Henrik Dobbe Flemmen

ROVTIO: RObust Visual Thermal Inertial Odometry

Masteroppgave i Kybernetikk og robotikk
Veileder: Kostas Alexis
Juli 2021

Henrik Dobbe Flemmen

ROVTIO: RObust Visual Thermal Inertial Odometry

Masteroppgave i Kybernetikk og robotikk
Veileder: Kostas Alexis
Juli 2021

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for teknisk kybernetikk

NTNU
Kunnskap for en bedre verden

ROVTIO: RObust Visual Thermal Inertial Odometry

Henrik Dobbe Flemmen

July 2021

MASTER THESIS
Department of Engineering Cybernetics
Norwegian University of Science and Technology

Kostas Alexis

Preface

Trondheim, 23.7.2021

Henrik D. Flemmen

Henrik Dobbe Flemmen

Acknowledgments

First of all I would like to thank my supervisor Kostas Alexis for his guidance and cooperation in this project. Furthermore I would like to thank Nikhil Khedekar for technical help and guidance with this project. I also want to thank Mihir and the rest of ARL for collecting the datasets that I used.

Furthermore, I wish to thank my family for support while working with and writing about this project. I also wish to thank Howard for helping me with proofreading my thesis.

H.D.F.

Executive Summary

This thesis introduces and evaluates an algorithm to estimate the motion of a flying robot using a visual camera, a thermal camera and inertial sensors. Using both a visual camera and a thermal camera improves robustness to degraded conditions for either of the cameras and the inertial sensors improve robustness to quick motions and complement the cameras. Despite both visual cameras and thermal cameras being camera-type sensors, the data they provide are fundamentally different. This makes the fusion of these two modalities non-trivial and an open problem how to fuse them.

A suggested algorithm to solve this problem is developed and released as one of the first open source of its kind, to the best of the authors knowledge.

Contents

Chapter 1

Introduction

The recent years have seen autonomous micro aerial vehicles (MAVs) being used in more and more different scenarios. In particular there is a drive to use them in conditions which are difficult or dangerous for humans to operate in. Reliable autonomous flight with a MAV relies on the MAV being able to measure its position relative to the surroundings. In many cases this can be resolved by global navigation satellite system (GNSS) based navigation. However, GNSS relies on a clear line of sight to the sky, which prohibits efficient use indoors, underground or in other similar scenarios. The desire to deploy MAVs in such scenarios has led to the development of alternative options. One of the options that has proved popular is based on using cameras. The development of visual odometry (VO) methods for accurate dead reckoning allows cheap lightweight sensors to replace the need for a GNSS receiver with access to satellite signals. VO methods typically work by tracking features in the scene and simultaneously estimating the 3D geometry of these points and the camera locations. Monocular VO alone can only estimate the motion up to scale and either another camera in a stereo setup or another sensor is needed. A common sensor to add due to its complementary nature to the camera is an inertial measurement unit (IMU). It measures the acceleration and the angular rate of the MAV and can observe the scale of the trajectory. Due to the quick drift of position estimates from an IMU, feedback from other sensors is needed for inertial navigation, e.g. from a camera. With an IMU and a camera the MAV can use visual inertial odometry (VIO) [1] methods to track its motion up to scale. VIO methods work well when there is sufficient illumination in the environment. With darkness, the camera cannot observe the scene and therefore cannot track it. A possible solution to this problem could be to use thermal cameras instead of the visible-light cameras. They are not limited by the illumination since all objects radiate thermal radiation proportional to their temperature and emissivity. With a thermal camera and an IMU a thermal inertial odometry (TIO) method can track the environment and the location of the MAV. Despite all objects emitting thermal radiation, it does not mean that there are good features to track in the environment. If the thermal camera registers only the same temperature at all pixels, it cannot be used to track the environment and camera motion. One solution to this problem could be to use both a thermal camera and a visual camera. Then the MAV can operate if either of the modalities are in good condition. This is core underlying motivation and goal of this thesis.

1.1 Objectives

There are two objectives of this thesis:

- Develop a robust odometry system based on a visual camera, a thermal camera and inertial sensors.
- Evaluate the developed system and compare it to other relevant odometry methods.

1.2 Contributions

This thesis contributes an open source visual thermal inertial odometry algorithm which can be used by the robotics community[1]. To the authors best knowledge, this is among the first attempts to release an open-source package for combined visual-thermal-inertial odometry estimation. It is thoroughly evaluated presenting its potential and core limitations. The remaining contributions are then the analysis of the problem and an approach to visual, thermal, inertial navigation which presented several challenges and suggestions for future development.

1.3 Outline

The remainder of this thesis is structured as follows: Chapter 2 introduces preliminary topics needed for VIO/TIO and evaluation of VIO/TIO. Chapter 3 presents earlier work on the subject. Chapter 4 introduces ROVIO[1] which is the base method for the developed approach. Chapter 5 describes the method developed in this thesis and the changes from original ROVIO. Chapter 6 evaluates ROVTIO and compares it with relevant methods. Chapter 8 concludes and sums up this thesis and gives recommendations for future work.

[1]https://github.com/ntnu-arl/rovtio

Chapter 2

Background

2.1 Camera models

To be able to relate camera measurements to the 3D geometry of the real world any visual navigation system relies on a geometrical model relating a 2D pixel coordinate to a corresponding 3D bearing vector. In addition to this, some algorithms use a photometric model that models the relation between the real world radiation and the detected color in the image.

2.1.1 Geometric camera models

Geometric camera models model where a visible point in the 3D world will end up in the image. Typically denoted as

$$\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2 \tag{2.1}$$

A common and simple geometrical camera model is the pinhole model [2].

$$\pi({}_{C}\mathbf{p}) = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} f_x \dfrac{{}_{C}p_x}{{}_{C}p_z} + c_x \\ f_y \dfrac{{}_{C}p_y}{{}_{C}p_z} + c_y \end{bmatrix}, \quad {}_{C}\mathbf{p} = \left[{}_{C}p_x, {}_{C}p_y, {}_{C}p_z\right]^T \tag{2.2}$$

Where

$\pi(.)$ is the projection function

${}_{\mathcal{B}}\mathbf{r} \in \mathbb{R}^3$ is a 3D point in the real world expressed in the camera (C) coordinate system.

$\mathbf{u} = [u_x, u_y]$ is the corresponding pixel location ${}_{C}\mathbf{p}$ will project to.

f_x, f_y, c_x, c_y is the intrinsic parameters of the model which needs to be fitted to the specific lens.

While this model can describe many lenses with sufficient accuracy, many other common lenses have nonlinear effects in the projection. In particular wide angle lenses will often have significant distortion compared to the pinhole model. Two common solutions to this issue exist: Either use other models that capture these effects better or have a separate undistortion step before the projection with the pinhole model.

Several models exist, but only the Kannala-Brandt/Equidistant/Fisheye model is included here since that is the one used in this project. The Kannala-Brandt model[3] is sometimes called the equidistant model when used as a distortion model for the pinhole model. In the opencv documentation it is used as a distortion model for the pinhole camera model and called the fisheye model. Despite the many different names and the difference between being a stand alone model or a distortion model, all the variations are mathematically equivalent.[4]. While being mathematically equivalent, variations in the naming conventions for the parameters have been observed by the author of this thesis.

The Kannala-Brandt model as formulated by [4] as a standalone model:

$$\mathbf{u} \in \mathbb{R}^2 = [u_x, u_y]^T, {}_C\mathbf{p} \in \mathbb{R}^3 = [{}_Cp_x, {}_Cp_y, {}_Cp_z] \tag{2.3a}$$

$$\mathbf{u} = \pi({}_C\mathbf{p}) \tag{2.3b}$$

$$r = \sqrt{{}_Cp_x^2 + {}_Cp_y^2} \tag{2.3c}$$

$$\theta = atan2(r, {}_Cp_z) \tag{2.3d}$$

$$\theta_d = \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) \tag{2.3e}$$

$$u_x = f_x \frac{\theta_d}{r} {}_Cp_x + c_x \tag{2.3f}$$

$$u_y = f_y \frac{\theta_d}{r} {}_Cp_y + c_y \tag{2.3g}$$

$$\tag{2.3h}$$

Where

${}_C\mathbf{p} \in \mathbb{R}^3$: A 3D world point to be projected, expressed in the camera ($\mathcal{C}$) coordinate system.

$\mathbf{u} \in \mathbb{R}^2$: The projection of the point ${}_C\mathbf{p}$.

f_y, f_x, c_x, c_y: Intrinsic parameters of the model, similar to the pinhole parameters.

k_1, k_2, k_3, k_4: Also intrinsic parameters of the model, often denoted "distortion coefficients".

As earlier mentioned the model is the equivalent of the equidistant model/fisheye model, but they cannot represent points behind or directly to the side of the camera, since they have an intermediate representation as the pinhole model.

It is generally feasible to use a lens with a large field of view(FoV) and a suited camera model, compared to using a lens with low distortion and a simple camera model [5].

2.1.1.1 Calibration

The process of finding the intrinsic parameters for a lens is called camera calibration. There are various procedures that can be used, but a quite simple one is to use several images of a planar calibration pattern, then simultaneously optimize over the pose of the pattern and the intrinsic of the camera. The details of this are out of scope for this thesis, but a few relevant tools are opencv [2], basalt [6] and kalibr [7]. The two latter can also calibrate camera-imu extrinsics, which is discussed later.

2.1.2 Photometric camera models

While geometric lens models model where in the image a 3D point will be projected to, photometric camera models model the color. Specifically they model the excitation value at the pixel as a function of the original radiation.

There are different effects that can be modeled in a photometric camera model. Two common ones are the nonlinear response function of the camera sensor and the vignetting of the lens. In most cameras the pixel values are a nonlinear function of the incoming radiation. It is mostly a monotonic function which makes the image look nice to the human eye, but it might impact tracking algorithms. Many lenses, in particular wide angle lenses, are darker on the edge of the image than in the center, an effect called vignetting. As opposed to the nonlinear response function, this effect must be compensated separately for each pixel.

There are different ways of modeling and calibrating these effects. Some approaches use parametric representations of the vignetting and the response function. A few are outlined in [8]. According to [9] parametric models are often not capable of capturing the complex distortions of lenses and dense methods are preferred instead. Dense methods instead create a lookup table for all pixels and radiance values. I.e. a lookup table for all values between 0 and 255 for the nonlinear response function and a lookup table for all pixels for the vignetting. Dense methods often require setups with uniform illumination, but [9] propose a method that instead requires a large number of images.

2.2 Inertial navigation

Inertial measurment units (IMUs) consist (at least) of a 3-axes accelerometer and a 3-axes attitude rate sensor (ARS). With these two, the IMU can measure the 3 degree of freedom (DoF) linear acceleration and the 3 DoF angular velocity. One could in theory integrate these measurements to get the position and attitude. However, in practice this will quickly lead to drift in the estimate.

To be able to use IMUs to estimate position the estimator needs an additional position estimator to correct for the drift and estimate the model parameters. A common sensor model is:

$$
\begin{aligned}
{}_{\mathcal{B}}\mathbf{a}_{imu} &= \mathbf{q}_{\mathcal{B}\mathcal{G}}({}_{\mathcal{G}}\mathbf{a}_{\mathcal{G}\mathcal{B}} - {}_{\mathcal{G}}\mathbf{g}) + {}_{\mathcal{B}}\mathbf{b}_{acc} + {}_{\mathcal{B}}\mathbf{w}_{acc} && \text{(2.4a)}\\
{}_{\mathcal{B}}\dot{\mathbf{b}}_{acc} &= {}_{\mathcal{B}}\mathbf{w}_{b,acc} && \text{(2.4b)}\\
{}_{\mathcal{B}}\boldsymbol{\omega}_{imu} &= {}_{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{G}\mathcal{B}} + {}_{\mathcal{B}}\mathbf{b}_{ars} + {}_{\mathcal{B}}\mathbf{w}_{ars} && \text{(2.4c)}\\
{}_{\mathcal{B}}\mathbf{b}_{ars} &= {}_{\mathcal{B}}\mathbf{w}_{b,ars} && \text{(2.4d)}
\end{aligned}
$$

where ${}_{\mathcal{B}}\mathbf{a}_{imu} \in \mathbb{R}^3$ is the linear acceleration measurement from the IMU and ${}_{\mathcal{B}}\boldsymbol{\omega}_{imu} \in \mathbb{R}^3$ is the rotation rate measurement from the IMU. ${}_{\mathcal{G}}\mathbf{a}_{\mathcal{G}\mathcal{B}}$ is the true acceleration from the inertial frame to the IMU frame expressed in the inertial frame and ${}_{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{G}\mathcal{B}}$ is the true angular rate between the inertial frame and the imu frame. $\mathbf{q}_{\mathcal{B}\mathcal{G}} \in SO(3)$ is the relative rotation from the IMU frame($\mathcal{B}$) to the inertial frame(NED,$\mathcal{G}$) and ${}_{\mathcal{G}}\mathbf{g} \in \mathbb{R}^3 = [0,0,9.81]^T$ is the gravity expressed in the inertial frame. ${}_{\mathcal{B}}\mathbf{b}_{acc} \in \mathbb{R}^3$ is the linear bias, ${}_{\mathcal{B}}\mathbf{w}_{acc} \in \mathbb{R}^3$, ${}_{\mathcal{B}}\mathbf{w}_{acc} \in \mathbb{R}^3$ is the linear sensor noise and ${}_{\mathcal{B}}\mathbf{w}_{b,acc} \in \mathbb{R}^3$ is the bias drift. ${}_{\mathcal{B}}\mathbf{b}_{ars}$ is the angular bias, ${}_{\mathcal{B}}\mathbf{w}_{ars}$ is the angular sensor noise and ${}_{\mathcal{B}}\mathbf{w}_{b,ars}$ is the angular drift.

From this model one can see why integrating the IMU measurements will lead to drift. Integrating the measurements with a nonzero error in the estimate for the biases will lead the estimate to be increasingly wrong with time. The biases are slowly random varying offsets which need to be estimated to be able to use IMUs for navigation.

To use IMUs for navigation we need a somewhat larger state than only the kinematic system since we also need to estimate the biases.

$$
\mathbf{x} = \begin{bmatrix} {}_{\mathcal{G}}\mathbf{p}_{\mathcal{G}\mathcal{B}} \\ {}_{\mathcal{G}}\mathbf{v}_{\mathcal{G}\mathcal{B}} \\ {}_{\mathcal{B}}\mathbf{b}_{acc} \\ {}_{\mathcal{B}}\mathbf{q}_{\mathcal{G}\mathcal{B}} \\ {}_{\mathcal{B}}\mathbf{b}_{ars} \end{bmatrix} = \begin{bmatrix} \text{The position of the IMU expressed in world coordinates.} \\ \text{The velocity of the IMU expressed in world coordinates.} \\ \text{The bias of the accelerometer in the IMU-frame.} \\ \text{The attitude of the IMU in the the world expressed in the IMU frame.} \\ \text{The bias of the ARS expressed in the IMU-frame.} \end{bmatrix} \qquad (2.5)
$$

If the state is known, future states can be found from the inertial measurements using the strapdown equations:

$$
\begin{align}
{}_{\mathcal{G}}\dot{\mathbf{p}}_{\mathcal{G}\mathcal{B}} &= {}_{\mathcal{G}}\mathbf{v}_{\mathcal{G}\mathcal{B}} \tag{2.6a}\\
{}_{\mathcal{G}}\dot{\mathbf{v}}_{\mathcal{G}\mathcal{B}} &= \mathbf{q}_{\mathcal{G}\mathcal{B}}({}_{\mathcal{B}}\mathbf{a}_{imu} - {}_{\mathcal{B}}\mathbf{b}_{acc} - {}_{\mathcal{B}}\mathbf{w}_{acc}) + {}_{\mathcal{G}}\mathbf{g} \tag{2.6b}\\
{}_{\mathcal{B}}\dot{\mathbf{b}}_{acc} &= {}_{\mathcal{B}}\mathbf{w}_{b,acc} \tag{2.6c}\\
\dot{\mathbf{q}}_{\mathcal{G}\mathcal{B}} &= {}_{\mathcal{B}}\boldsymbol{\omega}_{imu} - {}_{\mathcal{B}}\mathbf{b}_{ars} - {}_{\mathcal{B}}\mathbf{w}_{ars} \tag{2.6d}\\
{}_{\mathcal{B}}\dot{\mathbf{b}}_{ars} &= {}_{\mathcal{B}}\mathbf{w}_{b,ars} \tag{2.6e}
\end{align}
$$

where the symbols are the same as in eq. 2.4a and eq. 4.1. The noise parameters are typically wrongly assumed to be 0, which leads to errors that need to be corrected for by other sensors.

2.2.1 Visual inertial calibration

To accurately use IMUs for visual inertial navigation there is a set of intrinsic parameters for the sensors that needs to be determined. The noise parameters (${}_{\mathcal{B}}\mathbf{w}_{acc}$,${}_{\mathcal{B}}\mathbf{w}_{b,acc}$, ${}_{\mathcal{B}}\mathbf{w}_{ars}$ and ${}_{\mathcal{B}}\mathbf{w}_{b,ars}$) are needed to know the expected uncertainty of the IMU. These can typically be found in the datasheet for the sensor.

The geometrical transformation from the IMU relative to the camera is also needed. Some visual inertial methods can estimate it online, but this requires the motion of the system to make the transformation observable. The state of the art way of estimating this transformation is to do some high dynamic movement while tracking a fixed marker with the camera. Then the kinematic state of the camera relative to the world frame can be found from the marker and the kinematic state of the IMU can be estimated from the motion. The most common tools for this are the Kalibr toolbox [1] and the calibration tools from Basalt [2]. Both of these use a spline based continuous time representation of the state to align the trajectories from visual and inertial measurements.

2.3 Iterated extended Kalman filter

The iterated extended Kalman filter (IEKF) is based on the exteded Kalman filter (EKF), which in turn is based on the linear Kalman filter. All of these are bayesian filters. Here is an introduction to Kalman filtering based on [10].

[1]https://github.com/ethz-asl/kalibr
[2]https://gitlab.com/VladyslavUsenko/basalt/-/blob/master/doc/Calibration.md

2.3.1 Linear Kalman filter

The discrete linear Kalman filter assumes a linear process model where the state at timestep k, $\mathbf{x}_k$ and process noise $\mathbf{w}_k$ are the only values that influence the next state:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{w}_k \tag{2.7}$$

The matrix $\mathbf{A}$ is then the state transition matrix. The goal of the Kalman filter is then to state $\mathbf{x}_k$ by a measurement $\mathbf{y}_k$. The linear Kalman filter assumes a linear measurement model:

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{v}_k \tag{2.8}$$

where the matrix $\mathbf{C}$ is the measurement matrix and $\mathbf{v}_k$ is the measurement noise.

Both the process noise $\mathbf{w}_k$ and the measurement noise $\mathbf{v}_k$ are assumed to be white noise processes and are assumed to be uncorrelated to each other.

The Kalman filter needs the state to be observable given the measurement and also assumes the distribution over the processes noise and measurement noise to be known. It is common to assume them to be white gaussian noise processes.

The Kalman filter solves the estimation problem recursively and is usually separated into two steps: The prediction and the update. Both of these steps change both the estimate for the state and also an estimate for the covariance of the state.

The update step assumes a prior estimate for the state, $\hat{\mathbf{x}}_k^-$ with the corresponding covariance $\mathbf{P}_k^-$ and updates the state and covariance with the measurement. We distinguish between the the prior which is denoted by the superscript "-" and the posterior without the superscript. The update step calculates the posterior state and covariance from the priors and the measurement.

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k^-) \tag{2.9a}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{C})\mathbf{P}_k^- \tag{2.9b}$$

$\hat{\mathbf{x}}_k$ is then the resulting estimate for the timestep k, $\mathbf{y}_k$ is the measurement at timestep k and $\mathbf{K}_k$ is the Kalman gain and calculated as

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{C}^T(\mathbf{C}\mathbf{P}_k^-\mathbf{C}^T + \mathbf{R})^{-1} \tag{2.10}$$

where $\mathbf{R}$ is the covariance matrix of the measurement noise $\mathbf{v}_k$. Here only the result is included; why this is statistically optimal is shown in [10]. Note that the update step does not propagate time, it only works on the same timestep k. It takes the priors, $\hat{\mathbf{x}}_k^-$ and $\mathbf{P}_k^-$ and the measurement $\mathbf{y}_k$ and calculates the posterior estimate $\hat{\mathbf{x}}_k$ and covariance $\mathbf{P}_k$.

The prediction step gives an estimate on a future state from the current state and the transition matrix.

$$\hat{\mathbf{x}}_{k+1}^{-} = \mathbf{A}\hat{\mathbf{x}}_k \tag{2.11a}$$
$$\hat{\mathbf{P}}_{k+1}^{-} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q} \tag{2.11b}$$

where $\mathbf{Q}$ is the covariance matrix of the process noise $\mathbf{w}_k$. The state estimate is here propagated to the next timestep and the uncertainty of the estimate is increased by the process noise.

The equations in this subsection are for a linear time-invariant system, but the same theory can be applied if $\mathbf{A}$, $\mathbf{C}$, $\mathbf{R}$ and $\mathbf{Q}$ are dependent on time k.

2.3.2 Extended Kalman filter

The linear Kalman filter assumes a linear system model, but in general real world systems are non-linear. One solution to this is to repeatedly linearize the system around the current estimate for the state and use the linearization for the updates. This is called the extended Kalman Filter (EKF). [11]

The EKF now assumes a general continuous nonlinear function for state transitions and the measurement function:

$$\dot{\mathbf{x}}(t) = \mathbf{f}_c(\mathbf{x}, t) + \mathbf{w}_c(t) \tag{2.12a}$$
$$\mathbf{y}_c(t) = \mathbf{h}_c(\mathbf{x}, t) + \mathbf{v}_c(t) \tag{2.12b}$$

where $\mathbf{x}(t)$ is the state, $\mathbf{f}_c(\mathbf{x}, t)$ is a nonlinear function describing the behavior of the system,, $\mathbf{w}_c(t)$ is white process noise, $\mathbf{y}_c(t)$ is the measurement, $\mathbf{h}_c(\mathbf{x}, t)$ is a nonlinear continuous function describing the measurement from the state and $\mathbf{v}_c(t)$ is white measurement noise. The subscript c denotes that it is continuous.

This continuous model is important to be able to differentiate it, but we will also use the discretized version:

$$\dot{\mathbf{x}}_{k+1} = \mathbf{f}(\mathbf{x}_k) + \mathbf{w}_k \tag{2.13a}$$
$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k \tag{2.13b}$$

For the derivation I refer the reader to other literature, e.g. [10] or [11] and only present the result here.

The update is similar to the linear version, except that we use the jacobian of the functions as the system matrices:

$$\mathbf{x}_k = \mathbf{x}_k^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k^-)) \tag{2.14a}$$

$$\mathbf{K}_k = \mathbf{P}_k^- \left(\frac{\partial \mathbf{h}_c}{\partial \mathbf{x}}(\mathbf{x}_k^-)\right)^T \left(\frac{\partial \mathbf{h}_c}{\partial \mathbf{x}}(\mathbf{x}_k^-)\mathbf{P}_k^- \left(\frac{\partial \mathbf{h}_c}{\partial \mathbf{x}}(\mathbf{x}_k^-)\right)^T + \mathbf{R}_k\right)^{-1} \tag{2.14b}$$

$$\mathbf{P}_k = \left(\mathbb{I} - \mathbf{K}_k \frac{\partial \mathbf{h}_c}{\partial \mathbf{x}}(\mathbf{x}_k^-)\right) \mathbf{P}_k^- \tag{2.14c}$$

$\mathbf{R}_k$ is assumed to be the measurement noise covariance matrix. As in the linear case it is assumed to be known and with known covariance structure, often gaussian. $\frac{\partial \mathbf{h}_c}{\partial \mathbf{x}}(\mathbf{x}_k^-)$ is the jacobian of the measurement function evaluated at $\mathbf{x}_k^-$. $\mathbf{K}_k$ is still the Kalman gain, but is now only valid around the linearization at $\mathbf{x}_k^-$. The superscript $^-$ still denotes the prior estimate and the states without the superscript is the posterior estimate.

The prediction step is now following the nonlinear function to propagate the state, but to propagate the covariance we need the linearization:

$$\mathbf{x}_{k+1}^- = \mathbf{f}(\mathbf{x}_k) \tag{2.15a}$$

$$\mathbf{P}_{k+1}^- = \frac{\partial \mathbf{f}_c}{\partial \mathbf{x}}(\mathbf{x}_k)\mathbf{P}_k \left(\frac{\partial \mathbf{f}_c}{\partial \mathbf{x}}(\mathbf{x}_k)\right)^T + \mathbf{Q}_k \tag{2.15b}$$

$\mathbf{Q}_k$ is assumed to be the covariance matrix for the process noise, $\frac{\partial \mathbf{f}_c}{\partial \mathbf{x}}(\mathbf{x}_k)$ is the jacobian of $\mathbf{f}$ and the rest is the same as above.

The EKF handles nonlinear system dynamics and measurements by linearizing the current estimate around the current state. This is accurate if the update frequency is high enough and the system state is accurately estimated.

2.3.3 Iterated extended Kalman filter

Due to linearization errors the EKF is less accurate when the estimate is poor. This may lead to a worse estimate and then divergence. The Iterated extended Kalman filter (IEKF) tries to mitigate this by iterative linearizing around the updated estimate in the update step. This makes the estimate more precise and thereby the filter more robust.

The linear kalman filter and the EKF have closed form solutions for the prediction and the update step, but the IEKF iterates in the update step. The initialization for the iterations is then

$$\mathbf{x}_{k,0} = \mathbf{x}_k^- \tag{2.16a}$$

$$\mathbf{P}_{k,0} = \mathbf{P}_k^- \tag{2.16b}$$

The iterations are then equivalent of doing the update step several times. I.e. the i'th iteration will become:

$$\mathbf{x}_{k,i+1} = \mathbf{x}_{k,i} + \mathbf{K}_{k,i}(\mathbf{y}_k - \mathbf{h}(\mathbf{x}_{k,i})) \tag{2.17a}$$

$$\mathbf{K}_{k,i} = \mathbf{P}_{k,i}^- \left(\frac{\partial \mathbf{h}_c}{\partial \mathbf{x}}(\mathbf{x}_{k,i})\right)^T \left(\frac{\partial \mathbf{h}_c}{\partial \mathbf{x}}(\mathbf{x}_{k,i})\mathbf{P}_{k,i}\left(\frac{\partial \mathbf{h}_c}{\partial \mathbf{x}}(\mathbf{x}_{k,i})\right)^T + \mathbf{R}_k\right)^{-1} \tag{2.17b}$$

$$\mathbf{P}_{k,i+1} = \left(\mathbb{I} - \mathbf{K}_{k,i}\frac{\partial \mathbf{h}_c}{\partial \mathbf{x}}(\mathbf{x}_{k,i})\right)\mathbf{P}_{k,i} \tag{2.17c}$$

When the change in the state between the iterations gets too small, the iterations stop and we set

$$\mathbf{x}_k = \mathbf{x}_{k,l} \tag{2.18a}$$

$$\mathbf{P}_k = \mathbf{P}_{k,l} \tag{2.18b}$$

if there were l iterations needed. When we define it this way, the prediction step is the same as with the standard EKF.

The IEKF linearizes the nonlinear dynamics closer to the true state, but the uncertainty is still propagated with linearized equations which may not capture the underlying uncertainty well.

2.4 Evaluation of VIO/TIO algorithms

Benchmarking visual navigation methods is a mature but complex topic with different metrics and which are benchmarking different qualities of the algorithms. This section distinguishes between quality measures and metrics. Quality measures are qualities which can be used to assess performance of the algorithm, e.g. accuracy and robustness. Metrics are a specific evaluation criteria which is comparable between different results, eg. ATE and RPE. Accuracy of the estimated trajectory is a common quality to assess since it relatively easily can be evaluated quantitatively. Robustness is an important quality of visual navigation methods, but is difficult to quantitatively measure. Still, it is nevertheless quite common to assess in some quantitative way or qualitatively.

2.4.1 Accuracy

Accuracy relates the estimated trajectory and the true trajectory. Higher accuracy means less deviation in the estimated trajectory from the true trajectory. It is theoretically possible to measure the accuracy of the estimated map, but due to the difficulty of collecting a ground truth for the map it is more common to measure the accuracy for the trajectory.

2.4.1.1 ATE: Absolute trajectory error

ATE is a common accuracy metric for evaluating the overall trajectory against ground truth. First the trajectories are aligned since the ground truth trajectory and the estimated trajectory in general are referenced in different coordinate systems. According to [12] the alignment is calculated with the method from [13], which is closed form and minimizes the least squares difference between the matched points of the trajectories. Using the formulation from [12] the ATE can be formally written as:

$$\mathbf{F}_i = \mathbf{Q}_i^{-1}\mathbf{S}\mathbf{P}_i \tag{2.19a}$$

$$ATE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} ||\text{trans}(\mathbf{F}_i)||^2} \tag{2.19b}$$

where $S \in SE(3)$ is the transformation which is aligning the estimated trajectory $\mathbf{P}_1, ..., \mathbf{P}_N \in SE(3)$ and the ground truth trajectory $\mathbf{Q}_1, ..., \mathbf{Q}_N \in SE(3)$. trans(.) is a function extracting translational position in $\mathbb{R}^3$ from the pose. Notice that formulation calculates the mean of the translation. It is possible to formulate the same for the rotation, but this is more common. Rotational errors in dead reckoning methods are expected to create errors in the positions in the following frames; however this is not expected the other way around.

2.4.1.2 RPE: Relative pose error

When using ATE to evaluate dead reckoning methods (E.g. VIO/TIO) the errors made in the beginning of the trajectory will be influencing the error metric more than those late in the trajectory since all future poses will be referenced to the erroneous early poses. Relative pose error(RPE) tries to reduce this impact by only measuring the additional error over a given interval. This means that the RPE is dependent on the choice of this interval, and that RPEs calculated over different intervals cannot be directly compared.

With the same notation as for ATE the RPE can be formally defined as [12]:

$$\mathbf{E}_i = \left(\mathbf{Q}_i^{-1}\mathbf{Q}_{i+\Delta}\right)^{-1}\left(\mathbf{P}_i^{-1}\mathbf{P}_{i+\Delta}\right) \tag{2.20a}$$

$$RPE(\Delta) = \sqrt{\frac{1}{N-\Delta}\sum_{i=1}^{N-\Delta} \|\mathrm{trans}(\mathbf{E}_i)\|^2} \tag{2.20b}$$

where $\mathbf{P}_i$ and $\mathbf{Q}_i$ are estimated pose and ground truth as in the previous paragraph. Δ is the interval RPE is estimated over expressed in number of frames.

2.4.2 Robustness

Robustness is a quality measure which assesses the system's ability to provide odometry under increasingly difficult conditions, here called degragations. Imaging odometry systems are prone to settings which make odometry more difficult, which means that there are different types of robustness. I.e. some methods can be robust to some degreagations, but susceptible to others.

Here follows a list of common degregations for visual navigation systems.

Low light Low light typically reduces the number of features that could be used for tracking and increases the signal to noise ratio(SNR). The thermal equivalent is typically not that the radiation is too low to excite the sensor, but that the whole scene has the same temperature and emissivity which gives the image a single uniform value.

Obscurants Obscurants is a collective term for non stationary gasses or particles that are visible in the image instead of the scene behind. Examples include smoke, dust and fog, which are all obscurants, but have different effects on visual navigation algorithms. Some, but not all obscurants are transparent in the IR spectrum, but not in the visual spectrum.

Repetitive scenes Repetitive textures in a scene will be problematic for algorithms that rely on pattern matching. In addition, repetitive environments will be problematic for loop closures.

Fast motions Some types of algorithms will struggle with fast motions since fast motions require the algorithm to match features over a larger distance.

Degenerate motions If a camera is doing pure rotations, i.e. only rotating, not translating, it is impossible to estimate the depth in the scene. It is still possible to estimate the rotation, but it differs to which extent algorithms make use of this.

Degenerate scenes Different algorithms may make different assumptions about the scene structure, which may or may not be fulfilled. E.g. the 8-point algorithm [8] assumes that the points do not lie in a plane. This means that algorithms that rely on the 8-point algorithm or the similar 5-point algorithm [14] will struggle in planar scenes.

Not all of these are equally relevant for any given scenario. This thesis will focus on improving robustness to low light and flat thermal by combining both visual and thermal data in the same odometry algorithm.

2.4.3 Integrity

Integrity is a quality measure measuring the system's ability to warn the user that it is no longer providing accurate odometry. I.e. a system with high integrity will be able to detect it so the user can act accordingly. It is not as common to assess the integrity in VIO/TIO literature as with robustness and this thesis will not focus on it either.

2.4.4 Continuity

Continuity assesses how long time the system provides between it warning the user about a failure and the failure is too critical to provide odometry. For visual inertial navigation systems this can often be the duration which the IMU readings are accurate after loss of visual tracking. This can in some systems be important to know the length of critical maneuvers where loss of odometry would be unacceptable. This is however not a common focus in the MAV visual/thermal odometry literature.

2.4.5 Latency

Latency is a quality measure which measures the delay from the algorithm receiving a sensor reading to it outputting the related localization. Alternatively it can measure the delay from the sensor capture until it is processed. This is highly relevant in systems where the localization from the navigation system is used in feedback control.

This can be measured as a duration in seconds, and is a quite common metric in visual (inertial) navigation literature.

Variations in latency are also very relevant in addition to the mean or median value.

2.5 Notation

This thesis will mainly use the notation from [1] since it is a central reference for this thesis.

A point i a 3D coordinate system $\mathcal{D}$ is given by

$$_{\mathcal{D}}\mathbf{r} \in \mathbb{R}^3 \tag{2.21}$$

2.5.1 Rotations

As in [1] a rotation $\mathbf{q}_{\mathcal{GD}}$ is defined as an abstract function which is applied to a vector such that

$$ {}_{\mathcal{G}}\mathbf{r} = \mathbf{q}_{\mathcal{GD}}({}_{\mathcal{D}}\mathbf{r}) \tag{2.22}$$

if the frame $\mathcal{G}$ has the same origin as $\mathcal{D}$. The rotation can per definition not scale the length of vectors.

If we define $\mathbf{C}(.)$ to be the rotation matrix of a rotation such that

$$\mathbf{C} : SO(3) \rightarrow \mathbb{R}^{3\times 3} \tag{2.23}$$

then the application of a rotation is equivalent to

$$ {}_{\mathcal{G}}\mathbf{r} = \mathbf{C}(\mathbf{q}_{\mathcal{GD}})_{\mathcal{D}}\mathbf{r} \tag{2.24}$$

Note that [1] does not use rotation matrices to represent rotations, but unit quaternions.

Nevertheless, compositions of rotations can be defined by rotation matrices. The matrices in $SO(3)$ have a unit determinant and have unit length columns. Compostions of rotations is equivalent to

$$\mathbf{C}(\mathbf{q}_{\mathcal{GD}}\mathbf{q}_{\mathcal{DA}}) = \mathbf{C}(\mathbf{q}_{\mathcal{GD}})\mathbf{C}(\mathbf{q}_{\mathcal{DA}}) \tag{2.25}$$

The rotations are elements in the lie group $SO(3)$, which implicates the corresponding lie algebra $so(3)$. $so(3)$ is isomorphic to $\mathbb{R}^3$ and [1] uses this to define the exponential and logarithmic maps directly to the $\mathbb{R}^3$ representation:

$$exp : \mathbb{R}^3 \rightarrow SO(3) \tag{2.26a}$$
$$log : SO(3) \rightarrow \mathbb{R}^3 \tag{2.26b}$$
$$exp(\boldsymbol{\theta}_{\mathcal{RD}}) = \mathbf{q}_{\mathcal{RD}} \tag{2.26c}$$
$$log(\mathbf{q}_{\mathcal{RD}}) = \boldsymbol{\theta}_{\mathcal{RD}} \tag{2.26d}$$

Note that this is not the normal exponential map applied to a rotation matrix in $SO(3)$, but defined as a function that first does the matrix exponential and then extracts the vector from the skew symetric matrix.

[1] uses the lie algebra of $SO(3)$, $so(3)$, to define infinitesimal rotations and change in rotations.

To express operations on manifolds using the notation from vector spaces, [1] define the $\boxplus$ and $\boxminus$ operators.

The boxplus operator for $SO(3)$ is defined as

$$\boxplus : SO(3) \times \mathbb{R}^3 \rightarrow SO(3) \tag{2.27a}$$
$$\mathbf{q} \boxplus \boldsymbol{theta} = exp(\theta)\mathbf{q} \tag{2.27b}$$

and the boxminus operator is

$$\boxminus : SO(3) \times SO(3) \rightarrow \mathbb{R}^3 \tag{2.28a}$$
$$\mathbf{q}_1 \boxminus \mathbf{q}_2 = log(\mathbf{q}_1(\mathbf{q}_2)^{-1} \tag{2.28b}$$

2.5.2 Transformations

The notation in [1] does not define homogeneous transformations, but the convention used in this thesis is defined here. Same as with rotations in $SO(3)$ transformations are defined as abstract functions that can be applied to vectors.

$$_{\mathcal{G}}\mathbf{r} = {}_{\mathcal{G}}\mathbf{T}_{\mathcal{GD}}({}_{\mathcal{D}}\mathbf{r}) \tag{2.29}$$

The transformation $\mathbf{T} \in SE(3)$, or pose, consists of two parts one rotation $\mathbf{q} \in SO(3)$ and one translation $\mathbf{p} \in \mathbb{R}^3$.

$$_{\mathcal{G}}\mathbf{T}_{\mathcal{GD}} = (\mathbf{q}_{\mathcal{GD}}, {}_{\mathcal{G}}\mathbf{p}_{\mathcal{GD}}) \tag{2.30}$$

A transformation is equivalent to a homogeneous transformation matrix, similarly to rotations and rotation matrices:

$$\mathbf{C} : SE(3) \rightarrow \mathbb{R}^{4\times4} {}_{\mathcal{G}}\mathbf{T}_{\mathcal{GD}} = (\mathbf{q}_{\mathcal{GD}}, {}_{\mathcal{G}}\mathbf{p}_{\mathcal{GD}}) \tag{2.31a}$$
$$\mathbf{C}({}_{\mathcal{G}}\mathbf{T}_{\mathcal{GD}}) = \begin{bmatrix} \mathbf{C}(\mathbf{q}_{\mathcal{GD}}) & {}_{\mathcal{G}}\mathbf{p}_{\mathcal{GD}} \\ 0,0,0 & 1 \end{bmatrix} \tag{2.31b}$$

Similarly to the rotations, the homogeneous transformation matrix can be used to define the composition of transformations.

$$\mathbf{C}({}_{\mathcal{G}}\mathbf{T}_{\mathcal{GD}\mathcal{D}}\mathbf{T}_{\mathcal{DA}}) = \mathbf{C}({}_{\mathcal{G}}\mathbf{T}_{\mathcal{GD}})\mathbf{C}({}_{\mathcal{D}}\mathbf{T}_{\mathcal{DA}}) \tag{2.32}$$

Also the lie group $SE(3)$ has a lie algebra $se(3)$. $se(3)$ is isomorphic to $\mathbb{R}^6$, so we define the exponential and logaritmic maps directly from the lie group to the vector representation.

$$exp : \mathbb{R}^6 \rightarrow SE(3) \tag{2.33a}$$
$$log : SE(3) \rightarrow \mathbb{R}^6 \tag{2.33b}$$
$$exp([_{\mathcal{G}}\mathbf{r}_{\mathcal{G}\mathcal{D}}, \boldsymbol{\theta}_{\mathcal{R}\mathcal{D}}]^T) = {_{\mathcal{G}}\mathbf{T}_{\mathcal{G}\mathcal{D}}} \tag{2.33c}$$
$$log({_{\mathcal{G}}\mathbf{T}_{\mathcal{G}\mathcal{D}}}) = [_{\mathcal{G}}\mathbf{r}_{\mathcal{G}\mathcal{D}}, \boldsymbol{\theta}_{\mathcal{R}\mathcal{D}}]^T \tag{2.33d}$$

Also for $SE(3)$ can we define the $\boxplus$ and $\boxminus$ operators.
The boxplus operator for $SE(3)$ is defined as

$$\boxplus : SE(3) \times \mathbb{R}^6 \rightarrow SE(3) \tag{2.34a}$$
$$\mathbf{T} \boxplus [\mathbf{r}, \boldsymbol{theta}]^T = exp([\mathbf{r}, \boldsymbol{theta}]^T)\mathbf{T} \tag{2.34b}$$

and the boxminus operator is

$$\boxminus : SE(3) \times SE(3) \rightarrow \mathbb{R}^6 \tag{2.35a}$$
$$\mathbf{T}_1 \boxminus \mathbf{T}_2 = log(\mathbf{T}_1(\mathbf{T}_2)^{-1} \tag{2.35b}$$

2.5.3 Reference frames

This thesis uses the same conventions for the reference frames as [1].

Table 2.1: Reference frame subscripts

Symbol	Frame
$\mathcal{G}$	World frame, equivalent of NED, assumed to be inertial
$\mathcal{B}$	IMU frame
$\mathcal{C}$	camera frame
$\mathcal{D}$	Arbitrary frame
$\mathcal{A}$	Current frame, most recent frame
$\mathcal{R}$	Reference frame

Chapter 3

Related work

There is work from several related fields which is relevant for this thesis. Most notable is the work in Visual Odometry(VO)/Visual Simulationous Localisation and mappping(VSLAM), Visual Inertial Odometry(VIO)/Visual Inertilal SLAM(VISLAM), Thermal inertial odomerty(TIO) and Visual Thermal Odometry(VTO).

3.1 Estimation formulation

There are two common common ways of formulating the problem of pose estimation from sequential images: Direct and indirect. The authors of DSO [15] define the difference by whether the method has an intermediate step between the raw sensor data and the estimator. Direct methods minimize the difference between the expected and observed photometric pixel values. Indirect methods have a preceding processing step which tracks keypoints and then minimizes the reprojection error of these keypoints.

3.1.1 Direct methods

The definition from [15] states that direct methods use raw sensor values directly in the state estimator. For cameras and visual navigation this means that the position of the camera is estimated from the pixel intensities. In practice that typically means that they minimize the photometric error:

$$E_{photo} = I_{ref}(\mathbf{u}) - I_c(\pi({}_{C}T_{C\mathcal{R}}(\pi^{-1}(\mathbf{u}, d)))) \tag{3.1}$$

where E_{photo} is the photometric error, $I_{ref}(\mathbf{v})$ is the pixel intensity at pixel $\mathbf{v} \in \mathbb{R}^2$ for the reference image and $I_c(\mathbf{v})$ is the same for the current image. $\pi(.) : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ is the projection function and $\pi^{-1}(.) : (\mathbb{R}^2, \mathbb{R}) \rightarrow \mathbb{R}^3$ is the unprojection function with known depth. ${}_{C}T_{C\mathcal{R}}(.) \in SE(3)$ transforms a 3D feature point from the reference frames coordinate system to the current frame. It is common to include uncertainties in this error, but these are left out here for simplicity.

There are various ways of formulating this photometric error. Some formulations include extra brightness parameters to account for global changes in illumination, some account for other warping models with skew or they can represent the reference intensity differently than a location in a full image. The only thing that matters for it to be direct is that the state of the camera, ${}_{C}T_{C\mathcal{R}}$, is estimated from this error metric.

3.1.2 Indirect methods

According to [15] indirect methods are characterized by doing an intermediate step with the raw sensor data and then using the results from that to estimate the state. The typical way of doing this is by first establishing correspondences between keypoints in the images and then using these correspondences to estimate the state. The first step is often KLT-tracking[6] [16] or ORB-matching[17], but it can be anything that gives a set of correspondences between the frames. Furthermore, it does not need to be keypoints, it can be line segments, curves or other trackable features.

These correspondences are then used to formulate a probabilistic estimation problem to estimate the state. The typical error that is then minimized in the estimation for keypoints is then the reprojection error:

$$E_{reprojection} = \mathbf{u}_{ref} - \pi({}_{C}T_{C\mathcal{R}}(\pi^{-1}(\mathbf{u}_c, d))) \tag{3.2}$$

where $E_{reprojection}$ is the reprojection error, $\mathbf{u}_{ref} \in \mathbb{R}^2$ is the location of the feature in the reference image, $\mathbf{u}_c \in \mathbb{R}^2$ is the corresponding feature in the current image, ${}_{C}T_{C\mathcal{R}}$ is the relative transformation between the cameras that took the two images and $\pi(.) : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ is the projection function. Also here it is common to include uncertainties, but again these are left out for simplicity.

Even if the raw photometric values from the image are used to establish the correspondences, the method is indirect if the camera transformations ${}_{C}T_{C\mathcal{R}}$ are estimated from the correspondences alone.

3.1.3 Semi direct methods

Several methods exist that use both or are difficult to classify as either direct or indirect. There is less consensus about what makes a method semi-direct, but the SVO[18] type of semi direct methods first aligns the camera to the map using direct alignment, then updates the map (and the camera pose again with the map) using an indirect formulation.

3.2 Estimation horizon

Filter-based methods maintain an estimate over the state with the corresponding uncertainty of the state. An alternative, graph-based methods, to filter based methods is to store the measurements in a graph and infer the state at each step from this graph. I.e. the filter-based methods estimate the next state based only on the previous state estimate and the most recent measurement:

$$\hat{\mathbf{x}}_k = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k) \tag{3.3}$$

where $\hat{\mathbf{x}}_k$ is the state estimate at time k, $\mathbf{u}_k$ is the measurement at time k and $\mathbf{f}(.)$ is the estimation function.

The contrast is then to graph-based methods that do not use the previous state, but only earlier measurements:

$$\hat{\mathbf{x}}_k = \mathbf{f}(\mathbf{u}_k, \mathbf{u}_{k-1}, ..., \mathbf{u}_0) \tag{3.4}$$

where $\hat{\mathbf{x}}_k$ is the estimate at time k, $\mathbf{u}_k$ is the measurement at time k and $\mathbf{f}(.)$ is the estimation function.

Graph-based implementations tend to be more computationally demanding than filter-based methods. Graph-based methods solve a nonlinear optimization problem which in general is more demanding than the filters. The underlying graph is in general very sparse, so the optimization problem can be solved more efficiently. [19] Filter-based methods can also be based on nonlinear optimization, but still tend to be less computationally expensive due to the smaller state.

Most graph-based methods add only a selection of the captured images to the graph to keep the size of the graph bounded and to allow the estimation to run at a lower frequency than the frame rate. The selected frames are called keyframes. This is also favorable because triangulating points in the environment are more precise with larger distance between the camera positions at the time of capture. The disadvantage of increasing distance between the frame poses is that it becomes more difficult to make the associations between the frames. This means that the keyframe selection criteria is a tradeoff between ensuring sufficient distance between the frames to have well defined triangulated points, ensuring sufficiently small distance to robustly make associations between the frames and the runtime.

The middle ground between graph optimizing the whole trajectory from the beginning to the current state and filtering approaches is smoothing approaches. They maintain a limited size graph, typically of the N most recent keyframes, to be able to use the benefits of graph based methods while keeping the runtime bounded. In table 3.1 are grouped with the graph based methods, but examples include DSO[15] and OKVIS[19]. They maintain a sliding window over recent, representative keyframes and align new frames to the map made from these keyframes. It is common to marginalize out the old keyframes that are removed from the map, but this comes at the cost of reducing the

Table 3.1: Visual navigation methods by common classifications. Underlined methods also use an IMU or are capable of using an IMU to improve robustness and estimate scale. With the exception of DTAM and KTIO all methods listed here are open source.

		Filter based methods	Graph based methods
Direct methods	Dense/Semi-Dense	DVO [24]	LSD-SLAM [25] DTAM [26]
	Sparse	ROVIO [1]	DSO [15] DSM [27] LDSO [28]
Semi-direct methods	Direct tracking and keypoint based mapping		SVO [18] LCSD-SLAM [29] KTIO [30]
Indirect methods	Descriptor based	MSCKF [31]	OKVIS [19] ORBSLAM [17]
	KLT-based	RVIO[32] LARVIO [33]	PTAM [34] Basalt [6] VINS-fusion [35] Kimera [16]

sparsity that was a benefit of the graph based methods.

Other graph based variants is the incremental variants which only updates a subpart of the graph when a new measurement is added. This saves runtime and increases accuracy compared with the standard marginalization. Examples include [20], [21] and [22].

[23] analyzed the difference in performance can computational cost and concluded that graph-based methods were favorable over EKF-based filtering methods for visual slam.

Table 3.1 tries to show the common variants and give some examples.

3.3 Duration of associations

Different visual navigation methods can to various extents recognize earlier visited locations to compensate for drift. All common methods can associate new images to the most recent preceding images, but not all "remember" the historical images independent of their age. Using the definition from [17] we can classify the different forms of classifications into short term associations, mid term associations and long term associations.

Short term associations are correspondences between the most recent image and the images from the last few seconds. It is then fair to assume small drift and almost the same viewpoint.

Mid term associations are correspondences between the most recent image and images of any age as long as the tracked features have not drifted too far. How far a feature has to drift varies from the implementations, but the point is that an inaccurate guess for the location of the feature should be sufficient to make a useful association. This differs from the short term associations, which also assumed small drift, in that mid term associations can associate features of arbitrary age, while short term associations are limited to associations over a short time interval.

Long term associations are correspondences between images of arbitrary age with arbitrary drift. They are also called loop closures and rely only on visual place recognition.

Methods that rely only on short term associations are called visual odometry (VO) methods and methods that use short term associations and either both or one of the other are called visual SLAM (VSLAM) methods.

VO methods, using only short term associations, are dead reckoning methods meaning that the estimate will drift with time. The other two types of associations can to some degree reduce this drift. Long term associations can recover much more drift than mid term associations, but since they are purely based on recognizing earlier visited locations by their appearance, they are prone to false associations in repetitive environments. Since false loop closures would severely impact the quality of the map it is quite common to impose additional checks before a loop closure candidate is accepted, which improves precision at the cost of recall.

Table 3.2 displays which types of associations a selection of methods use. It is only the ORBSLAM[17] variants and LCSD-SLAM, which are built on ORBSLAM2[36] which use both mid term associations and long term associations in the same method. DSO [15] was developed into a VSLAM system separately in DSM [27], which added mid term associations, and LDSO [28] which added long term associations.

A type of association which could have been included as a separate column in table 3.2 is relocalization. That allows a VSLAM method to refind the map it earlier tracked, but lost track of. This requires a visual place recognition functionality similar to long term associations. This similarity is exploited by e.g. ORBSLAM[17] and VINS-Fusion [35] which use the same visual place recognition scheme for loop closure and relocalization. Not all methods with loop closure implement relocalization, and some methods, eg. PTAM, implement relocalization, but not long term associations.

Table 3.2: Table over used which types of associations the methods from table 3.1 use. An x indicates that the method in this row uses the type of association in this column.

	Short term associations	Mid term associations	Long term associations
DVO [24]	x		
LSD-SLAM [25]	x		x
DTAM [26]	x		
ROVIO [1]	x		
DSO [15]	x		
DSM [27]	x	x	
LDSO [28]	x		x
SVO [18]	x		
LCSD-SLAM [29]	x	x	x
KTIO [30]	x		
MSCKF [31]	x		
OKVIS [19]	x		
ORBSLAM [17]	x	x	x
LARVIO [33]	x		
PTAM [34]	x	x	
Basalt [6]	x		x
VINS-fusion [35]	x		x
Kimera [16]	x		x

Chapter 4

ROVIO : RObust Visual Inertial Odometry

ROVIO [37], [1] is a visual odometry estimation system designed for indoor use with focus on creating a robust visual inertial odometry estimator. The ROVIO framework is used as a starting point for the method developed in this project and significant amounts of the original ROVIO functionality are retained. Therefore this chapter will introduce and describe the original(multi modal changes) ROVIO.

4.1 Overview

ROVIO is a direct robocentric filtering based visual inertial odometry estimator based on an Iterated Extended Kalman filter. For each new image it estimates the new state by maximizing the maximum a posterori(MAP) estimate with nonlinear optimization. The prior for the estimate is obtained by integrating the motion obtained by the IMU from the previous state. Figure 4.1 shows a high level diagram of the dataflow through ROVIO.

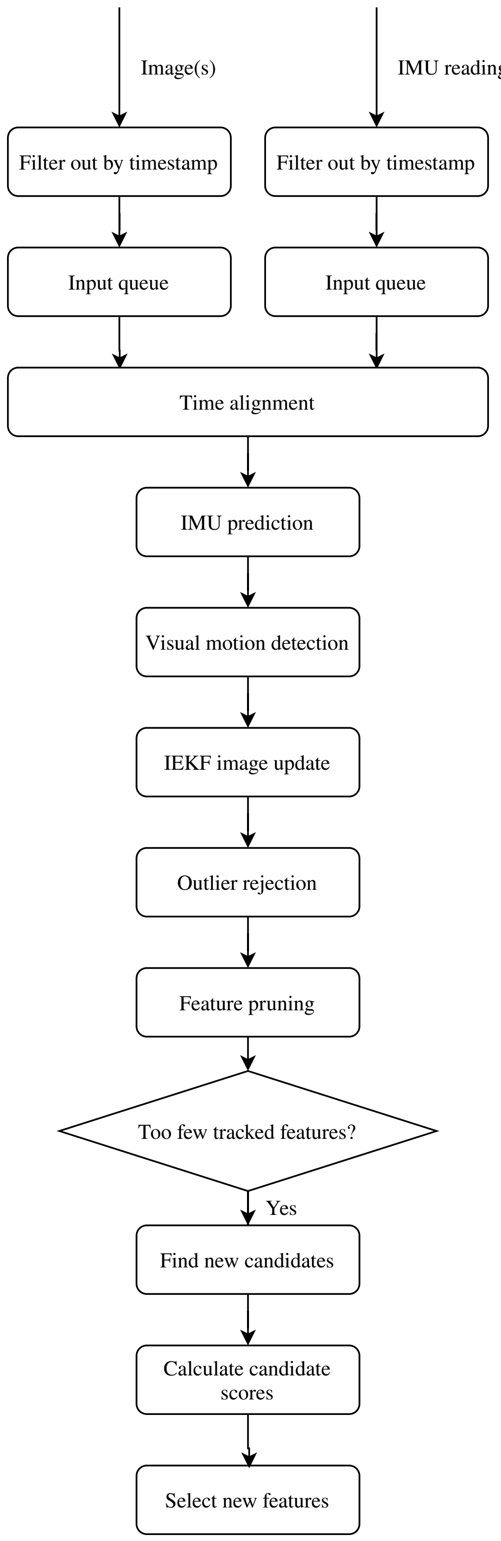


Figure 4.1: An overview showing the dataflow of ROVIO.

4.2 State

The rovio filter state consists of these subparts

$$\mathbf{x} = \begin{bmatrix} {}_{\mathcal{B}}\mathbf{r}_{\mathcal{GB}} \in \mathbb{R}^3 \\ {}_{\mathcal{B}}\mathbf{v}_{\mathcal{B}} \in \mathbb{R}^3 \\ \mathbf{q}_{\mathcal{GB}} \in SO(3) \\ {}_{\mathcal{B}}\mathbf{b}_{f} \in \mathbb{R}^3 \\ {}_{\mathcal{B}}\mathbf{b}^{i}_{\omega} \in \mathbb{R}^3 \\ {}_{\mathcal{B}}\mathbf{b}_{\mathcal{BC}} \in \mathbb{R}^3 \\ \mathbf{q}_{\mathcal{CB}} \in SO(3) \\ {}_{\mathcal{C}}\boldsymbol{\mu}_1 \in S^2 \\ \vdots \\ {}_{\mathcal{C}}\boldsymbol{\mu}_N \in S^2 \\ \rho_1 \in \mathbb{R} \\ \vdots \\ \rho_N \in \mathbb{R} \end{bmatrix} = \begin{bmatrix} \text{The world position of the IMU relative to the starting point.} \\ \text{The velocity of the IMU expressed in IMU coordinates.} \\ \text{The attitude of the IMU stored as a quaternion.} \\ \text{The bias of the accelerometer in the IMU-frame.} \\ \text{The bias of the ARS expressed in the IMU-frame.} \\ \text{The position of the camera relative to the IMU in IMU coordinates.} \\ \text{The rotation from the camera to the IMU stored as a quaternion.} \\ \\ \text{The bearing vectors for the } N \text{ tracked features.} \\ \\ \\ \text{The distances to the } N \text{ tracked features.} \\ \\ \end{bmatrix} \tag{4.1}$$

The number of parameters for the attitudes is 4, since they are stored as quaternions and the number of parameters for the bearing vectors is 3 due to the singular free bearing vector representation ROVIO use. This makes the total size of the filter

$$3+3+4+3+3+3+4+3N+N = 23+4N \tag{4.2}$$

with N tracked features.

The pose of the camera relative to the IMU can be frozen to a precalibrated value if known, reducing the filter size by 7.

4.3 Feature management

4.3.1 Feature score

ROVIO [1] assigns a score for how easy each feature is to track based on the texture around the feature. It is based on the shi-tomasi score, but adapted to consider the patch at several image pyramid levels.

It calculates an approximation of the hessian of a patch by the sum of the jacobians in that patch.

The score is based on the eigenvalues of the matrix defined in eq. 4.3 which contains the sum of the jacobians in that patch.

$$\mathbf{H}_l = \sum_{\mathbf{p} \in \text{Patch}} \begin{bmatrix} \left(\frac{\partial I_l}{\partial x}(\mathbf{p})\right)^2 & \left(\frac{\partial I_l}{\partial x}(\mathbf{p})\frac{\partial I_l}{\partial y}(\mathbf{p})\right)^2 \\ \left(\frac{\partial I_l}{\partial y}(\mathbf{p})\frac{\partial I}{\partial x}(\mathbf{p})\right)^2 & \left(\frac{\partial I}{\partial y}(\mathbf{p})\right)^2 \end{bmatrix} \tag{4.3}$$

where I_l is the image at pyramid level l and $\mathbf{p}$ is all the pixel locations in the patch. Notice that this is not the real hessian with the double derivatives, but an approximation.

The resulting approximated hessian for each patch is a weighted sum of the hessian at all the relevant pyramid levels:

$$\mathbf{H} = \sum_{l=\text{min level}}^{\text{max level}} 0.25^l \mathbf{H}_l \tag{4.4}$$

The resulting score is then calculated from the eigenvalues as:

$$\lambda_1 = \mathbf{H}[0,0] + \mathbf{H}[1,1] + \sqrt{(\mathbf{H}[0,0] + \mathbf{H}[1,1])^2 - 4 * (\mathbf{H}[0,0] * \mathbf{H}[1,1] - \mathbf{H}[0,1]^2)} \tag{4.5a}$$

$$\lambda_2 = \mathbf{H}[0,0] + \mathbf{H}[1,1] - \sqrt{(\mathbf{H}[0,0] + \mathbf{H}[1,1])^2 - 4 * (\mathbf{H}[0,0] * \mathbf{H}[1,1] - \mathbf{H}[0,1]^2)} \tag{4.5b}$$

$$s = \frac{\lambda_1 + \lambda_2}{2} \tag{4.5c}$$

where λ_1 and λ_2 are the eigenvalues, s is the patch score and $\mathbf{H}[i,j]$ denotes the element of $\mathbf{H}$ at row i and column j.

This score is used both in the feature selection and in the outlier rejection.

4.3.2 Feature selection

ROVIO[1] selects features in two steps: first it finds a set of candidates using the OpenCV implementation of the FAST[38] feature detector. Then these candidates are evaluated by the score described in sec. 4.3.1. Then ROVIO selects those M candidates with the highest score, where M is the needed new features.

4.3.2.1 Ensuring spatial distribution

To ensure that the features are distributed throughout the image ROVIO does not only use the features with the highest score, but also considers the distance to existing features. First ROVIO creates buckets based on the score of all the features. The range of the buckets is defined from the maximum score and the minimum score of all the candidates. The exact order of the candidates that end up in the same buckets is not considered, so candidates with a similar score have the same priority in the feature selection. Then it loops over all the tracked features and moves each candidate down to a lower bucket if it is close. How many buckets the candidate is moved down depends on the distance between the candidate and the feature. With the default settings candidates at the exact same location as a feature are moved down to the lowest bucket. Then features are selected by selecting randomly from the top non-empty bucket and initialized. Each time a candidate is selected as a feature all the other candidates are revisited and moved down if it is close. This is repeated until there are no more features to be found or the maximum number of features is reached.

4.3.2.2 Insertion into the filter

New features are immediately inserted into the filter with the median scene depth and a high variance. If the median depth is unknown, they are inserted at a preset depth. This allows features to be used for tracking from the very beginning, but makes the filter vulnerable in scenes with large variations in depth. The variation in depth is often limited in indoor scenes, but in outdoor scenes this might be problematic.

4.3.3 Feature tracking

Since ROVIO[1] is a direct method, features are only implicitly tracked as a part of the filter itself. The features are represented relative to the camera pose and only updated when the camera pose is updated. This means that they are not tracked individually, but only as a part of the state of the robot as a whole. During the IMU based prediction step the depth and feature locations are moved according to the estimated motion and added uncertainty. During the the update step the photometric error of the patches are used in the MAP update.

4.3.4 Outlier detection

ROVIO [1] employs three methods for outlier rejection: One Mhalanobis based rejection on the innovation residual, a threshold on the photometric error and a quality check by comparing the photometric error on nearby sampled locations.

Mahalanobis distance is a statistical measure for deviation from the mean of a distribution, corrected for the variance.

Photometric error threshold simply removes tracked features where the total photometric error from the prediction is over a fixed threshold. This has the unwanted side effect that features with large variance are more easily discarded.

Cornerness check is a condition placed on each feature that nearby locations needs to make the photometric error increase significantly in at least two of four directions. The nearby locations is by default a fixed distance up, down, left and right.

The Mahalanobis distance is similar to the eucledian distance metric, but weighted by the covariance:

$$d(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T \Sigma^{-1} (\mathbf{x}-\mathbf{y})} \tag{4.6}$$

where $d(.)$ is the Mahalanobis distance, $\mathbf{x}$ is the predicted innovation, $\mathbf{y}$ is the obtained innovation and Σ is the covariance of the predicted innovation. This is the standard way of defining Mahalanobis distance, but in the code it seems like the square root is left out. This is equivalent to the standard version if the threshold is also squared.

The cornerness check samples four nearby points around the feature and evaluates the residual at those points. If the residual is similar to the features residual, the tracker will struggle to differentiate between them. I.e. the feature is difficult to track since the neighborhood of the feature looks similar to the feature itself.

4.3.5 Feature pruning

That a feature is rejected by the outlier criteria in sec. 4.3.4, failed to converge or is out of frame does not necessarily mean it is discarded immediately. ROVIO [1] considers the features performance over time to decide if it should be pruned. ROVIO uses 3 statistical measures for removing features:

Global quality The ratio of tracking success of this feature.

Local quality The ratio of tracking success for this feature in the N most recent frames.

Local visibility The ratio the feature appearing in the frame in the last M frames.

N and M are here constants which are set at compile time and are in general not the same.

The global quality is calculated straight forward as

$$\text{Global quality} = \frac{\text{Tracked count}}{\text{Total count since feature initialization}} \tag{4.7}$$

The local qualities are approximated by a filtering approach instead of a sliding window. For each new frame:

$$\text{Local quality} = \text{Local quality} * \left(1 - \frac{1}{N}\right) + \text{Tracked} * \frac{1}{N} \tag{4.8}$$

where "Tracked" is 1 if the feature is successfully tracked this frame and 0 elsewise.

Similarly the local visibility is calculated for each new frame as:

$$\text{Local visibility} = \text{Local visibility} * \left(1 - \frac{1}{M}\right) + \text{Visible} * \frac{1}{M} \tag{4.9}$$

where "Visible" is 1 if the feature is the camera's FoV and 0 elsewhere.

These statistics are then combined into a single criteria for whether the the feature should be discarded:

$$\text{Local quality} * \text{Local visibility} > U - (U - L) * \text{Global quality} \tag{4.10}$$

If this is true, then the feature is kept. U and L are constants relating the global quality to the other two. It is designed this way to be more permissive in the local metrics if the overall global quality is good.

ROVIO does not only remove features to get rid of bad features. To keep the runtime bounded there is a fixed limit for how many features which can tracked at the same time. To make space for new features, which is needed when the camera moves to new environments, ROVIO therefore removes features that failed tracking based on the above criteria if the total number of tracked features is high.

4.4 Update step

The measurement ROVIO [1] uses for the update is a direct update from the photometric values in the images. I.e. the residual from eq. 2.17a , $(\mathbf{y}_k - \mathbf{h}(\mathbf{x}_{k,i}))$, is the photometric distance between the values reprojected from a previous image and the pixel intensities of the current image.

[39] showed that the IEKF update step is equivalent to a recursive gauss-newton maximum likelihood estimate optimization. I.e. the update step from sec. 2.3.3 is equivalent to optimizing

$$\arg\min_{\mathbf{x}_k} \|\mathbf{x}_k - \mathbf{x}_k^-\|_{(\mathbf{P}_k^-)^{-1}} + \|\mathbf{y}_k - \mathbf{h}(\mathbf{x}_{k,i})\|_{\mathbf{R}_k} \tag{4.11}$$

Notice that we can use the posterior estimates for the states and covariances in the cost function since they are updated every step in the iteration. The left part includes the known information about the state from the prediction step and the right part includes the measurement.

The innovation term, $\mathbf{y}_k - \mathbf{h}(\mathbf{x}_{k,i})$ of ROVIO formulated as the photometric difference between each reference patch and the current image:

$$e(\mathbf{p}, {}_{\mathcal{R}}I, {}_{\mathcal{A}}I, \mathbf{D}) = \sum_{l \in \{l_{min},\dots,l_{max}\}} \sum_{p_j \in \text{patch}} {}_{\mathcal{R}}I_l(\mathbf{p}_j) - a\,{}_{\mathcal{A}}I_l(\mathbf{p}s_l + \mathbf{D}\mathbf{p}_j) - b \tag{4.12}$$

where

$e_{l,j}(.)$ is the error for the patch pixel j at image pyramid level l].

$\mathbf{p}$ is the point in the image ${}_{\mathcal{A}}$ which corresponds to the patch location.

${}_{\mathcal{R}}I_l$ is the reference patch at pyramid level l. I.e. the extracted image pyramid where each level is $6x6$ pixels.

${}_{\mathcal{A}}I_l$ is the current image pyramid at level l. This is not only a patch, but all pixels in the whole image.

$\mathbf{D}$ is the linear warping matrix.

$\{l_{min}, ..., l_{max}\}$ are the image pyramid levels used. The default is $l_{min} = 1$ and $l_{max} = 2$.

$\mathbf{p}_j$ is the patch pixel locations. Together they make up the whole 6x6 patch.

a and b are affine brightness parameters to account for change in exposure or illumination between the frames. They are estimated each frame, but not a part of the state.

s_l is the scale factor to account for the fewer pixels in the subsampled images on the higher levels of the image pyramid.

The error term $e(.)$ from eq. 4.12 is not mimimized as is written here. The measurement vector $\mathbf{y}_k$ from eq. 4.11 the stacked ${}_{\mathcal{R}}I_l(\mathbf{p}_j)$ from all the patches and $a{}_{\mathcal{A}}I_l(\mathbf{p}s_l + \mathbf{D}\mathbf{p}_j) - b$ is stacked to form the corresponding $\mathbf{h}(\mathbf{x}_{k,i})$.

Note that the innovation term in eq. 4.12 considers all the image pyramid levels at once. I.e. it is not a scheme where the finer levels are initialized by the result from the coarser ones, but instead ROVIO considers the full image pyramid each iteration.

Since the state contains structures which are not manifolds, e.g. rotations and bearing vectors, the implementation is formulated with the $\boxplus$ and $\boxminus$ operators. This allows ROVIO to use a formulation similar to that of vector spaces while remaining in the valid lie groups. For the exact formulation and the all the different state transition equations please see [1].

The implementation of ROVIO does the update per feature, which implicitly assumes that the features are uncorrelated. This is in general not the case since the feature patches may overlap. The issue is reduced by extracting features sufficiently far away from other features, but in environments with few features will they still be close. The change of perspective may also move features which initially were apart in image closer to each other.

The mimization of eq. 4.11 is done with Gauss-newton optimization, which as shown by [39] is equivalent to repeatedly doing the EKF update as in sec. 2.3.3. To do this ROVIO needs to calculate the image gradients, which is done in the simple manner of

$$\frac{\partial I}{\partial x}(x,y) = \frac{I(x+1,y) - I(x-1,y)}{2} \tag{4.13a}$$

$$\frac{\partial I}{\partial y}(x,y) = \frac{I(x,y+1) - I(x,y-1)}{2} \tag{4.13b}$$

where I is the image and $I(x,y)$ is the pixel at integer location (x,y). I.e. it does the diagonal pixels of $I(x+1,y+1)$.

If a patch fails to converge ROVIO will try again multiple times with starting points spread out along the uncertainty of the patch location. This is an alternative to using a higher image pyramid level which is guided by the estimated uncertainty.

4.5 Notes on the multi camera case

In VIO it is in general feasible to have two(or more) cameras with overlapping field of view(FoV) since it makes the depth observable and makes it possible to initialize from only a single view. Typically, these stereo cameras are configured such that all cameras trigger at the same time. ROVIO supports stereo cameras with or without overlapping FoV and integrates them into a joint filter update. It defers processing an image until it has a pair and then it processes them both together.

Chapter 5

ROVTIO : RObust Visual Thermal Inertial Odometry

ROVTIO is developed as a part of this thesis and a fork of ROVIO [1] and with elements from ROTIO [40]. The main changes from ROVIO are the adaptions required for unsynchronized cameras, the use of full bit depth 16bit radiometric images in the filter update and the necessary logic for fusing the two modalities.

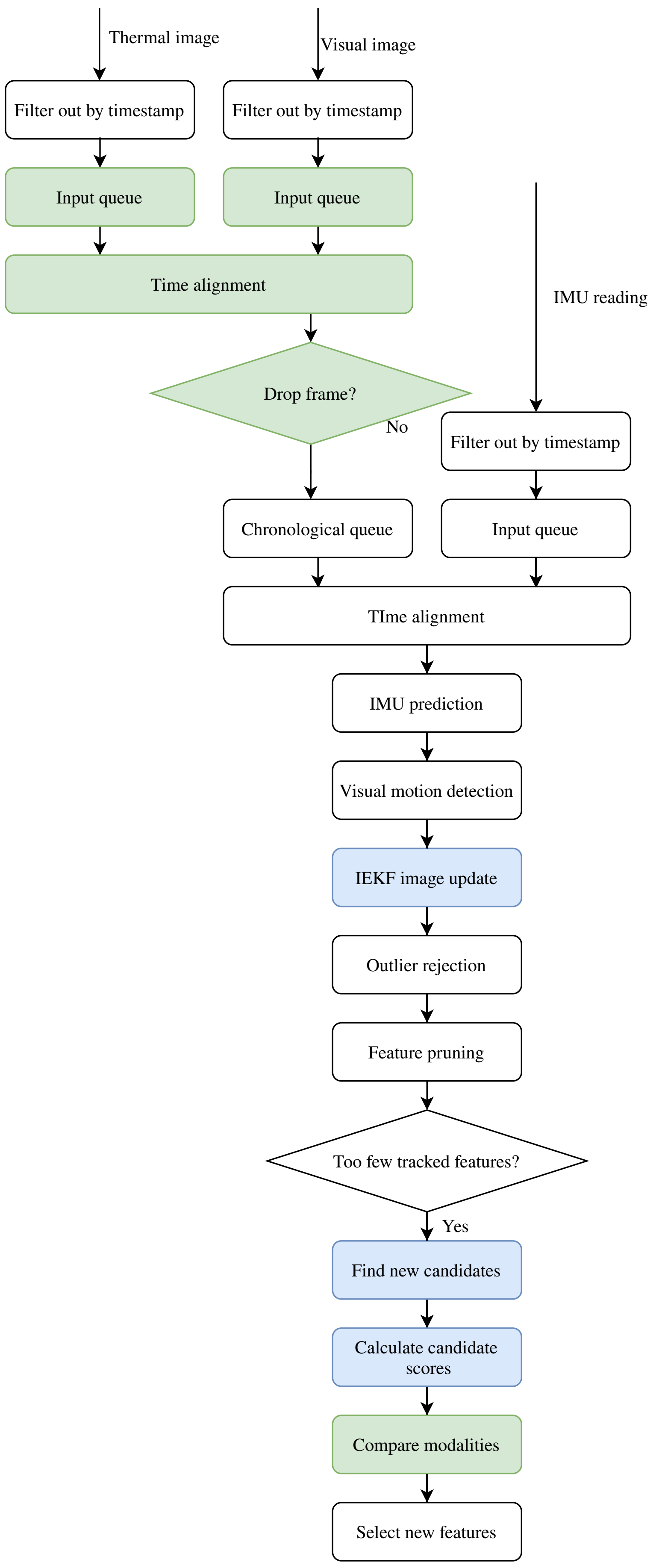


Figure 5.1: An overview showing the modules from original ROVIO which is modified in ROVTIO.

5.1 Tracking on 16bit images

Thermal cameras often obtain radiometric images at 14bit color depth per pixel, which makes it necessary to use two byes to represent each pixel. ROVIO and most other VIO algorithms are made to work with 8bit images, which then are not directly compatible with the 16bit representation. This could be resolved by rescaling the 16bit image to 8bit, but that would be a suboptimal loss of information. Selecting a fixed interval will then need to be very large to account for both the warmest and coldest objects in the area of operation. Linearly rescaling a large interval, or in the extreme case the whole original 14bit representation, will reduce the contrast. In many environments the temperature in the scene is generally the same or very similar, which leads to low contrast in the original 14bit image. Further degrading this will reduce the quality of feature tracking [30]. A common alternative, which also often is implemented in the camera, is to dynamically change the rescaling interval by considering the maximum and minimum reading of the image which is being rescaled. This will reduce the loss of contrast to the minimum available without clipping. However, this also means that the intensity will change between consecutive when warm or cold objects leave the field of view(FoV). Motivated by this and the similar discussion in [30] the tracking of ROVIO is adapted to use 16bit images without rescaling.

ROVIO originally uses floating points to represent the patches it uses for tracking, so the minimum change needed to account for 16bit images is just the extraction of these patches. This solution to the technical problem of color depth in the images, however, cloaks the the persistent issue that the values in the images of different modalities are in a range which differs with orders of magnitude. In theory the thermal images represented by floating points could be rescaled without loss of contrast, but this does not mean that it will be comparable to visual images. In fact, in the data analyzed in this thesis the variations in the images seem to be more comparable when the thermal images are unscaled. The raw intensity is in order of magnitude higher, but the gradients are sometimes in the same range.

5.2 Time alignment of separate image streams

ROVIO originally uses stereo images which are exposed at the same time, i.e. the timestamps are the same. It is possible to synchronize the triggers of thermal and visual cameras, but that is not done in all robots. Specifically, the cameras do not trigger simultaneously in the data considered in this thesis nor in the open VIVID [41] dataset.

Every step in the algorithm, feature detection and tracking, is modified to only use one of the images in the stereo pair and to keep track of which of them that is updating. The incoming images are left in two input queues until none of them are empty. Then the oldest of them is taken out and inserted into the next queue for alignment with the IMU. This is repeated until one of them is empty at each new update. If one of the image streams stops providing images, the other queue will be emptied without waiting for the stopped one. This may happen if one of them has a sensor failure or during FFC for the thermal camera.

In difficult conditions ROVIO requires more runtime to complete an update due to more iterations and more searching. Also, if ROVIO is running on a computer that runs other software, the execution may be interleaved by other computationally demanding tasks. Both of these may cause ROVIO to run slower than real time for short amounts of time. This would then cause the queues for aligning the IMU and the images to build up and increase the latency of the estimator. Control systems are in many cases very vulnerable to high latency, which may severely degrade control performance. To avoid these spikes in latency ROVTIO stops adding new images to the alignment queue if the queue is starting to fill up. To reduce the negative impact on the estimator performance, only frames from the camera with the least amount of tracked features are dropped.

5.3 Selecting the most reliable modality

Due to runtime limitations, ROVIO[1] and ROVTIO are limited to a fixed set of features. If ROVTIO just selects features the same way ROVIO does, they will end up rather arbitrarily divided between the modalities. This may not be optimal, and therefore the division of features between the modalities is explicitly calculated.

Automatically deciding which of the frames that contains the most useful information for tracking is non-trivial, since the two cameras provide information from two different modalities. This information is not directly comparable due to the different color depths. To get around this, ROVTIO uses several indicators when deciding the best modality.

The first one is the area of the image which is found suitable for tracking. I.e. the modality where the most feature candidates of a minimum quality can be found. This is in theory comparable between the modalities since both sensors are projective imaging sensors. However, this does not escape the underlying problem, since we need to find a fixed threshold for what is the minimum accepted quality. This is suboptimal since the threshold only can be valid in a range of scenarios. If both modalities are very good, the threshold will not differentiate well between them, since both will have many features above. This is not a severe issue; if both modalities are good both can be used. It is worse if both modalities have few features above the threshold. Then this indicator becomes more a check for which modality that has the strongest features instead of which modality that has the most features. In theory we could reduce this issue by having an adaptive threshold which moves according to the performance of the best performing modality. However, such a scheme will be very difficult to tune since the thresholds may not change with the same area and may not be changing linearly.

To save runtime it uses the same features and scores which are used in the feature detection to evaluate which of the modalities that are best suited for tracking. The feature score which needs to be above the given threshold is the same as used for feature extraction and is described in sec. 4.3.1. One could consider to use the feature scores directly, but they appear to be very different between the two modalities and not really comparable.

In addition to this indicator which considers the most recent frame, ROVTIO implements two indicators based on the tracking history. One is the average feature lifetime. I.e. the average time since first initialized over all existing features. This is based on the hypothesis that a modality which is currently good for tracking will have long feature tracks. If both modalities are so good for tracking that the features live until they leave the FoV, this score should on average be neutral. The downside is that since it is only neutral on average, it may lead to some randomness in the result.

The second historical indicator is an approximation of the number of possible features for the N last seconds. I.e. it is considering the history of the first indicator. Instead of keeping track of all the previous scores it approximates them by recursively estimating the approximate average.

$$\text{Historical modality score} = \frac{\text{Time since last update}}{N}\text{Modality score} \tag{5.1}$$

This last indicator may help to smooth out short term changes in performance and make the algorithm prefer the modality which is historically performing best. The downside is that it will induce an unwanted hysteresis when switching between the modalities.

So far this section has described three indicators for which modality is the best: The modality score based on the detected features in the frames, the average feature lifetime and the historical modality score. First, all of these are reformulated into a relative score compared to the other modality. I.e

$$\text{Score} = \frac{\text{Indicator value}}{\text{Sum of indicator value of both modalities}} \tag{5.2}$$

Combining these in a total score can be done in different ways. After some trial and error, ROVTIO ended up with three cases. If very few features are found in total, then no restrictions are imposed and all possible features from both modalities are used. If that is not the case, and the difference in "Historical feature score" is large enough to be decisive, the total score is

$$\text{Total score} = \text{Historical feature score}^2 \cdot \text{Modality score} \tag{5.3}$$

otherwise, the total score also includes the "Average feature lifetime":

$$\text{Total score} = \text{Average feature lifetime}^2 \cdot \text{Historical feature score}^2 \cdot \text{Modality score} \tag{5.4}$$

Both are then scaled by a fixed scalar to reasonable range of scoring. Then the total score is multiplied by the maximum number of features it is possible to add to get the number of features to add from this modality.

Results during the development of this modality selection scheme indicate that the accuracy increases if ROVTIO relies only on a single modality instead of using two modalities. Partly based on those indications and on the fact that the scheme described above worked better than the alternatives, this formula for combining the different indicators is nonlinear. If a modality is a little good, it quickly gets a high score and if it is a little bad, it quickly gets a low score. This is why the total score is multiplicative instead of additive and why some of the factors are squared.

Chapter 6

Evaluation

This chapter evaluates the method on a series of datasets collected from a robot with a visual camera, a thermal camera, an IMU and a high precision optical tracking system for accurate ground truth. The developed method is compared with a similar method using a visual camera, ROVIO [1], and an adaption of ROVIO for thermal cameras called ROTIO [42]

6.1 Datasets

The datasets used in this evaluation are collected by the Autonomous robots lab in the spring 2021 using a custom-made aerial robot. They are several sequences collected indoors in a room with a Vicon tracking system. The illumination varies between the sequences to give varying conditions for the visual camera and there is a varying amount of heated objects to provide different conditions for the thermal camera. Table 6.1 gives an overview over the characteristics of the different sequences.

The Vicon optical tracking system requires sufficiently many observations of the robot at all times. Some places during the trajectory the robot is partly outside of the FoV of the optical tracking cameras, or they are occluded. If not enough of the markers are detected by enough of the cameras, the rotation is estimated wrong. This shows up as spikes in the rotation error graphs in this chapter. To reduce this problem the ground truth data is preprocessed to leave out the largest spikes, but not all of them could be removed. This phenomenon can be recognized by thin spikes in rotation error that occur at the same place for all three methods.

The utilized aerial robotic scouts called “Charlie” are based around a DJI Matrice M100 quadrotor platform and integrate a multi-modal sensor fusion solution combining LiDAR (Ouster OS-1), a visual (FLIR Blackfly), a thermal camera (FLIR Tau2 LWIR) and inertial (VectorNav VN-100) estimation. The aerial robot relies on Model Predictive Control (MPC) for its automated operation and its controller and path tracker subscribe to the data provided by its multi-modal localization and mapping system [40] and provide references to the onboard controller [43]. All the processing takes place onboard and in real-time based on an Intel NUC-i7 (NUC7i7BNH) computer that further interfaces the attitude and thrust control unit from DJI. The LiDAR sensor integrated, the Ouster OS-1, provides a horizontal and vertical field of view of $F_H = 360°$, $F_V = 30°$ respectively and a maximum range of 100m. It is noted that the data from the LiDAR are not used in this study. The visual FLIR Blackfly camera model type is BFS-U3-16S2M-CS and recorded at 720×540 resolution for purposes of efficiency. The thermal FLIR Tau2 camera has a resolution of 640×512 pixels.

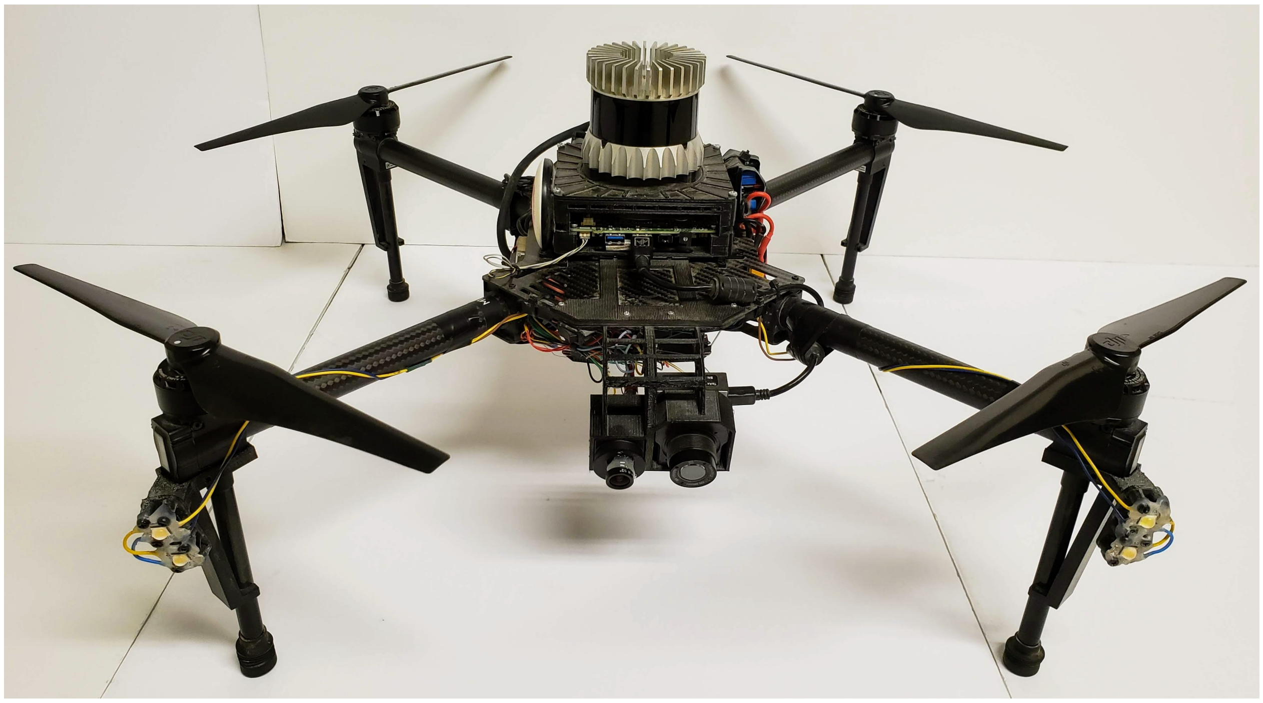


Figure 6.1: Illustration of the robot used to collect the datasets. The thermal and visual cameras are visible in the front and the LIDAR is at the top. The IMU is not visible, but below the Intel NUC. Courtesy of the autonomous robots lab.

The parameters that exist in ROVIO, ROTIO and ROVTIO due to similarity between the methods are tuned to the same values.

Datasetname	Enviroment	Illumination	Platform	Motion
Sync	Indoors, highly augmented thermal	Partial	Flying quadcopter	Slow in the beginning, then aggressive
LT1	Indoors, some thermal augmentation	Partial	Flying quadcopter	Slow in the beginning, then aggressive, little yaw
LT2	Indoors, some thermal augmentation	Partial	Flying quadcopter	Slow in the beginning, then very aggressive, little yaw
LT3	Indoors, some thermal augmentation	Partial	Flying quadcopter	Aggressive manual flight
ALT1	Indoors, no thermal augmentation	Global	Handheld	Handheld walking motion
ALT2	Indoors, no thermal augmentation	Global	Handheld	Handheld walking motion

The *sync* dataset is characterized by the very strong thermal augmentations and partial illumination. The camera tuns away from the illuminated part of the scene which makes visual tracking difficult.

lt1, *lt2* and *lt3* have some heated objects in the scene, but less than the *sync* dataset. The scene is partially illuminated, but the camera motion in *lt1* and *lt2* keeps the illuminated parts in the frame during the whole sequence. *lt3* alternates between light and dark frames.

The last two sequences are captured handheld instead of flying and are in a fully illuminated room with no heated objects to assist thermal tracking. The strongest thermal features are from the windows in the scene, but they are relatively far away from the camera compared to the visual features.

Figure 6.2 shows a screenshot from the *alt2* sequence. All of the datasets are collected in the same room and this shows the room quite well. The illumination is only representative for *alt1* and *alt2*.

The illumination in the other sequences is more similar to the one shown in figure 6.3. There it is an illuminated part in the center and darkness to the left and right. In *lt3* and *sync* the camera turns away from the illuminated part in the center, which shows up as in figure 6.4.

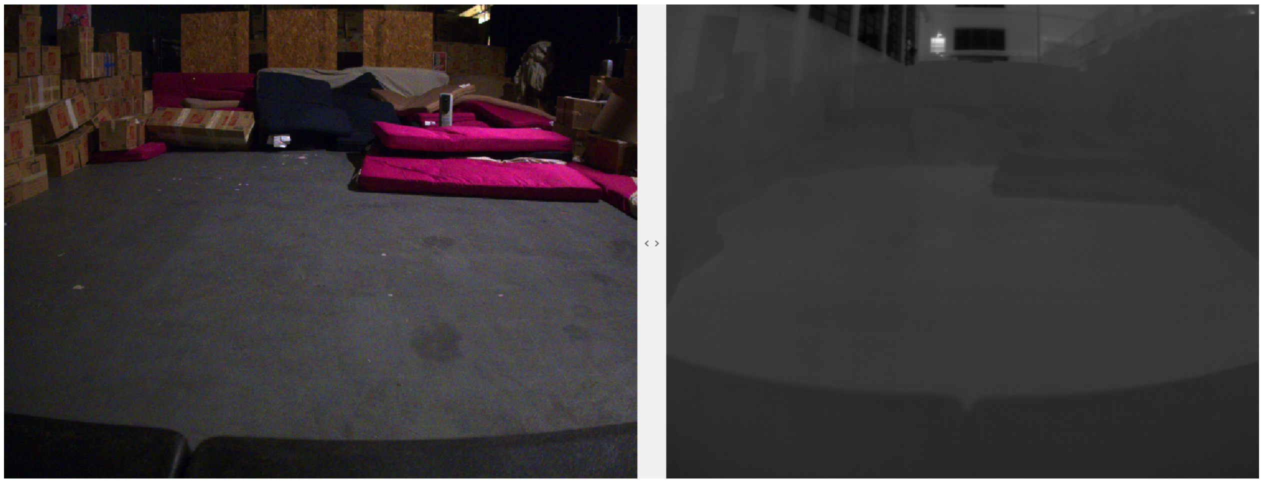


Figure 6.2: Screenshot of the beginning of the *alt2* sequence. Visual to the left and thermal to the right.

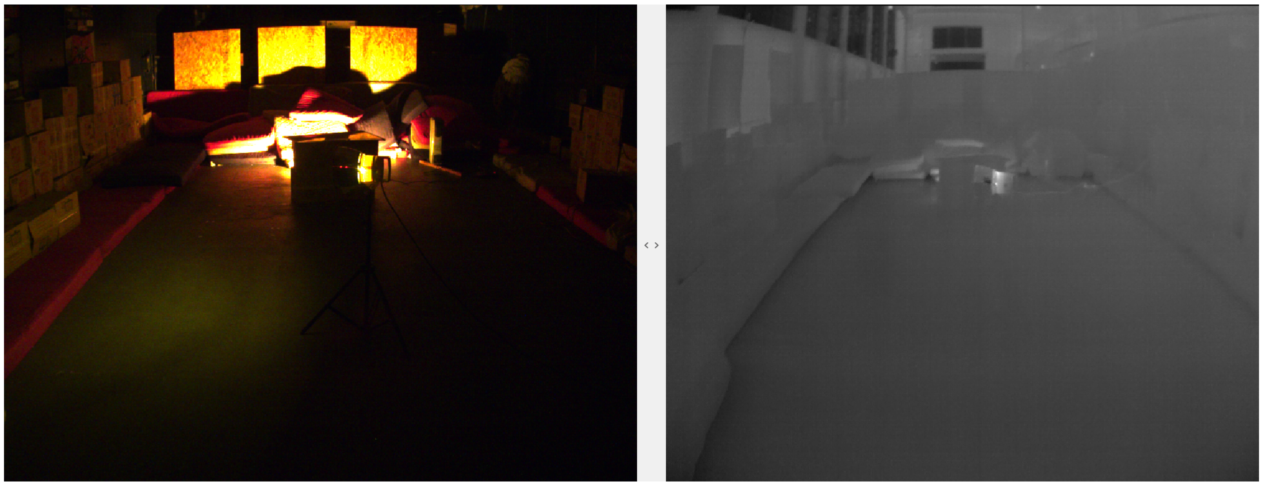


Figure 6.3: Screenshot of the beginning of the *lt3* sequence. Visual to the left and thermal to the right.

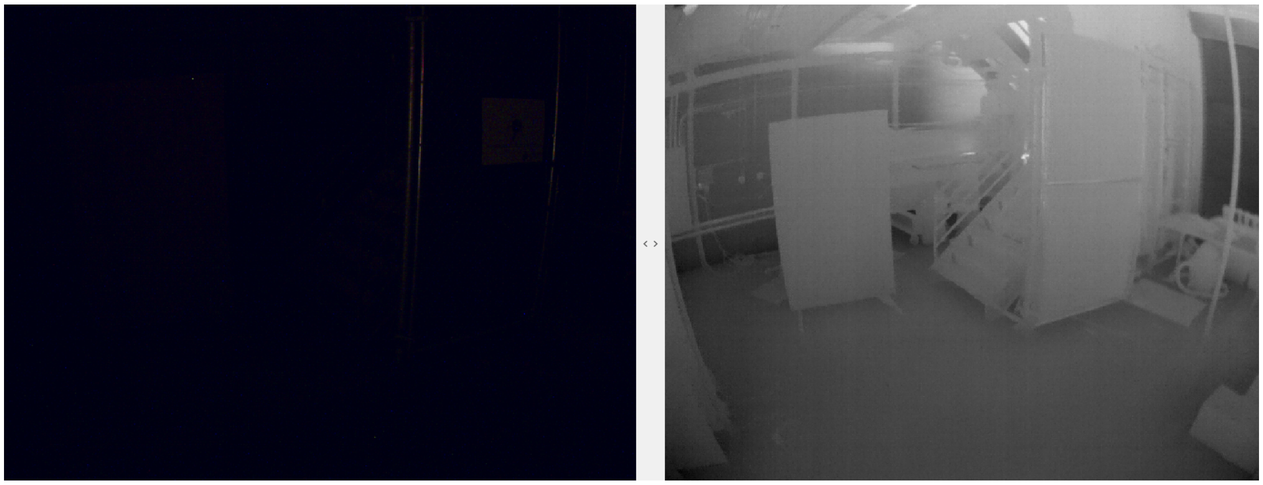


Figure 6.4: Screenshot from the *lt3* sequence. It is taken after a while when the cameras are facing away from the illuminated part. Visual to the left and thermal to the right.

6.2 Overview over the results

Figure 6.5 and figure 6.6 show an overview over the results of the different methods at the different datasets. Figure 6.5 shows the mean error of the translational part of the pose and figure 6.6 shows the mean error of the rotational part.

When ROVIO (and ROTIO and ROVTIO) lose track of too many features, the filter will continue providing localization data, but it will be very wrong. It will often estimate increasingly high velocities, which makes the tracking worse and puts the estimate further and further away from the truth. This is what happened in the *sync* dataset and the reason for ROVIOs poor performance there.

The methods compared here are all odometry methods, so it makes sense to also consider the RPE. The ATE metric includes the accumulated drift which gives relatively higher impact to the estimation errors early in the sequence compared to the estimation errors at the end. Figure 6.7 and figure 6.8 show the mean RPE with over a distance of 3 m.

In many cases slow drift in the error is less bad than abrupt spikes in the error. One possible way of quantifying this could be to consider the maximum RPE over a short distance. The largest RPE would be the worst spot during the sequence for e.g. a feedback control system. The performance of the worst spot must be above the minimum requirement for the system to be reliable. This is however also prone to large movement in the beginning, before the filter has converged, and issues with the vicon coverage. Figure 6.9 and figure 6.10 show the maximum RPE over 3m for the different sequences.

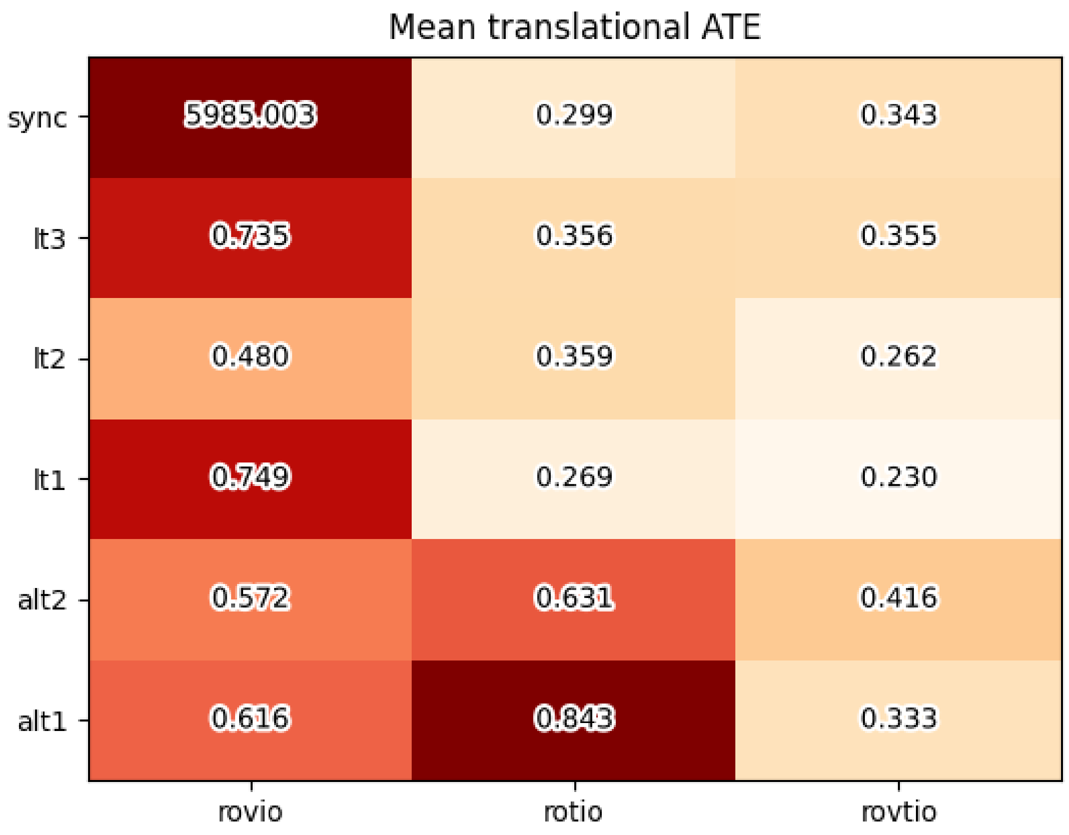


Figure 6.5: The mean translational ATE for the various datasets and the three different methods methods in meters.

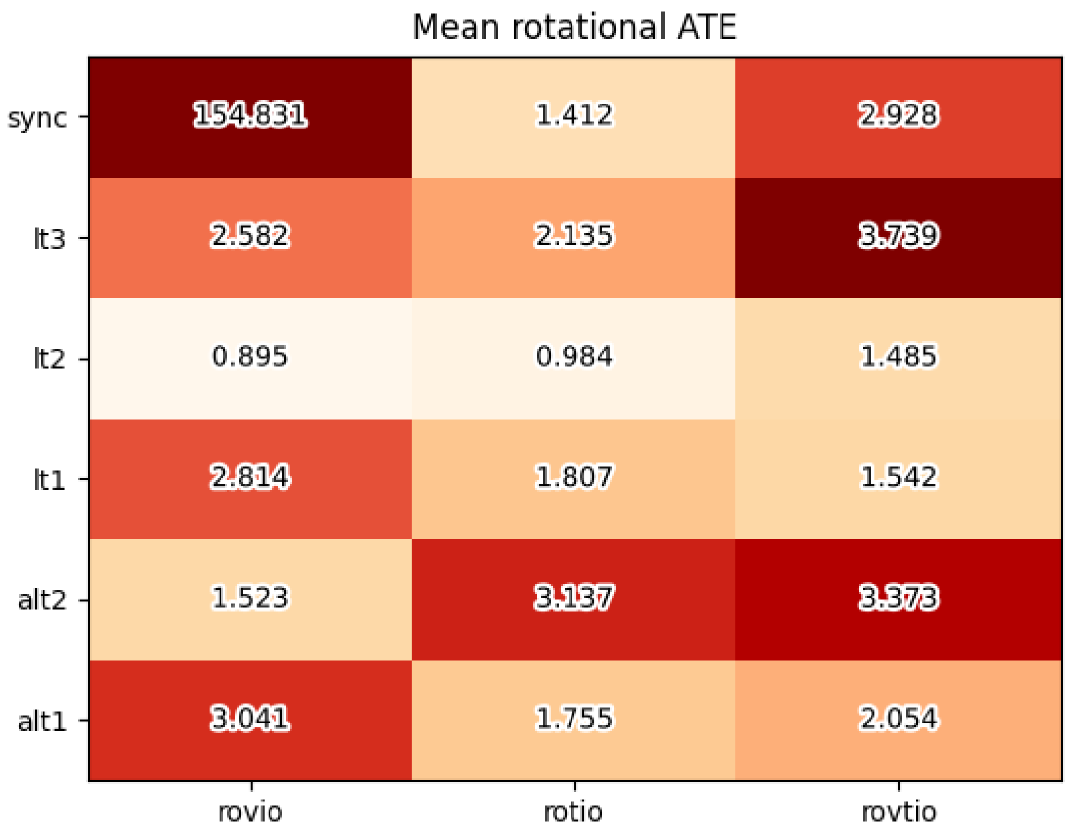


Figure 6.6: The mean rotational ATE for the various datasets and the three different methods methods in degrees.

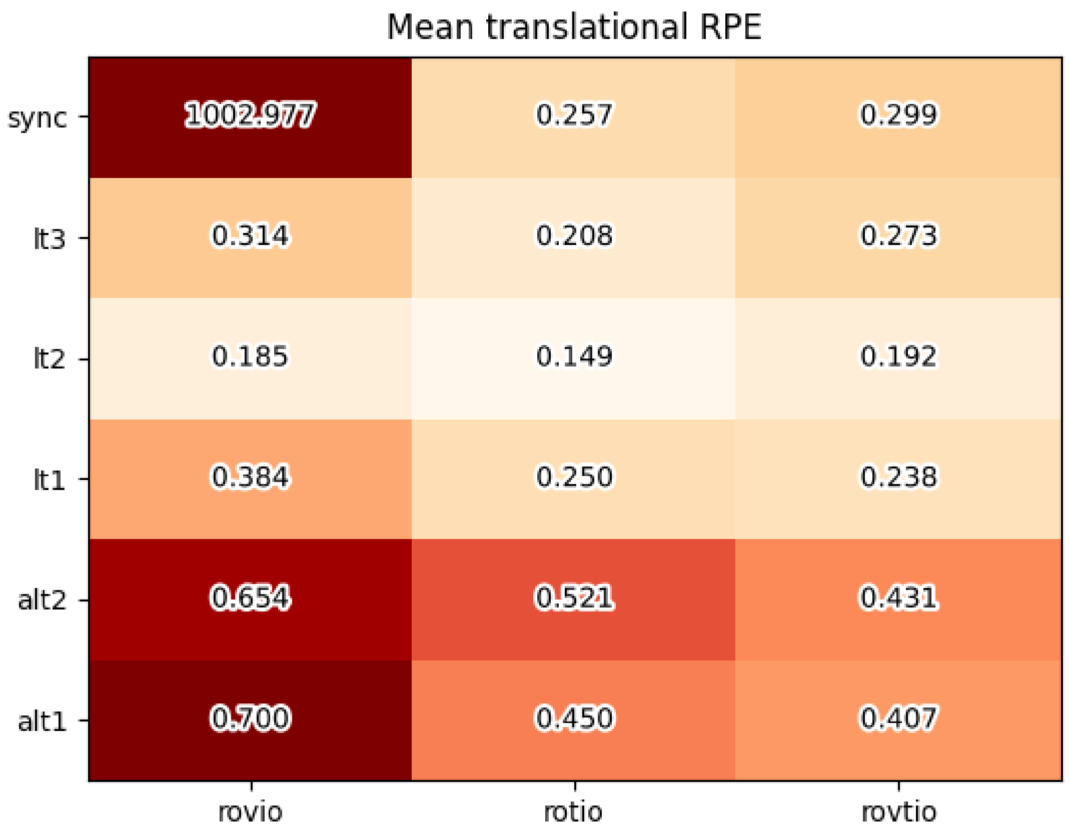


Figure 6.7: The mean translational RPE for the various datasets and the three different methods methods in meters.

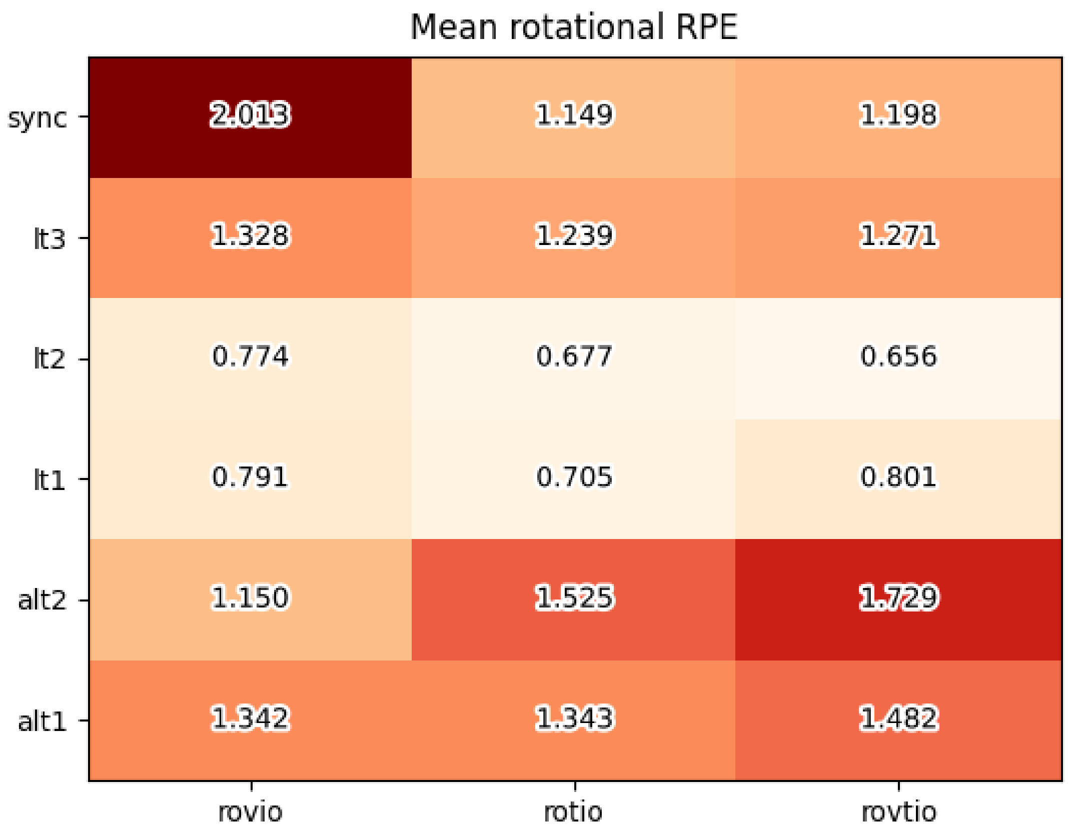


Figure 6.8: The mean rotational RPE for the various datasets and the three different methods methods in degrees.

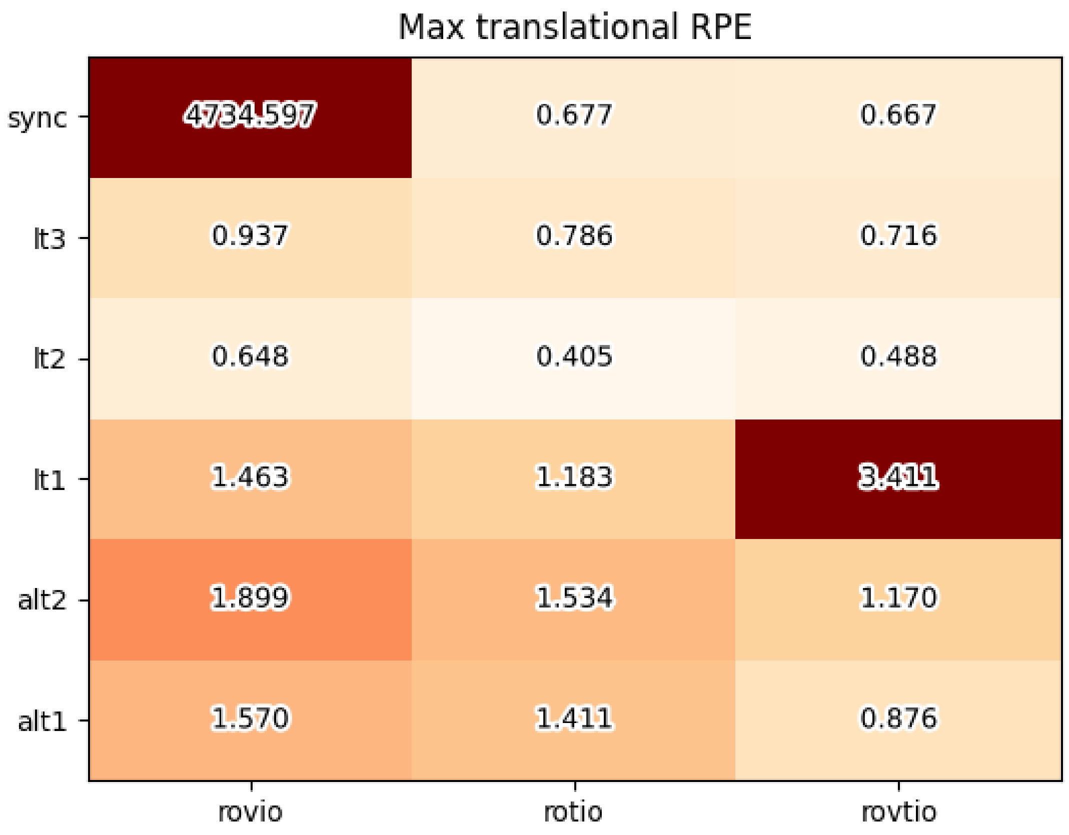


Figure 6.9: The maximum translational RPE for the various datasets and the three different methods methods in meters.

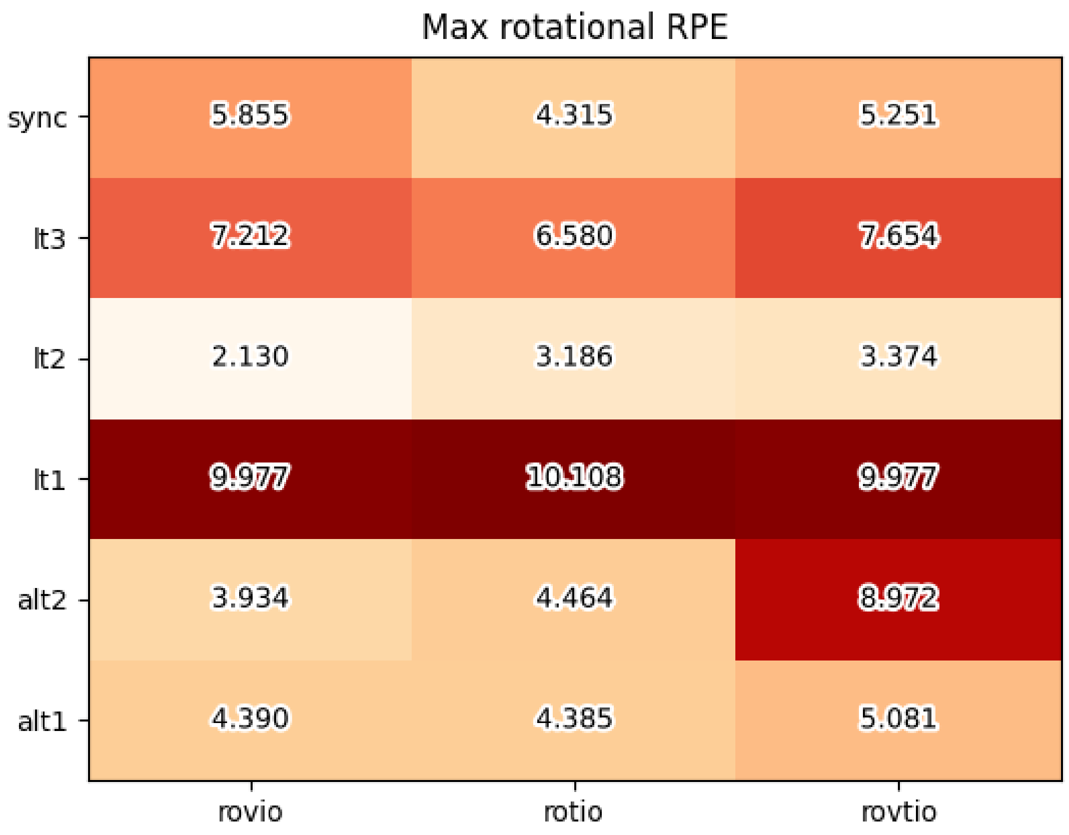


Figure 6.10: The maximum rotational RPE for the various datasets and the three different methods methods in degrees.

6.3 Artificial degregation

The datasets considered in this thesis are collected indoors to be able to collect reliable ground truth. This does, however, impose some restrictions on the diversity of the environment. The focus of this thesis is degraded environments. Visually degrading an indoors environment is relatively easy, the light can be turned off. Thermally degrading an indoors environment is less easy. The windows, walls and objects will have different emissivity and create trackable features. In addition it is difficult to create controlled variations of degregation in real conditions. Both of these motivate an evaluation on artificially degraded data with gradual increase in degregation.

A very common form of of visual degregation is lack of illumination in the scene. An effect similar to low light can be artificially achieved by subtracting a fixed value β from all pixels in the image. All texture which is darker than β will disappear and cannot be used for tracking. The texture with magnitude larger than β is still as distinct as it was and still as good for tracking.

$$\forall x, y | I(x, y) = I_{raw}(x, y) - \beta \tag{6.1}$$

Unlike visual cameras, thermal cameras are not dependent on illumination to work. Thermally degraded environments typically have a uniform temperature and similar emissivity. The artificial degregation of the thermal images is therefore not created the same was as for the visual images, but instead is created by multiplying each pixel with a scalar $\alpha \in [0, 1]$. This will reduce the strength of all features, which in theory should make them worse to track. However, the noise is also reduced, so the quality of the features remains the same. The score given to each feature (see sec. 4.3.1) will be reduced in these degraded images, so all of these methods will extract less features than in the equivalent non-degraded ones.

$$\forall x, y | I(x, y) = \alpha I_{raw}(x, y), \alpha \in [0, 1] \tag{6.2}$$

Neither of these degregations, and in particular the thermal degregation, accurately simulates physical degregation. However, they both reduce the number of tracked features. I.e. the impact these degregations have on the algorithms might be similar to the impact of environments with few features.

Figures 6.11-6.16 show the performance in ATE and RPE of the three compared algorithms. The selected range of degregation means that it starts at a significant level of degregation.

Several of the cells in figure 6.11 have errors in the hundreds. This is because these algorithms use the filter state to assist in the feature tracking. When the estimated velocities are wrong, tracking fails and the velocities increase more and more. The result is a long trajectory racing towards infinity. This leads to high translational errors, but for sequences without much rotational motion, the rotational error might be smaller.

Unsurprisingly the performance drops along the axis of increasing degreagation for the sin-

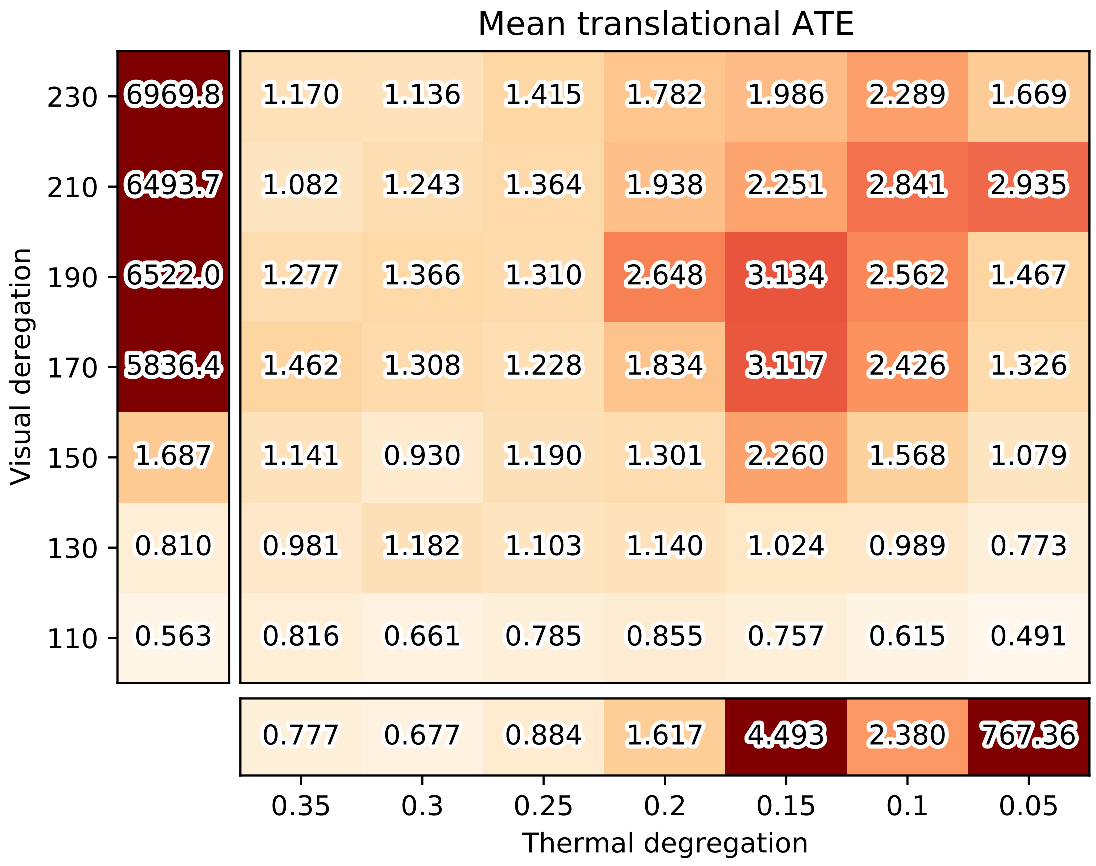


Figure 6.11: The mean translational ATE on the *lt3* sequence for different forms of artificial and thermal degregation in meters. The error is given in meters. The magnitude of the visual degregation is subtracted from each pixel in the visual image and the magnitude of the thermal degregation is multiplied with each pixel in the thermal image. The left column is the results of ROVIO, the bottom row is the results of ROTIO and the block in the middle is the results of ROVTIO.

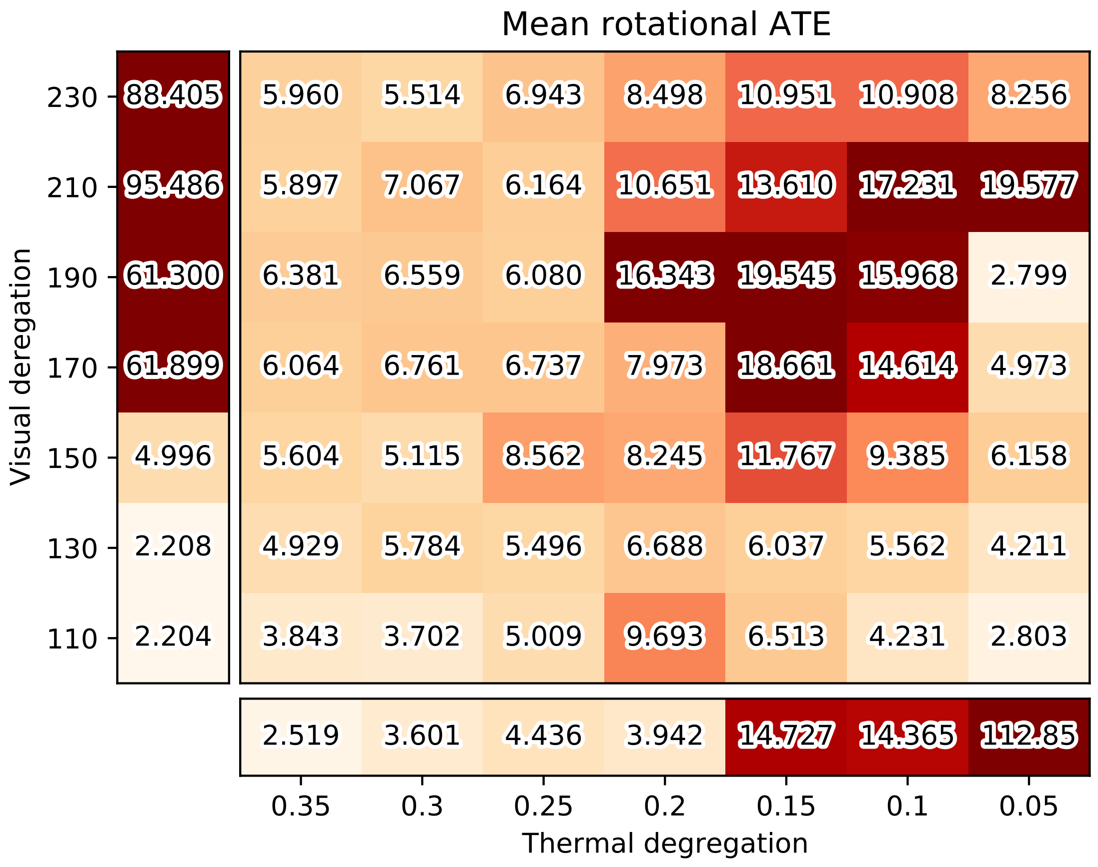


Figure 6.12: The mean rotational ATE on the *lt3* sequence for different forms of artificial and thermal degregation in degrees. The magnitude of the visual degregation is subtracted from each pixel in the visual image and the magnitude of the thermal degregation is multiplied with each pixel in the thermal image. The left column is the results of ROVIO, the bottom row is the results of ROTIO and the block in the middle is the results of ROVTIO.

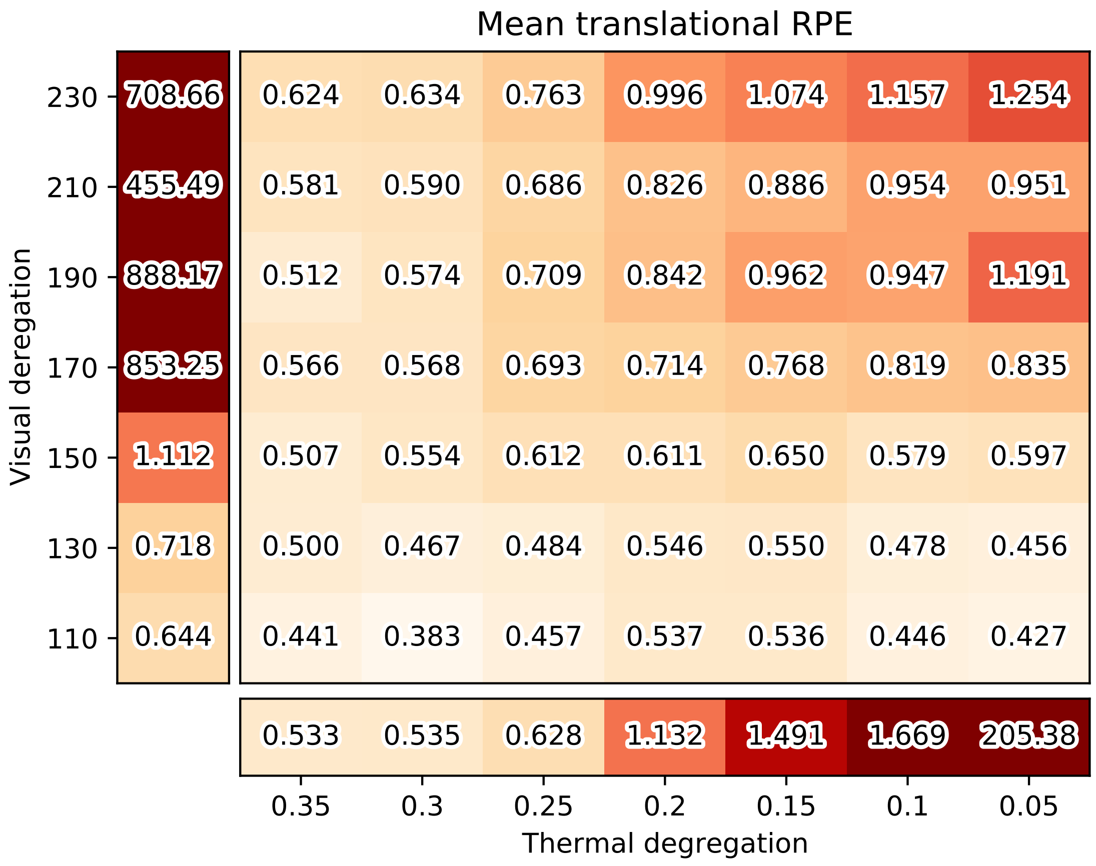


Figure 6.13: The mean translational RPE over 3m on the *lt3* sequence for different forms of artificial and thermal degregation in meters. The magnitude of the visual degregation is subtracted from each pixel in the visual image and the magnitude of the thermal degregation is multiplied with each pixel in the thermal image. The left column is the results of ROVIO, the bottom row is the results of ROTIO and the block in the middle is the results of ROVTIO.

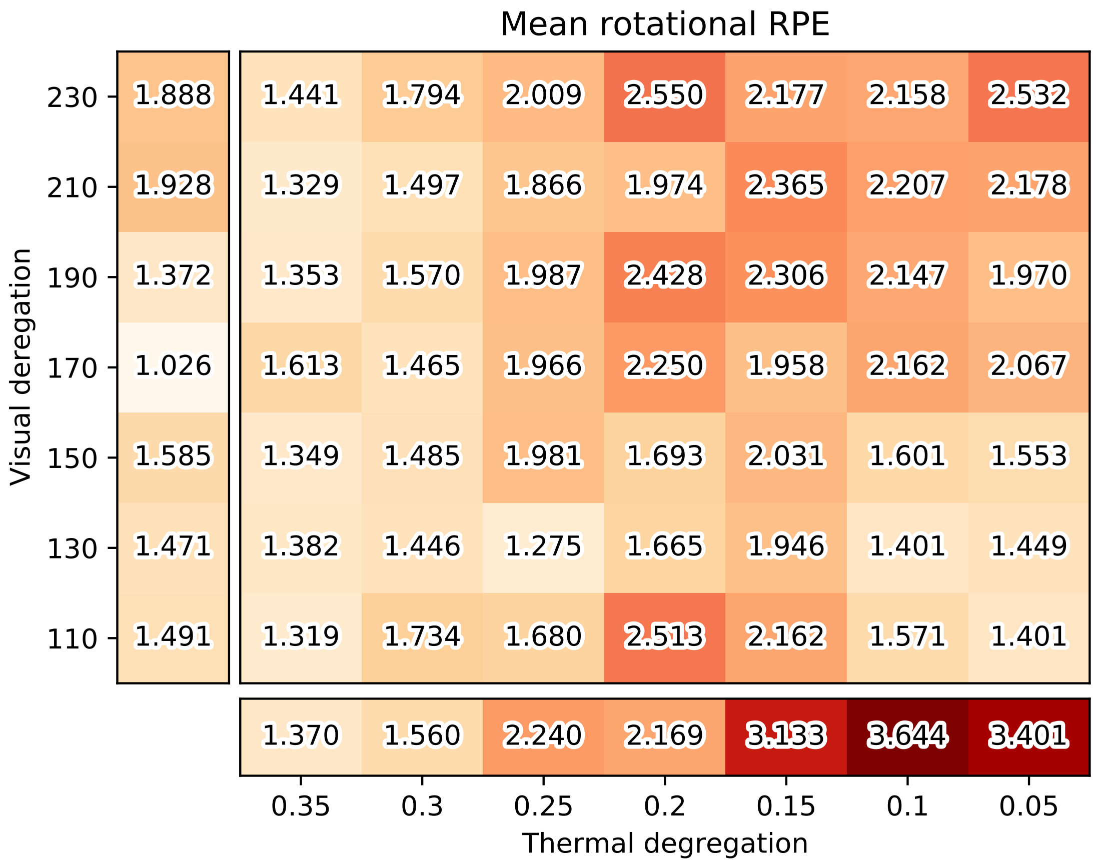
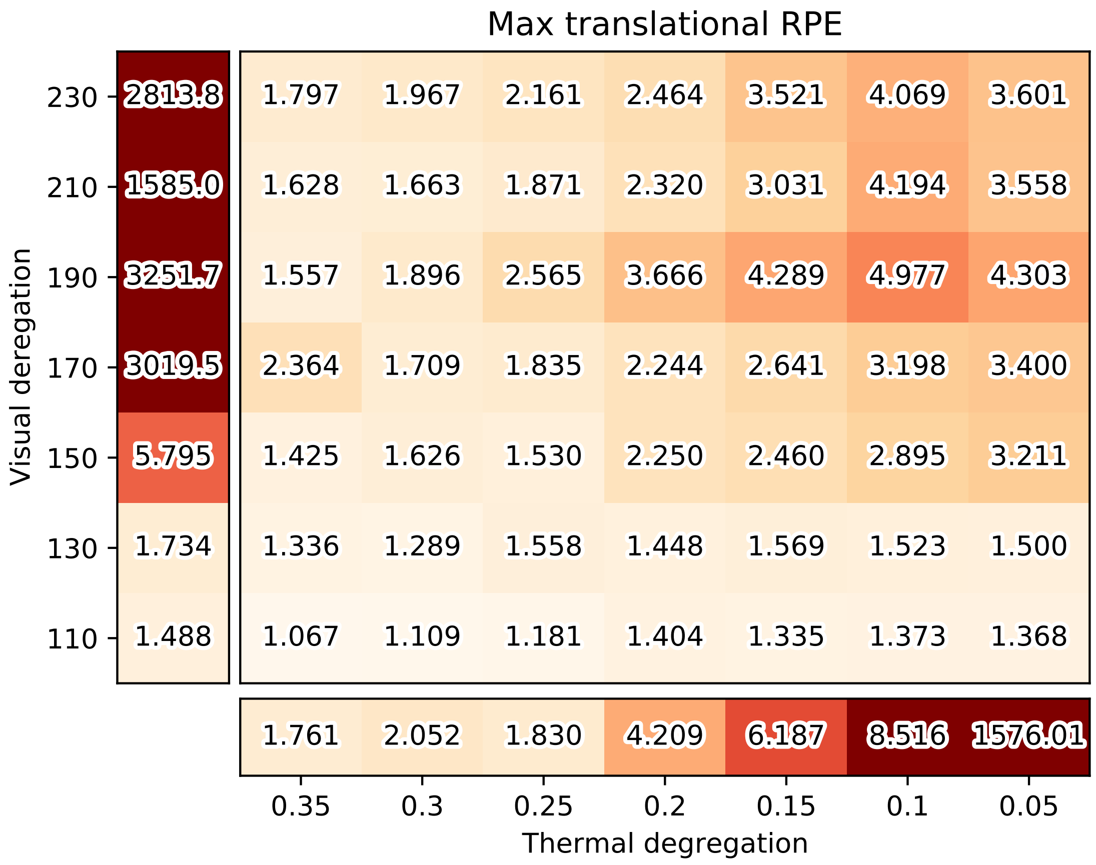
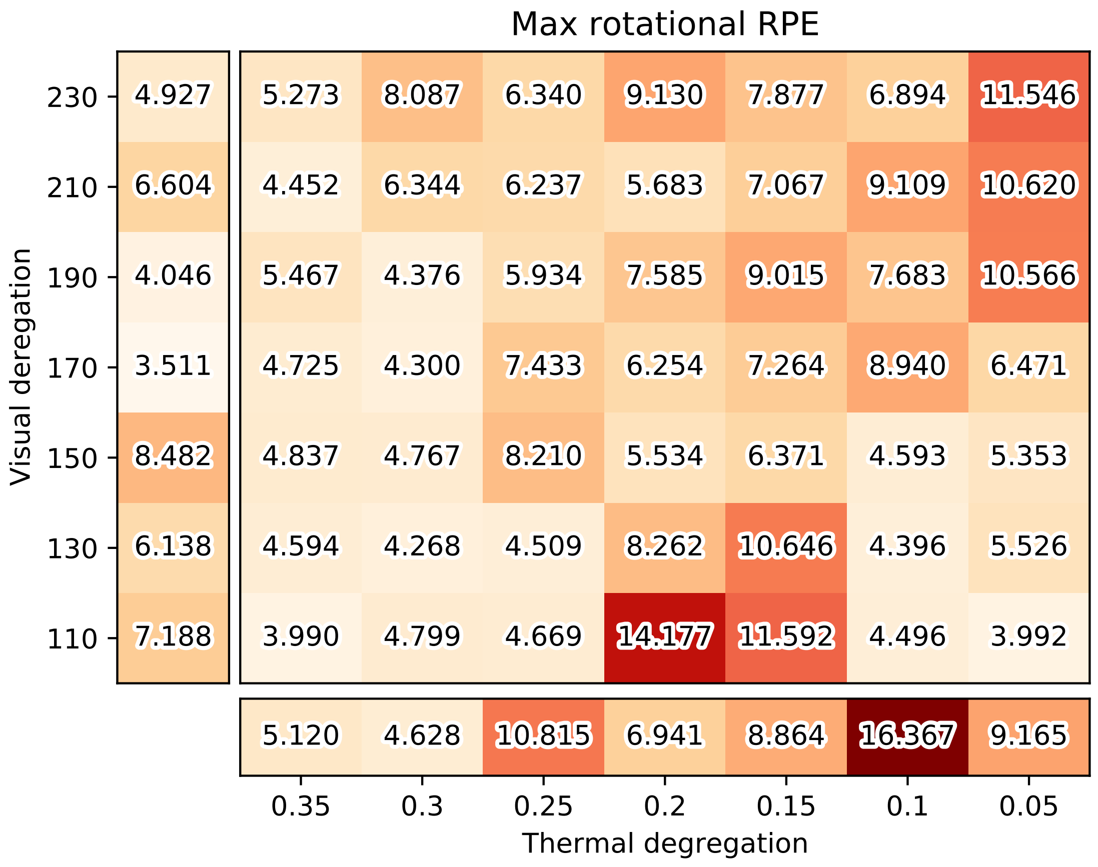


Figure 6.14: The mean rotational RPE on the *lt3* sequence for different forms of artificial and thermal degregation in degrees. The magnitude of the visual degregation is subtracted from each pixel in the visual image and the magnitude of the thermal degregation is multiplied with each pixel in the thermal image. The left column is the results of ROVIO, the bottom row is the results of ROTIO and the block in the middle is the results of ROVTIO.



Visual deregation	ROVIO	0.35	0.3	0.25	0.2	0.15	0.1	0.05
230	2813.8	1.797	1.967	2.161	2.464	3.521	4.069	3.601
210	1585.0	1.628	1.663	1.871	2.320	3.031	4.194	3.558
190	3251.7	1.557	1.896	2.565	3.666	4.289	4.977	4.303
170	3019.5	2.364	1.709	1.835	2.244	2.641	3.198	3.400
150	5.795	1.425	1.626	1.530	2.250	2.460	2.895	3.211
130	1.734	1.336	1.289	1.558	1.448	1.569	1.523	1.500
110	1.488	1.067	1.109	1.181	1.404	1.335	1.373	1.368
ROTIO		1.761	2.052	1.830	4.209	6.187	8.516	1576.01

Figure 6.15: The max translational RPE over 3m on the *lt3* sequence for different forms of artificial and thermal degregation in meters. The magnitude of the visual degregation is subtracted from each pixel in the visual image and the magnitude of the thermal degregation is multiplied with each pixel in the thermal image. The left column is the results of ROVIO, the bottom row is the results of ROTIO and the block in the middle is the results of ROVTIO.



Visual deregation	ROVIO	0.35	0.3	0.25	0.2	0.15	0.1	0.05
230	4.927	5.273	8.087	6.340	9.130	7.877	6.894	11.546
210	6.604	4.452	6.344	6.237	5.683	7.067	9.109	10.620
190	4.046	5.467	4.376	5.934	7.585	9.015	7.683	10.566
170	3.511	4.725	4.300	7.433	6.254	7.264	8.940	6.471
150	8.482	4.837	4.767	8.210	5.534	6.371	4.593	5.353
130	6.138	4.594	4.268	4.509	8.262	10.646	4.396	5.526
110	7.188	3.990	4.799	4.669	14.177	11.592	4.496	3.992
ROTIO		5.120	4.628	10.815	6.941	8.864	16.367	9.165

Figure 6.16: The max rotational RPE on the *lt3* sequence for different forms of artificial and thermal degregation in degrees. The magnitude of the visual degregation is subtracted from each pixel in the visual image and the magnitude of the thermal degregation is multiplied with each pixel in the thermal image. The left column is the results of ROVIO, the bottom row is the results of ROTIO and the block in the middle is the results of ROVTIO.

gle modality algorithms and along the diagonal of increasing both degregations for ROVTIO. It is perhaps more surprising that for the rows of low visual degregation and columns of low thermal degregation, the performance does not decrease monotonically. E.g. for a visual degregation of 110, the worst mean translational RPE is with 0.2 in thermal degregation. Both more and less thermal degregation leads to lower mean translational RPE. When both modalities have low degregation, both are good and can be useful for tracking. When the degreation in one of the modalities increases, the quality of the feature tracks in that modality gets worse, which leads to poorer performance. When the degregation gets even worse, the features in the most degraded modality will not be detected and ROVTIO relies on the least degraded modality. Just relying on the least degraded modality gives less poor feature tracks and the better performance observed in this table.

6.4 Handoff

One of the scenarios where it is beneficial to use both thermal and visual cameras is in environments which alternate between untrackable in one of the modalities while the other still works. One interesting question is then how the algorithm responds to change in the dominating modality; this change is here called a “handoff”.

To emulate this I artificially create these conditions by stopping the two image streams one at the time. This is not a completely realistic test since it is much easier to detect the handoff when the image stream is completely stopped. Furthermore, in many real world handoffs the number of features in the former modality will decline gradually, instead of this abrupt removal.

The trajectory used for the handoff is shown in figure 6.17.

Figure 6.19 illustrates when the handoff happens. The blue line is the tracked visual features and the yellow line is the tracked thermal features. The yellow line starts and stays at 0 for a while until the blue line stops. This is because ROVTIO does not start using the thermal features in the beginning

Figure 6.18 shows that the translational error drifts almost one meter after the handoff, while the rotational error hardly drifts at all. This might be because the motion during the handoff is mostly translating away from the observed scene without rotation.

This handoff was from visual being good to thermal being good. It is interesting to see if it is different the other way too. Figure 6.20 shows the trajectory when transitioning from thermal to visual.

Figure 6.21a shows that the translational drift before the handoff is very similar to the drift in the other case(fig. 6.18a) before the handoff, but the increase in error after the handoff is much smaller. The probable reason is shown in fig. 6.22 where the number of features is changing more gradually.

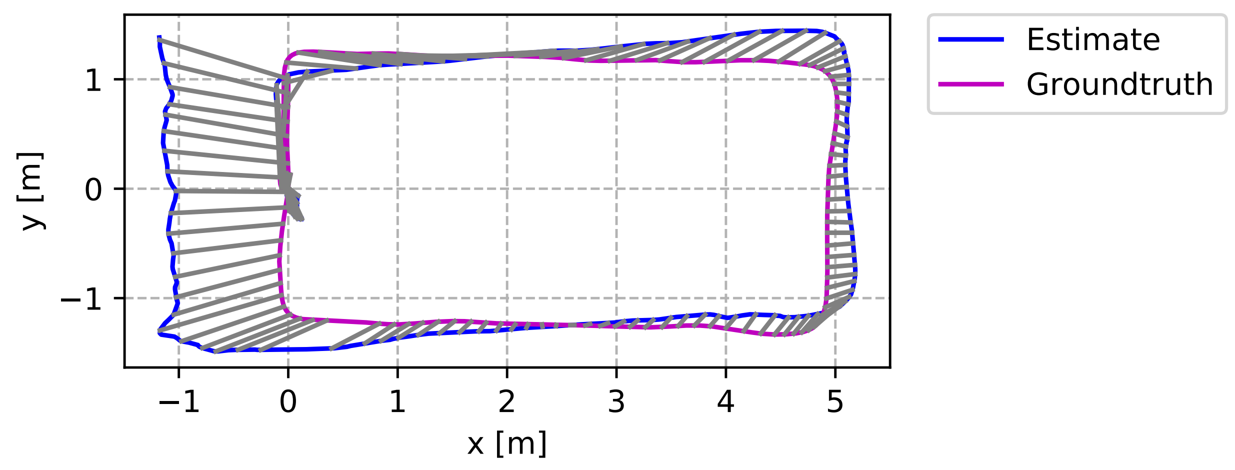
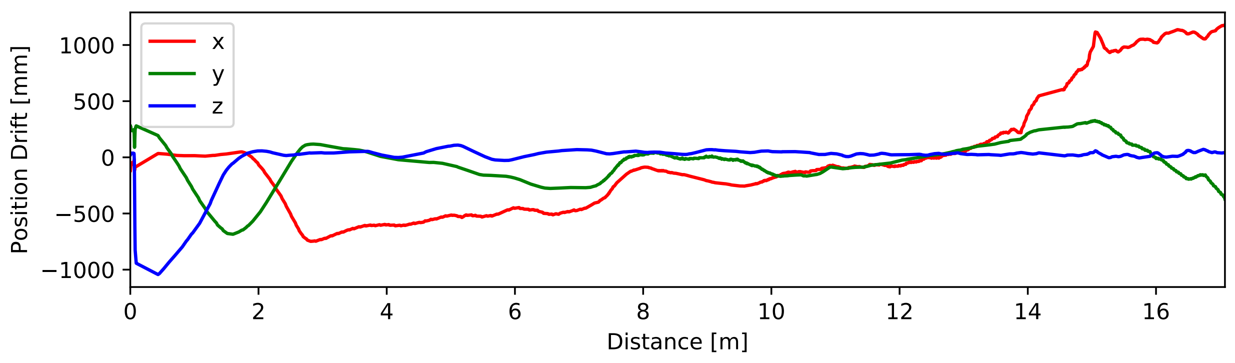
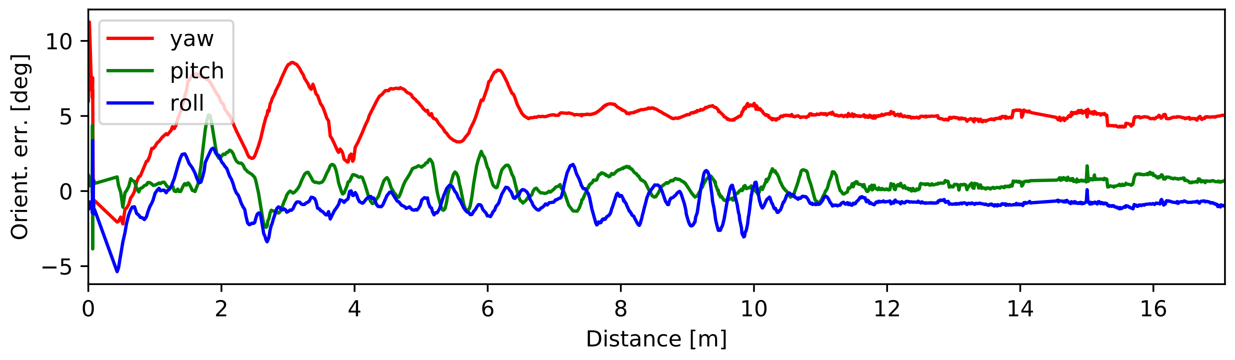


Figure 6.17: The topside view of the trajectory from the handoff. The visual modality was interrupted at the lower side of the rectangle.



(a) Translational drift



(b) Rotational drift

Figure 6.18: The translational drift and the rotational drift during the handoff

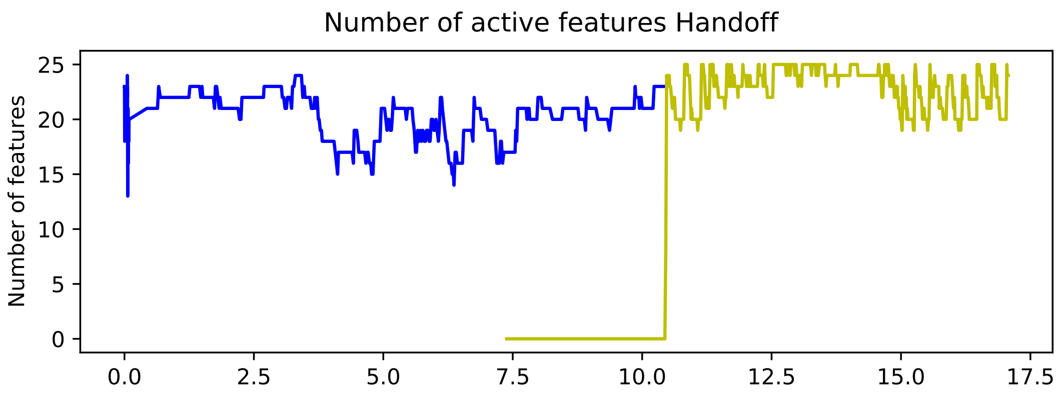


Figure 6.19: The number of tracked features per modality during the handoff.

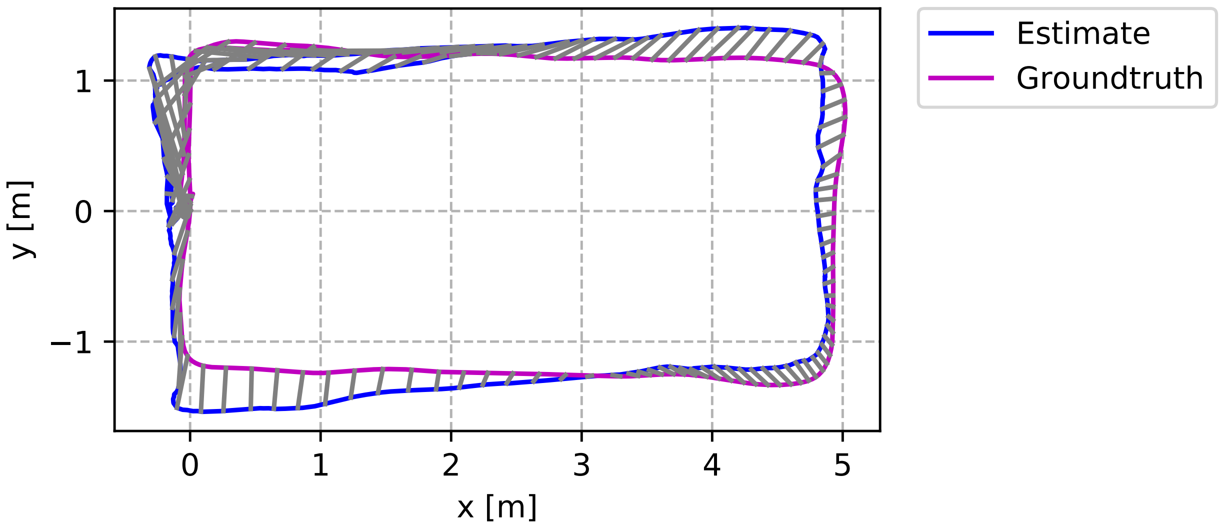
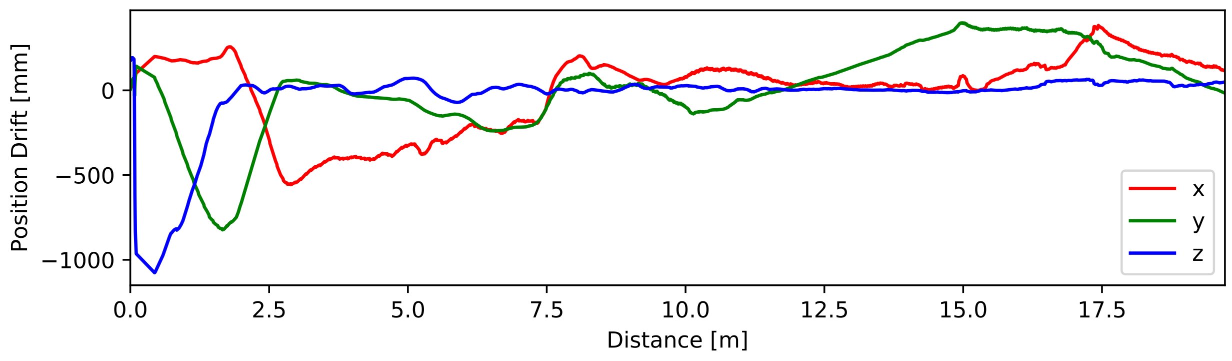
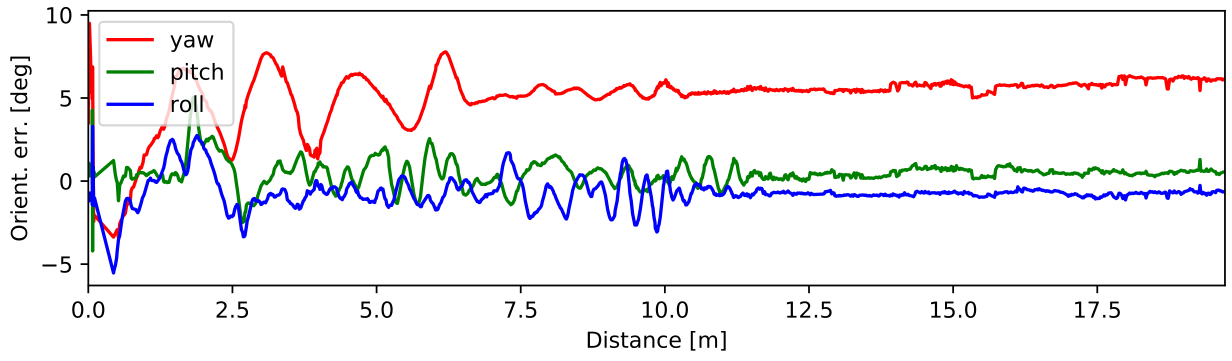


Figure 6.20: The topside view of the trajectory from the handoff. The visual modality was interrupted at the lower side of the rectangle.



(a) Translational drift



(b) Rotational drift

Figure 6.21: The translational drift and the rotational drift during the handoff

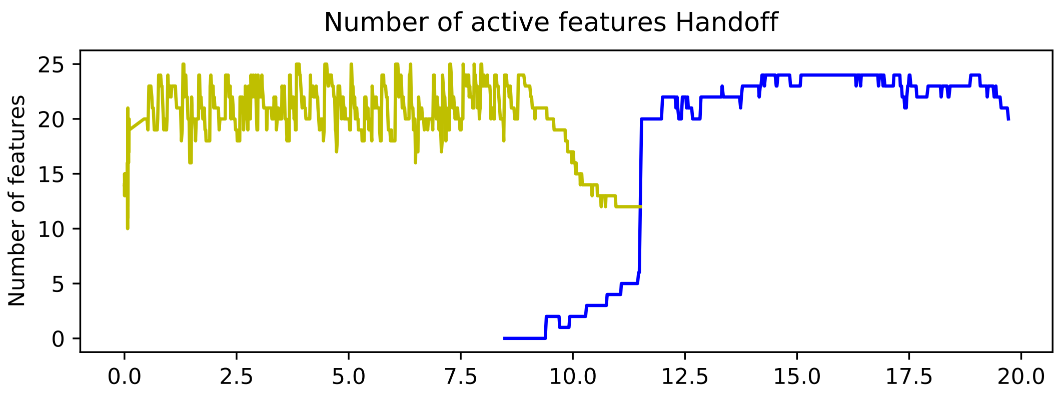


Figure 6.22: The number of tracked features per modality during the handoff. X-axis in meters

6.5 SYNC

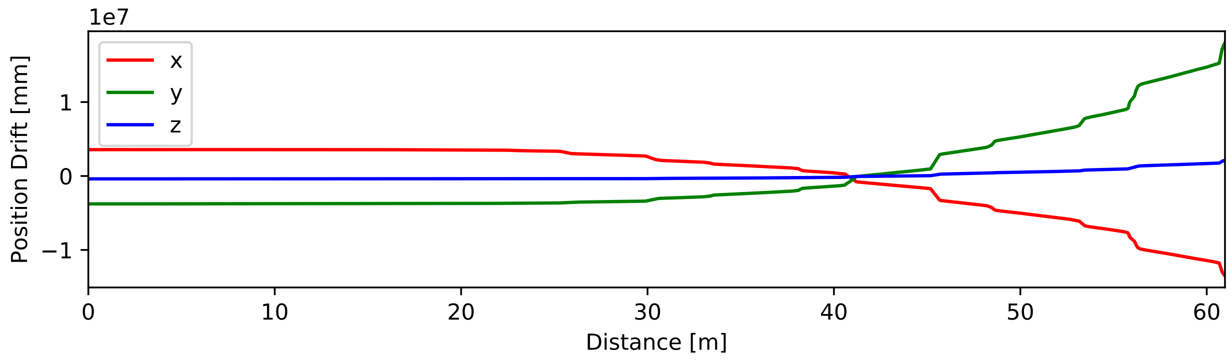
In figure 6.5 we see that the ATE of ROVIO is far higher than the other two. This is because the filter lost track early in the sequence and never recovered the correct velocity. The beginning of the sequence is only partially illuminated and not enough for ROVIO to initialize a consistent map before it drifted away.

The *sync* sequence is partially illuminated with a lamp that illuminates around half of the scene. The other half is partially completely black with almost no trackable visual features. Several heaters and a heated cable are placed in the scene to make thermal tracking easier. These provide very strong thermal features.

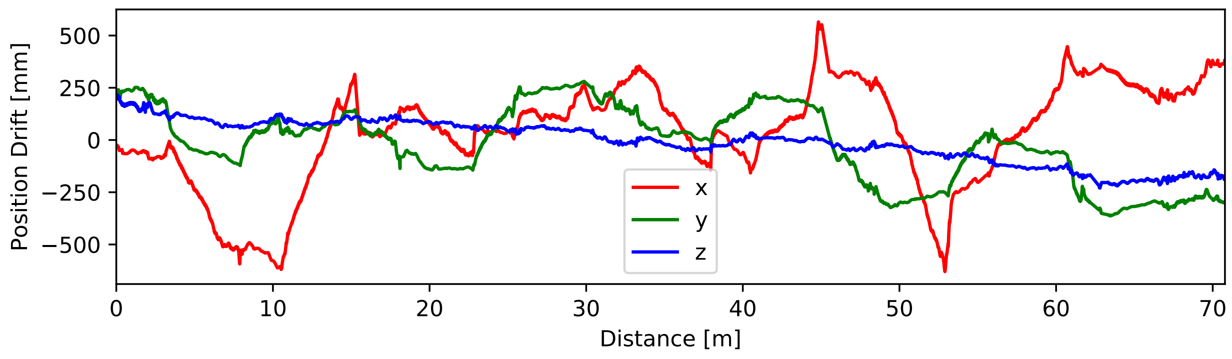
The impression from figure 6.23 and 6.24 is that ROVIO failed tracking early and the other two are quite similar. ROTIO is slightly better, note the scale of the y-axis.

Figure 6.25 shows the average feature lifetime of all the successfully tracked features. The first seconds is before takeoff, which means that the camera is stationary and features do not appear nor disappear. This leads to the linearly increasing feature lifetime in the beginning. ROVTIO has a few situations where all the thermal features disappear. This is because ROVTIO drops all the thermal features during FCC and instead relies on visual tracking.

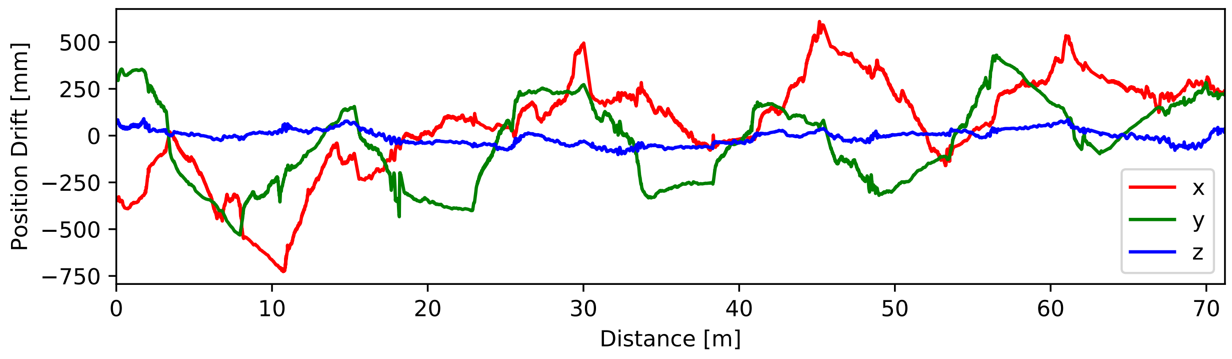
Figure 6.25 shows that ROVTIO is using visual features where ROVIO also finds visual features and thermal features elsewhere. This leads ROVTIO to switch back and forth between the two modalities when the camera moves through the scene.



(a) ROVIO

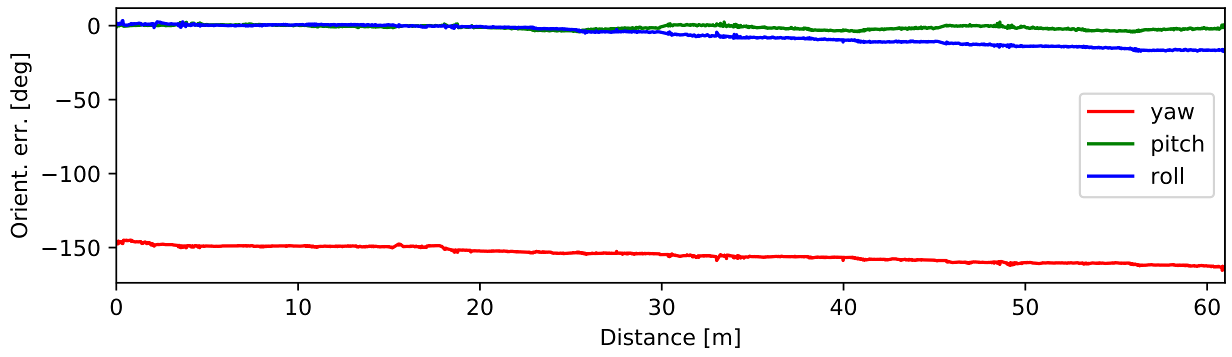


(b) ROTIO

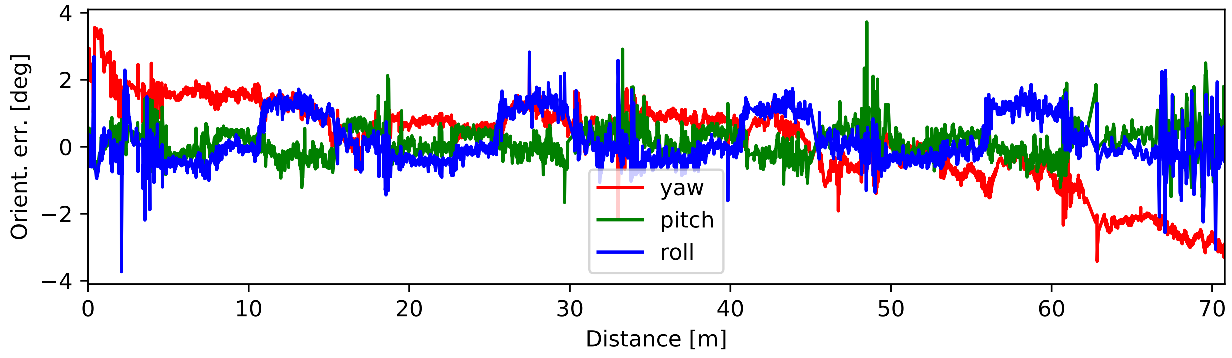


(c) ROVTIO

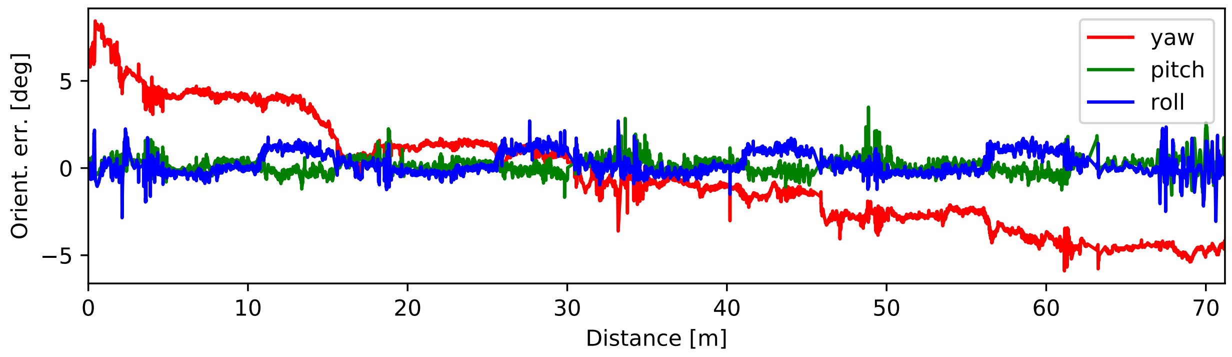
Figure 6.23: The translational deviation from ground truth compared with distanced traveled for the *sync* dataset. Notice the scale on the y-axis which goes up to 10 km for ROVIO.



(a) ROVIO

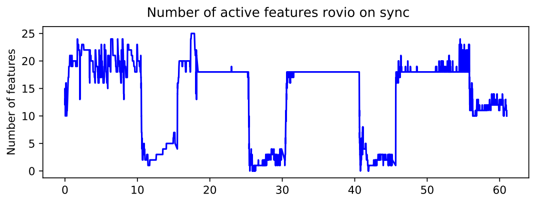


(b) ROTIO

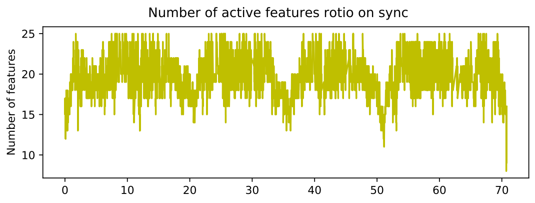


(c) ROVTIO

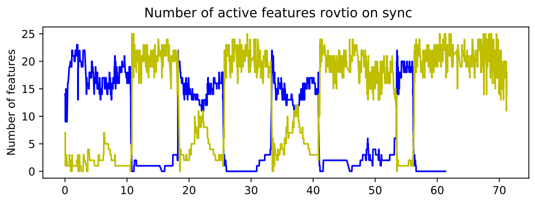
Figure 6.24: The rotational deviation from ground truth compared with distanced traveled for the *sync* dataset.



(a) ROVIO

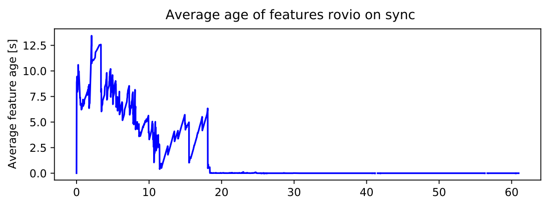


(b) ROTIO

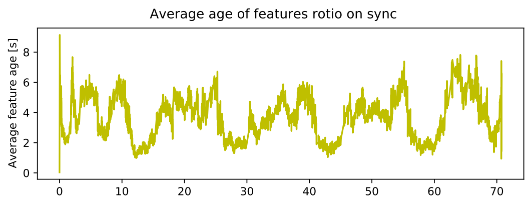


(c) ROVTIO

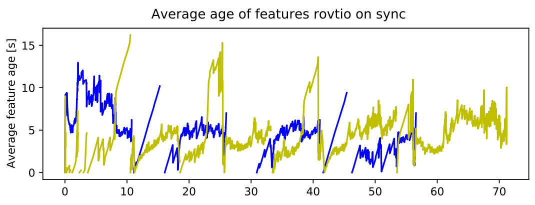
Figure 6.25: The number of tracked features over time *sync* dataset. Blue is the number of visual features and yellow is the number of thermal features.



(a) ROVIO



(b) ROTIO



(c) ROVTIO

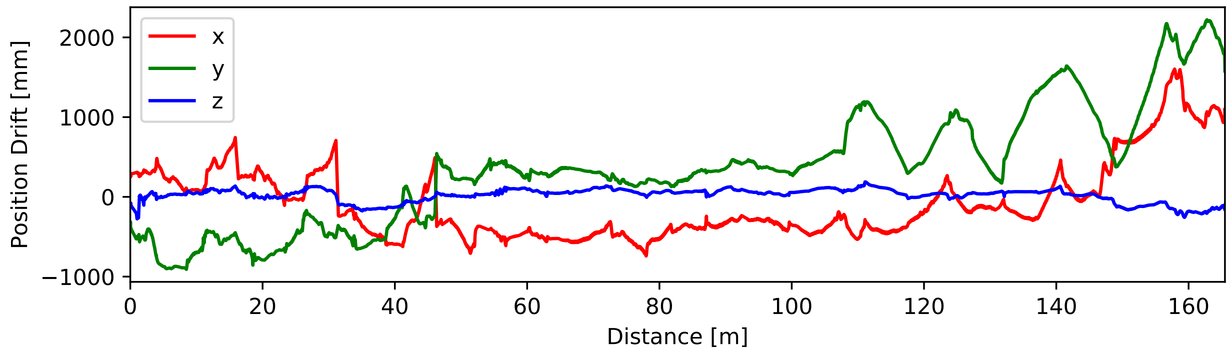
Figure 6.26: The average age of the features in the scene at any given time during the *sync* dataset. The discontinous jumps in the graph occurs when there is no tracked features. Blue is the average lifetime of the of the visual features and yellow is the average lifetime of the thermal features.

6.6 LT3

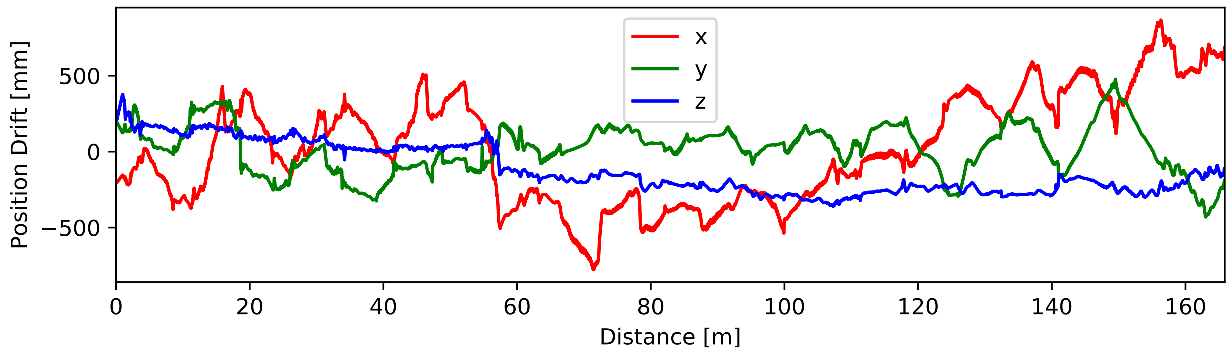
The *lt3* sequence has some thermal augmentation, but less than the *sync* sequence. In isolation this makes it more difficult to use the thermal modality for tracking, but the conditions are still good. The illumination in the scene is a single lamp illuminating parts of it. In parts of the sequence the cameras are pointed away from the illuminated part, which makes visual tracking more difficult. Nevertheless, ROVIO finishes the whole sequence without losing track as in the *sync* sequence.

The large spikes in the beginning in fig. 6.28 are due to a failure in the vicon tracking which was not fully filtered out. This also results in other smaller spikes in that figure and other figures in this chapter.

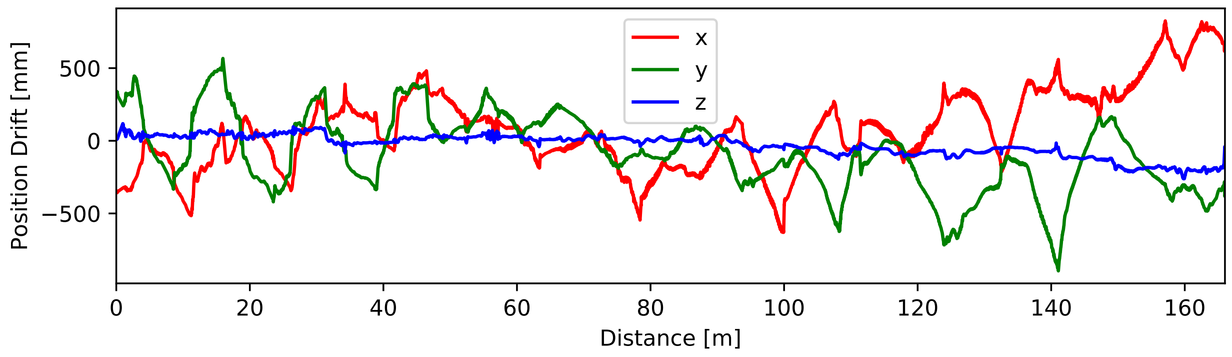
Figure 6.27 shows the deviation from ground truth for the three different methods.



(a) ROVIO

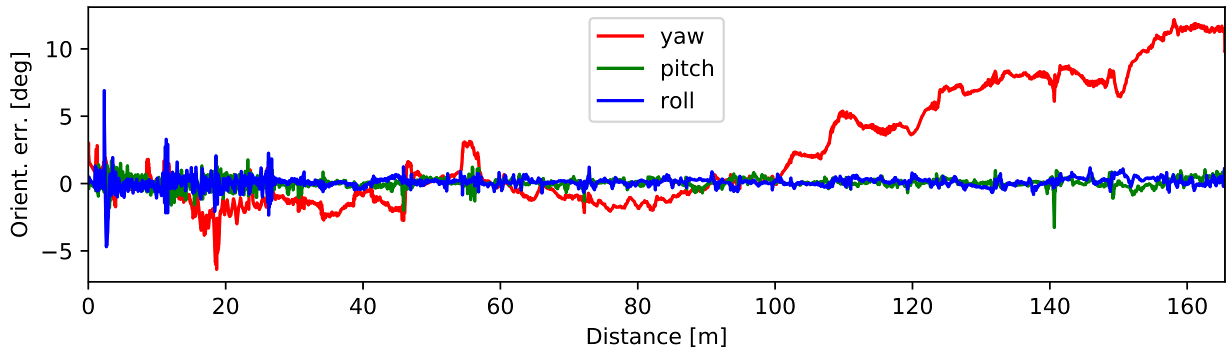


(b) ROTIO

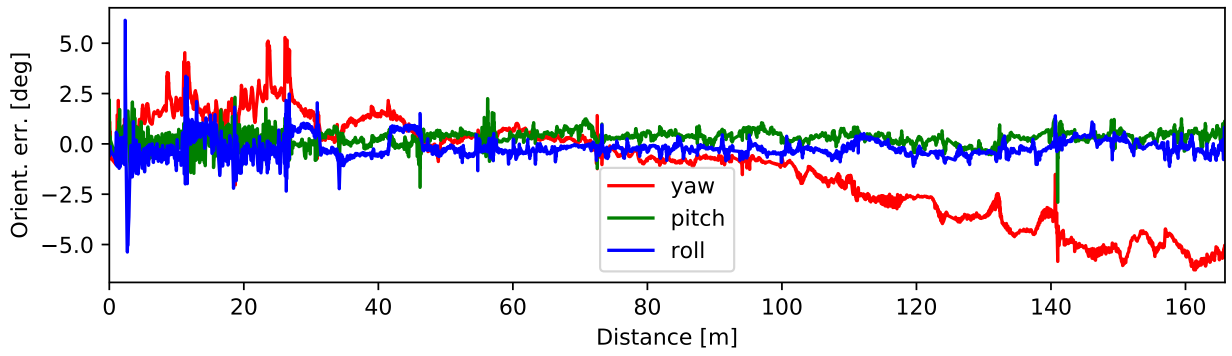


(c) ROVTIO

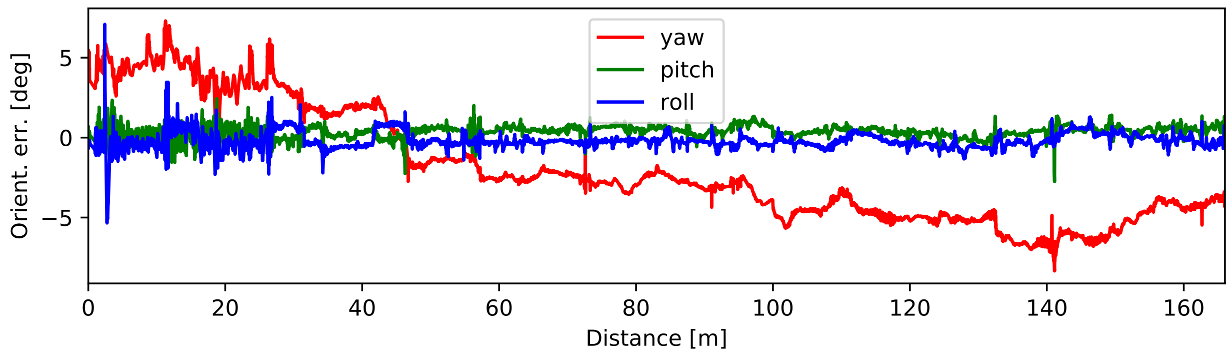
Figure 6.27: The translational deviation from ground truth compared with distanced traveled for the *lt3* dataset.



(a) ROVIO

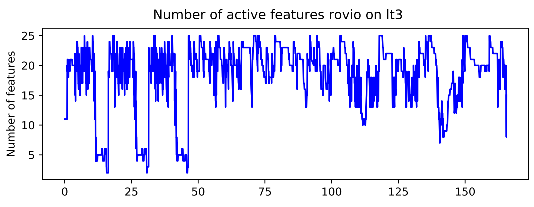


(b) ROTIO

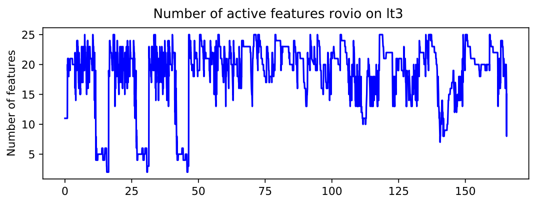


(c) ROVTIO

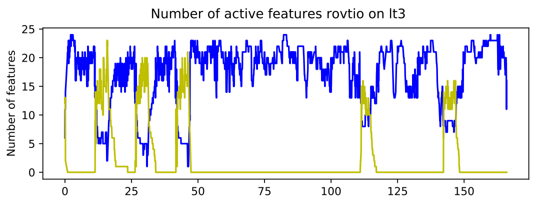
Figure 6.28: The rotational deviation from ground truth compared with distanced traveled for the *lt3* dataset.



(a) ROVIO

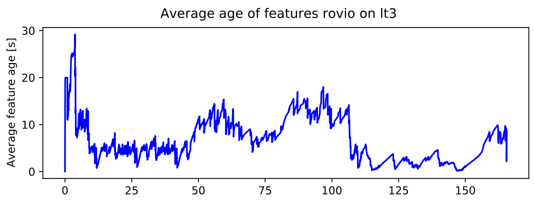


(b) ROTIO

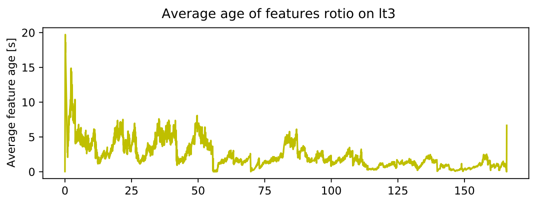


(c) ROVTIO

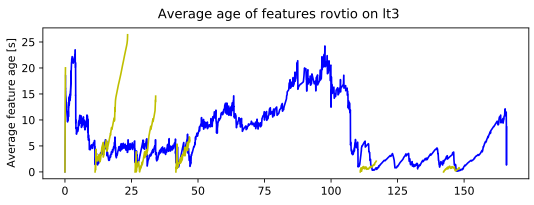
Figure 6.29: The number of tracked features over time *lt3* dataset.



(a) ROVIO



(b) ROTIO



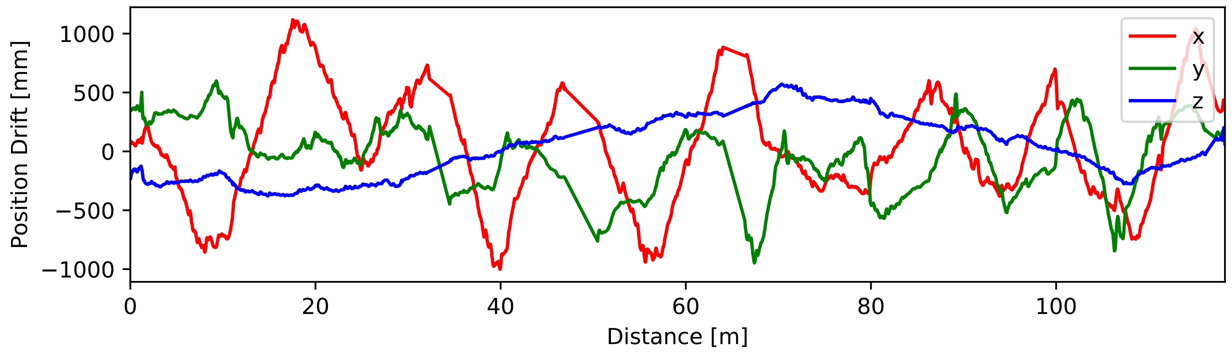
(c) ROVTIO

Figure 6.30: The average age of the features in the scene at any given time during the *lt3* dataset. The discontinous jumps in the graph occurs when there is no tracked features.

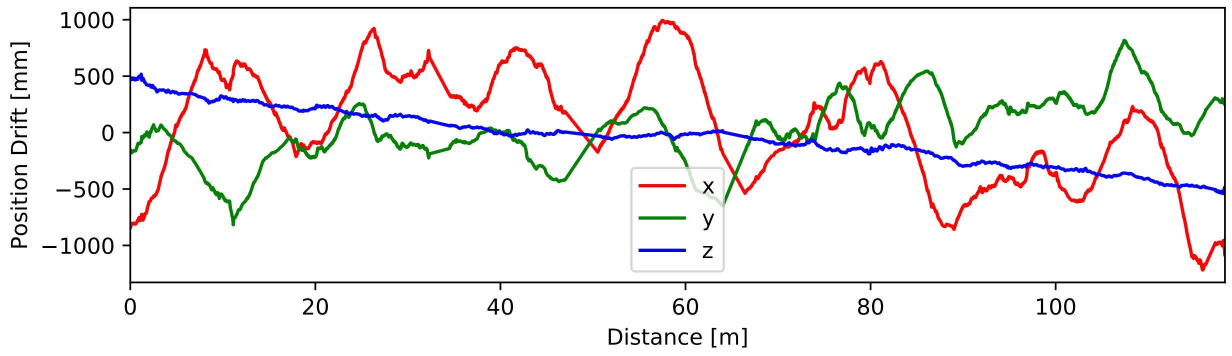
6.7 ALT2

The *alt2* sequence has good illumination, but no thermal augmentation. It is still indoors so there are quite a lot of good thermal features. However, there are none of the very strong ones which are in the sequences with thermal augmentation.

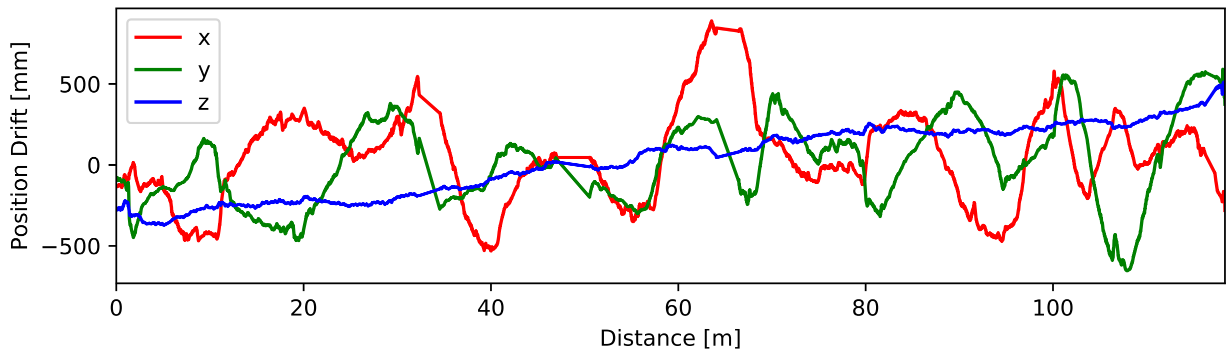
Figures 6.31-6.34 show the errors and the feature statistics during the run. An interesting result to notice is that ROVTIO after approximately 100m does a sharp turn away from the true yaw. Figure 6.34c shows that this coincides with a gradual shift from visual to thermal and an abrupt change back to visual again.



(a) ROVIO

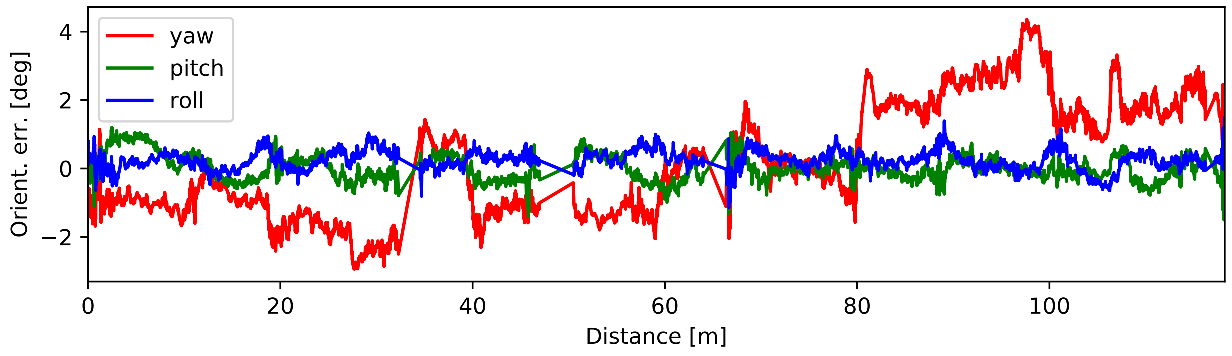


(b) ROTIO

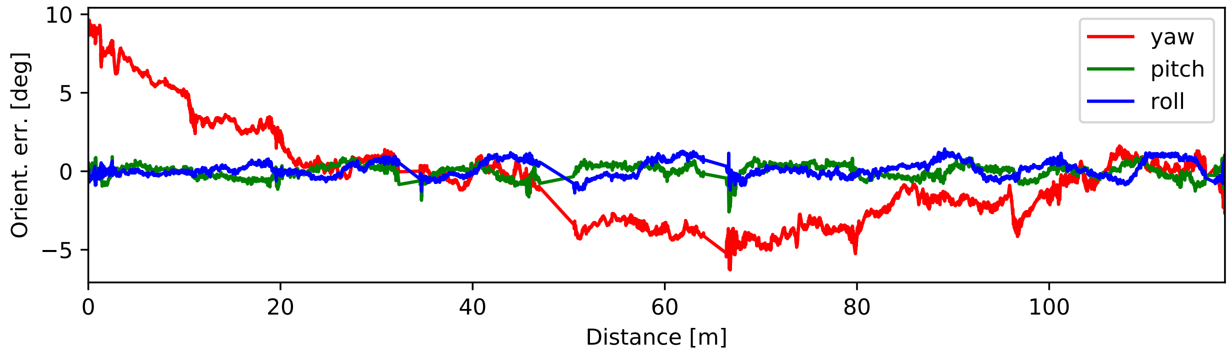


(c) ROVTIO

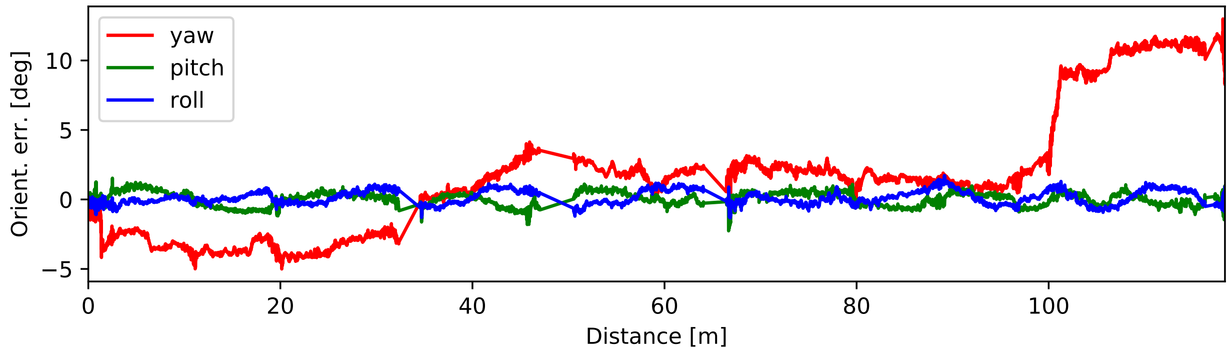
Figure 6.31: The translational deviation from ground truth compared with distanced traveled for the *alt2* dataset.



(a) ROVIO

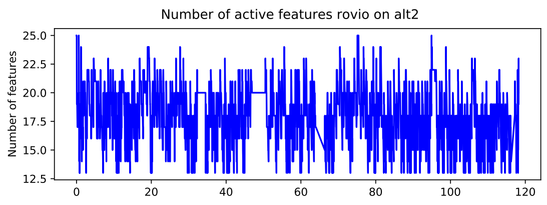


(b) ROTIO

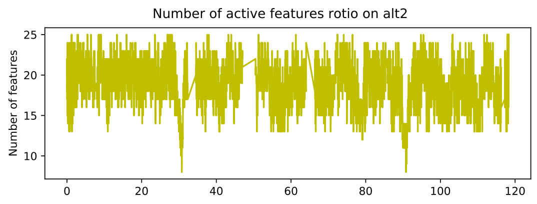


(c) ROVTIO

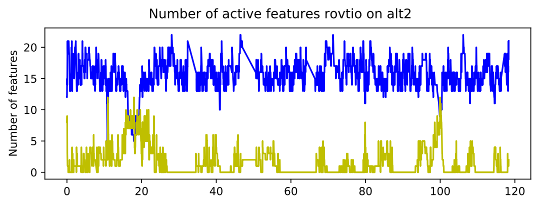
Figure 6.32: The rotational deviation from ground truth compared with distanced traveled for the *alt2* dataset.



(a) ROVIO

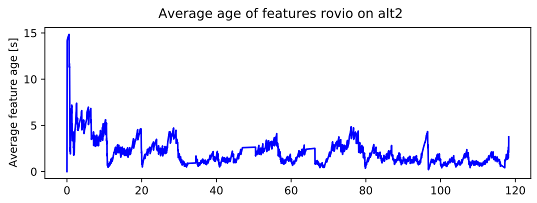


(b) ROTIO

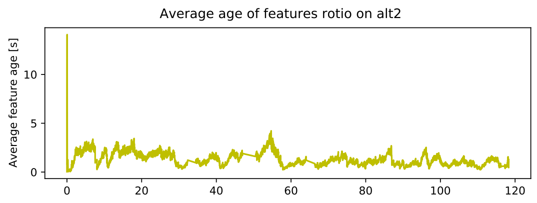


(c) ROVTIO

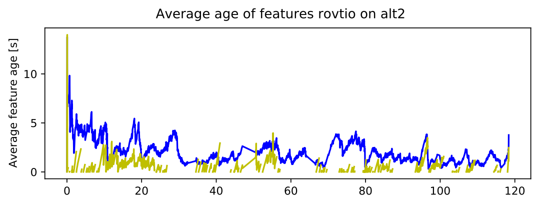
Figure 6.33: The number of tracked features over time *alt2* dataset.



(a) ROVIO



(b) ROTIO



(c) ROVTIO

Figure 6.34: The average age of the features in the scene at any given time during the *alt2* dataset. The discontinous jumps in the graph occur when there are no tracked features.

6.8 Direct and indirect

ROVIO, and thereby ROVTIO, also implements an indirect mode where the update instead is based on KLT tracking and a traditional reprojection error update. It uses the same feature detection, same pixels and same gradients as the direct update, but uses them first to track the location of the patches and then the optical flow to calculate the movement. This should in theory give a good comparison between the two estimation formulations. The update noise is not directly comparable, but approximately tuned to best performance in both cases.

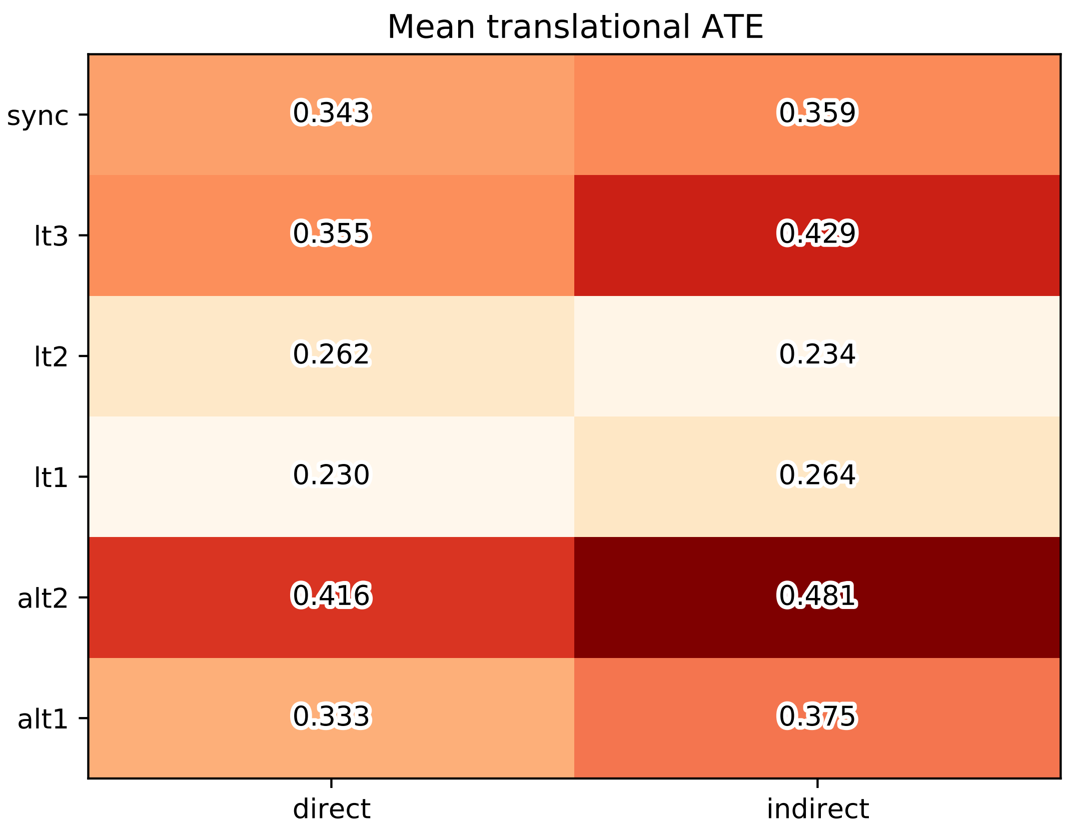


Figure 6.35: The mean translational ATE in meters for the direct and indirect formulation of ROVIO on the different datasets.

Figures 6.35-6.40 show the results on the different datasets. We see that the direct approach performs slightly better than the indirect approach by most measures. The exceptions are the rotational error metrics in some of the sequences.

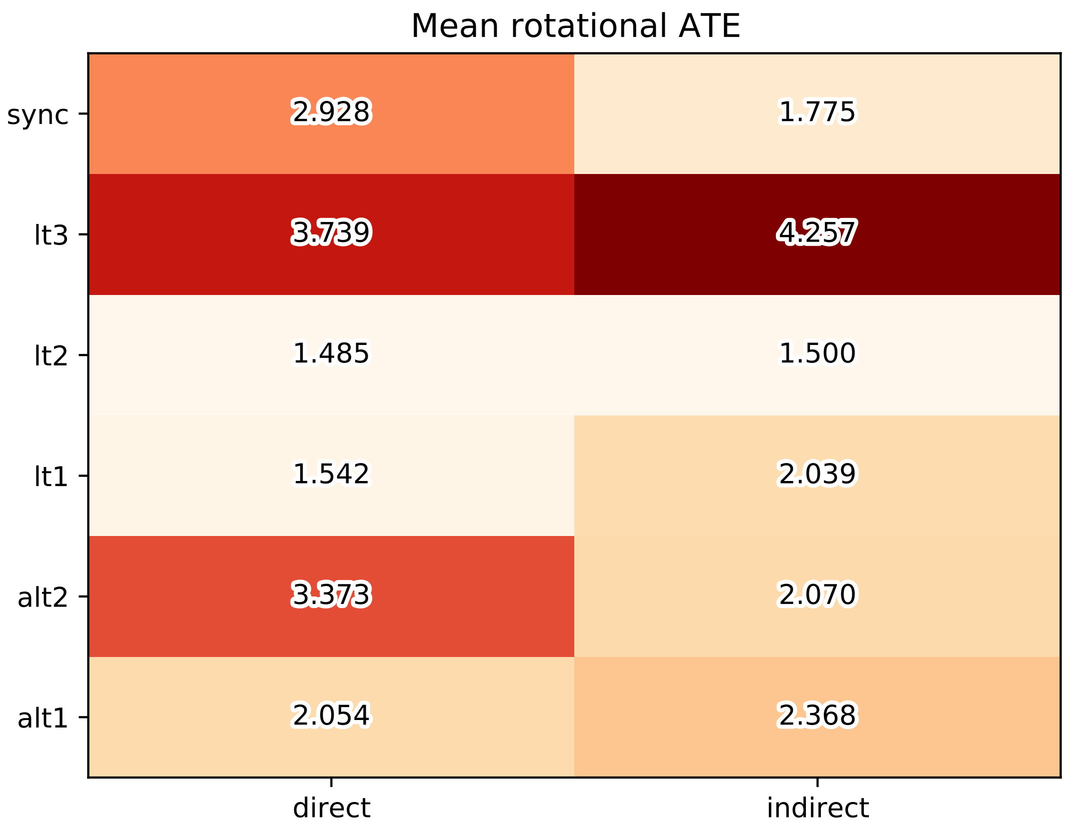


Figure 6.36: The mean rotational ATE in degrees for the direct and indirect formulation of ROVIO on the different datasets.

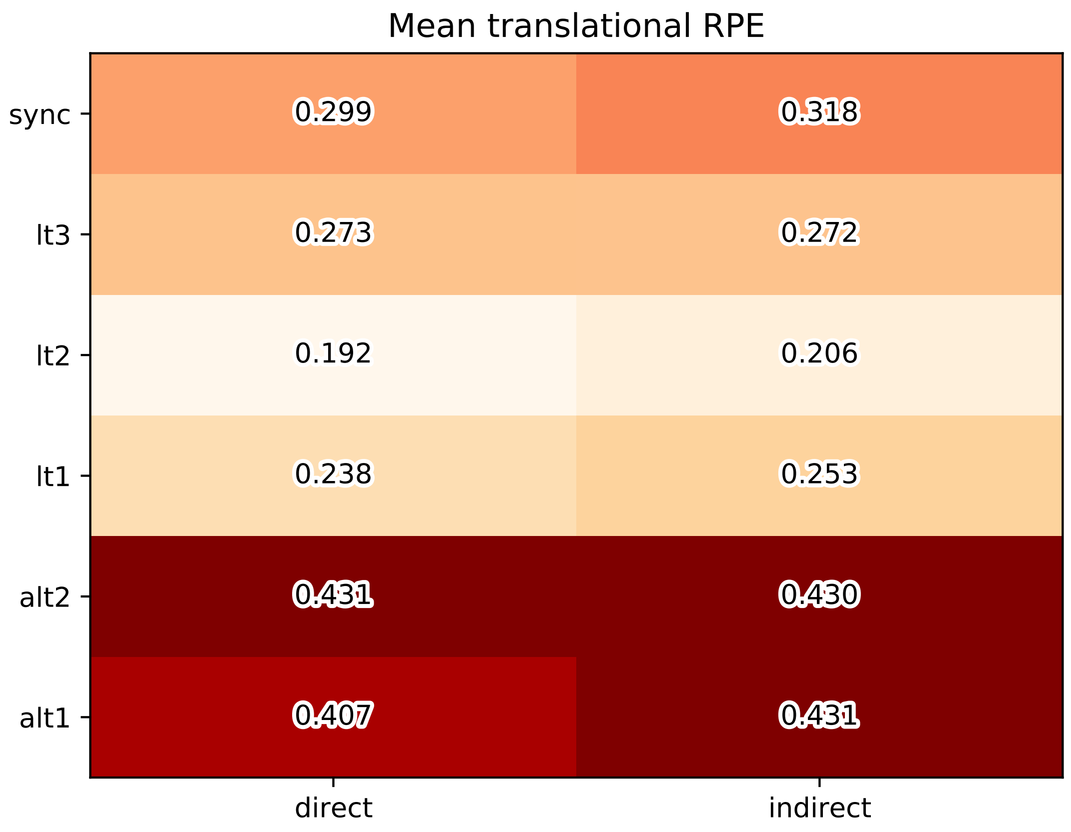


Figure 6.37: The mean translational RPE in meters over a 3m interval for the direct and indirect formulation of ROVIO on the different datasets.

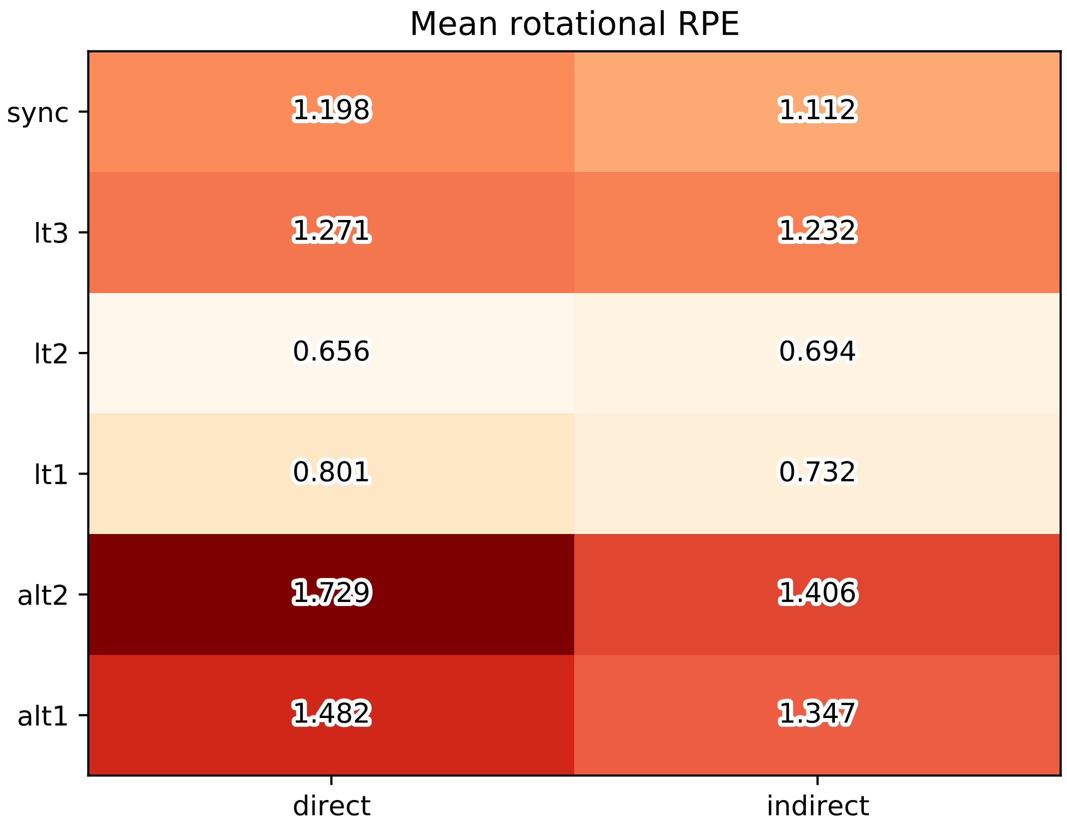


Figure 6.38: The mean rotational RPE for the direct and indirect formulation of ROVIO on the different datasets.

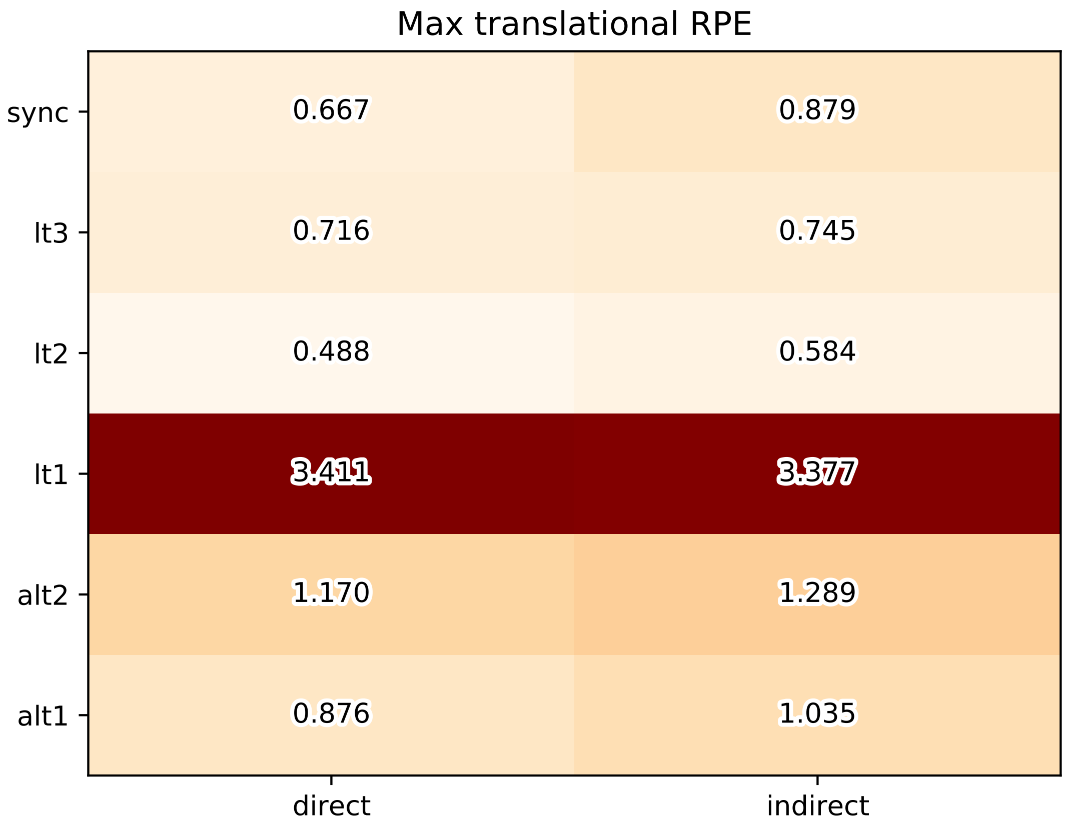


Figure 6.39: The max translational RPE in meters over a 3m interval for the direct and indirect formulation of ROVIO on the different datasets.

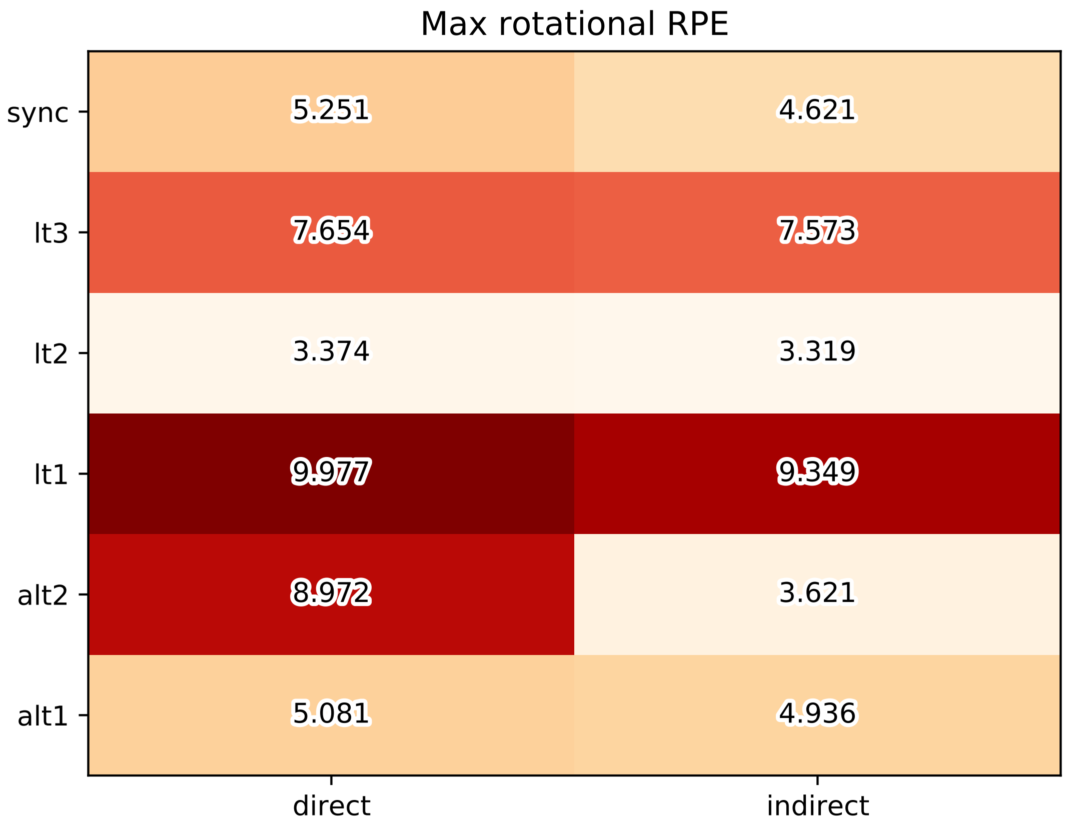


Figure 6.40: The max rotational RPE in degrees over a 3m interval for the direct and indirect formulation of ROVIO on the different datasets.

Chapter 7

Discussion

7.1 Selecting a starting point.

For this project it is desirable to start with an existing method to save some time implementing it. There are several relevant methods that can be used, which are listed in table 3.1 and table 3.2.

The goal is to develop an odometry estimation algorithm using inertial measurements, visual images and thermal images for greater robustness. Starting out with a visual inertial odometry algorithm is then feasible, since it already implements some of the key aspects.

The next question is whether a graph based method or a filter based method is the most feasible. [23] argues that graph based methods are the better choice, but there are several differences between this case and their assumptions. The first difference is that this case is visual inertial, but [23] analyzes a pure visual scenario. Adding inertial measurements to the graph in the form of preintegrated factors [44] significantly increases the computational load of the system [17]. However, it ingrates very well in the prediction step of filtering methods. Secondly, [23] only considers EKFs, but other more sophisticated filters have been developed, e.g. IEKF from [1]. There has also been significant progress for graph based methods, most notably the incremental methods [21]. The third difference which favors filter based methods is the need for a computationally fast algorithm. Since the adaption will process the thermal images and the visual images in separate updates, the algorithm needs to be fast enough to process them at a rate twice the normal framerate. The discussion in [23] argues that there are indications suggesting that filtering methods are the most feasible when the processing budget is limited.

It is not obvious whether a direct, indirect or semi-direct method is the most feasible. The trend in the visual navigation community seems to be going away from dense direct methods to sparse direct methods [15] [27] [45]. Earlier work indicates that descriptors made for visual images are not directly transferable to thermal images [46]. This leaves us with direct sparse algorithms and klt-based indirect algorithms, which both may be feasible options.

Several of the methods mentioned in chapter 3 are various types of VSLAM methods which also use either mid term associations or long term associations. This is out of scope for this thesis, but these methods can still be used as a starting point.

ROVIO is selected as the starting point after these considerations and good results in the preliminary tests. Earlier work has also been done on ROVIO with the Autonomous robots lab, which simplifies the creation of ROVTIO.

7.2 Evaluation

7.2.1 Artificial degregation

The main strong and persistent pattern observable from the artificial degregation study is that more degregation leads to worse metric results. The single modality methods have a gradual change of worsening performance with increasing degregation and ROVTIO gets gradually worse when both get worse. This is particularly visible in figure 6.13. This is to be expected as the different algorithms have less features to track. There are some outliers in the others, and the column with a thermal degregation 0.05 could appear to be better than the columns immediately to the left in fig. 6.11, but this might be a dataset specific issue from which not many conclusions can be drawn.

The second observation we can make is that ROVTIO is persistently performing better that the two single modalities. This indicates that ROVTIO can exploit the data from both modalities even when both are poor or one is poor and the other is good.

The last observation from this experiment to mention is that ROVTIO shows robustness to one of the modalities being too degraded for the single modality keep track. I.e. if one of the modalities has low degregation, ROVTIO will perform relatively well, at least compared to when both are good. This indicates ROVTIO has some added robustness compared to ROVIO and ROTIO. Also when both modalities fail to track, i.e. the upper half of the right most column in fig. 6.13, the error is far smaller than for the single modality methods. 30% mean RPE drift is quite high and in most scenarios unacceptable. Still, the filter does not completely lose track and diverges as in the case of ROVIO and ROTIO. While still not useful, it indicates that some robustness is gained in this case too. Future development may be able to exploit this better and achieve acceptable accuracy.

7.2.2 Handoff

The results in section 6.4 show that the handoff from thermal to visual has a far smaller negative impact than the handoff from visual to thermal. The main reason is probably that from thermal to visual it finds a few features before thermal dies, but from visual to thermal there is a brief moment where it has to reinitialize. From figure 6.17 it seems as if the scale is suffering from the handoff and needs to be reinitialized. This supports the hypothesis of the few features obtained during the handoff being the difference. These features can then be initialized with the accurate scale from the existing thermal map and then be used afterwards. For the case of visual to thermal there are no new features and the scale has to converge from the initial guess. In theory, the correct scale and depth of the features could have been obtained from the estimated motion from the IMU. The biases of the IMU have converged and the motion could probably have estimated relatively accurately for the next seconds. ROVTIO (as with ROVIO), however, does not do this, it initialized all features at a fixed depth the frame they are observed. The depth of the features probably converges faster with the already estimated biases compared to the initial initialization, but not fast enough to avoid a significantly reduced performance.

The relevance of this experiment to real world handoffs is not one to one. The abrupt stop of the image stream makes the handoff easier to detect for the algorithm, so it does not try to keep finding features in the declining modality. Perhaps more importantly it goes the other way. The handoff in the sequences used in this thesis occurs due to the camera moving between regions with different illumination. Then there is time during the transition where the old modality is getting worse and ROVTIO starts finding features in the new modality. It also means that features that were good enough for initialization can be tracked until they leave the field of view.

Despite the constraints of this experiment it indicates that it is feasible with a few features in the new modality during a handoff. One could then argue that ROVTIO should keep a minimum of features in each modality at all times, instead of possibly relying on only one of them. Then ROVTIO would always be prepared for a handoff.

7.2.3 Direct and indirect

We see from the figures 6.35-6.40 that the direct approach in total performs slightly better than the indirect approach in estimating translation. This is in line with the results from the original ROVIO paper [1]. For rotation the overall impression is the opposite, that the indirect approach slightly outperforms the direct approach. Poor estimates of rotation will also impact the estimated translation in the long run, but it does not seem to change the trend for these sequences. The difference in rotation error is relatively small and the sequences are relatively short.

The issue of fusing the two different modalities that are not directly comparable could in theory have been mitigated by using an indirect method. Then we are no longer fusing measurements from different sensors, but bearing vector correspondences in pixels. This experiment, however, indicates that this does not help, and that the direct fusion approach is still feasible.

Chapter 8

Conclusions, Discussion, and Recommendations for Further Work

8.1 Summary and Conclusions

This thesis introduces, describes and evaluates ROVTIO, which can fuse asynchronous thermal, visual and inertial measurements for estimating the odometry of the robot. The evaluation indicates that ROVTIO can use both modalities to improve the accuracy in degraded coditions. The robustness is increased for ROVTIO compared to ROVIO/ROTIO by being able to rely on the other modality when one modality is poor. The integrity is nonexistent for both of ROVTIO and ROVIO, it has not been a focus for this thesis. Neither has the continuity, but it is still slightly better than nothing due to the IMU,the same as ROVIO. The latency is not studied extensively in this thesis, but it was experimentally observed that the latency of ROVTIO is higher than that of ROVIO.

The handoffs study indicates that ROVTIO does not handle handoffs from visual to thermal well. This is to some extent visible in the real world sequences which contain handoffs, but the resulting drift is much smaller. I.e. it seems like it handles real world handoffs better than the artificial ones. This can probably be reduced by always keeping a few features in both modalities or by reducing the influence of the “Historical modality score” indicator in the modality selection. Experience from the development of ROVTIO indicates that this will lead to a slight reduction in the accuracy.

In the debate between direct methods and indirect methods, this thesis indicates that direct methods are favorable. This is, however, the same result that ROVIO [1] had, so the result has to be considered to be expected. However, this thesis indicates that direct is the better option also on thermal and also when fusing both.

8.2 Further Work

This project identified several interesting direction for future work:

1. The work in this thesis was constrained by the sensor platform used in the assessed datasets. The most significant constraint is that the visual camera and thermal camera are not triggering simultaneously. If the cameras had been triggered simultaneously the update step could have been implemented as a joint update of both modalities. This would lead to an even update rate, it would be a smaller compromise to differentiate between the modalities, the computational load would be smaller since the update overhead could be done rarer. This would probably also in isolation make the update more precise and increase accuracy.

2. ROVTIO (and ROVIO) initialize the map by initializing the features at a fixed depth and then require the feature depth to converge to the true depth during operation. This is nice because ROVIO can start providing localization data from the second frame, but the reliability of the localization before the convergence of the filter is poor. It is also observed that this approach is susceptible to divergence in the beginning. E.g., the failure of ROVIO on the *sync* sequence was in the beginning before the filter had converged. It is an interesting line of future work to explore an explicit SfM-based visual thermal inertial initialization procedure. E.g. adapt something like the ORBSLAM3 initialization [47] or the VINS-mono initialization [48].

3. This final point is not really a future work direction in visual thermal inertial fusion, but only in thermal (inertial) navigation. Thermal cameras are in theory factory-calibrated to give the correct temperature, but in some of the sequences, lens artifacts are visible. Typically there was a little vignetting in the corner(s) of the image. The thermal images also have visible, persistent noise along the rows and columns of the pixel array. It would be an interesting future study to explore options for photometric/radiometric calibration of the thermal cameras. This is typically done by the manufacturer, but it would be interesting to see whether better performance can be obtained.

Of course, there are many other things that can be improved upon ROVTIO, including the modality selection criteria and runtime optimizations. However, the ones listed above are in the opinion of the author the most interesting self-contained aspects to study next.

Appendix A

Acronyms

ARS Attitude rate sensor

ATE Absolute trajectory error

DSO Direct sparse odometry [15]

DoF Degree of freedom

EKF Extended kalman filter

FAST Fast accelerated segment test [38]

FoV Field of view

GNSS Global navigation satellite system

IEKF Iterated extended kalman filter

IMU Inertial measurement unit

MAP Maximum a posteriori

MAV Micro aerial vehicle

NED North East Down

ROVIO Robust visual inertial odometry [1]

RPE Relative pose error

SLAM Simultaneous localization and mapping

SNR Signal to noise ratio

TIO Thermal inertial odometry

TO Thermal odometry

VIO Visual inertial odometry

VO Visual odometry

VSLAM Visual simultaneous localization and mapping

Bibliography

[1] Michael Bloesch, Michael Burri, Sammy Omari, Marco Hutter, and Roland Siegwart. Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback. *The International Journal of Robotics Research*, 36(10):1053–1072, 2017. doi: 10.1177/0278364917728574. URL https://doi.org/10.1177/0278364917728574.

[2] OpenCV. Camera calibration and 3d reconstruction. URL https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html?highlight=findchess.

[3] J. Kannala and S. Brandt. A generic camera calibration method for fish-eye lenses. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 1, pages 10–13 Vol.1, 2004. doi: 10.1109/ICPR.2004.1333993. URL https://ieeexplore.ieee.org/document/1333993?arnumber=1333993.

[4] V. Usenko, N. Demmel, and D. Cremers. The double sphere camera model. In *Proc. of the Int. Conference on 3D Vision (3DV)*, September 2018. URL https://vision.in.tum.de/research/vslam/double-sphere.

[5] Zichao Zhang, Henri Rebecq, Christian Forster, and Davide Scaramuzza. Benefit of large field-of-view cameras for visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[6] V. Usenko, N. Demmel, D. Schubert, J. Stueckler, and D. Cremers. Visual-inertial mapping with non-linear factor recovery. *IEEE Robotics and Automation Letters (RA-L) Int. Conference on Intelligent Robotics and Automation (ICRA)*, 5(2):422–429, 2020. doi: 10.1109/LRA.2019.2961227. URL https://vision.in.tum.de/research/vslam/basalt.

[7] Joern Rehder, Janosch Nikolic, Thomas Schneider, Timo Hinzmann, and Roland Siegwart. Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4311, 2016. doi: 10.1109/ICRA.2016.7487628. URL https://ieeexplore.ieee.org/document/7487628.

[8] Richard Szeliski. *Computer Vision: Algorithms and Applications 2nd Edition*. Springer, 2020. URL http://szeliski.org/Book/. preprint.

[9] J. Engel, V. Usenko, and D. Cremers. A photometrically calibrated benchmark for monocular visual odometry. In *arXiv:1607.02555*, July 2016. URL https://vision.in.tum.de/_media/spezial/bib/engel2016monodataset.pdf.

[10] Robert Grover Brown and Patrick Y. C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering*, volume Fourth edition. John Wiley Sons, Inc, 2012. ISBN 978-0-470-60969-9.

[11] Gavriel A. Terejanu. Extended kalman filter toutorial. URL https://cse.sc.edu/~terejanu/files/tutorialEKF.pdf.

[12] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012. URL https://vision.in.tum.de/_media/spezial/bib/sturm12iros.pdf.

[13] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of The Optical Society of America A-optics Image Science and Vision*, 4:629–642, 1987.

[14] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:756–770, 2004.

[15] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, mar 2018. URL https://vision.in.tum.de/_media/spezial/bib/engel_et_al_pami2018.pdf.

[16] Antoni Rosinol, M. Abate, Yun Chang, and L. Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1689–1696, 2020. URL https://arxiv.org/abs/1910.02490.

[17] Carlos Campos, Richard Elvira, Juan J. Gomez, Jose M. M. Montiel, and Juan D. Tardos. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *arXiv preprint arXiv:2007.11898*, 2020. URL https://arxiv.org/pdf/2007.11898.pdf.

[18] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, 2014. doi: 10.1109/ICRA.2014.6906584. URL http://rpg.ifi.uzh.ch/docs/ICRA14_Forster.pdf.

[19] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015. doi: 10.1177/0278364914554813. URL https://doi.org/10.1177/0278364914554813.

[20] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008. doi: 10.1109/TRO.2008.2006706. URL https://ieeexplore.ieee.org/document/4682731.

[21] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012. doi: 10.1177/0278364911430419. URL https://doi.org/10.1177/0278364911430419.

[22] Haomin Liu, Mingyu Chen, Guofeng Zhang, Hujun Bao, and Yingze Bao. Ice-ba: Incremental, consistent and efficient bundle adjustment for visual-inertial slam. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1974–1982, 2018. doi: 10.1109/CVPR.2018.00211. URL https://ieeexplore.ieee.org/document/8578309.

[23] H. Strasdat, J. Montiel, and A. Davison. Visual slam: Why filter? *Image Vis. Comput.*, 30: 65–77, 2012. URL http://www.doc.ic.ac.uk/~ajd/Publications/strasdat_etal_ivc2012.pdf.

[24] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for rgb-d cameras. In *International Conference on Robotics and Automation (ICRA)*, May 2013. URL https://vision.in.tum.de/_media/spezial/bib/kerl13icra.pdf.

[25] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*, September 2014. URL https://vision.in.tum.de/research/vslam/lsdslam.

[26] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327, 2011. doi: 10.1109/ICCV.2011.6126513. URL https://ieeexplore.ieee.org/document/6126513.

[27] J. Zubizarreta, I. Aguinaga, and J. M. M. Montiel. Direct sparse mapping. *IEEE Transactions on Robotics*, 36(4):1363–1370, 2020. doi: 10.1109/TRO.2020.2991614. URL https://ieeexplore.ieee.org/document/9102352.

[28] X. Gao, R. Wang, N. Demmel, and D. Cremers. Ldso: Direct sparse odometry with loop closure. In *International Conference on Intelligent Robots and Systems (IROS)*, October 2018. URL https://vision.in.tum.de/research/vslam/ldso.

[29] S. H. Lee and Javier Civera. Loosely-coupled semi-direct monocular slam. *IEEE Robotics and Automation Letters*, 4:399–406, 2019. URL https://arxiv.org/abs/1807.10073v1.

[30] Shehryar Khattak, Christos Papachristos, and Kostas Alexis. Keyframe-based thermal–inertial odometry. *Journal of Field Robotics*, 37(4):552–579, 2020. doi: https://doi.org/10.1002/rob.21932. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21932.

[31] Anastasios I. Mourikis and Stergios I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572, 2007. doi: 10.1109/ROBOT.2007.364024. URL https://ieeexplore.ieee.org/document/4209642.

[32] Zheng Huai and Guoquan Huang. Robocentric visual-inertial odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6319–6326, 2018. doi: 10.1109/IROS.2018.8593643. URL https://ieeexplore.ieee.org/document/8593643.

[33] Xiaochen Qiu, Hai Zhang, Wenxing Fu, Chenxu Zhao, and Yanqiong Jin. Monocular visual-inertial odometry with an unbiased linear system model and robust feature tracking front-end. *Sensors*, 19(8), 2019. ISSN 1424-8220. doi: 10.3390/s19081941. URL https://www.mdpi.com/1424-8220/19/8/1941.

[34] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007. URL http://www.robots.ox.ac.uk/~gk/publications/KleinMurray2007ISMAR.pdf.

[35] Tong Qin, Shaozu Cao, Jie Pan, and Shaojie Shen. A general optimization-based framework for global pose estimation with multiple sensors, 2019.

[36] Raul Mur-Artal and Juan D. Tardos. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. doi: 10.1109/TRO.2017.2705103. URL https://ieeexplore.ieee.org/document/7946260.

[37] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. Robust visual inertial odometry using a direct ekf-based approach. In *2015 IEEE/RSJ International Conference on Intelligent Robots*

and Systems (IROS), pages 298–304, 2015. doi: 10.1109/IROS.2015.7353389. URL https://ieeexplore.ieee.org/document/7353389.

[38] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32:105–119, 2010. doi: 10.1109/TPAMI.2008.275. URL http://lanl.arXiv.org/pdf/0810.2434.

[39] B.M. Bell and F.W. Cathey. The iterated kalman filter update as a gauss-newton method. *IEEE Transactions on Automatic Control*, 38(2):294–297, 1993. doi: 10.1109/9.250476. URL https://www.cc.gatech.edu/~bboots3/STR-Spring2018/readings/IKF_GaussNewton.pdf.

[40] Shehryar Khattak, Huan Nguyen, Frank Mascarich, Tung Dang, and Kostas Alexis. Complementary multi-modal sensor fusion for resilient robot pose estimation in subterranean environments. In *2020 IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, 2020.

[41] JJ Yeong Sang Park, Youngsik Shin, and Ayoung Kim. Vivid: Vision for visibility dataset. In *2019 IEEE Int. Conf. Robot. Automat. Workshop (ICRAW)*, 2019. URL https://irap.kaist.ac.kr/publications/alee-2019-icra-ws.pdf.

[42] Shehryar Khattak, Frank Mascarich, Tung Dang, Christos Papachristos, and Kostas Alexis. Robust thermal-inertial localization for aerial robots: A case for direct methods. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1061–1068. IEEE, 2019.

[43] Tung Dang, Marco Tranzatto, Shehryar Khattak, Frank Mascarich, Kostas Alexis, and Marco Hutter. Graph-based subterranean exploration path planning using aerial and legged robots. *Journal of Field Robotics*, 37(8):1363–1388, 2020.

[44] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2017. doi: 10.1109/TRO.2016.2597321. URL https://ieeexplore.ieee.org/document/7557075.

[45] Trym Vegard Haavardsholm. Ttk21 introduction to visual simultaneous localization and mapping (vslam). 2020.

[46] S. Khattak, C. Papachristos, and K. Alexis. Visual-thermal landmarks and inertial fusion for navigation in degraded visual environments. In *2019 IEEE Aerospace Conference*, pages

1–9, 2019. doi: 10.1109/AERO.2019.8741787. URL https://ieeexplore.ieee.org/document/8741787.

[47]

[48] Tong Qin and Shaojie Shen. Robust initialization of monocular visual-inertial estimation on aerial robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4225–4232, 2017. doi: 10.1109/IROS.2017.8206284. URL https://ieeexplore.ieee.org/abstract/document/8206284.