	0.266	0.258	0.251	0.242	0.185
nviews=3	0.271	0.258	0.246	0.237	0.224	0.175
nviews=5	0.268	0.25	0.236	0.224	0.21	0.171
nviews=7	0.263	0.239	0.221	0.208	0.194	0.169
nviews=40	0.22	0.177	0.167	0.164	0.162	0.157

Table C.1: Ground truth error vs. number of iterations and number of view (N=Newton, GN=Gauss-Newton, LM=Levenberg-Marquardt, GD=Gradient Descent)

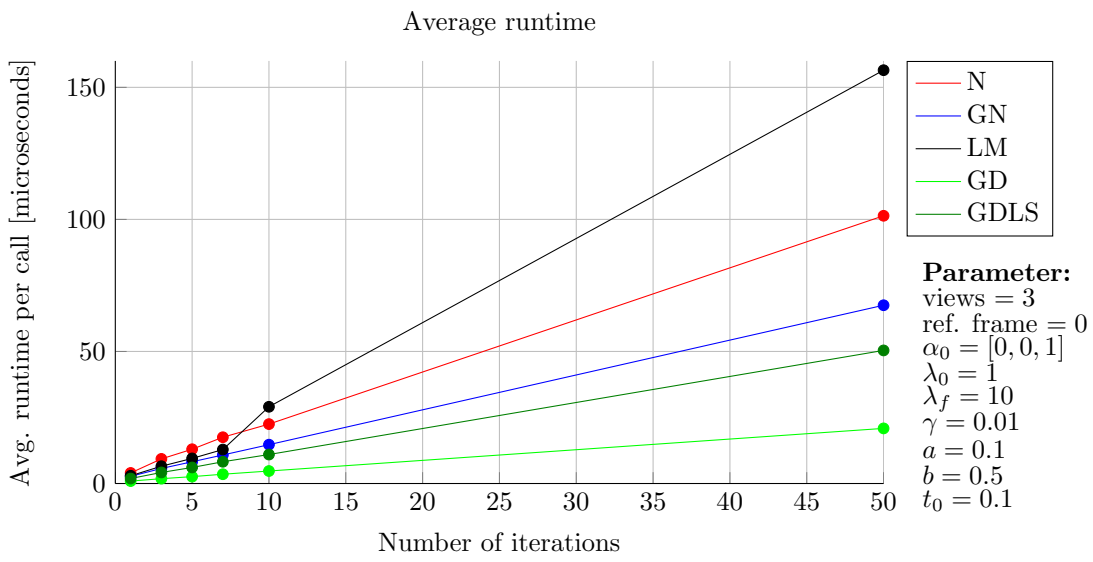


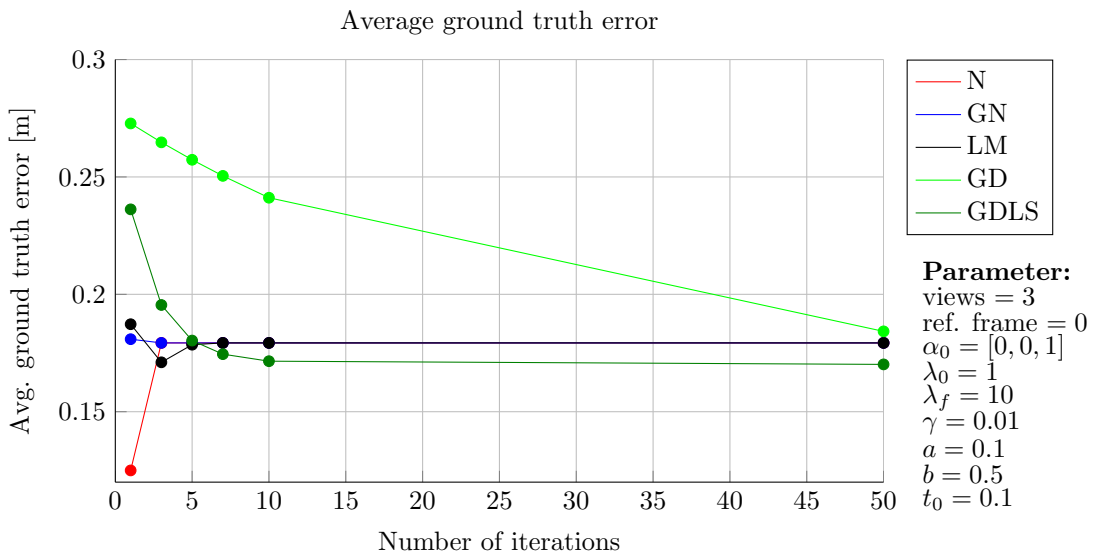

Feature noise
Number of samples
3,000
2,000
1,000
0
−5
−4
−3
−2
−1
0
1
2
3
4
5
Noise on feature observation in image [pixel]
$\cdot 10^{-2}$

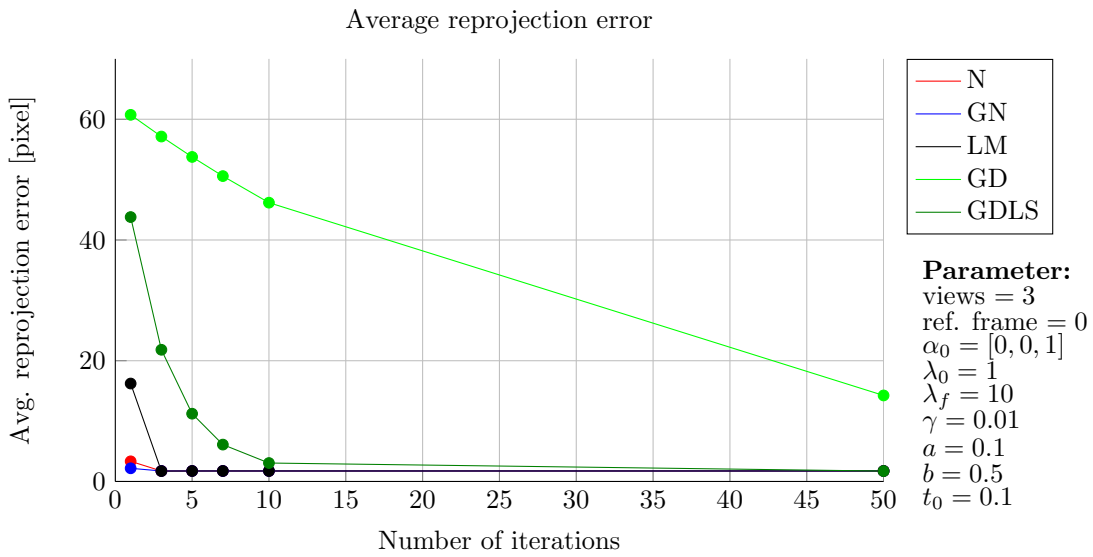
C.7.1 3 views


Ground truth error
Ground truth error [m]
1
0.8
0.6
0.4
0.2
0
iter=1
iter=3
iter=5
iter=7
iter=10
iter=50
GDLS
GD
LM
GN
N
Parameter:
views = 3
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

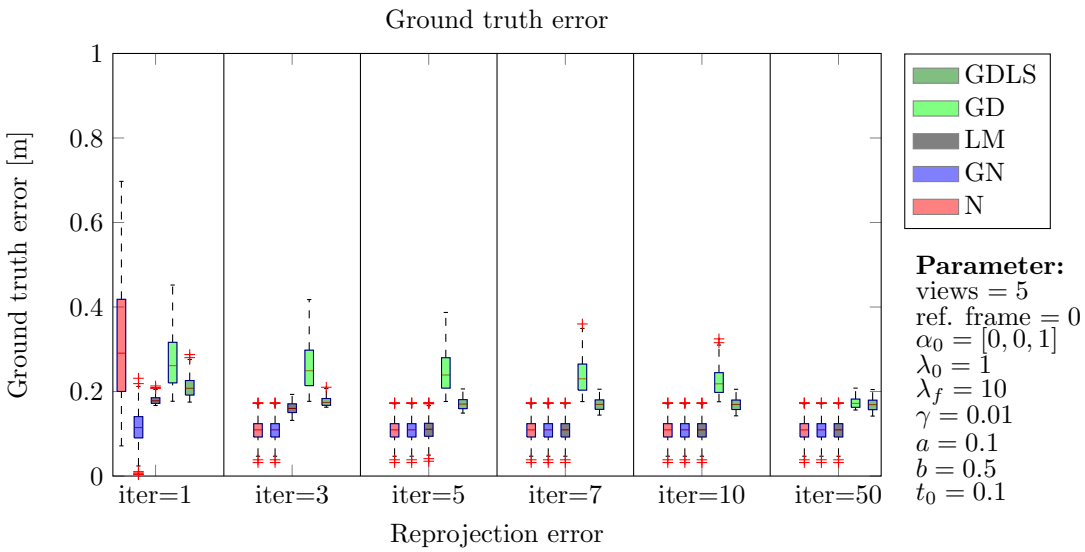

Reprojection error
Reprojection error [pixel]
6
4
2
0
iter=1
iter=3
iter=5
iter=7
iter=10
iter=50
GDLS
GD
LM
GN
N
Parameter:
views = 3
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

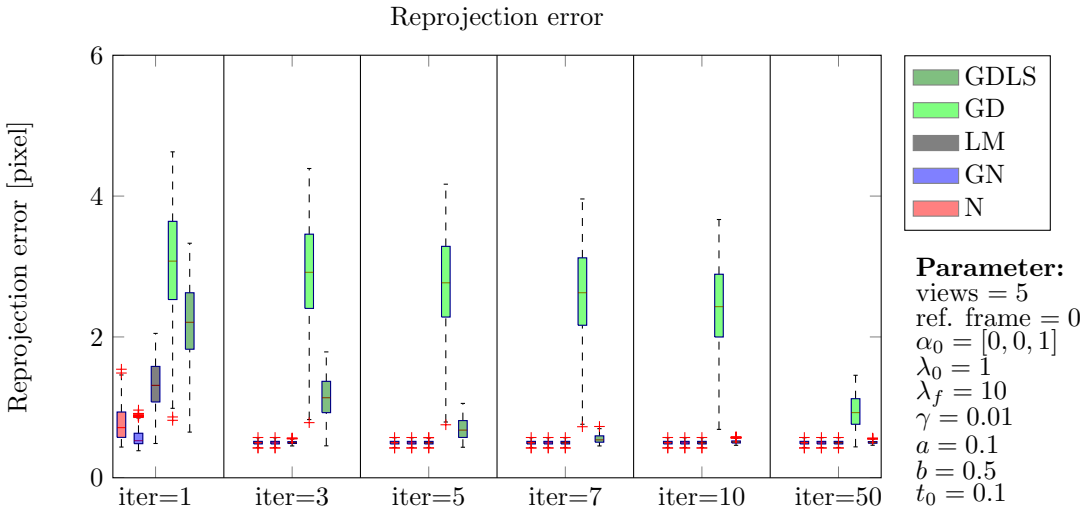

Average runtime
Avg. runtime per call [microseconds]
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 3
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

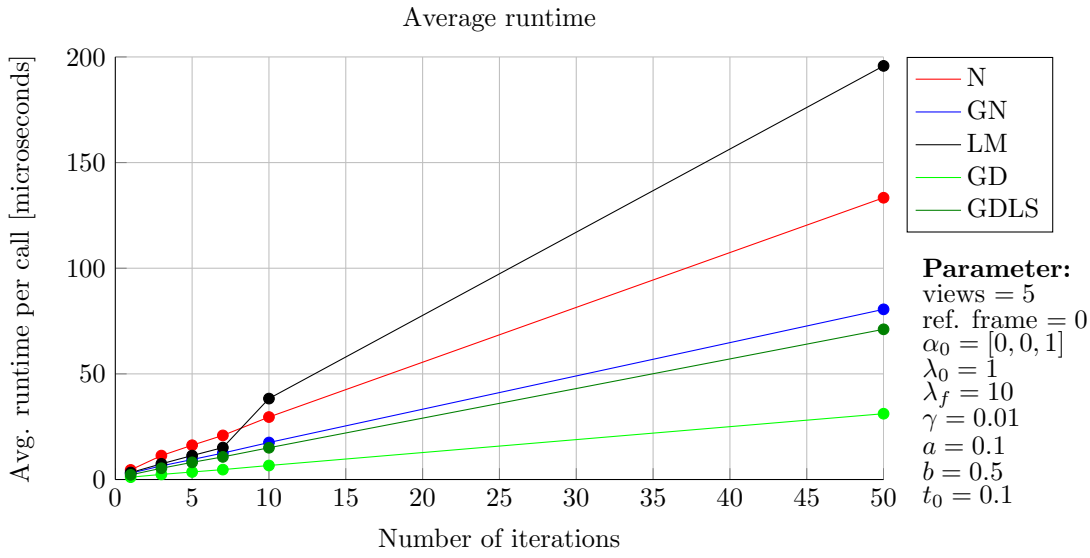

Average ground truth error
Avg. ground truth error [m]
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 3
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

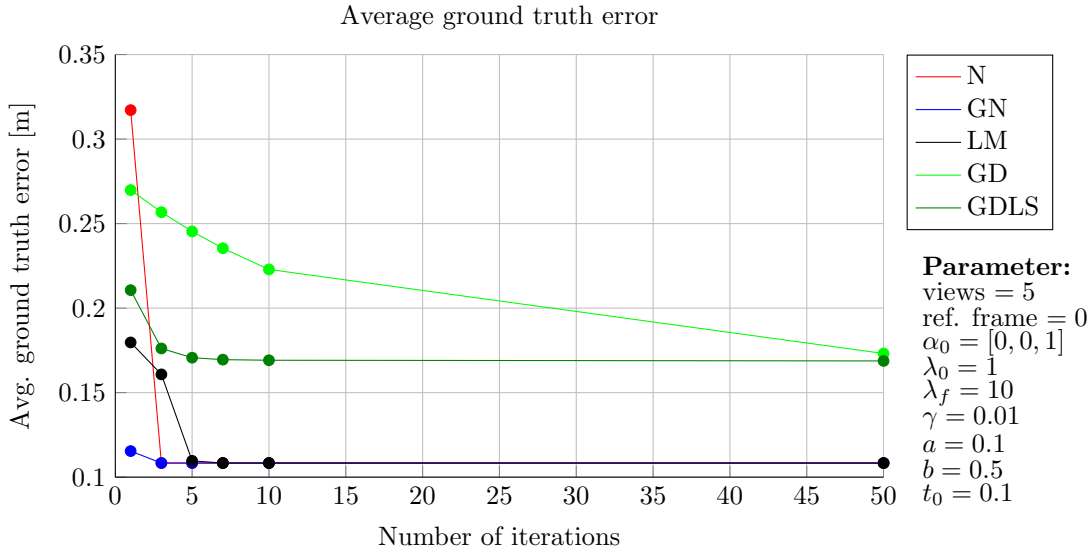

Average reprojection error
Avg. reprojection error [pixel]
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 3
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

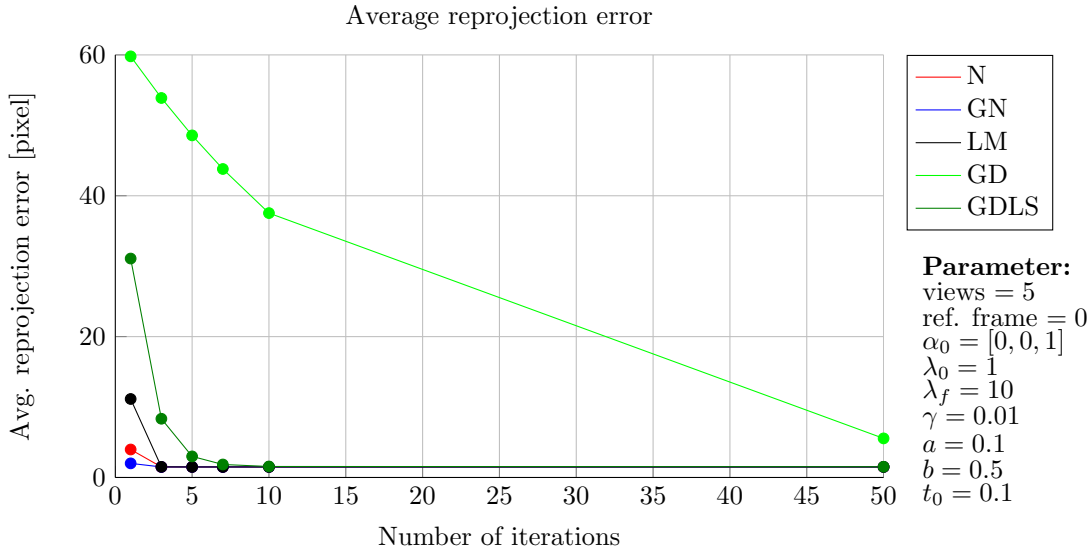
C.7.2 5 views


Ground truth error
Ground truth error [m]
1
0.8
0.6
0.4
0.2
0
iter=1
iter=3
iter=5
iter=7
iter=10
iter=50
GDLS
GD
LM
GN
N
Parameter:
views = 5
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

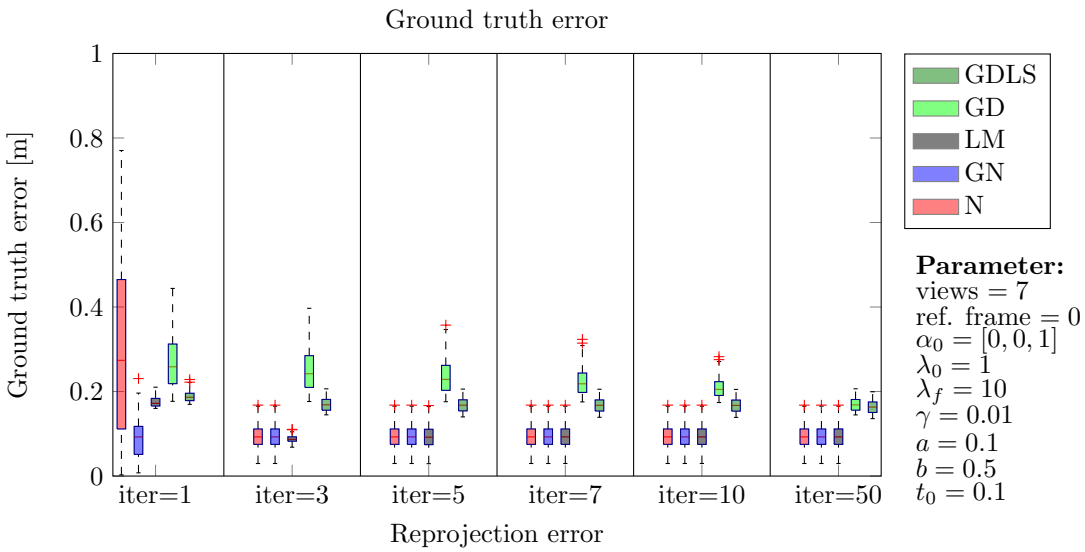

Reprojection error
Reprojection error [pixel]
6
4
2
0
iter=1
iter=3
iter=5
iter=7
iter=10
iter=50
GDLS
GD
LM
GN
N
Parameter:
views = 5
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$


Average runtime
Avg. runtime per call [microseconds]
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 5
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

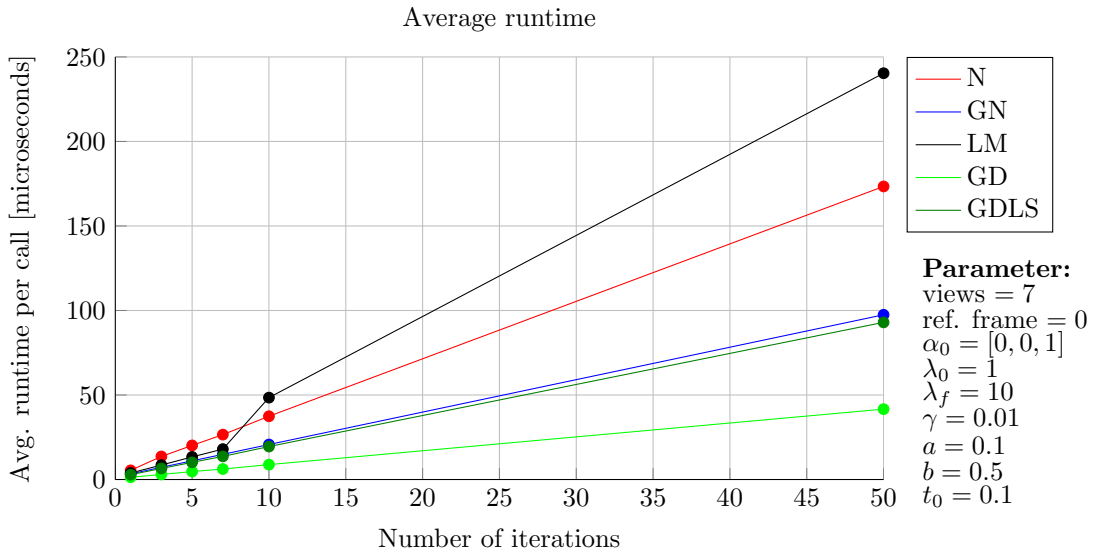

Average ground truth error
Avg. ground truth error [m]
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 5
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

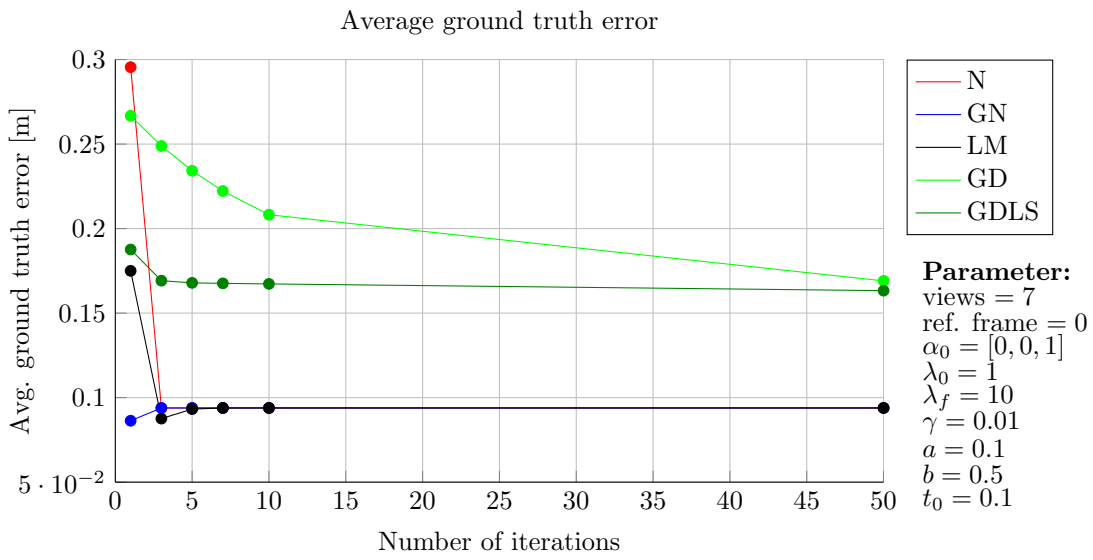

Average reprojection error
Avg. reprojection error [pixel]
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 5
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

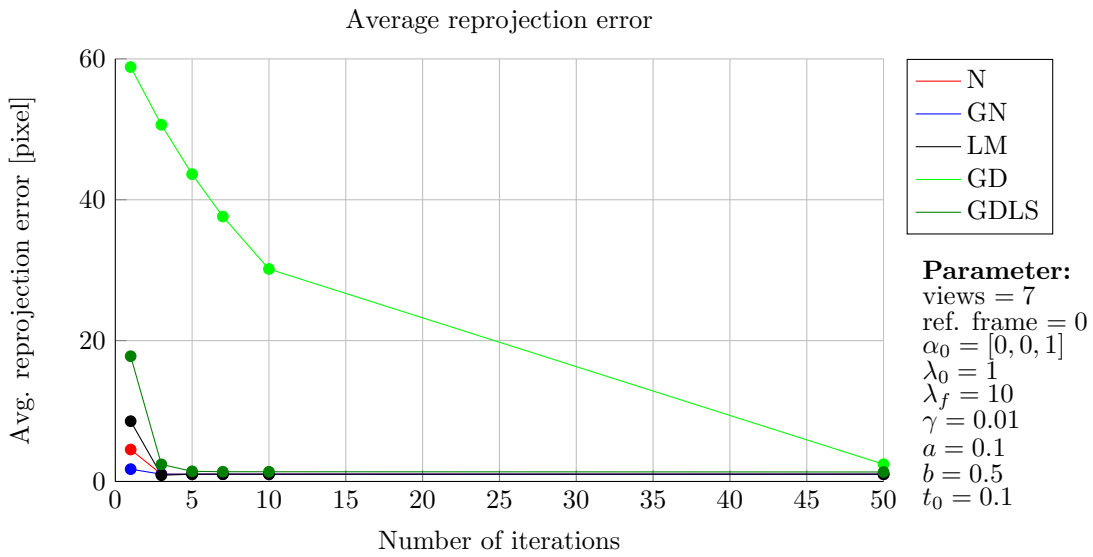
C.7.3 7 views


Ground truth error
Ground truth error [m]
1
0.8
0.6
0.4
0.2
0
iter=1
iter=3
iter=5
iter=7
iter=10
iter=50
GDLS
GD
LM
GN
N
Parameter:
views = 7
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

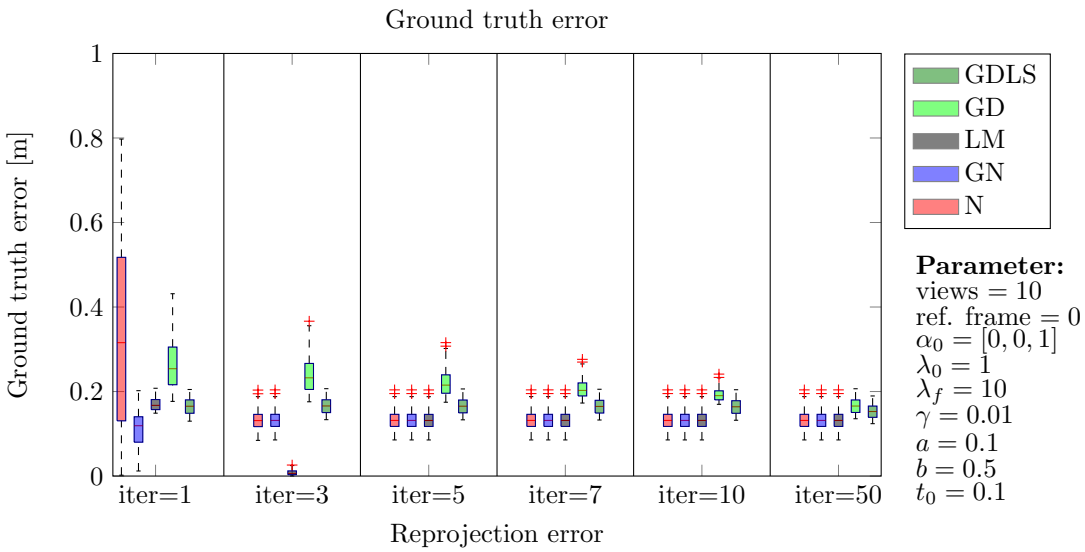

Reprojection error
Reprojection error [pixel]
6
4
2
0
iter=1
iter=3
iter=5
iter=7
iter=10
iter=50
GDLS
GD
LM
GN
N
Parameter:
views = 7
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

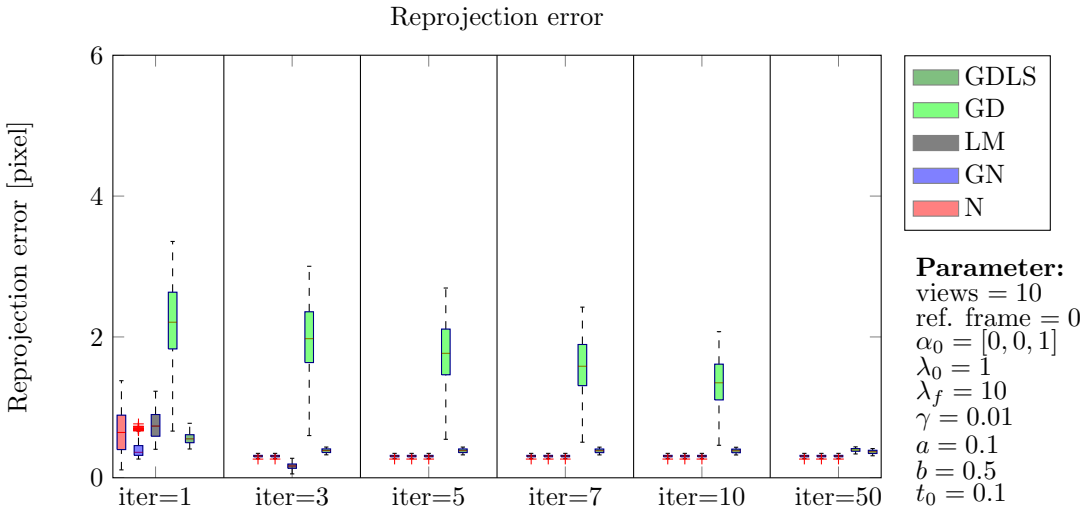

Average runtime
Avg. runtime per call [microseconds]
250
200
150
100
50
0
0 5 10 15 20 25 30 35 40 45 50
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 7
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

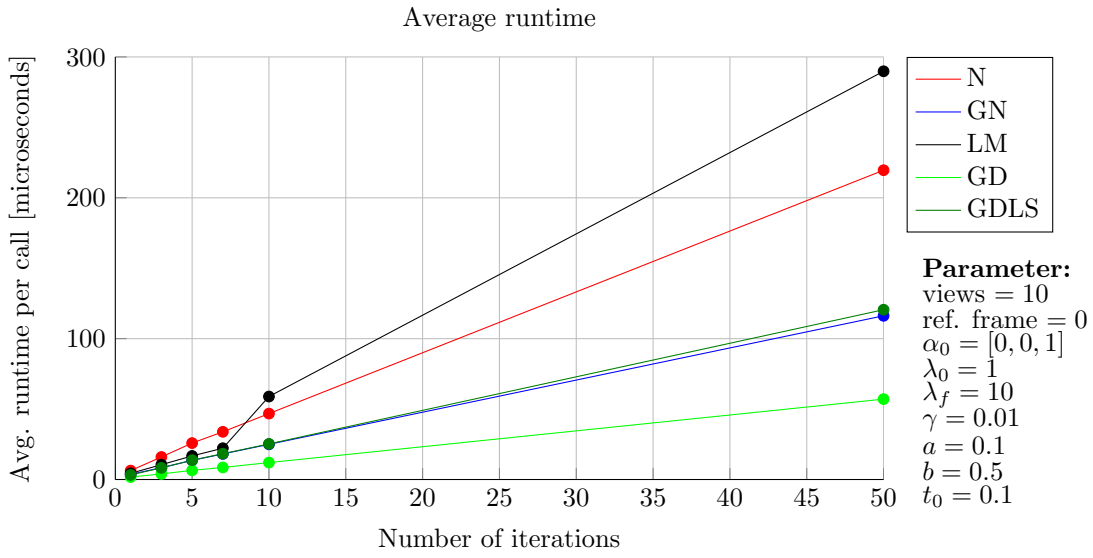

Average ground truth error
Avg. ground truth error [m]
0.3
0.25
0.2
0.15
0.1
$5 \cdot 10^{-2}$
0 5 10 15 20 25 30 35 40 45 50
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 7
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

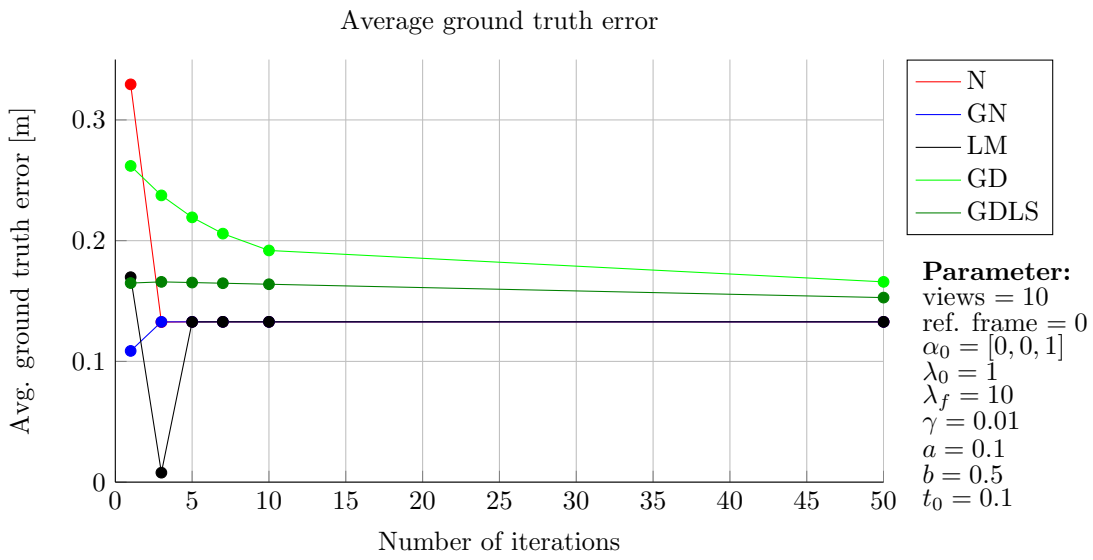

Average reprojection error
Avg. reprojection error [pixel]
60
40
20
0
0 5 10 15 20 25 30 35 40 45 50
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 7
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

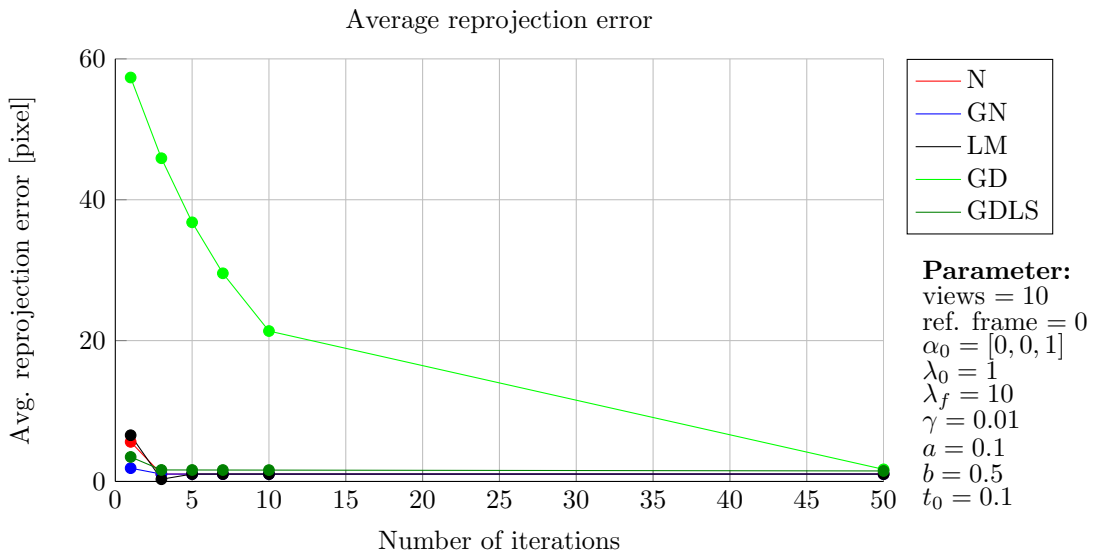
C.7.4 10 views


Ground truth error
Ground truth error [m]
1
0.8
0.6
0.4
0.2
0
iter=1
iter=3
iter=5
iter=7
iter=10
iter=50
GDLS
GD
LM
GN
N
Parameter:
views = 10
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

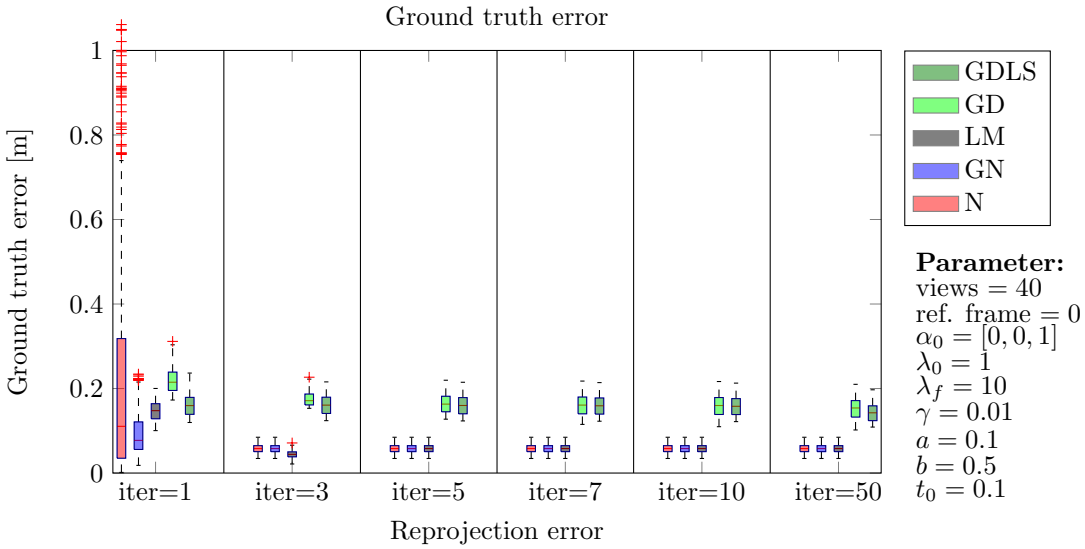

Reprojection error
Reprojection error [pixel]
6
4
2
0
iter=1
iter=3
iter=5
iter=7
iter=10
iter=50
GDLS
GD
LM
GN
N
Parameter:
views = 10
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

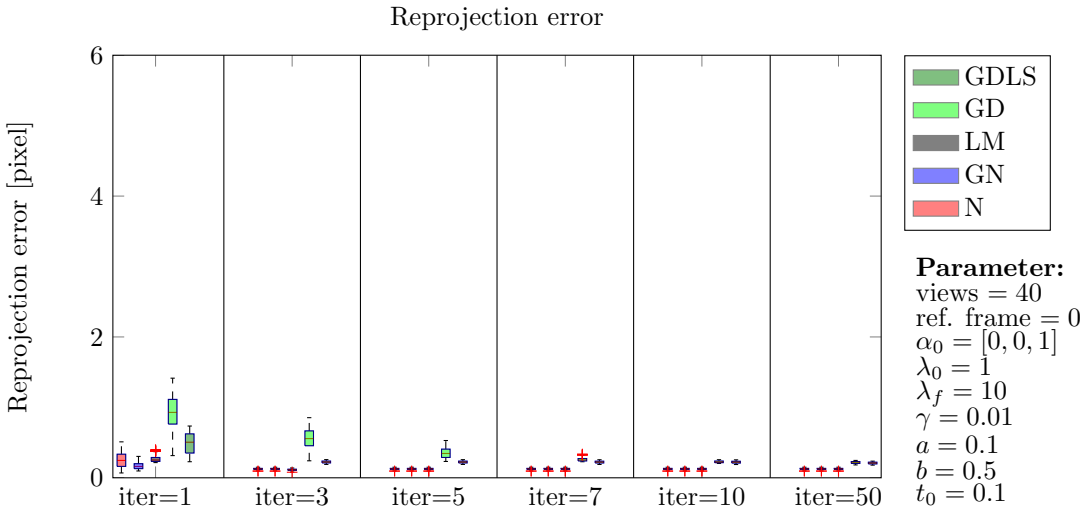

Average runtime
Avg. runtime per call [microseconds]
300
200
100
0
0 5 10 15 20 25 30 35 40 45 50
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 10
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

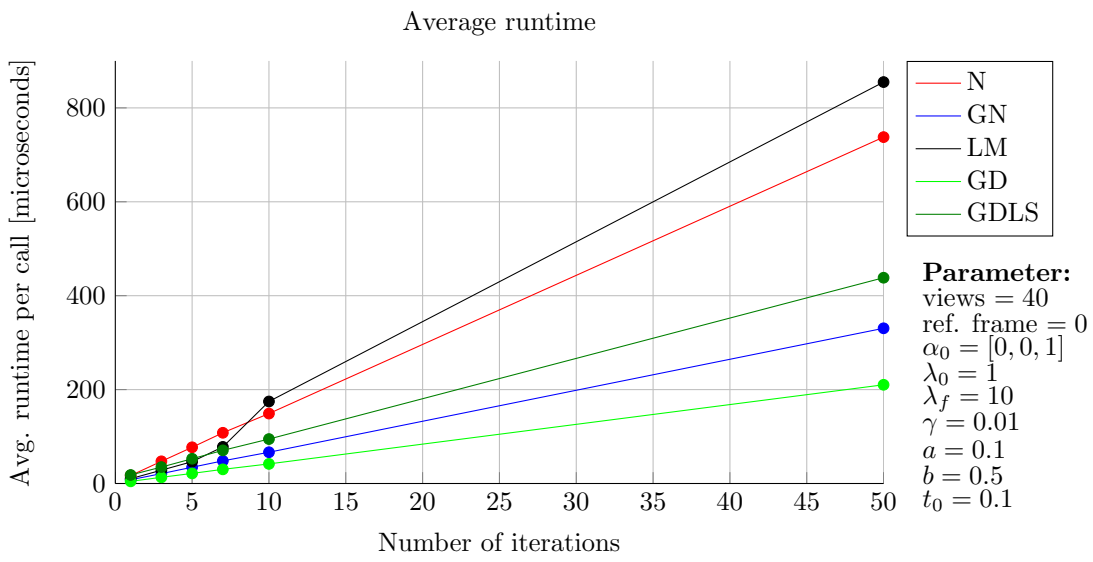

Average ground truth error
Avg. ground truth error [m]
0.3
0.2
0.1
0
0 5 10 15 20 25 30 35 40 45 50
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 10
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

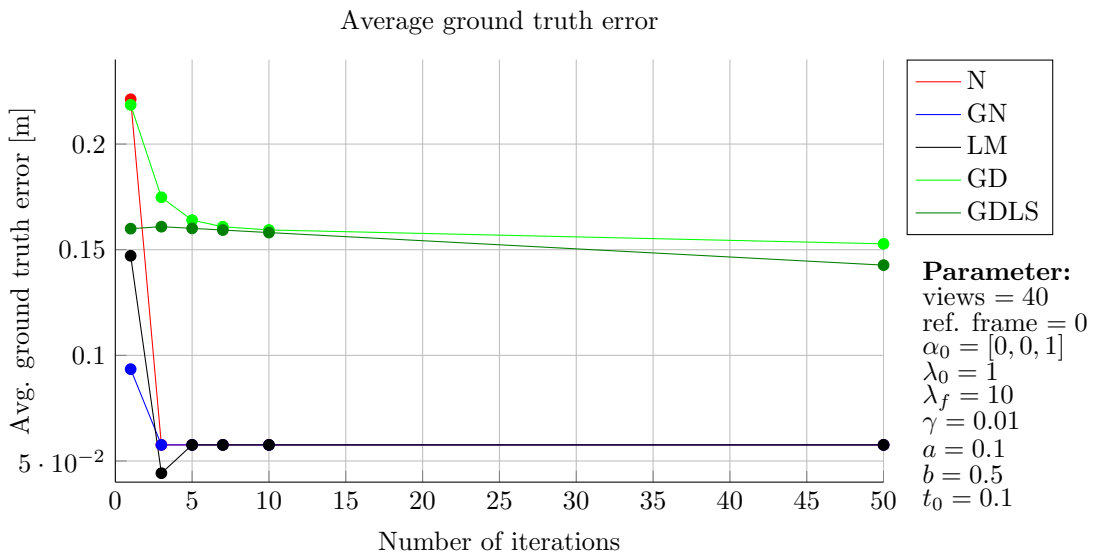

Average reprojection error
Avg. reprojection error [pixel]
60
40
20
0
0 5 10 15 20 25 30 35 40 45 50
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 10
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

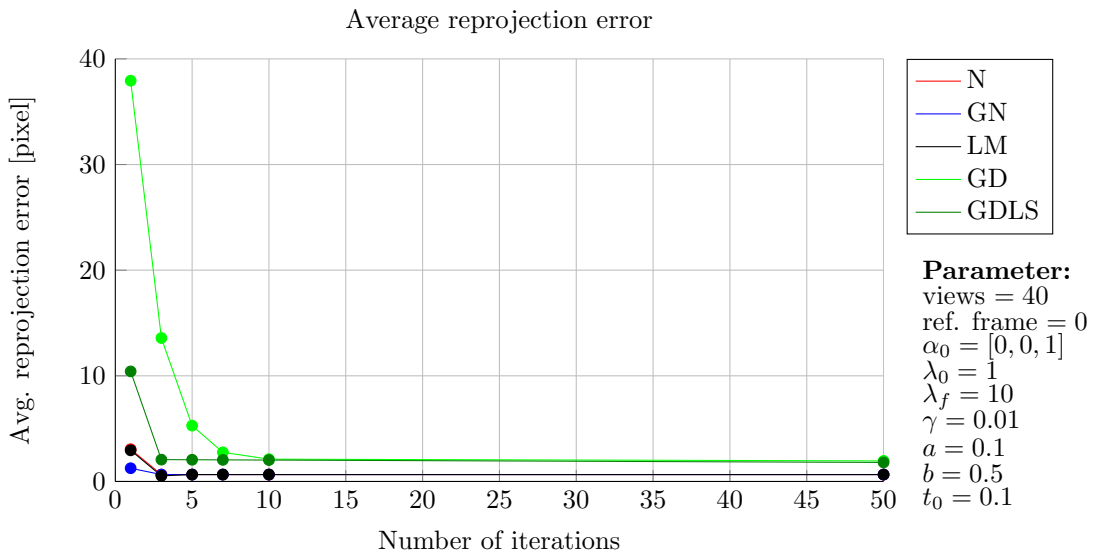
C.7.5 40 views


Ground truth error
Ground truth error [m]
1
0.8
0.6
0.4
0.2
0
iter=1
iter=3
iter=5
iter=7
iter=10
iter=50
GDLS
GD
LM
GN
N
Parameter:
views = 40
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$


Reprojection error
Reprojection error [pixel]
6
4
2
0
iter=1
iter=3
iter=5
iter=7
iter=10
iter=50
GDLS
GD
LM
GN
N
Parameter:
views = 40
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$


Average runtime
Avg. runtime per call [microseconds]
0
200
400
600
800
0 5 10 15 20 25 30 35 40 45 50
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 40
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$


Average ground truth error
Avg. ground truth error [m]
$5 \cdot 10^{-2}$
0.1
0.15
0.2
0 5 10 15 20 25 30 35 40 45 50
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 40
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$


Average reprojection error
Avg. reprojection error [pixel]
0
10
20
30
40
0 5 10 15 20 25 30 35 40 45 50
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 40
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

C.8 Ground truth error vs. number of views (noise)

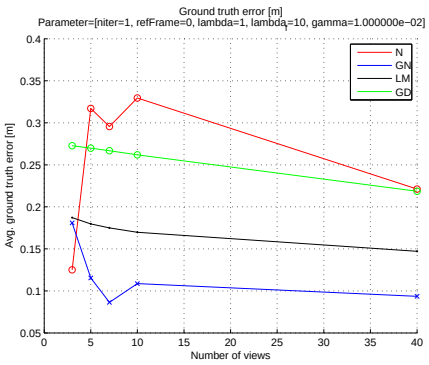


Figure C.30: 1 iteration

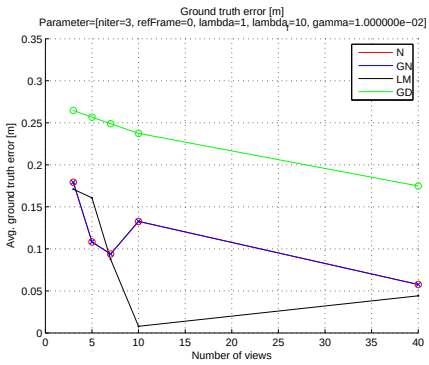


Figure C.31: 3 iterations

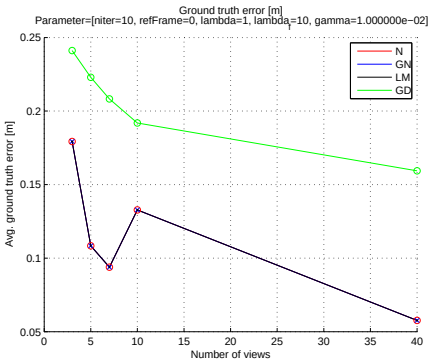


Figure C.32: 10 iterations

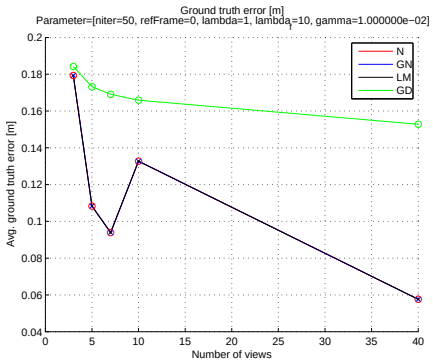
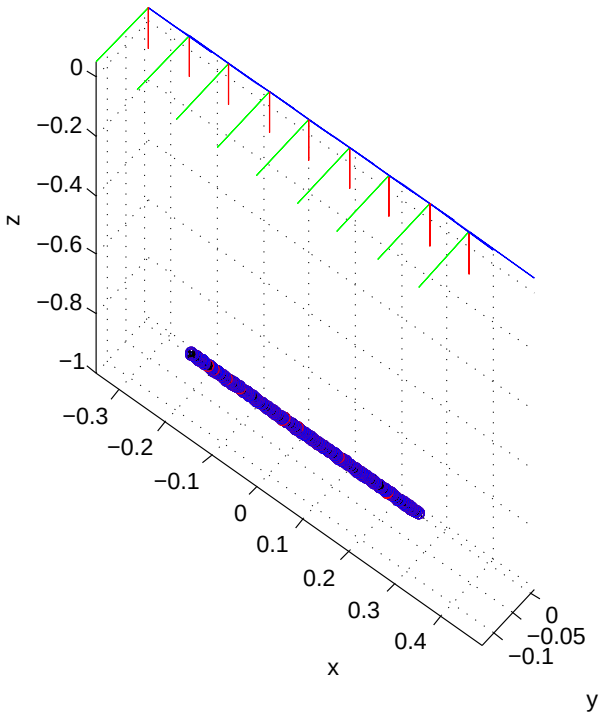
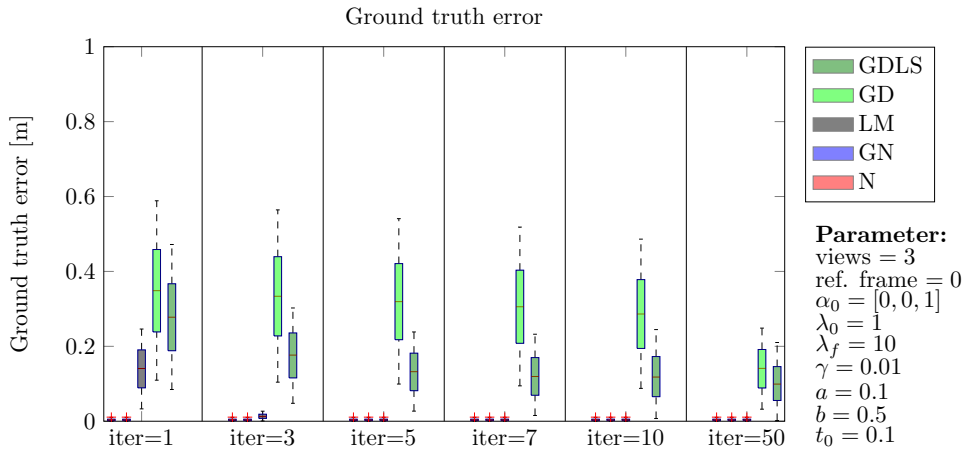


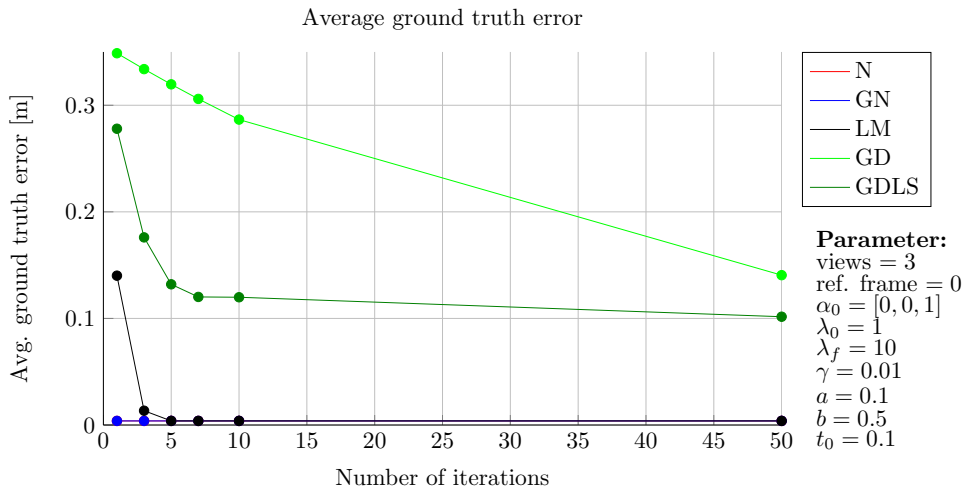
Figure C.33: 50 iterations

C.9 Simulation: Constant distance, different number of camera poses in between

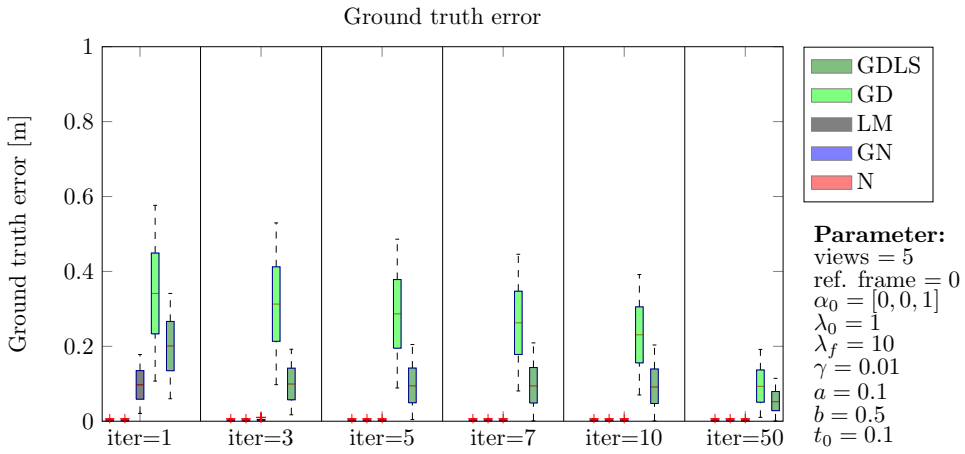


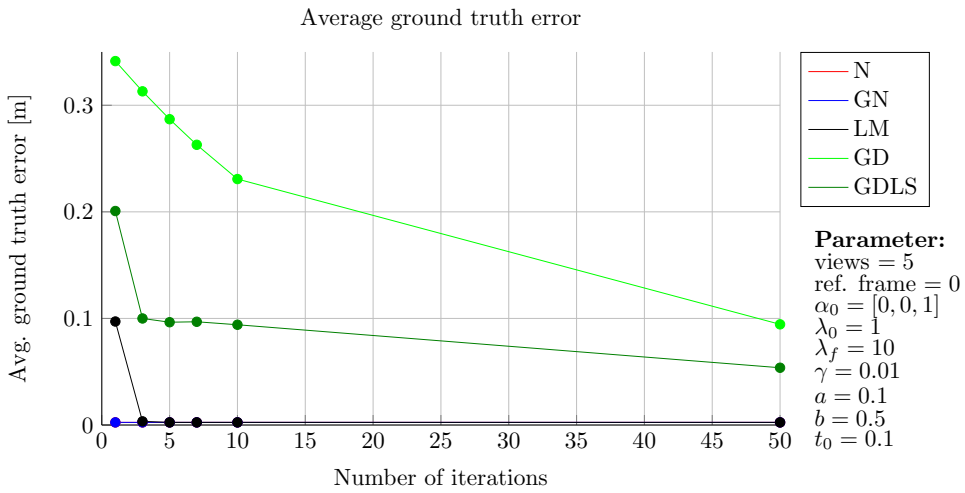
C.9.1 3 views


Ground truth error
Ground truth error [m]
1
0.8
0.6
0.4
0.2
0
iter=1
iter=3
iter=5
iter=7
iter=10
iter=50
GDLS
GD
LM
GN
N
Parameter:
views = 3
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

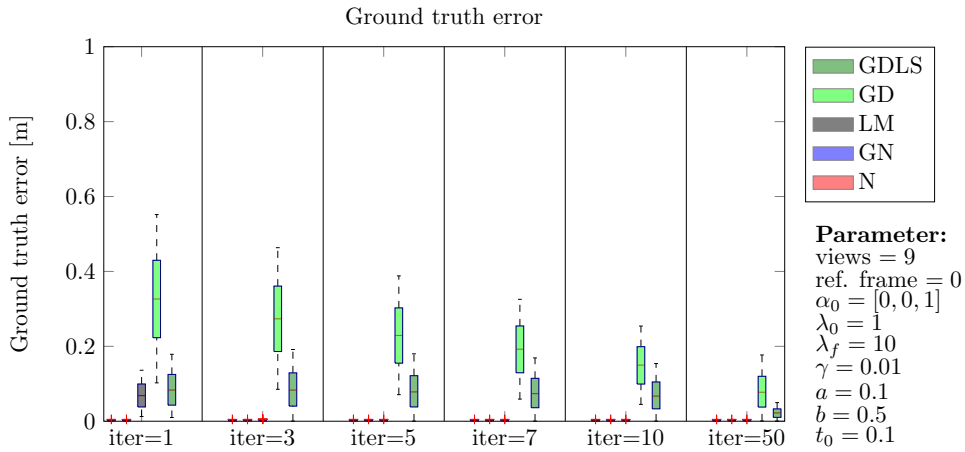

Average ground truth error
Avg. ground truth error [m]
0.3
0.2
0.1
0
0
5
10
15
20
25
30
35
40
45
50
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 3
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

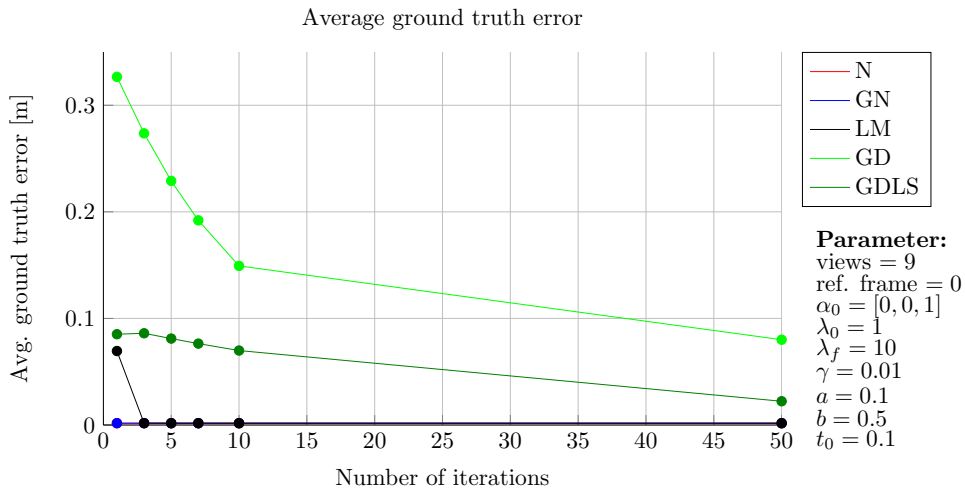
C.9.2 5 views


Ground truth error
Ground truth error [m]
1
0.8
0.6
0.4
0.2
0
iter=1
iter=3
iter=5
iter=7
iter=10
iter=50
GDLS
GD
LM
GN
N
Parameter:
views = 5
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

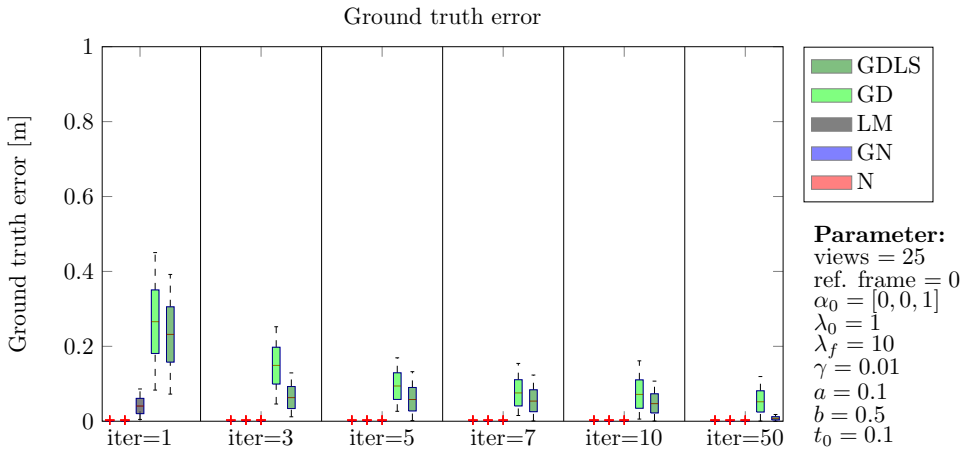

Average ground truth error
Avg. ground truth error [m]
0.3
0.2
0.1
0
0
5
10
15
20
25
30
35
40
45
50
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 5
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

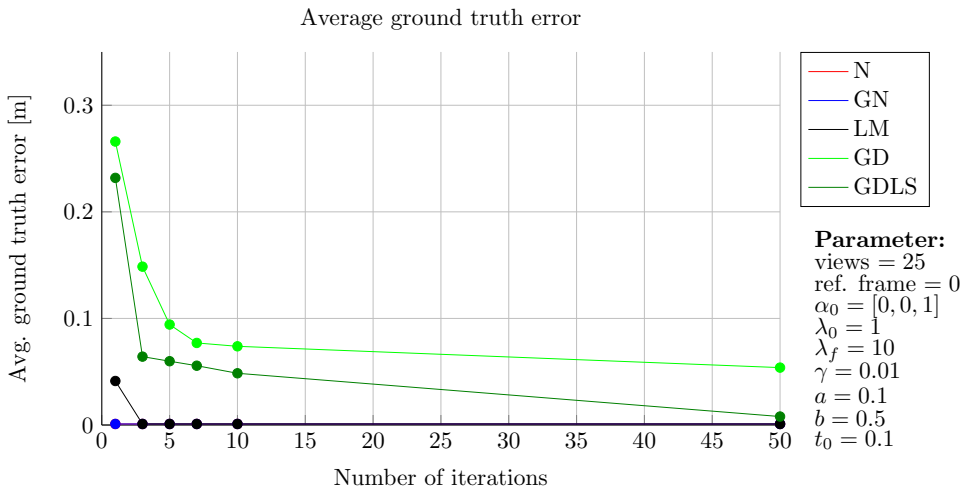
C.9.3 9 views


Ground truth error
Ground truth error [m]
1
0.8
0.6
0.4
0.2
0
iter=1
iter=3
iter=5
iter=7
iter=10
iter=50
GDLS
GD
LM
GN
N
Parameter:
views = 9
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

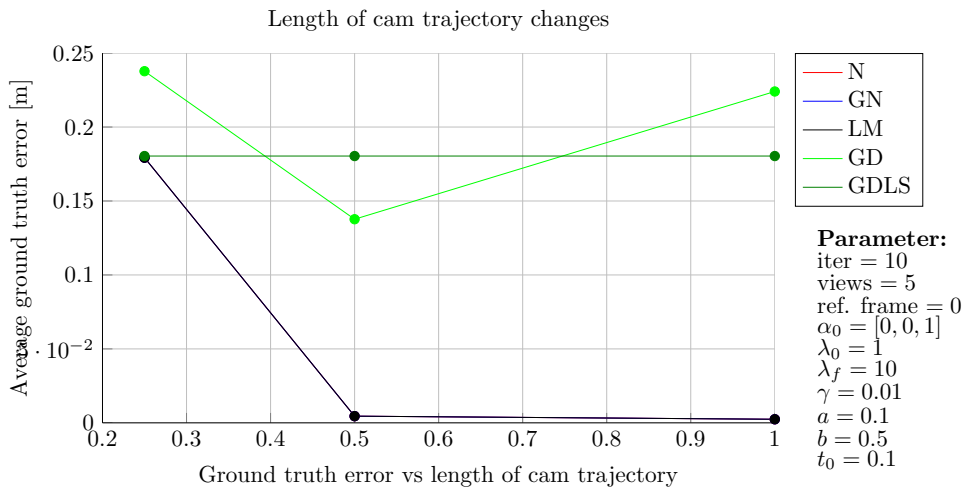

Average ground truth error
Avg. ground truth error [m]
0.3
0.2
0.1
0
0
5
10
15
20
25
30
35
40
45
50
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 9
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

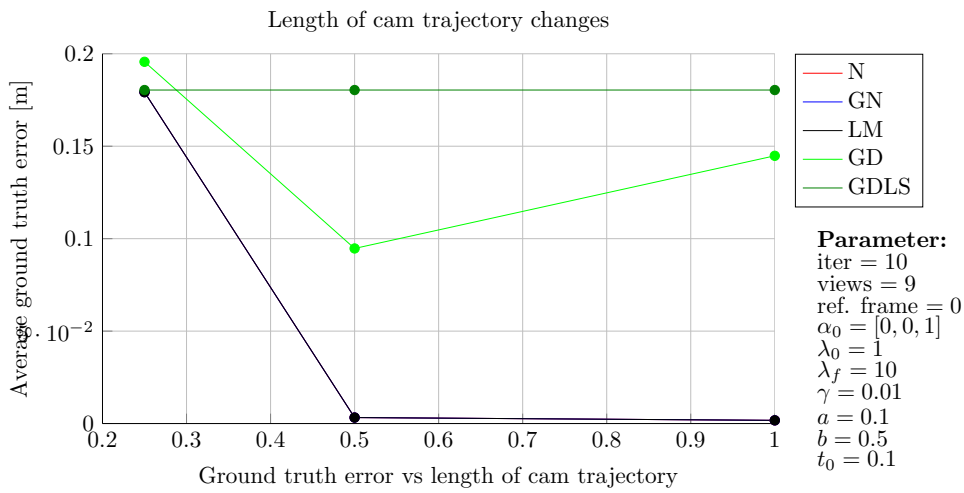
C.9.4 25 views

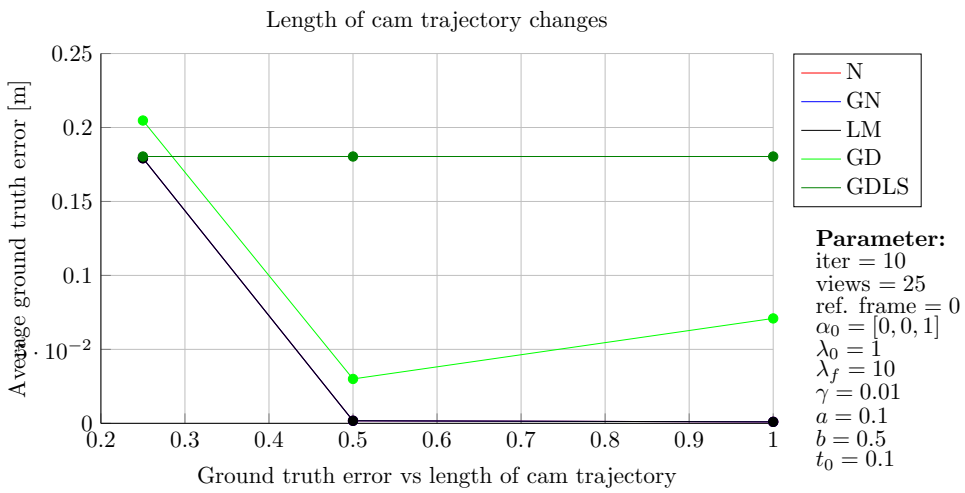

Ground truth error
Ground truth error [m]
1
0.8
0.6
0.4
0.2
0
iter=1
iter=3
iter=5
iter=7
iter=10
iter=50
GDLS
GD
LM
GN
N
Parameter:
views = 25
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$


Average ground truth error
Avg. ground truth error [m]
0.3
0.2
0.1
0
0
5
10
15
20
25
30
35
40
45
50
Number of iterations
N
GN
LM
GD
GDLS
Parameter:
views = 25
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

C.10 Constant number of camera frames, length of camera trajectory changes






Length of cam trajectory changes
Average ground truth error [m]
0.25
0.2
0.15
0.1
$\cdot 10^{-2}$
0
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Ground truth error vs length of cam trajectory
N
GN
LM
GD
GDLS
Parameter:
iter = 10
views = 25
ref. frame = 0
$\alpha_0 = [0, 0, 1]$
$\lambda_0 = 1$
$\lambda_f = 10$
$\gamma = 0.01$
$a = 0.1$
$b = 0.5$
$t_0 = 0.1$

Appendix D

Pinhole model calibration

Pinhole model calibrated using ROS/OpenCV [1]:

```
[image]

width
752

height
480

[narrow_stereo]

camera matrix
395.848120 0.000000 358.533742
0.000000 397.135809 228.498856
0.000000 0.000000 1.000000

distortion
-0.287020 0.064190 -0.001523 -0.000553 0.000000

rectification
1.000000 0.000000 0.000000
0.000000 1.000000 0.000000
0.000000 0.000000 1.000000

projection
277.782867 0.000000 357.530939 0.000000
0.000000 347.031342 223.195059 0.000000
0.000000 0.000000 1.000000 0.000000
```

[1] http://wiki.ros.org/camera_calibration

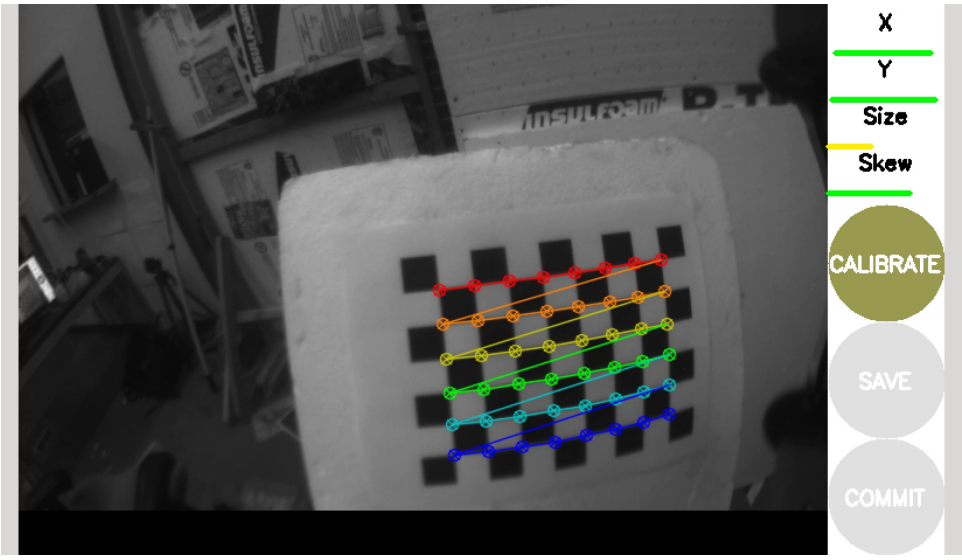


Figure D.1: Calibration sequence: Pinhole camera model

Appendix E

ATAN model calibration

ATAN model calibrated using `ethzasl_ptam`[1]:

```
params = [0.459985 0.718351 0.480415 0.484343 0.940115]
```

Figure E.1: Calibration sequence: ATAN camera model

[1] `https://github.com/ethz-asl/ethzasl_ptam`

Appendix F

Camera-IMU calibration

Spatial and temporal calibration of the Camera-IMU system using `Kalibr` toolbox [24].

```
cam0:
  T_cam_imu:
  - [0.7001070735670, -0.7130844115403, 0.03688776978428, 0.07022976807384]
  - [-0.7138617058209, -0.7001472119814, 0.013976641837782, 0.021997656888638]
  - [0.01586034375047, -0.03611791207748, -0.9992216700628, -0.0009858418187710]
  - [0.0, 0.0, 0.0, 1.0]
  camera_model: pinhole
  distortion_coeffs: [-0.28702, 0.06419, -0.001523, -0.000553]
  distortion_model: radtan
  intrinsics: [395.84812, 397.135809, 358.533742, 228.498856]
  resolution: [752, 480]
  rostopic: camera/image_raw
  timeshift_cam_imu: 0.0
```

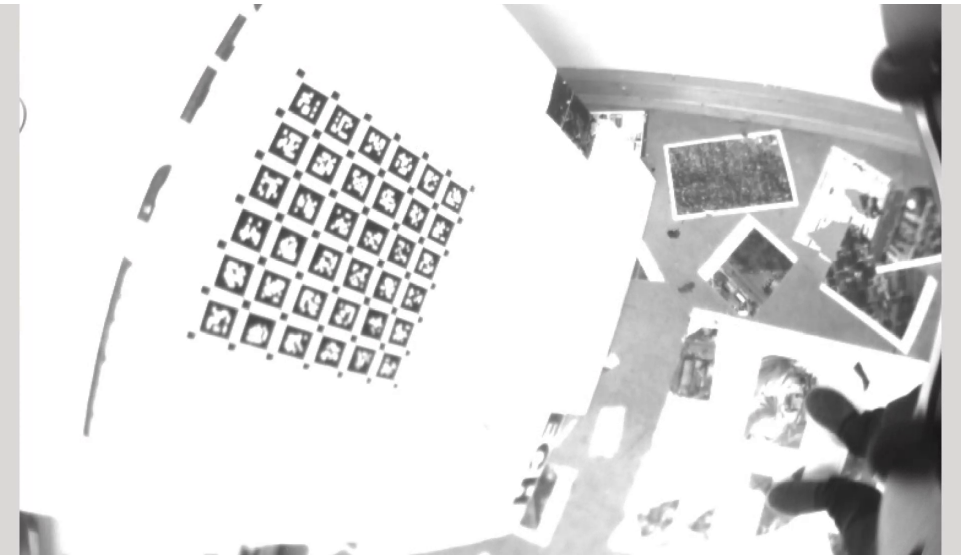


Figure F.1: Calibration sequence: Camera-IMU calibration

Appendix G

Asctec accelerometer datasheet

Low Cost ±1.5 *g* Tri Axis Accelerometer with Ratiometric Outputs

MXR9500G/M

FEATURES

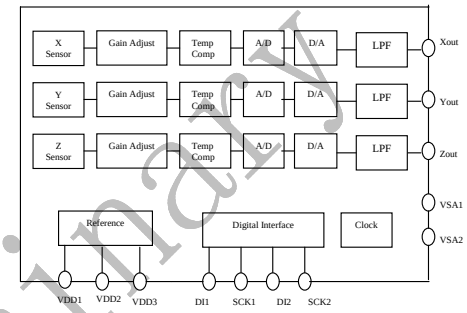
Low cost
RoHS compliant
Resolution better than 1 milli-*g*
Tri-axis accelerometer in a single package.
On chip mixed signal processing
No moving parts
No loose particle issues
>50,000 *g* shock survival rating
SMT package: 7mm X 7mm X 1.8mm
2.7V to 3.6V single supply continuous operation
No adjusting external components needed

APPLICATIONS

GPS – Electronic Compass Tilt Correction/Navigation
Consumer – LCD projectors, pedometers, blood pressure Monitor, digital cameras/MP3 players
Information Appliances – Computer Peripherals/PDA's/Mouse Smart Pens/Cell Phones
Gaming – Joystick/RF Interface/Menu Selection/Tilt Sensing
Security – Gas Line/Elevator/Fatigue Sensing

GENERAL DESCRIPTION

The MXR9500G/M is a low cost, tri axis accelerometer fabricated on a standard, submicron CMOS process. It is a complete sensing system with on-chip mixed signal processing. The MXR9500G/M measures acceleration with a full-scale range of ±1.5 *g* and a sensitivity of 500mV/g @3.0V at 25°C. It can measure both dynamic acceleration (e.g. vibration) and static acceleration (e.g. gravity). The MXR9500G/M design is based on heat convection and requires no solid proof mass. This eliminates stiction and particle problems associated with competitive devices and provides shock survival greater than 50,000 *g*, leading to significantly lower failure rate and lower loss due to handling during PCB assembly and at customer field application.



FUNCTIONAL BLOCK DIAGRAM

The MXR9500G/M provides three ratiometric analog outputs that are set to 50% of the power supply voltage at zero g.

The Max. noise floor is 1 mg/$\sqrt{Hz}$ allowing signals below 1 milli-*g* to be resolved at 1 Hz bandwidth. The MXR9500G/M is packaged in a hermetically sealed, surface mount LCC 16pins package (7 mm x 7 mm x 1.8 mm height) and is operational over a -40°C to +85°C (M) and 0°C to +70°C (G) temperature range.

I^2C fast mode interface is soon available in the next generation chip.

©MEMSIC, Inc.
800 Turnpike St., Suite 202, North Andover, MA 01845
Tel: 978.738.0900 Fax: 978.738.0196
www.memsic.com

MXR9500G/M SPECIFICATIONS (Measurements @ 25°C, Acceleration = 0 *g* unless otherwise noted; VDD1,VDD3 = 3.0V unless otherwise specified)

Parameter	Conditions	Min	Typ	Max	Units
Measurement Range[1]	Each Axis	±1.5			*g*
Nonlinearity	Best fit straight line		0.5	1.0	% of FS
Alignment Error[2]	X, Y-axis Z-axis		± 1.0 ± 3.0		degrees degrees
Transverse Sensitivity[3]			± 2.0		%
Sensitivity		475	500	525	mV/g
Sensitivity Change Over Temperature	Δ from 25°C		15		%
Zero *g* Offset Bias Level		1.45	1.50	1.55	V
		-0.1	0.0	+0.1	g
Zero *g* Offset TC	Δ from 25°C, based on 500mV/g X,Y-axis Z-axis		 1.0 1.5		 m*g*/°C m*g*/°C
Normal Output Range	Output High Output Low	2.8		 0.2	V V
Noise Density, RMS	X,Y-axis Z-axis		0.6 0.9		$mg/\sqrt{Hz}$ $mg/\sqrt{Hz}$
Resolution	@1Hz BW		0.5	1	mg
Frequency Response	@-3dB		17		Hz
Output Drive Capability	@2.7V-3.6V			100	µA
Turn-On Time[4]			75		mS
Operating Voltage Range		2.7	3.0	3.6	V
Supply Current			4.2		mA
Power Down Current				0.1	uA
Operating Temperature Range	MXR9500G MXR9500M	0 -40		+70 +85	°C °C

NOTES

[1] Guaranteed by measurement of initial offset and sensitivity.

[2] Alignment error is specified as the angle between the true and indicated axis of sensitivity.

[3] Cross axis sensitivity is the algebraic sum of the alignment and the inherent sensitivity errors.

[4] Output settled to within ±17mg.

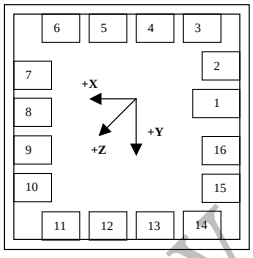
ABSOLUTE MAXIMUM RATINGS*

Supply Voltage (V_{DD}) ……...-0.5 to +7.0V
Storage Temperature …….……-65°C to +150°C
Acceleration ……..50,000 *g*

*Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; the functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Pin Description: LCC-16 Package

Pin	Name	Description
1	NC	Do Not Connect
2	Zout	Z Channel Output
3	VSA2	Connect to Ground
4	VDD1	2.7V to 3.6V
5	DI1	Power Down Pin
6	SCK1	Connect to Ground
7	NC	Do Not Connect
8	VSA1	Connect to Ground
9	NC	Do Not Connect
10	NC	Do Not Connect
11	VDD2	2.7V to 3.6V
12	Yout	Y Channel Output
13	Xout	X Channel Output
14	VDD3	2.7V to 3.6V
15	SCK2	Connect to Ground
16	DI2	Power Down Pin



(Top View)

(Side View)

Ordering Guide

Model	Temperature Range	Package
MXR9500GZ	0 to 70°C	LCC16, RoHS compliant
MXR9500MZ	-40 to 85°C	LCC16, RoHS compliant

All parts are shipped in tape and reel packaging.
Caution: ESD (electrostatic discharge) sensitive device.

THEORY OF OPERATION

The MEMSIC device is a complete tri-axis acceleration measurement system in a single package fabricated on CMOS IC process. The device operation is based on heat transfer by natural convection and operates like other accelerometers having a proof mass except it is a gas in MEMSIC sensor.

Heat source, centered in the silicon chip is suspended across a cavity. Equally spaced aluminum/polysilicon thermopiles (groups of thermocouples) are located equidistantly on all four sides of the heat source. Under zero acceleration, a temperature gradient is symmetrical about the heat source, so that the temperature is the same at all four thermopiles, causing them to output the same voltage.

Acceleration in any direction will disturb the temperature profile, due to free convection heat transfer, causing it to be asymmetrical. The temperature, and hence voltage output of the four thermopiles will then be different. The differential voltage at the thermopile outputs is directly proportional to the acceleration. Please visit the MEMSIC website at www.memsic.com for a picture/graphic description of the free convection heat transfer principle.

MXR9500G/M PIN DESCRIPTIONS

VDD1, VDD2, VDD3– These pins are the supply input for the circuits and the sensor heater in the accelerometer. The DC voltage should be between 2.7 and 3.6 volts. Refer to the section on PCB layout and fabrication suggestions for guidance on external parts and connections recommended.

VSA1, VSA2– These pins are ground pin for the accelerometer.

SCK1, SCK2– These pins are for factory used only, should be connect to ground.

DI1, DI2– These pins are the power down control pin. Pull these pins HIGH will put the accelerometer into power down mode. When the part goes into power down mode, the total current will be smaller than 0.1uA at 3V.

In normal operation mode, this pin should be connected to Ground.

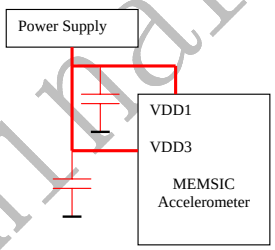
Xout – This pin is the analog output of the X-axis acceleration sensor.

Yout – This pin is the analog output of the Y-axis acceleration sensor

Zout – This pin is the analog output of the Z-axis acceleration sensor.

POWER SUPPLY NOISE REJECTION

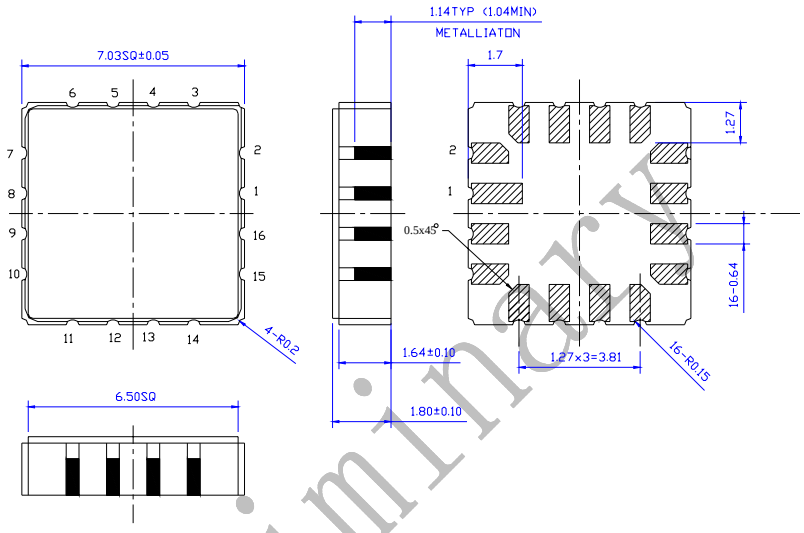
One capacitor is recommended for best rejection of power supply noise. The capacitor should be located as close as possible to the device supply pin (VDD1,VDD3). The capacitor lead length should be as short as possible, and surface mount capacitor is preferred. For typical applications, the capacitor can be ceramic 0.1 µF.



PCB LAYOUT AND FABRICATION SUGGESTIONS

1. It is best to solder a 0.1uF capacitor directly across VDD1, VSA1 and VDD3, VSA2 pin.
2. Robust low inductance ground wiring should be used.

PACKAGE DRAWING



Appendix H

Asctec gyroscope datasheet

ANALOG DEVICES

±300°/sec Yaw Rate Gyro

ADXRS610

FEATURES

Complete rate gyroscope on a single chip
Z-axis (yaw rate) response
High vibration rejection over wide frequency
2000 *g* powered shock survivability
Ratiometric to referenced supply
5 V single-supply operation
105°C operation
Self-test on digital command
Ultrasmall and light (< 0.15 cc, < 0.5 gram)
Temperature sensor output
RoHS compliant

APPLICATIONS

Vehicle chassis rollover sensing
Inertial measurement units
Platform stabilization

GENERAL DESCRIPTION

The ADXRS610 is a complete angular rate sensor (gyroscope) that uses the Analog Devices, Inc. surface-micromachining process to create a functionally complete and low cost angular rate sensor integrated with all required electronics on one chip. The manufacturing technique for this device is the same high volume BiMOS process used for high reliability automotive airbag accelerometers.

The output signal, RATEOUT (1B, 2A), is a voltage proportional to angular rate about the axis normal to the top surface of the package. The output is ratiometric with respect to a provided reference supply. A single external resistor can be used to lower the scale factor. An external capacitor sets the bandwidth. Other external capacitors are required for operation.

A temperature output is provided for compensation techniques. Two digital self-test inputs electromechanically excite the sensor to test proper operation of both the sensor and the signal conditioning circuits. The ADXRS610 is available in a 7 mm × 7 mm × 3 mm BGA ceramic package.

FUNCTIONAL BLOCK DIAGRAM

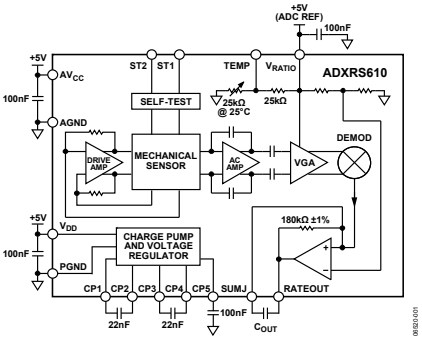


Figure 1.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 **www.analog.com**
Fax: 781.461.3113

TABLE OF CONTENTS

REVISION HISTORY

2/10—Rev. 0 to Rev. A

Updated Outline Dimensions ... 11
Changes to Ordering Guide ... 11

4/07—Revision 0: Initial Version

SPECIFICATIONS

All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed.

T_A = −40°C to +105°C, V_S = AV_{CC} = V_{DD} = 5 V, V_{RATIO} = AV_{CC}, angular rate = 0°/sec, bandwidth = 80 Hz (C_{OUT} = 0.01 µF), I_{OUT} = 100 µA, ±1 *g*, unless otherwise noted.

Table 1.

Parameter	Conditions	ADXRS610BBGZ Min	Typ	Max	Unit
SENSITIVITY[1]	Clockwise rotation is positive output				
Measurement Range[2]	Full-scale range over specifications range	±300			°/sec
Initial and Over Temperature	−40°C to +105°C	5.52	6	6.48	mV/°/sec
Temperature Drift[3]			±2		%
Nonlinearity	Best fit straight line		0.1		% of FS
NULL[1]					
Null	−40°C to +105°C	2.2	2.5	2.8	V
Linear Acceleration Effect	Any axis		0.1		°/sec/*g*
NOISE PERFORMANCE					
Rate Noise Density	T_A ≤ 25°C		0.05		°/sec/√Hz
FREQUENCY RESPONSE					
Bandwidth[4]		0.01		2500	Hz
Sensor Resonant Frequency		12	14.5	17	kHz
SELF TEST[1]					
ST1 RATEOUT Response	ST1 pin from Logic 0 to Logic 1	−650	−450	−250	mV
ST2 RATEOUT Response	ST2 pin from Logic 0 to Logic 1	250	450	650	mV
ST1 to ST2 Mismatch[5]		−5		+5	%
Logic 1 Input Voltage		3.3			V
Logic 0 Input Voltage				1.7	V
Input Impedance	To common	40	50	100	kΩ
TEMPERATURE SENSOR[1]					
V_{OUT} at 25°C	Load = 10 MΩ	2.35	2.5	2.65	V
Scale Factor[6]	@25°C, V_{RATIO} = 5 V		9		mV/°C
Load to V_S			25		kΩ
Load to Common			25		kΩ
TURN-ON TIME	Power on to ±½ °/sec of final			50	ms
OUTPUT DRIVE CAPABILITY					
Current Drive	For rated specifications			200	µA
Capacitive Load Drive				1000	pF
POWER SUPPLY					
Operating Voltage (V_S)		4.75	5.00	5.25	V
Quiescent Supply Current			3.5	4.5	mA
TEMPERATURE RANGE					
Specified Performance		−40		+105	°C

[1] Parameter is linearly ratiometric with V_{RATIO}.
[2] The maximum range possible, including output swing range, initial offset, sensitivity, offset drift, and sensitivity drift at 5 V supplies.
[3] From +25°C to −40°C or +25°C to 105°C.
[4] Adjusted by external capacitor, C_{OUT}. Reducing bandwidth below 0.01 Hz does not reduce noise further.
[5] Self-test mismatch is described as (ST2 + ST1)/((ST2 − ST1)/2).
[6] For a change in temperature from 25°C to 26°C. V_{TEMP} is ratiometric to V_{RATIO}. See the Temperature Output and Calibration section for more details.

ABSOLUTE MAXIMUM RATINGS

Table 2.

Parameter	Rating
Acceleration (Any Axis, 0.5 ms)	
Unpowered	2000 *g*
Powered	2000 *g*
V_{DD}, AV_{CC}	−0.3 V to +6.0 V
V_{RATIO}	AV_{CC}
ST1, ST2	AV_{CC}
Output Short-Circuit Duration (Any Pin to Common)	Indefinite
Operating Temperature Range	−55°C to +125°C
Storage Temperature Range	−65°C to +150°C

Stresses above those listed under the Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Drops onto hard surfaces can cause shocks of greater than 2000 *g* and can exceed the absolute maximum rating of the device. Exercise care during handling to avoid damage.

RATE SENSITIVE AXIS

The ADXRS610 is a Z-axis rate-sensing device (also called a yaw rate sensing device). It produces a positive going output voltage for clockwise rotation about the axis normal to the package top, that is, clockwise when looking down at the package lid.

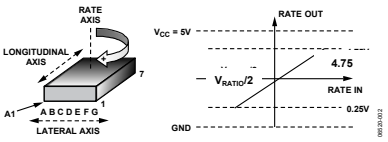


Figure 2. RATEOUT Signal Increases with Clockwise Rotation

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

PIN CONFIGURATION AND FUNCTION DESCRIPTIONS

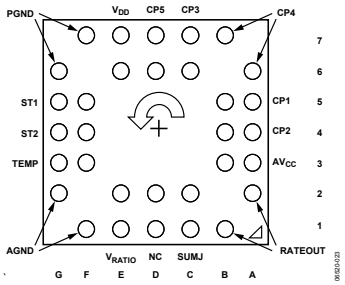


Figure 3. Pin Configuration

Table 4. Pin Function Descriptions

Pin No.	Mnemonic	Description
6D, 7D	CP5	HV Filter Capacitor (0.1 µF).
6A, 7B	CP4	Charge Pump Capacitor (22 nF).
6C, 7C	CP3	Charge Pump Capacitor (22 nF).
5A, 5B	CP1	Charge Pump Capacitor (22 nF).
4A, 4B	CP2	Charge Pump Capacitor (22 nF).
3A, 3B	AV_{CC}	Positive Analog Supply.
1B, 2A	RATEOUT	Rate Signal Output.
1C, 2C	SUMJ	Output Amp Summing Junction.
1D, 2D	NC	No Connect.
1E, 2E	V_{RATIO}	Reference Supply for Ratiometric Output.
1F, 2G	AGND	Analog Supply Return.
3F, 3G	TEMP	Temperature Voltage Output.
4F, 4G	ST2	Self-Test for Sensor 2.
5F, 5G	ST1	Self-Test for Sensor 1.
6G, 7F	PGND	Charge Pump Supply Return.
6E, 7E	V_{DD}	Positive Charge Pump Supply.

TYPICAL PERFORMANCE CHARACTERISTICS

N > 1000 for all typical performance plots, unless otherwise noted.

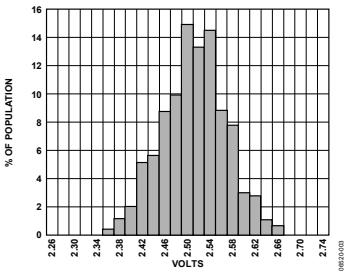


Figure 4. Null Output at 25°C (V_{RATIO} = 5 V)

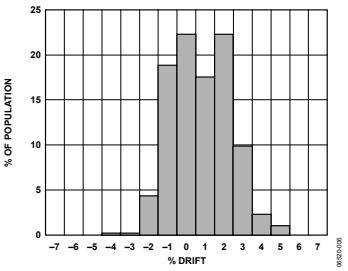


Figure 7. Sensitivity Drift over Temperature

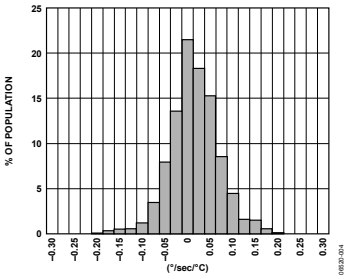


Figure 5. Null Drift over Temperature (V_{RATIO} = 5 V)

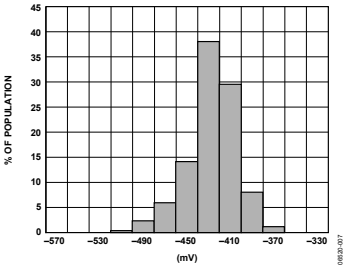


Figure 8. ST1 Output Change at 25°C (V_{RATIO} = 5 V)

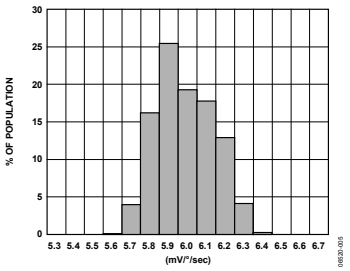


Figure 6. Sensitivity at 25°C (V_{RATIO} = 5 V)

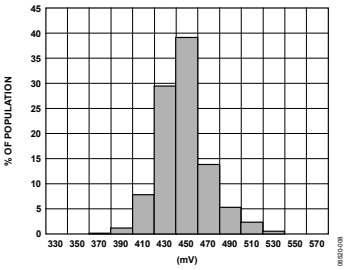


Figure 9. ST2 Output Change at 25°C (V_{RATIO} = 5 V)

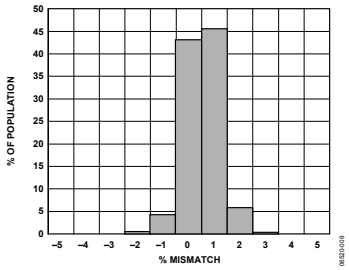


Figure 10. Self-Test Mismatch at 25°C (V_{RATIO} = 5 V)

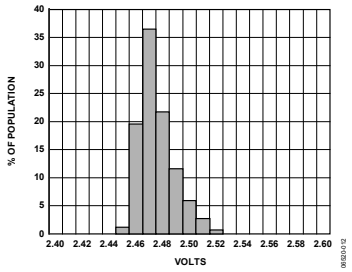


Figure 13. V_{TEMP} Output at 25°C (V_{RATIO} = 5 V)

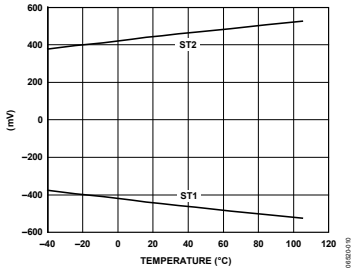


Figure 11. Typical Self-Test Change over Temperature

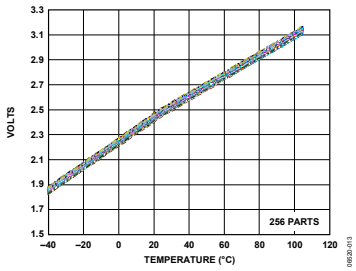


Figure 14. V_{TEMP} Output over Temperature (V_{RATIO} = 5 V)

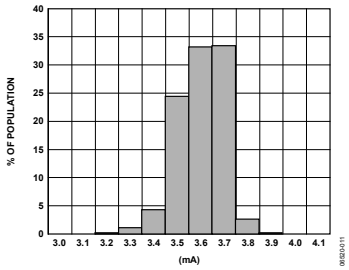


Figure 12. Current Consumption at 25°C (V_{RATIO} = 5 V)

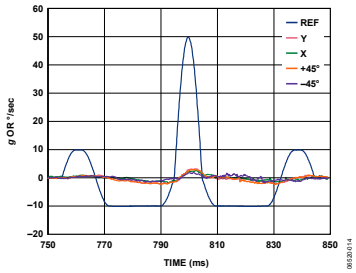


Figure 15. g and g × g Sensitivity for a 50 g, 10 ms Pulse

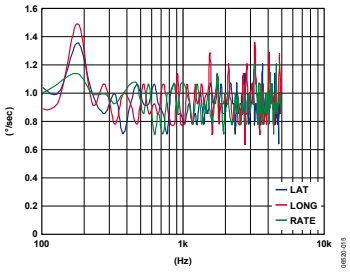


Figure 16. Typical Response to 10 g Sinusoidal Vibration (Sensor Bandwidth = 2 kHz)

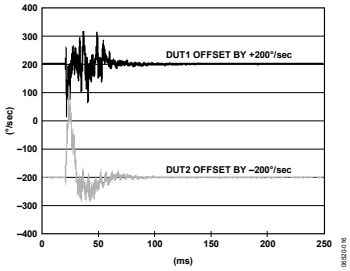


Figure 17. Typical High g (2500 g) Shock Response (Sensor Bandwidth = 40 Hz)

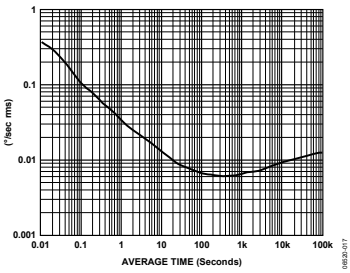


Figure 18. Typical Root Allan Deviation at 25°C vs. Averaging Time

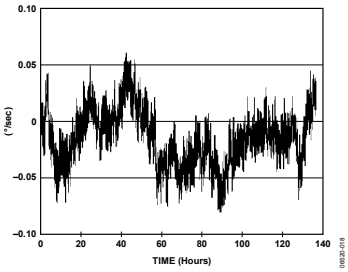


Figure 19. Typical Shift in 90 sec Null Averages Accumulated over 140 Hours

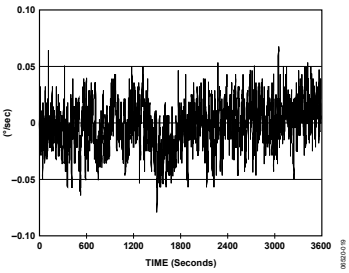


Figure 20. Typical Shift in Short Term Null (Bandwidth = 1 Hz)

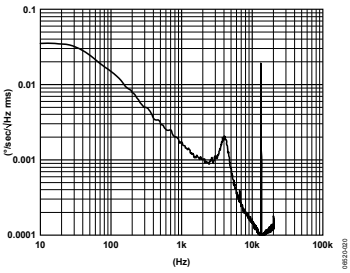


Figure 21. Typical Noise Spectral Density (Bandwidth = 40 Hz)

THEORY OF OPERATION

The ADXRS610 operates on the principle of a resonator gyro. Two polysilicon sensing structures each contain a dither frame that is electrostatically driven to resonance, producing the necessary velocity element to produce a Coriolis force during angular rate. At two of the outer extremes of each frame, orthogonal to the dither motion, are movable fingers that are placed between fixed pickoff fingers to form a capacitive pickoff structure that senses Coriolis motion. The resulting signal is fed to a series of gain and demodulation stages that produce the electrical rate signal output. The dual-sensor design rejects external *g*-forces and vibration. Fabricating the sensor with the signal conditioning electronics preserves signal integrity in noisy environments.

The electrostatic resonator requires 18 V to 20 V for operation. Because only 5 V are typically available in most applications, a charge pump is included on-chip. If an external 18 V to 20 V supply is available, the two capacitors on CP1 through CP4 can be omitted and this supply can be connected to CP5 (Pin 6D, Pin 7D). Note that CP5 should not be grounded when power is applied to the ADXRS610. Although no damage occurs, under certain conditions the charge pump may fail to start up after the ground is removed without first removing power from the ADXRS610.

SETTING BANDWIDTH

External Capacitor C_{OUT} is used in combination with the on-chip R_{OUT} resistor to create a low-pass filter to limit the bandwidth of the ADXRS610 rate response. The –3 dB frequency set by R_{OUT} and C_{OUT} is

$$f_{OUT} = \frac{1}{\left(2 \times \pi \times R_{OUT} \times C_{OUT}\right)}$$

and can be well controlled because R_{OUT} has been trimmed during manufacturing to be 180 kΩ ±1%. Any external resistor applied between the RATEOUT pin (1B, 2A) and SUMJ pin (1C, 2C) results in

$$R_{OUT} = \frac{\left(180\ \text{k}\Omega \times R_{EXT}\right)}{\left(180\ \text{k}\Omega + R_{EXT}\right)}$$

In general, an additional hardware or software filter is added to attenuate high frequency noise arising from demodulation spikes at the gyro's 14 kHz resonant frequency (the noise spikes at 14 kHz can be clearly seen in the power spectral density curve shown in Figure 21). Typically, this additional filter's corner frequency is set to greater than 5× the required bandwidth to preserve good phase response.

Figure 22 shows the effect of adding a 250 Hz filter to the output of an ADXRS610 set to 40 Hz bandwidth (as shown in Figure 21). High frequency demodulation artifacts are attenuated by approximately 18 dB.

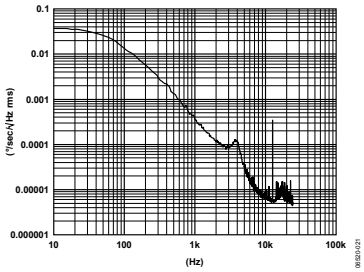


Figure 22. Noise Spectral Density with Additional 250 Hz Filter

TEMPERATURE OUTPUT AND CALIBRATION

It is common practice to temperature-calibrate gyros to improve their overall accuracy. The ADXRS610 has a temperature proportional voltage output that provides input to such a calibration method. The temperature sensor structure is shown in Figure 23. The temperature output is characteristically nonlinear, and any load resistance connected to the TEMP output results in decreasing the TEMP output and temperature coefficient. Therefore, buffering the output is recommended.

The voltage at the TEMP pin (3F, 3G) is nominally 2.5 V at 25°C, and V_{RATIO} = 5 V. The temperature coefficient is ~9 mV/°C at 25°C. Although the TEMP output is highly repeatable, it has only modest absolute accuracy.

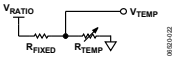


Figure 23. ADXRS610 Temperature Sensor Structure

CALIBRATED PERFORMANCE

Using a 3-point calibration technique, it is possible to calibrate the null and sensitivity drift of the ADXRS610 to an overall accuracy of nearly 200°/hour. An overall accuracy of 40°/hour or better is possible using more points.

Limiting the bandwidth of the device reduces the flat-band noise during the calibration process, improving the measurement accuracy at each calibration point.

ADXRS610 AND SUPPLY RATIOMETRICITY

The ADXRS610 RATEOUT and TEMP signals are ratiometric to the V_{RATIO} voltage, that is, the null voltage, rate sensitivity, and temperature outputs are proportional to V_{RATIO}. Thus, the ADXRS610 is most easily used with a supply-ratiometric ADC that results in self-cancellation of errors due to minor supply variations. There is some small error due to nonratiometric behavior. Typical ratiometricity error for null, sensitivity, self-test, and temperature output is outlined in Table 3.

Note that V_{RATIO} must never be greater than AV_{CC}.

Table 3. Ratiometricity Error for Various Parameters

Parameter	$V_S = V_{RATIO} = 4.75$ V	$V_S = V_{RATIO} = 5.25$ V
ST1		
Mean	−0.4%	−0.3%
Sigma	0.6%	0.6%
ST2		
Mean	−0.4%	−0.3%
Sigma	0.6%	0.6%
Null		
Mean	−0.04%	−0.02%
Sigma	0.3%	0.2%
Sensitivity		
Mean	0.03%	0.1%
Sigma	0.1%	0.1%
V_{TEMP}		
Mean	−0.3%	−0.5%
Sigma	0.1%	0.1%

NULL ADJUSTMENT

The nominal 2.5 V null is for a symmetrical swing range at RATEOUT (1B, 2A). However, a nonsymmetrical output swing may be suitable in some applications. Null adjustment is possible by injecting a suitable current to SUMJ (1C, 2C). Note that supply disturbances may reflect some null instability. Digital supply noise should be avoided particularly in this case.

SELF-TEST FUNCTION

The ADXRS610 includes a self-test feature that actuates each of the sensing structures and associated electronics as if subjected to angular rate. It is activated by standard logic high levels applied to Input ST1 (5F, 5G), Input ST2 (4F, 4G), or both. ST1 causes the voltage at RATEOUT to change about −0.5 V, and ST2 causes an opposite change of +0.5 V. The self-test response follows the viscosity temperature dependence of the package atmosphere, approximately 0.25%/°C.

Activating both ST1 and ST2 simultaneously is not damaging. ST1 and ST2 are fairly closely matched (±5%), but actuating both simultaneously may result in a small apparent null bias shift proportional to the degree of self-test mismatch.

ST1 and ST2 are activated by applying a voltage equal to V_{RATIO} to the ST1 and ST2 pins. The voltage applied to ST1 and ST2 must never be greater than AV_{CC}.

CONTINUOUS SELF-TEST

The on-chip integration of the ADXRS610 gives it higher reliability than is obtainable with any other high volume manufacturing method. In addition, it is manufactured under a mature BiMOS process with field-proven reliability. As an additional failure detection measure, a power-on self-test can be performed. However, some applications may warrant continuous self-test while sensing rate. Details outlining continuous self-test techniques are also available in a separate application note.

OUTLINE DIMENSIONS

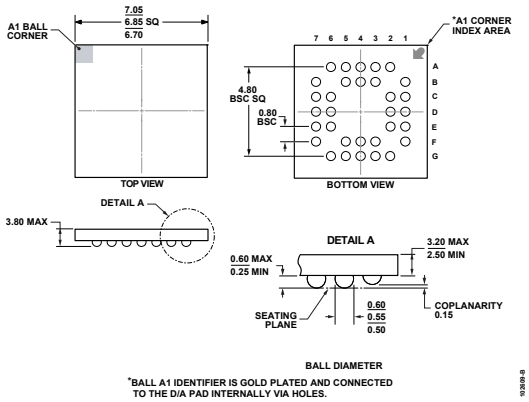


Figure 24. 32-Lead Ceramic Ball Grid Array [CBGA] (BG-32-3) Dimensions shown in millimeters

ORDERING GUIDE

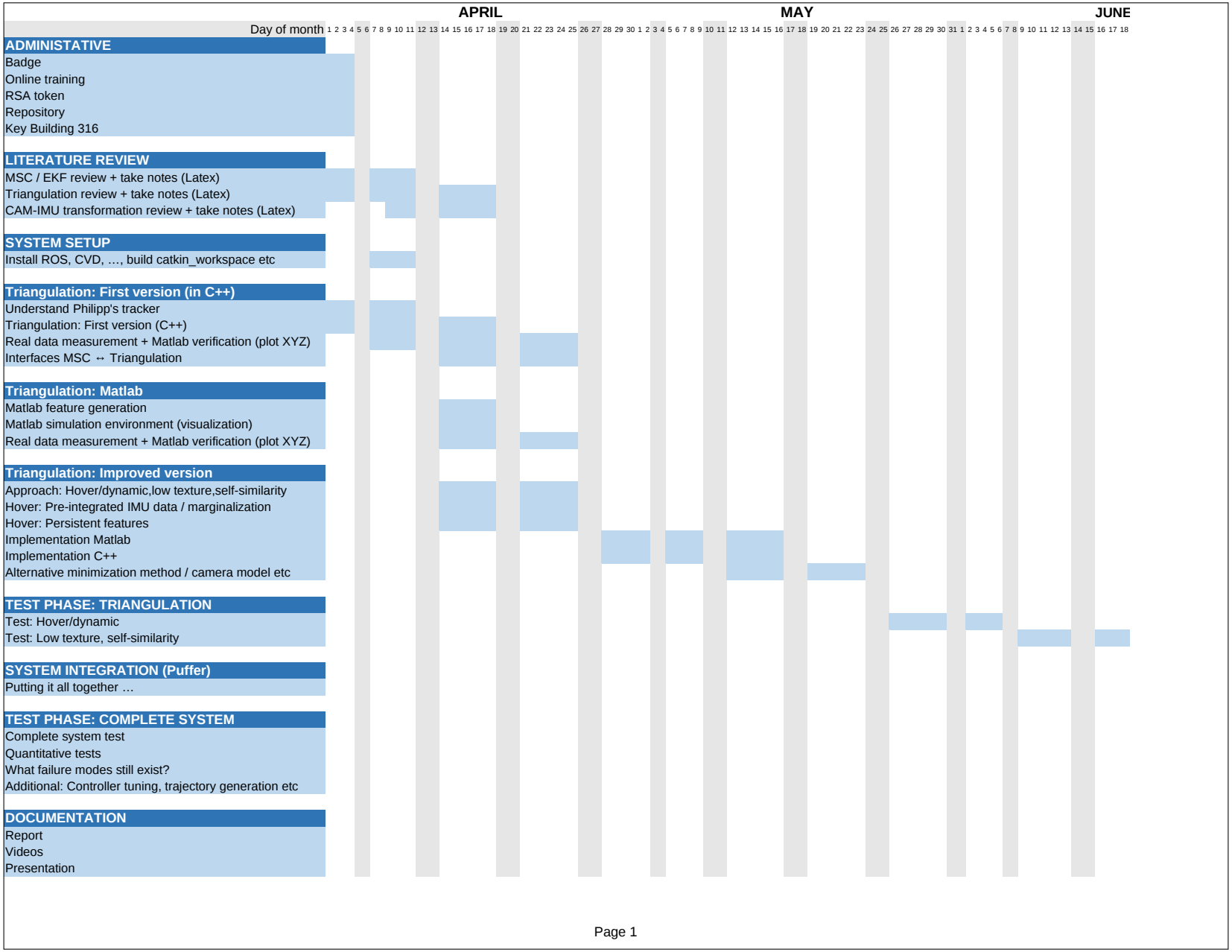
Model[1]	Temperature Range	Package Description	Package Option
ADXRS610BBGZ	–40°C to +105°C	32-Lead Ceramic Ball Grid Array (CBGA)	BG-32-3
ADXRS610BBGZ-RL	–40°C to +105°C	32-Lead Ceramic Ball Grid Array (CBGA)	BG-32-3
ADXRS610WBBGZA	–40°C to +105°C	32-Lead Ceramic Ball Grid Array (CBGA)	BG-32-3
ADXRS610WBBGZA-RL	–40°C to +105°C	32-Lead Ceramic Ball Grid Array (CBGA)	BG-32-3
EVAL-ADXRS610Z		Evaluation Board	

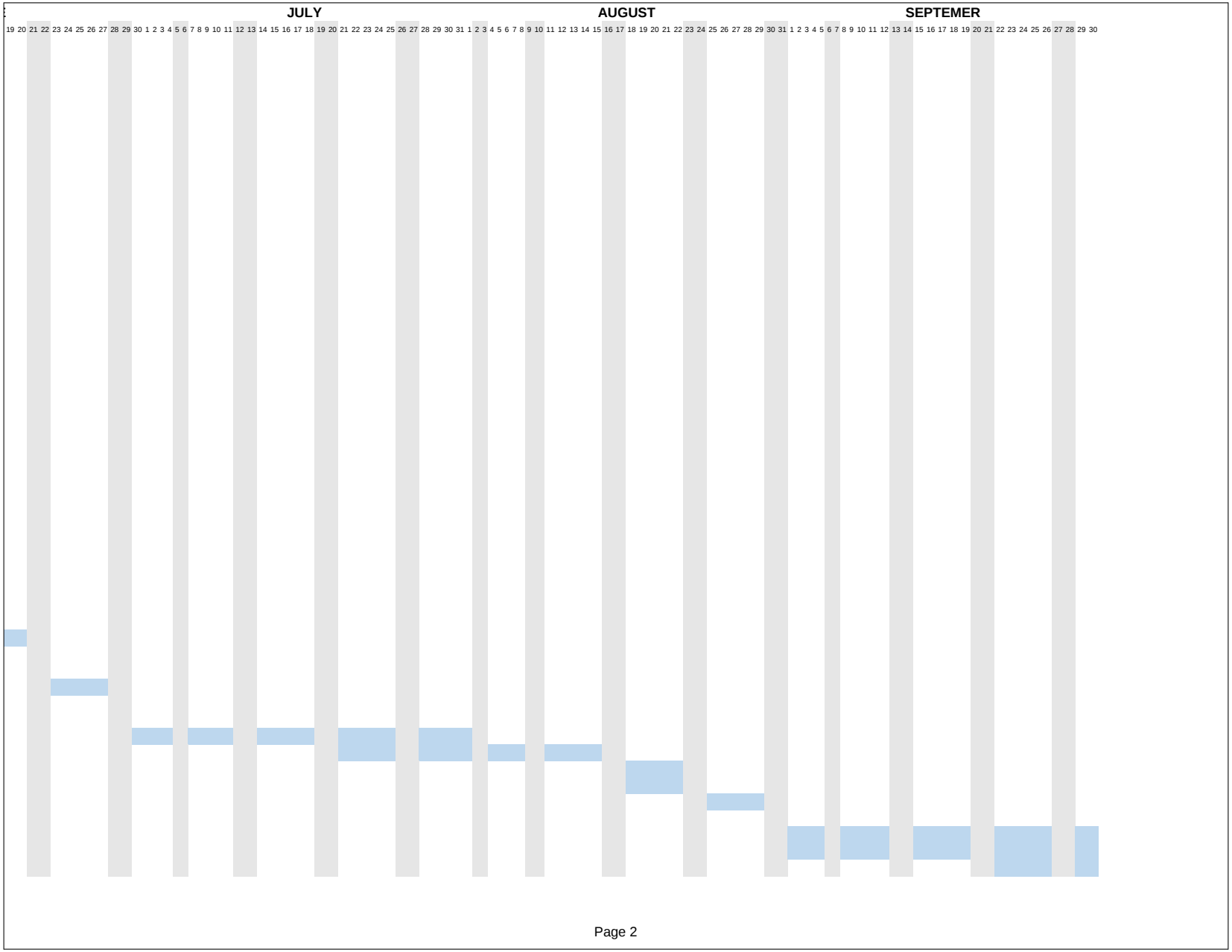
[1] Z = RoHS Compliant Part.

NOTES

Appendix I

Time plan


APRIL
MAY
JUNE
Day of month
ADMINISTATIVE
Badge
Online training
RSA token
Repository
Key Building 316
LITERATURE REVIEW
MSC / EKF review + take notes (Latex)
Triangulation review + take notes (Latex)
CAM-IMU transformation review + take notes (Latex)
SYSTEM SETUP
Install ROS, CVD, …, build catkin_workspace etc
Triangulation: First version (in C++)
Understand Philipp's tracker
Triangulation: First version (C++)
Real data measurement + Matlab verification (plot XYZ)
Interfaces MSC ↔ Triangulation
Triangulation: Matlab
Matlab feature generation
Matlab simulation environment (visualization)
Real data measurement + Matlab verification (plot XYZ)
Triangulation: Improved version
Approach: Hover/dynamic,low texture,self-similarity
Hover: Pre-integrated IMU data / marginalization
Hover: Persistent features
Implementation Matlab
Implementation C++
Alternative minimization method / camera model etc
TEST PHASE: TRIANGULATION
Test: Hover/dynamic
Test: Low texture, self-similarity
SYSTEM INTEGRATION (Puffer)
Putting it all together …
TEST PHASE: COMPLETE SYSTEM
Complete system test
Quantitative tests
What failure modes still exist?
Additional: Controller tuning, trajectory generation etc
DOCUMENTATION
Report
Videos
Presentation
Page 1


JULY
AUGUST
SEPTEMER
19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Page 2

Bibliography

[1] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2007)*, (Nara, Japan), November 2007.

[2] G. Nuetzi, S. Weiss, D. Scaramuzza, and R. Siegwart, "Fusion of imu and vision for absolute scale estimation in monocular slam," *Journal of Intelligent and Robotic Systems*, vol. 61, p. 287ÃćâĆňâĂIJ299, 2011.

[3] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli, and R. Siegwart, "Monocular vision for long-term micro aerial vehicle state estimation: A compendium," *Journal of Field Robotics*, vol. 30, no. 5, pp. 803–831, 2013.

[4] S. Weiss, R. Siegwart, and V. Kumar, *Vision Based Navigation for Micro Helicopters*. ETH, 2012.

[5] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *in Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 10–14, 2007.

[6] Ascending Technologies GmbH, *AscTec Hummingbird with AutoPilot, User's Manual*, 2012.

[7] MATRIX VISION GmbH, *mvBlueFox Datasheet*, 2014. `http://www.matrix-vision.com/USB2.0-single-board-camera-mvbluefox-mlc.html`.

[8] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation," Tech. Rep. 2005-002, University of Minnesota, Dept. of Comp. Sci. & Eng., Mar. 2005.

[9] A. Mourikis and U. of Minnesota, *Characterization and Optimization of the Accuracy of Mobile Robot Localization*. University of Minnesota, 2008.

[10] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johnson, A. Ansar, and L. Matthies, "Vision-aided inertial navigation for spacecraft entry, descent, and landing," *IEEE Transactions on Robotics*, vol. 25, pp. 264–280, April 2009.

[11] S. R. A. J. N. Trawny, A.I. Mourikis and J. Montgomery, "Vision-aided inertial navigation for pin-point landing using observations of mapped landmarks," *Journal of Field Robotics*, vol. 24, no. 5, pp. 257–378, 2007.

[12] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *International Journal of Robotics Research*, vol. 32, pp. 690–711, May 2013.

[13] F. Devernay and O. Faugeras, "Straight lines have to be straight," in *In SPIE, volume 2567*, 2001.

[14] D. Scaramuzza, *Omnidirectional Vision: from Calibration to Root Motion Estimation.* PhD thesis, Swiss Federal Institute of Technology Zurich (ETHZ), February 2008.

[15] E. Rosten and T. Drummond, "Machine learning for high speed corner detection," in *9th European Conference on Computer Vision*, May 2006.

[16] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, pp. 105–119, 2010.

[17] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models.* SpringerVerlag, 2003.

[18] M. Li and A. I. Mourikis, "Optimization-based estimator design for vision-aided inertial navigation," in *Proceedings of Robotics: Science and Systems*, (Sydney, Australia), July 2012.

[19] D. Bardsley and B. Li, "3D Reconstruction Using the Direct Linear Transform with a Gabor Wavelet Based Correspondence Measure," tech. rep.

[20] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[21] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics).* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[22] D. G. Kottas, K. J. Wu, and S. I. Roumeliotis, "Detecting and dealing with hovering maneuvers in vision-aided inertial navigation systems.," in *IROS*, pp. 3172–3179, IEEE, 2013.

[23] M. Li and A. I. Mourikis, "Optimization-based estimator design for vision-aided inertial navigation: Supplemental materials," in *Proceedings of Robotics: Science and Systems*, (Sydney, Australia), July 2012.

[24] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 1280–1286, Nov 2013.