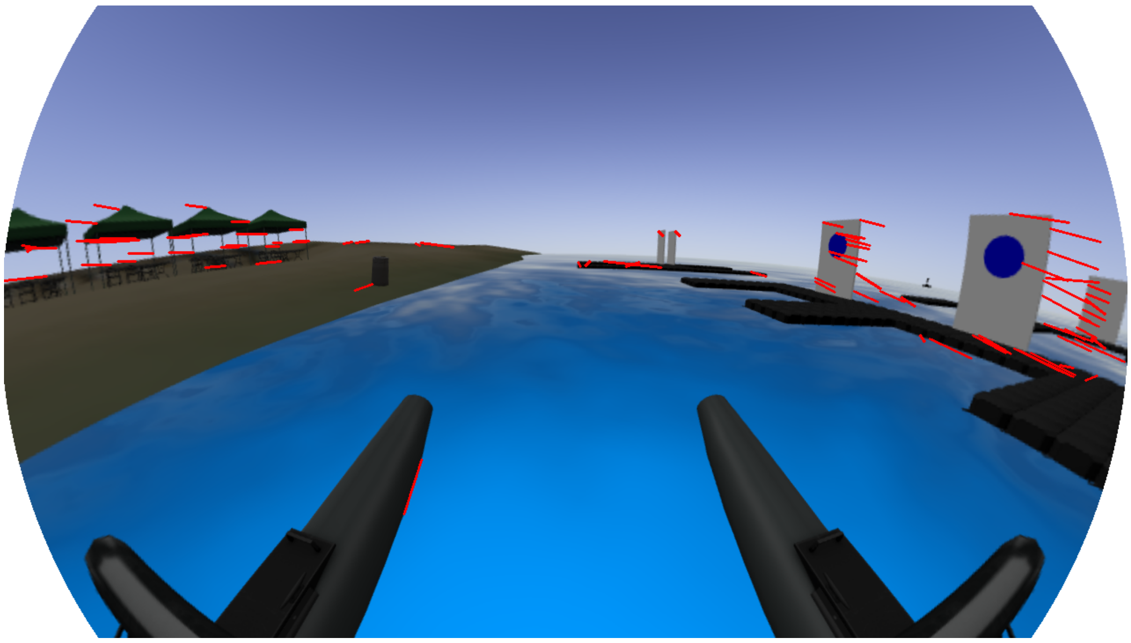
CHALMERS
UNIVERSITY OF TECHNOLOGY



Automatic Calibration of Multiple Fisheye Cameras

Trajectory based Extrinsic Calibration of a Multicamera Rig

Master's thesis in Systems, Control and Mechatronics

OLA SÆTHERN & DAVID HEIMAN

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021:38

Automatic Calibration of Multiple Fisheye Cameras

Trajectory based Extrinsic Calibration of a Multicamera Rig

DAVID HEIMAN
OLA SAETHERN

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Adaptive Systems Research Group
Chalmers University of Technology
Gothenburg, Sweden 2021

Automatic Calibration of Multiple Fisheye Cameras
Trajectory based Extrinsic Calibration of a Multicamera Rig
Ola Sæthern
David Heiman

Supervisor: Yaowen Xu, CPAC Systems AB
Examiner: Peter Forsberg, Department of Mechanics and Maritime Sciences

Master's Thesis 2021:38
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Adaptive Systems Research Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Feature tracking visualized in Gazebo VRX Simulator, with a customized setup of vehicle and objects.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Automatic Calibration of Multiple Fisheye Cameras
Trajectory based Extrinsic Calibration of a Multicamera Rig
David Heiman
Ola Sæthern
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

This thesis presents a new technique for automatic extrinsic calibration of fisheye cameras on boats. In contrast to most of the existing methods that require overlapping field of view, our technique does not, which opens up for completely free placement of cameras. The method selected for this work was trajectory based calibration, meaning that the fixed relative pose between the cameras can be determined from the trajectories of the cameras. For development, Gazebo VRX simulator was used to generate the data for calibration, and KITTI odometry to evaluate algorithms. The method for generating the trajectories was visual inertial odometry and ground truth trajectories were extracted from the simulator. The calibration method was selected based on the availability of sensors onboard boats, for the objective to generate accurate trajectories. It was found that the developed trajectory based calibration of relative pose between cameras is working, given an accurate trajectory. The method was independent of which algorithm is used to obtain the trajectory. However, there were difficulties getting visual inertial odometry alone to generate trajectories accurate enough for the simulated data sets.

Keywords: VO, VIO, visual odometry, extrinsic calibration, trajectory based calibration, marine, VRX, Gazebo, Computer Vision.

Acknowledgements

This thesis has been done as a final project to our master in Systems, Control and Mechatronics at Chalmers University of Technology. The work was carried out on behalf of CPAC Systems in spring 2021. We want to especially thank our supervisor Yaowen for guidance and brilliant ideas. We want to thank our examiner Peter for actively being present with years of experience with examination of thesis workers. Also for the inspiring talks on the weekly meetings with both Yaowen and Peter. And we want to thank CPAC systems for the opportunity!

Ola Sæthern and David Heiman, Gothenburg, June 2021

Thesis advisor: Yaowen Xu, CPAC Systems AB
Thesis examiner: Peter Forsberg, Department of Mechanics and Maritime Sciences

Contents

List of Figures

List of Tables

Nomenclature

BRIEF	Binary Robust Independent Elementary Feature
CRF	Corner Response Function
DLT	Direct Linear transform
DoG	Difference of Gaussian
EKF	Extended Kalman Filter
EXIF	Exchangeable Image file Format
FAST	Features from Accelerated and Segments Test
FOV	Field Of View
GNSS	Global Navigation Satellite System
IMU	Inertial Measuring Unit
MLE	Maximum Likelihood Estimation
RANSAC	Random Sample Consensus
ROI	Region Of Interest
ROS	Robot Operating System
SIFT	Scale-invariant Feature Transform
SLAM	Simultaneous Locating and Mapping
SVD	Singular Value Decomposition
VIO	Visual Inertial Odometry
VO	Visual Odometry

1

Introduction

In current years, the utilization of image sensors has increased for motor vehicles [17]. Advanced driver assistance systems is one example of its usefulness, helping with driving and parking for increased safety. For increased robustness, a combination with other sensors such as LiDAR and radar can be employed. Those sensors can support the camera in situations with poor visibility or tough light conditions, e.g. direct sunlight, night time and snow. Implementation of camera functions can contribute to the safety of autonomous vehicles, where awareness of surroundings is critical. The awareness is needed to react to events and objects near the vehicle. Image sensors can also be used to reveal blind spots that would otherwise be invisible to the driver.

However, high performance camera calibration is necessary for high quality computer vision applications, and in particular to fully utilize the potential of a camera as a sensor. In practise, calibration is divided into intrinsic and extrinsic. The intrinsic calibration aims at finding the correct way to undistort the camera image whereas the extrinsic to find the correct coordinates of the camera in some known coordinate system. Whenever there is a need to relate the camera to another coordinate system, e.g. when working with stereo cameras, the transform between those coordinate systems is needed. The transform is the cameras extrinsic parameters. For detailed explanations of these concepts, see Section 2.1 Since extrinsic calibration entails finding the relative pose of the cameras with respect to a base, it is necessary for image stitching, 3D reconstruction and more. To do extrinsic calibration, high quality odometry data and good key-frame matching can be used to calculate the transform between the cameras and base frame. Several approaches to gather odometry data are available depending on the sensor modalities: Visual odometry (VO), visual inertial odometry (VIO), laser sensors, global navigation satellite systems as well as loosely or tightly coupled combinations of these.

The focus of this thesis is to build a framework that automatically calibrates the extrinsics of a rig of cameras by driving the vessel around. Inspiration from the different algorithms mentioned in 1.1 has contributed to creating a combination of techniques that result in good trajectories, considering difficulties imposed by the marine simulation environment.

1.1 Related work

Camera calibration is a well investigated field of computer vision, with many approaches and usages. The intrinsic calibration methods are mostly the same, and uses a known pattern e.g. a chessboard. There exists a large variety in camera models, Mei [2] and Kannala-Brandt [29] are some of them. Mei, used in OpenCV, and Kannala-Brandt can handle a wide variety of cameras. MATLAB's calibration toolbox is developed by Scaramuzza [3] and use the Scaramuzza model. The common approach when doing visual odometry is to rectify the fisheye images by normalizing rectifying, using the parameters found doing intrinsic calibration. In direct visual odometry [22], however, greater field of view is reached by not converting fish-eye images to rectified images, at the cost of epipolar lines becoming epipolar curves for the plane-sweeping stereo matching. The approach is shown to have high performance for both motion estimate accuracy and point cloud quality.

The intrinsic parameters are initialized and optimized over several images to perceive a pattern like a chessboard in the right way, all vertical lines should meet in a point. The thesis was mostly concerned with the extrinsic calibration, which can be done in a few different ways. Hand eye calibration is commonly used in robotics when calibrating the perceiving camera to the end effector of a robotic manipulator. G. Carrera proposes a SLAM-based calibration method [19] for a multi camera rig on a small wheeled robot. G. Carrera's method requires the robot to make certain maneuvers which are hard to achieve on a boat.

CamOdoCal [4] uses visual odometry to get the pose and transforms of the cameras, key-frames are selected and used for feature matching between the different cameras to find a relative pose. CamOdoCal also makes use of loop closures which makes it very similar to visual-SLAM. To see the context of visual-SLAM vs. structure from motion (SFM) and VO, see Fig. 1.1. CamOdoCal is a fully automatic calibration pipeline for intrinsic and extrinsic calibration. The same team have also made a quicker variant using a prerecorded track in infrastructure based calibration [5], with similar results.

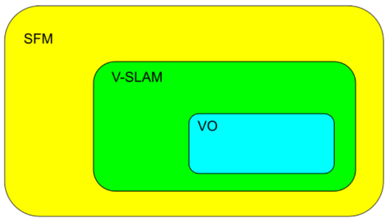


Figure 1.1: How the visual algorithms relate to each other.

In [23] another approach is chosen for doing extrinsic calibration, the existing methods of comparing different odometry trajectories come up short because it can be prone to failure, takes longer time, requires initial guesses and is generally constrained. The approach uses a common plane found in both cameras to get the extrinsics. With only three plane correspondences, good results are achieved. The drawback with [23] approach is that it requires RGB-D cameras.

While visual odometry can work just fine, for applications with sparse features and unreliable/changing features, it can be beneficial to add an additional sensor. In VINS-Mono [14] the integration of an IMU is introduced. The IMU always provides an estimate of pitch and roll which does not drift, along with some initial information about the position by integrating the acceleration twice (deteriorate quickly over time). On the bleeding edge of visual odometry research there is among others OV^{2}SLAM [13], which outperforms the previous best slam methods running real time. The initial approach in this thesis was to attain a trajectory in a similar way as CamOdoCal, but had the odometry aided with the addition of an IMU. The IMU addition was for robustness during time windows of poor features, which is usually the case in marine environments.

1.2 Purpose and goal

The primary goal was to replace the current manual calibration with an automatic calibration of a camera rig. The calibration was to work on a marine vessel of arbitrary size. A part of the purpose behind the task was that time needed for automatic calibration of a camera rig is a fraction of what manual calibration requires. The manual calibration done at time of writing required the boat to be on land to take measurements. Taking the boat on land was not always possible and always costly. A second part of the purpose was that with one algorithm, any marine vessel's camera rig could be calibrated, thereby creating a generalized solution where at present only a case-by-case solution exists. Furthermore, a generalized solution would allow re-calibration in case any camera suffered an off-set as time progressed. After implementation of the calibration, the next step was to use the camera rig to build and maintain a seamless 360° view of the surroundings and additional other views like disparity map, stixels [20] and segmentation if considered useful.

1.3 Problem statement

This thesis targets the problem of accurately calibrating a large camera rig, found on marine vessels. Challenges arise when trying to archive good visual odometry on water. On large vessels there are often multiple cameras arbitrarily placed, which is a factor factor that makes manual calibration a time consuming task, hence creating a need for automatic calibration.

1.4 Limitations

- No considerations were made for feature rich environments with high quality features, which might have been available in a natural harbor or a harbor with varied and high amount of objects.

- The computing power limited the frame rate for the recorded data sets, which had a negative impact on some algorithms.
- For some functions in the software, open-source code was used in order to save time.
- Intrinsics were assumed to be provided either by the camera manufacturer, or from the data sets.

1.5 Contribution

This thesis contributed with a specific calibration algorithm. Most noteworthy were the ability to use non-overlapping cameras and possibility to deploy in environments poor of features, e.g. marine environments. Although we have used existing software parts where possible, our method is unique and offer advantages over existing ditto. The advantages such as being able to use generically obtained trajectories for calibration. In addition, the pipeline we have created is simple to understand and to build upon and was implemented in Python and Robotic Operating System (ROS).

1.6 Thesis outline

The thesis is split into five chapters: introduction, theory, method, results and conclusion. The introduction is a short study of relevant work in similar areas and description of the problem, followed by the theory chapter describing the theory needed to solve the problem of automatic extrinsic calibration. In the methodology chapter, the approach taken and choice of hardware and software are covered. The chapter also covers how the results are collected. The result section compares how well different methods perform and the final chapter concludes the study.

2 Theory

This chapter contains explanations of theory segments necessary for understanding the method.

2.1 Calibration

To utilize the full potential of cameras as sensors, it is important to have well calibrated camera parameters and a correct transform (rotation and translation) between camera pairs in a rig. Furthermore, with well calibrated cameras it is possible to achieve good surround views, disparity calculations etc. In addition, camera matrices can be split into intrinsic and extrinsic parts as such:

$$P = K[R|t] \tag{2.1}$$

Here K is the intrinsic part and the R|t matrix the extrinsic part respectively. Calibration can essentially be regarded as to fit a model to match some a priori known properties.

2.1.1 Intrinsic calibration

Intrinsic calibration is to estimate the camera parameters K which normalizes the image. A normalized image has the principle axis in the image center. The coordinates are normalized and pixels are corrected for skewness, aspect ratio for non square pixels and distortions from the lens. Calibration process of camera intrinsics is usually carried out by capturing a known object, such as a checkerboard pattern, from several views. The inner and outer corners will be detected using an automatic corner detection algorithm or manually selected. The intrinsic parameters needed will differ depending on the camera model of choice, but the focal length and image center is usually initialized with the known values from Exchangeable Image File Format (EXIF) data of the images. The rest of the parameters for distortions, aspect ratio and skew are initially set to 0. The initial values of the intrinsic parameters are refined with a nonlinear optimization method. The objective of the optimization is to minimize the sum of the reprojection errors on all images. To find the intrinsic parameters of a camera, the camera first needs to be modelled. See Section 2.2 for an explanation of how modelling a camera can be done.

Since intrinsic parameters are typically provided by manufacturers, the parameters are assumed to be known in this case.

2.1.2 Extrinsic calibration

The goal of the extrinsic calibration is to estimate the relative position and orientation with respect to a global coordinate frame shown in Fig. 2.1. The estimation is done by finding the rotation matrices and translation vectors for the different cameras. There are a few ways to do this. Manually measuring can be very useful for simple camera rigs but is tedious work on large and complex rigs. There are some automatic ways of doing this with different constraints and methods. The different methods of solving automatic extrinsic calibration will be explained in chapter 3.

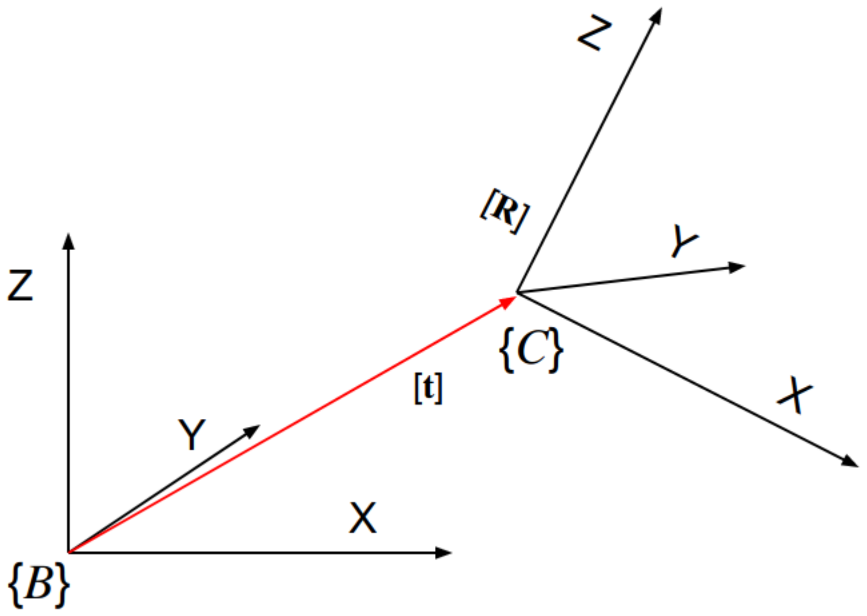


Figure 2.1: A visual interpretation of the camera extrinsics, a rotation [R] and translation [t] is applied on the Base frame {B} to reach the Camera frame {C}.

2.2 Camera models

Various lenses are modelled differently, to represent the projection from the 3D point coordinates to image plane coordinates. The most common approach is to use the pinhole camera model, which assumes no lens and is modelled after the first cameras ever made. Alternatively one can use Kannala-Brandt or Mei models, which better represent the fisheye lens.

2.2.1 Pinhole camera

The pinhole model considers focal lengths, skew, image center and aspect ratio. It assumes the camera center is small enough to be considered a point. The camera matrix of a pinhole camera is the following:

$$x = PX \tag{2.2}$$

$$P = K[R|t] = \begin{bmatrix} \gamma f & sf & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} [R|t] \tag{2.3}$$

Table 2.1: Description of the intrinsic parameters and common values for a camera.

Symbol	**Description**	**Typical value**
f	Focal length	3-1000 mm
s	Pixel skew, if pixel is a parallelogram	$f \tan \alpha \sim 0$
γ	Aspect ratio, for non square pixels	1
$[x_0, y_0]$	Image center	[w/2, h/2]

The normalized points x can then be corrected for radial distortion, which can be done by a simple polynomial model [7] as shown in Eq. (2.4).

$$\begin{aligned} \hat{x} &= x(1 + k_1 r + k_2 r^2) \\ \hat{y} &= y(1 + k_1 r + k_2 r^2) \end{aligned} \tag{2.4}$$

In Eq. (2.4), r is the squared distance from the origin $r = (x^2 + y^2)$. The sign of k_1 usually determines whether it is a barrel or pincushion type of radial distortion. The tangential distortion, on the other hand, is present when the lens is not parallel to the sensor and is often neglected. Despite the fact that the pinhole camera model is a crude approximation of cameras with lenses focusing the light, adjustable aperture and varying focal lengths, it can still handle distortions quite well.

2.2.2 Fisheye

Fisheye lenses are wide angle lenses with extreme field of view (FOV) going up to around 180°. Fisheye lenses is commonly seen in surveillance and rear view cameras because it covers large areas with few cameras. Fisheye cameras do not have a single viewpoint but the small cluster of viewpoints are close enough to be considered as one. Due to the extreme view they offer they can not be modeled with a pinhole camera model and a simple polynomial radial distortion model. Instead there exist a few other commonly used camera models that can capture everything from regular lenses to omnidirectional lenses.

2.2.2.1 Scaramuzza

The Scaramuzza model [3] can represent all view angles from narrow FOV to omnidirectional lenses up to 195°. In this report, we assume that the sensor is centered and aligned with the lens. The following model to capture the lens distortions is proposed:

$$P = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \\ f(\rho) \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \\ a_0 + a_1\rho + a_2\rho^2 + a_3\rho^3 + a_4\rho^4 \end{bmatrix}. \tag{2.5}$$

Here ρ is the distance between principle point and the pixel coordinate $\rho = \sqrt{u^2 + v^2}$. To speed up calibration fewer parameters should be used and a reasonable simplification for fisheye lenses is

$$\left.\frac{df}{d\rho}\right|_{\rho=0} = 0 \tag{2.6}$$

such that $a_1 = 0$. One could potentially choose a higher polynomial degree, but Scaramuzza experienced 4th order reached the best calibration results trade off.

2.2.2.2 Kannala-Brandt model

Kannala-Brandt model [29] is a camera model which covers most types of lenses, and handles fisheye lenses better than the pinhole model. It is used in Infrastructure-Based Calibration of a Multi-Camera Rig [5], where it is described as an accurate model for the fisheye lens, but with the downside of a high computation time. The perspective projection is described by 5 equations, where one is for pinhole model and the rest is for fisheye variants. [29] proposes a general polynomial function instead, to handle every fisheye camera type. For feasibility [5] proposes a reduced model for fisheye cameras, the reduced model ignores the tangential distortion and uses 8 parameters $[k1, k2, k3, k4, u_0, v_0, f_x, f_y]$. The general description of radial distortion for this camera model is

$$D_r(\alpha) = (a_1\theta + a_2\theta^3 + a_3\theta^5)(p_1cos(\varphi) + p_2sin(\varphi) + p_3cos(2\varphi) + p_4sin(2\varphi)) \tag{2.7}$$

and the tangential can be described by

$$D_t(\theta, \varphi) = (b_1\theta + b_2\theta^3 + b_3\theta^5)(q_1cos(\varphi) + q_2sin(\varphi) + q_3cos(2\varphi) + q_4sin(2\varphi)) \tag{2.8}$$

θ and φ are the two angles needed to describe the direction from which a light ray enters the fisheye lens, similar as one see in light refraction between air and water. The mapping from an incoming light ray to the image coordinate is then

$$\begin{pmatrix} x \\ y \end{pmatrix} = r(\theta) \begin{pmatrix} cos(\varphi) \\ sin(\varphi) \end{pmatrix} \tag{2.9}$$

where

$$r(\theta) = o_1\theta + o_2\theta^3 + o_3\theta^5 + o_4\theta^7 + o_5\theta^9 \ . \tag{2.10}$$

Describing the radius of the image coordinate from the principal point in this manner allows us to accurately handle various types of lenses.

2.2.2.3 Unified projection model

This model is often referred to as Mei camera model [2] and it treats the mirror and camera as one sensor. It is built for omnidirectional cameras and is similar to Scaramuzza, but with one important exception which is the assumption that for Mei there is only a single viewpoint for the fisheye. To model the radial distortion a three parameter model is used:

$$L(\rho) = 1 + k_1\rho^2 + k_2\rho^4 + k_5\rho^6 \tag{2.11}$$

The tangential distortion dx abides by Eq. (2.12).

$$dx = \begin{bmatrix} 2k_3xy + k_4(\rho^2 + 2x^2) \\ k_3(\rho^2 + 2y^2) + 2k_4xy \end{bmatrix} \tag{2.12}$$

2.2.2.4 Undistortion

When a camera captures an image the lens usually causes a distortion in the image to some extent. Distortion effect is undesirable and should be removed, first one have to normalize the image coordinates. Normalization is reached by multiplying with K^{-1} and returns the image with coordinates centered at 0 and with a range of ± 1. Secondly, one should apply the distortion from the lens, depending on which model is used (Mei, Scaramuzza, Pinhole). Applying the distortion will move all pixels, after which an interpolation method is applied to fill the empty spots with estimated values. For a fisheye type of lens, the corners will be most distorted and have a lower resolution. The reason is that they have the greatest distance from the center of the image, and the correlation is shown in Eq. (2.4). Finally, to get back to pixel coordinates again, the result is multiplied by K. Resulting in an undistorted image.

2.3 Stereo vision

Stereo vision is becoming increasingly more popular for depth estimates. The phenomena used to gain a depth estimate is disparity, the distance between the same point in two images, which is also one of many ways humans sense depth. In stereo cameras, the cameras are usually locked in place in one common plane. In Fig. 2.2 the geometry behind the concept is explained.

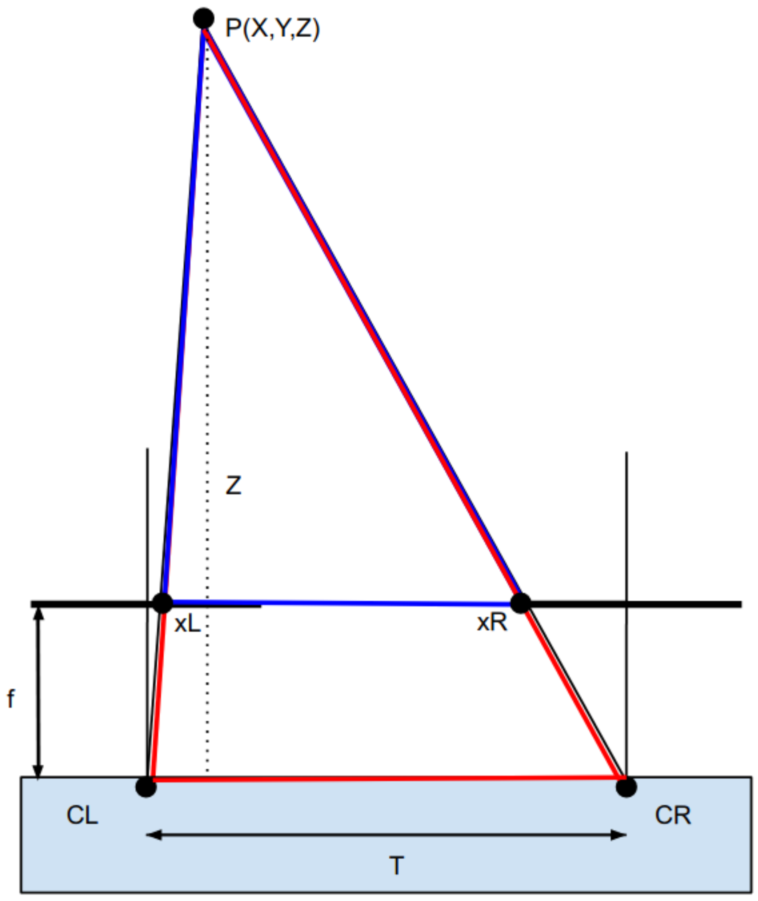


Figure 2.2: The derivation of the disparity depth estimate can be drawn from this figure. CR and CL is the camera centers, P is a point, xL and xR is the image point referring to the same point in the world frame. The red and blue triangle have the same relationship between height and width, $\frac{T}{Z} = \frac{T-xl+xr}{Z-f}$.

When manipulating the expression from the relation between the two triangles, one is left with an expression for depth Z:

$$Z = \frac{Tf}{xl - xr} \tag{2.13}$$

What determines the range and accuracy of the depth estimates is the resolution, lens and baseline between cameras T. Since xL and xR is measured in pixels, there is no continuous scale and the absolute max depth possible to calculate is Tf , when the disparity is 1 px. Another thing that might affect the accuracy is how well features is matched between stereo images, this varies with the texture of the surfaces of the measured objects. An interactive spreadsheet can display these relations in [30].

2.4 Feature extraction

To find and track objects in images, these objects are broken down into small parts such as corners or other features with characteristic properties. These features are first detected through tests of e.g. changing illumination, then described more specifically. The description is used to match features that have been detected in different images.

Good and reliable feature extraction is the foundation for many computer vision applications. When features are properly described you can find the same features in another image even if it is scaled, rotated or illuminated differently. Several methods exists for doing this, some of the most important is described in this section.

2.4.1 SIFT

Scale Invariant Feature Transform (SIFT) is a feature detection and description method consisting of the following steps [34].

Scale-space extrema detection

To find points of interest a search over all scales and image locations is performed. A difference of Gaussians (DoG) function is used. Define an image's scale space as

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.14}$$

where $G(x, y, \sigma)$ is a Gaussian of changeable scale

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} , \tag{2.15}$$

$I(x, y)$ is an input image and $*$ denotes a convolution. Then the *difference* of Gaussians becomes

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) . \tag{2.16}$$

In Eq. (2.16) $k = 2^{1/s}$ and s is the number of stages that each octave is divided into. With the DoG function a good approximation to the so called scale-normalized Laplacian of Gaussians can be made, which means that true scale-invariance is achieved. The scale space is contructed like a pyramid, with the bottom layer being an octave of full-size images with DoG from increasingly blurred images. Each octave is down-sampled by a factor of 2.

Keypoint localization

Extrema are found by iterating over each DoG image, checking if the center point in a 3-by-3 square is a minima or maxima and then comparing to the corresponding squares in the closest neighboring DoG images. If the contrast is higher, then a point of interest has been found. A certain threshold is set for how strong contrast is needed to consider the keypoint robust to noise and therefore worth keeping. Another robustness aspect of keypoints is whether or not they can be accurately localized, and one case where that is difficult is when a keypoint is along an edge. Along edges, the DoG function might yield a similar enough result further down the sides of the edge. To resolve that issue, keypoints with a ratio of principal curvatures above a decided threshold are rejected.

Orientation assignment

A feature orientation is assigned from local image properties. The descriptor becomes relative to the assigned orientation and not the rotation of the image, thus becoming rotation invariant. An orientation histogram is created from gradient orientations of sample points in the vicinity of the keypoint. The histogram covers 360° in 10° intervals.

Keypoint descriptor

Computation of descriptor by taking a 16 by 16 sample array divided into 4 by 4 subregions. For each sample point the magnitude and orientation is calculated. Then a Gaussian blur is applied on the sample array. For each subregion the updated values of sample points are accumulated, such that the magnitudes are added and the orientations are divided into 8 generalized directions. Hence, forming a 128 element feature descriptor.

2.4.2 FAST

Features from Accelerated Segment Test (FAST) [35] is a feature detector which compares a pixel's px intensity to 16 surrounding pixels placed more or less in a circle around the pixel. A least number m of these pixels that should be either less or more intense, for the pixel to be considered a corner can be chosen. A threshold is applied to ensure distinctiveness of the feature. A typical case is $m = 12$, and for that case the test can be accelerated by at first ascertaining that at least three of four pixels placed in compass directions from px were more or less intense. To avoid several pixels being detected adjacent to each other, non-maximum suppression can be applied.

2.4.3 BRIEF

Binary Robust Independent Elementary Features [36] (BRIEF) is a feature descriptor method, unlike SIFT the feature points are described as binary strings of test results where each test can be described by Eq. (2.17). In Eq. (2.17), the test result depends on the input smoothed patch p and the intensities $p(x), p(y)$ of p at given image coordinates x, y.

$$\tau(p; x, y) = \begin{cases} 1 & \text{if } p(x) < p(y) \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

Common descriptor sizes n_s are between 128 to 512 bits. The feature descriptor f_{n_s} can be composed according to Eq. (2.18).

$$f_{n_s}(p) := \sum_{1 \leq i \leq n_s} 2^{i-1} \tau(p; x_i, y_i) \tag{2.18}$$

2.5 Feature matching

The two most commonly used matchers are the brute force matcher and FLANN [37] based matcher. Brute force matcher as the name suggests goes through all keypoints in both images and matches the ones with the smallest distance i.e the best match. Distances are typically Hamming or L2-norm. L2- norm is the Euclidean distance and can be used for SIFT with its float value descriptors. Hamming distance is in this case the number of elements in a binary string that differs. It works well for feature descriptors such as BRIEF which descriptors are binary strings. Fast Library for Approximate Nearest Neighbor (FLANN) based matcher builds a kd-tree with descriptors and does not run through every possible descriptor to find a nearest neighbor, instead it finds an approximate best match via the kd-tree. Thus, becoming efficient for larger sets of descriptors compared to brute force matcher.

2.6 RANSAC

Random sample consensus [27] (RANSAC) is a model fitting algorithm, see Algorithm 1. Given the least number of data points needed to estimate a model, the data points that fit this model within a predefined threshold are called inliers. The best model is defined as the model with the most inliers. It is found through iteration until either enough iterations have passed, or until it is very likely that the best model has been found. The inlier ratio ε is the ratio of inliers compared to the total number of points.

Probability of picking n inliers randomly ε^n and probability of a sample containing at least one outlier $(1-\varepsilon^n)$ are used to determine when the algorithm can finish more rapidly.

Algorithm 1: RANSAC

```
Input: m data points;
Initialization: $k = 0$, $\varepsilon = \varepsilon_0$, $k_{max} = log(\eta)/log(1-\varepsilon^n)$;
while $k < k_{max}$ do
    Model estimation from n random data points;
    Estimate number of inliers;
    if $currInliers > bestInliers$ then
        Update best model;
        Update $\varepsilon$ = (number of inliers)/$m$;
        Update $k_{max} = log(\eta)/log(1-\varepsilon^n)$;
    end
end
Return: Best model
```

2.7 DLT

Direct linear transformation (DLT) is a simple way of finding a model. The first step is to compose a system matrix consisting of a number of linear equations, depending on the problem it needs a different amount of points and equations, e.g. camera resection $N >= 6$ and homography estimation $N >= 4$. The goal is to find the null space of the system matrix M, this is done with a singular value decomposition and the solution lies within the top N elements of the rightmost column of V (the right singular matrix). DLT can be used as a minimal solver in an RANSAC algorithm, to avoid using outliers in the estimate.

2.8 Epipolar geometry

Epipolar geometry is a way of relating two or more cameras to each other, it gives a way of perceiving the depth and gaining their relative pose to each other. When capturing an image of a scene, the information about the depth is lost. All the points lying on the line, projected by $\lambda x = PX$ (viewing ray) of camera 1, refers to the same image coordinate. With the addition of another camera, it is possible to get the correct depth of the 3D point from the matching point x_R. x_R can be quickly found by searching along the epipolar line and not the whole image, this is called the epipolar constraint. To find the epipolar line, the first camera is set to be in $[I|0]$ pose and the second camera is rotated and translated by $P2$. The scene point X can bee seen as a function of λ, so we get Eq. (2.19). Namely, the equation for the epipolar line.

$$P_2 X(\lambda) = \begin{bmatrix} R_2 & t_2 \end{bmatrix} \begin{bmatrix} \lambda x \\ 1 \end{bmatrix} = R_2 \lambda x + t_2 \tag{2.19}$$

In Fig. 2.3, the green triangle $O_L X O_R$ is called the epipolar plane. The projection of the other camera centers passes through the epipole $e_L = P_1 O_R$, the epipole is represented by the null space of the corresponding camera matrix. The epipole is always present, but can only be found in the image if the cameras can see each other. All the possible epipolar lines from different image coordinates passes through the epipole. A line can be described by two points, and the general line equation is $l^T x = ax + by + c = 0$. Another useful trick to know is that the cross product between two vectors give a perpendicular vector to both of them. Taking the cross product of two lines will also return the point of the intersection of the lines.

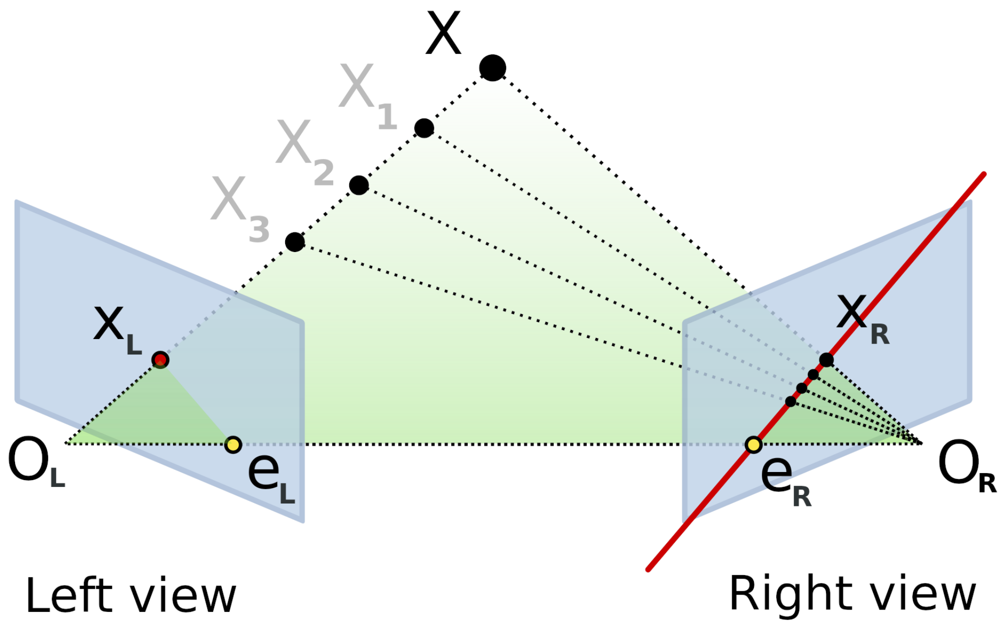


Figure 2.3: A visualization of epipolar geometry, xL and xR are the matched points in the image plane. The epipolar line is displayed in red from xR to eR(epipole) the intersection of the line between camera centers in the right image plane. The plane XO_LO_R is the epipolar plane. Note figure is taken from [33].

2.8.1 Essential matrix

An essential matrix is the foundation of epipolar geometry, it contains information about the rotation and translation between the cameras. As opposed to the fundamental matrix, the essential matrix can only handle normalized image coordinates. However, when the image is normalized and rectified then E = F. The essential matrix is defined by Eq. (2.20)

$$E = R[t]_\times \tag{2.20}$$

That satisfies the epipolar constraint.

$$x_{2_n}^T E x_{1_n} = 0 \tag{2.21}$$

Here, the notation of t is the skew symmetric matrix of t, i.e. a matrix composed of the cross product. The essential matrix has the following properties. Two constraints on the singular values, two of them should be equal, and one should be zero. These constraints should always be enforced. The determinant of E should be zero and the epipolar constraint should be fulfilled.
One can find the essential matrix in a few ways, if the fundamental matrix F and the camera intrinsics K is known then E can be found with Eq. (2.22)

$$E = K_2^T F K_1 \tag{2.22}$$

It can also be found directly from five or more point correspondences, using either RANSAC or singular value decomposition (SVD). Calculating it with RANSAC is

most efficient, a minimal model like the 5 point algorithm from David Nister [21] is applied to random samples and the solution from the samples that gave the most inliers is chosen. A solution from SVD can yield better results if the images are not too noisy.
From the essential matrix, the transform between the two camera centers can be recovered. The first camera is always set to be at 0 with no rotation, and then R and t can be found from the singular value decomposition of E, [U,S,V] = svd(E), where R and t are described by Eq. (2.23)

$$t = U[1:3,3],\ R = UWV^T \tag{2.23}$$

Here W is selected in such a way that it satisfies the constraints on the singular values of E:

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.24}$$

2.8.2 Fundamental matrix

The fundamental matrix is similar to the essential matrix, except the addition of the intrinsic parameters so it handles pixel coordinates directly.

$$F = K_2^{-1T} E K_1^{-1} \tag{2.25}$$

F is not restricted the same way with the singular values, but has a similar epipolar constraint.

$$u_2^T F u_1 = 0 \tag{2.26}$$

Similar to the essential matrix, the fundamental matrix is also rank deficient and its determinant is 0. To ensure the determinant is 0, the last singular value is set to 0. For finding the fundamental matrix having 7 degrees of freedom, the 7 or 8 point algorithm is chosen to solve for F. It can be done using RANSAC or DLT.

2.9 Triangulation

Triangulation is essentially to find the coordinate of a 3D scene point visible in two cameras. Triangulation is done by taking the matched points and set them up in a direct linear transform (DLT) equation. The goal is to solve Eq. (2.27) for the common 3D point X.

$$\lambda_1 x_1 = P_1 X ... \lambda_2 x_2 = P_2 X \tag{2.27}$$

To get rid of the scale λ, the cross product is used and we get Eq. (2.28)

$$x_1 \times (P_1 X) = 0 \tag{2.28}$$

Because the multiplicand and the multiplier have the same direction the cross product is equal to zero, writing out the cross product in Eq. (2.28) P^n is the n^{th} row of P.

$$\begin{aligned}(P_1^3X^T)y - (P_1^2X^T) = 0\\(P_1^1X^T) - (P_1^3X^T)x = 0\\(P_1^2X^T)x - (P_1^1X^T)y = 0\end{aligned} \tag{2.29}$$

From Eq. (2.29) we can factor out X and we get the DLT matrix $MX = 0$ of size $4N \times 4$(N number of cameras), finally the 3D scene point is found by taking the singular value decomposition (SVD) and decomposing the right singular vector.

2.10 Scale estimation

Scale estimation is needed because when using tools from epipolar geometry one get the transform between two cameras, but the translation is only defined up to scale. The scale can either be estimated from a ground truth, triangulated 3D points or from an additional sensor e.g. GNSS or IMU. For visual odometry image data is the only type of data available, the scale is estimated by the difference in the matched triangulated points from the current and previous frame.

$$\frac{\|t_{k-1,k}\|}{\|t_{k,k+1}\|} = \frac{\|X'_{0:k-1} - X_{0:k-1}\|}{\|X'_{1:k} - X_{1:k}\|} \tag{2.30}$$

In Eq. (2.30), X' represent the matched scene points in the new frame and X is from the old frame. From Eq. 2.30 you get an array of ratios, a common approach to get a relative scale is simply to select the median of the array as a way to reduce the effect of outliers.

One can aid the scale measurements either with adding another camera (stereo vision) or an IMU or GNSS sensor. Equipped with a calibrated stereo camera setup, one can calculate the disparity between matches, see Section 2.3, and get a good point cloud with correct depths. Then, applying the same method as for monocular vision on the 3D points gives the scale. With the addition of an IMU, one can loosely or tightly couple it to get better estimates of the metric scale. The accelerometer data is usually integrated in between the frames only and used together with a bias and measurement error model and a Kalman filter. The integration approach is shown in [25].

2.10.1 Bundle adjustment

Bundle adjustments (BA) refers to bundles of light that go from a feature point to a camera, and the aim is to reduce the reprojection errors. BA can be described as the problem of refining a visual reconstruction by optimizing the 3D scene points, relative motion parameters and viewing parameter estimates. Given normalized image points $x_{i,j}$, find 3D scene points $X_j = [X_j; 1]$ and camera matrices $P_i = [R_i; t_i]$. To reduce reprojection errors the cost function

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\left(x_{ij}^t - \frac{R_i^2X_j + t_i^1}{R_1'X_j + t_i^3}, x_{ij}^2 - \frac{R_1^2X_j + t_1^2}{R_i^3X_j + +_i^3}\right)\|^2 \tag{2.31}$$

is minimized. Here the subscript of a rotation matrix R specifies the camera and the superscript specifies the row of the rotation matrix. Bundle adjustment optimizes camera poses or scene points or both.

2.10.2 Loop closure

When a vessel needs to keep track of its position and pose only from camera feed there will be small errors, that will increase continuously without stop. Unless there is a way to recognize when the vessel comes back to a previously visited location, because then the difference in position and pose currently believed to exist can be erased while also correcting the locations visited since the vessel was last at the location revisited. To be able to make these adjustments a way to recognize that the vessel is in the same location as before needs to be known. The combination of detecting such occurrences and adjusting the positions and poses is called loop closure.

Loop closure detection can be divided into two parts. Visual place recognition and geometric verification. Different options for visual place recognition include bag of visual words, deep learning, visual feature matching, and kd-tree search. [8] The procedure to match images with bag of visual words is to first convert images to numerical vectors, and then compare these vectors in the bag of words space. [11] Benefits of bag of words is the speed and efficiency of the image matching. [11] The downside is perceptual aliasing, namely that two locations are erroneously found to be the same.

Image matching with KD-tree search can be summarized as follows. Key frames are images that are either spatially far apart or have limited overlap, where the first frame is always a key frame. For all such key frames the features are added to a feature pool D. Perform a nearest neighbor search, in the current KD-tree, for each feature in a current image. Then for the top K key frames, with the most matched features to a current image, do a similarity test checking whether a loop closure has been found. The similarity test uses feature matching on image level and is accurate. The benefits of this method is the accuracy of the image matching, while the problem can be scalability of the environment since no offline feature extraction is performed, thus creating computational speed issues for larger environments. [9]

2.11 IMU

An inertial measurement unit (IMU) is used to measure the specific force and angular velocity, and with these measurements the position is later calculated. In visual inertial odometry, sensor output from the cameras and the IMU can be either loosely coupled or tightly coupled. The difference of how the sensor data flows for the two approaches is illustrated in Fig. 2.4.

Inspiration for Fig. 2.4 was taken from [38], however different sensors were used and a simpler illustration was made. The point was that for loosely coupled sensor setups, the data from one sensor might go through an algorithm before being fused

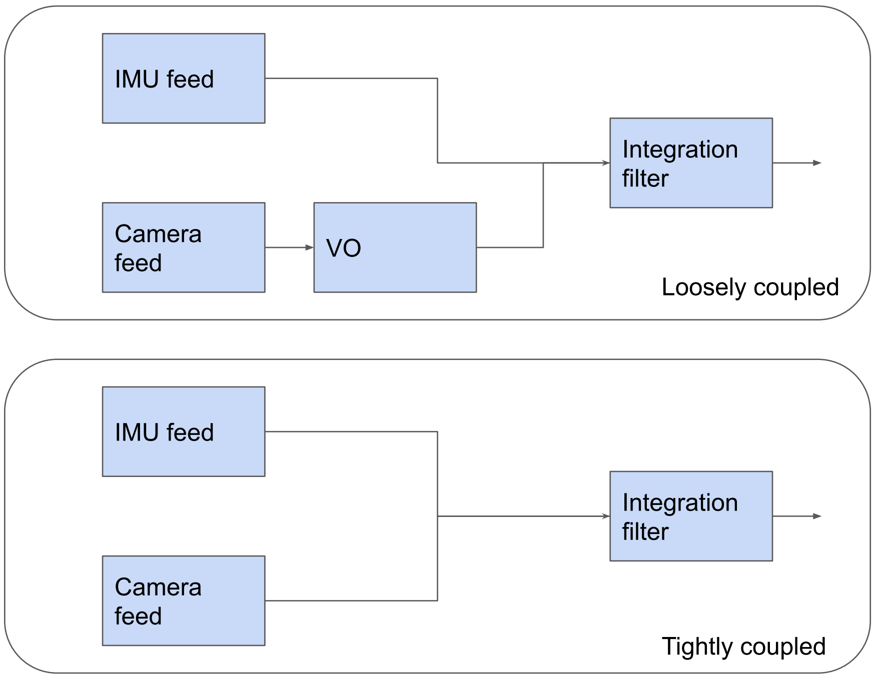


Figure 2.4: An overview of coupled systems between camera sensors and IMU sensors.

with data from another sensor. In the loosely coupled setting a common approach is to use the extended kalman filter (EKF), here the state propagation typically stems from IMU data and the pose calculated from camera feed is used for the update [14]. The EKF is a filtering method, while there are also optimization based techniques for visual inertial odometry.

2.12 Visual odometry

Visual odometry mono or stereo is a method for generating a trajectory solely based on observations from cameras. VO is a cheap and usually very good way to get accurate estimates of the pose. The approach has some shortcomings when it comes to robustness in featureless environments and when the object comes to a stop. The stereo methods can yield some improvements to the mono scheme because of better scale estimation. The simplest approach of visual odometry is described below.

1. Firstly features have to be matched between the current and previous frame.
2. After two frames are captured and matched, the essential (or fundamental matrix if the intrinsic is not available) is found according to description in Section 2.8.
3. From the essential matrix the pose of the current frame with respect to $[I|\mathbf{0}]$ is decomposed.
4. With the new pose the matched points are triangulated to a current 3D point cloud.

5. From the current point cloud and previous point cloud the relative scale is estimated as described in 2.10
6. Calculate the current total position and rotation:

$$\mathbf{R}_{totk} = \mathbf{R}_k \mathbf{R}_{totk-1} \tag{2.32}$$

,

$$\mathbf{Pose}_k = \mathbf{Pose}_{k-1} + \mathbf{R}_{totk} \mathbf{t}_k S \tag{2.33}$$

In addition to the simple approach, one can add several local and global optimizing steps and loop closure. However, logging frames, matches, poses and 3D points is needed then and it essentially becomes a SLAM algorithm.

2.13 Visual inertial odometry

Visual inertial odometry combines camera feed and IMU feed to estimate a trajectory. Where the visual odometry is less robust in certain situations, the addition of an IMU can provide extra stability. E.g. when the object travels through an area of few features, then it has support in the IMU measurements to know if it is changing its trajectory. To get the information about the position from the IMU, the accelerometer readings are integrated twice and this will accumulate errors quickly. The orientation can on the contrary be accurately estimated. Roll and pitch can be observed directly due to the gravity vector, and the yaw can be integrated from the gyroscope which is accurate over shorter periods.

2.14 Framework

ROS Robotic Operating System is a framework which is commonly run on embedded machines. It is structured in such a way that it is one master or core, with one or more nodes related to it. Each node typically performs some function and exchange messages in between the other nodes. The node can either publish or subscribe to a topic, e.g. in a car one node could measure the state of the vehicle and publish among other velocity messages to a topic. Then a cruise control node would subscribe to the same topic and use the velocity messages to control the acceleration needed to keep a constant velocity.

A nice feature of ROS is that all messages have a timestamp so it is easy to synchronize, and one can easily find the sensor updates in between each frame captured.

3 Methods

Camera calibration includes estimating parameters for intrinsics, distortion and extrinsics. The intrinsic parameters are $[f_x, f_y, c_x, c_y, s, \alpha]$ and the distortion parameters varies depending on the complexity of the lens. All the mentioned parameters are fitted by using images with known features, often images of chessboards. The extrinsic parameters are transformation matrix containing rotation matrix and translation vector. The rotation matrix from a base orientation, usually one of the cameras' orientations, to another camera's orientation. Likewise, the translation from a base position to a camera's position

Regarding extrinsic calibration, the most common situation is that a stereo camera is used, which shares a large portion of FOV and the features in this shared section are used to find the parameters. When there is no shared FOV between cameras, or the nature of the lens reduces the quality of the overlap beyond a certain degree, other methods of estimating extrinsic parameters are needed. Some categories that have proven to yield accurate results are trajectory, SLAM, planar and mirror based methods [24].

Mirror based methods require a calibration pattern such as a chessboard to be visible to all cameras with the help of mirrors. With static camera pose and calibration pattern, the mirror pose can be calculated and in turn the relative pose between cameras can be calculated. Problems with mirror based methods include obtaining high quality captures of calibration pattern due to mirror poses, where there is a difficulty in properly arranging mirror poses while handling constraints imposed by the physical environment [24]. Trajectory based methods cameras are fixed on a rig, with relative poses between cameras that do not change regardless of how the rig moves. An image sequence per camera is produced, during which time the rig has to experience complex movements or the performance can decrease [24]. SLAM based methods estimates the extrinsics after registering the scene details with each camera. Challenges include accurate 3D reconstruction of the scene [24].

Planar calibration [23] can be done if all cameras are RGB-D type. Plane calibration method require the cameras to view one plane from several views or have several planes visible in one image in each camera. The algorithm works by first giving an initial guess of the extrinsics to find plane correspondences in each camera and the normal vector of the plane visible in each camera. From the plane correspondences they do a MLE (maximum likelihood estimation) of first relative rotation then the translation. Plane calibration method is quick and reliable.

For applications with 360° view it is common to have some overlap between the different frames. With overlap it may be possible to estimate the essential matrix and from that recover the relative rotation between the two cameras and a translation up to scale. A problem with finding features on the edges of frames from cameras with a high field of view, is that the edges are of lower resolution due to the construction of the lens.

3.1 Choice of method

To the best of our knowledge there were not any papers on automatic calibration of non overlapping cameras for marine vessels to date. A consideration to take into account for extrinsic calibration on vessels is that water is not well suited for finding good features. When good features are not available one should have additional sensors along cameras to estimate the pose. A water vehicle does not have the same traction as a land going one, and the same motion models do not apply directly. Camera placement is not as well defined as in cars, and there are large varieties in size and construction.

The choice in this report was based on the need for a general solution, that worked for a lot of different setups. E.g. fisheye lenses, regular cameras and stereo cameras. It was also not desired to require external cameras, mirrors or calibration patterns. The planar calibration method would have been a good option. However, it would have limited us to RGB-D cameras.

The factors mentioned above lead the way to a trajectory based method. The trajectory based methods could be done on all cameras and the position estimates could be fused together with other available sensors on a vessel.

3.2 Simulator setup

For maximal flexibility and availability to get exactly the data set desired with no cost, a simulator was chosen. The selected simulator is based upon ROS and Gazebo, the simulator world and vessel was built by Open Robotics for their Virtual RobotX competition[26]. The platform based upon Marine Advanced Robotics vessels, is equipped with an IMU, LiDAR, cameras and GPS. For this work the original sensor package was modified. The boat was fitted with 4 cameras, placed 90° upon each other and every camera was aimed downwards with an angle of 15°. The new sensor setup was specified in a yaml file used to generate a new urdf file for the vessel. The camera sensor itself was replaced with a wide angle camera model with 180 ° FOV, which was done by modifying a camera sensor specification file wamw_camera.xacro. In Fig. 3.2 the modified sensor setup is shown in bird's eye and side view.

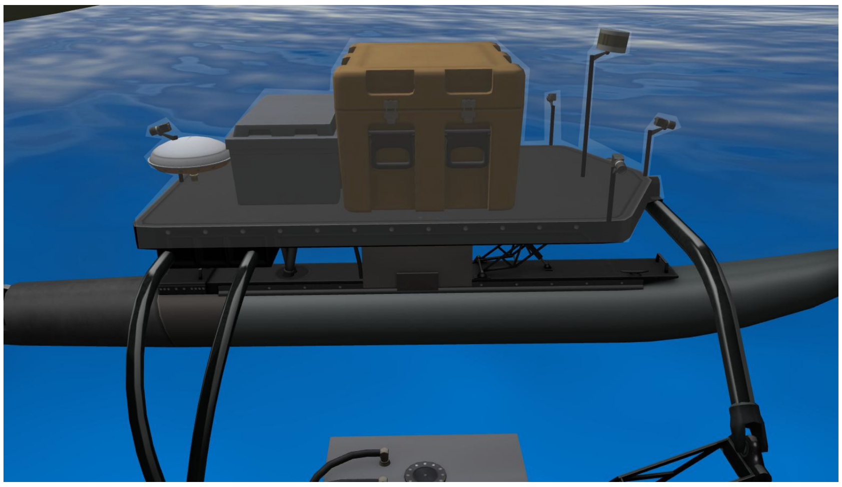


Figure 3.1: An overview of the sensor setup.

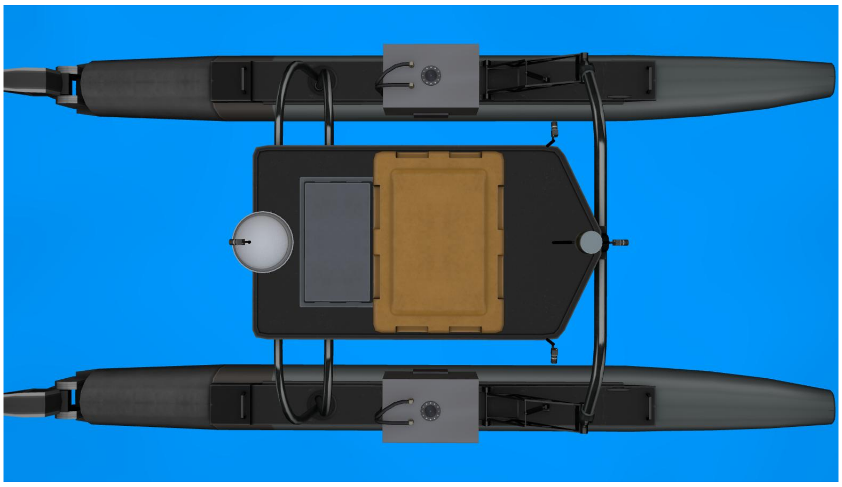


Figure 3.2: Bird's eye view of the sensor setup.

To enrich the water with features, several docks were added to the Gazebo world in order for all the cameras to have some features available.

3.2.1 Data recording

The data set was collected in ROS bag files. The ROS topics recorded are displayed in Table 3.1. The bag file was recorded while steered in a circuit with the keyboard and a localization node was running to get the /tf topic. With four cameras the laptops used in this thesis manage recording images at 6 FPS, IMU is kept at 15 Hz and tf at 100 Hz.

Table 3.1: The recorded ROS messages and a description.

ROS topics	Description
/tf	Dynamic transforms, world–>base, base–>proppellar
/tf_static	All static transforms
/wamv/sensors/cameras/front_camera/image_raw	RAW image data
/wamv/sensors/cameras/right_camera/image_raw	RAW image data
/wamv/sensors/cameras/back_camera/image_raw	RAW image data
/wamv/sensors/cameras/left_camera/image_raw	RAW image data
/wamv/sensors/cameras/front_camera/camera_info	Camera info: K, resolution, Distortion, ...
/wamv/sensors/imu/imu/data	Angular vel, Linear acc and timestamp

3.3 Detecting features on water

Compared to roads, water is not stationary in any way. The water surface is highly reflective and changes shape constantly. Because of the wide field of view, parts of the boat are visible in some of the cameras. Features tracked on the boat will distort the trajectory the boat takes, because the features tracked will appear to be stationary. The stationary features is a problem when there are few other features available and they will be considered inliers by the RANSAC algorithm. For that reason it was desired to have as few features as possible detected of the water and the vessel itself.

To avoid features on the vessel, a black mask was multiplied with the images. The black region will not be tracked by most trackers, but SIFT does sometimes find features along the edge. When SIFT was used, it was needed to define a region of interest (ROI) where the features were tracked. The good features were from objects relatively stationary on the water. E.g. docks, buoys, pillars or objects on the coastline.

The docks added to the simulator environment were identical to each other, so a constraint on the maximum distance between matches in two consecutive frames and a minimum of 2 pixels apart was added. The maximum distance was directly correlated to the frame rate and was adjusted accordingly.

3.4 Finding intrinsic and distortion parameters

The focus of this thesis was not about finding the intrinsic parameters. For data sets, e.g. KITTI odometry [28], the intrinsic parameters were provided. On the simulator, the camera intrinsic matrix was provided in the camera info message published from the camera sensor. The distortion parameters, resolution and FOV could be set in the sensor specification wamw_camera.xacro file.

3.5 Obtaining a trajectory

The method of choice depended on independent trajectories from all of the cameras. To calibrate the extrinsics of the cameras solely based on trajectories, they needed to be accurate.

3.5.1 Visual odometry

Visual odometry is a simple but effective way of obtaining a trajectory. The drawback of visual odometry is that it requires a good amount of features spread out evenly in the image. Because it could then:

- Track more features through a longer time before computing new features.
- Get a better estimate of the essential or fundamental matrix.

A commonly used trick is to exclude any non forward dominated movement. This constraint does not always hold, when turning in sharp corners the boat is sliding sideways a bit at the same time, and of course from the perspective of the camera on the stern of the boat, the motion is backward.

3.5.2 Visual inertial odometry

Visual inertial odometry was used to add stability to the visual odometry when there was a lack of features. To get the right yaw rotation, an IMU was used. The IMU was seated at the platform of the boat with its x axis in the driving direction of the boat and so the rotation around z axis represents heading the boat. The coordinate system of the gyroscope in the IMU was transformed to match the camera with Eq. (3.1)

$$R_{optical} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \tag{3.1}$$

The gyro was considered to be noise free in this application, a reasonable approximation when a bias compensated IMU is usually off by 1°/min and the sequences made in this report were typically < 1 min. The update frequency of the IMU was set in the IMU specification file. From the rosbag, that frequency f was read. There were typically 4-5 IMU updates per frame when IMU was running at 15 Hz, and they were integrated in between each frame to get the rotation between frames.

$$\mathbf{r}_{IMU} = \boldsymbol{\omega}\frac{1}{f} \tag{3.2}$$

The final rotation for a frame update was found with a simple complementary filter shown in Eq. (3.3)

$$\mathbf{r} = (1-\gamma)\mathbf{r}_{IMU} + \gamma\mathbf{r}_{cam} \tag{3.3}$$

Both the angular velocity measurements from the IMU and the poses were filtered with a short moving average filter, to get rid of the non smooth behavior.

3.5.3 Ground truth camera trajectories

For testing the extrinsic calibration algorithm, a ground truth trajectory was used. In order to synchronize the transformation needed by the algorithm and the data extracted from the simulator, an understanding of the simulator coordinate system is needed. Fig. 3.3 shows the relation between the simulator world coordinate system, and the vehicle. The vehicle has a base, from which further transformations were needed to reach each individual camera.

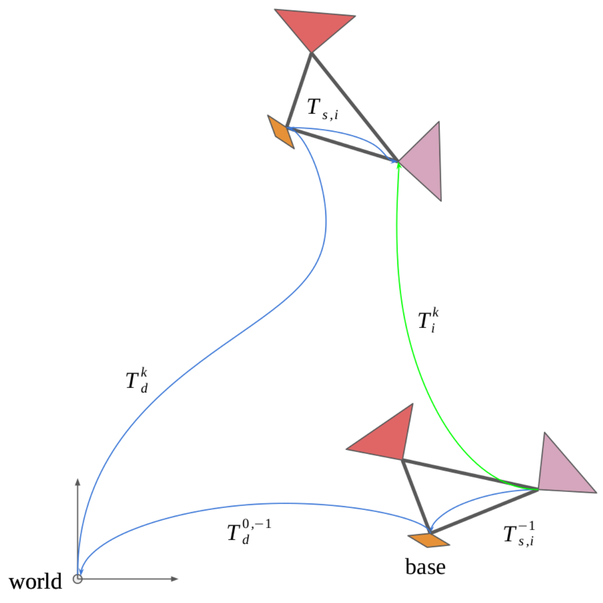


Figure 3.3: Ground truth transforms.

In a real world scenario, to estimate trajectories for each camera, the first motion T_i^1 for camera i would be set as

$$T_i^1 = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{3.4}$$

i.e. the origin. And the transformation T_i^k would represent a motion from the "origin", i.e. first camera frame of the same camera, to the k^{th} camera frame of that camera. Naturally, the pose of the k^{th} frame could then be calculated as a subsequent multiplication of each transform up until that frame. However, for the ground truth based on the simulator values, this generated issues. These issues stemmed from the representation of data in the simulator. The time-varying rotation and translation from world to vehicle base coordinates were read from the ROS topic tf with frame id wamv/odom and child frame id wamv/base_link. Static data was read from tf_static. These were defined when creating the vessel in the simulator. A list of the four used static links from vessel base to camera center are:

- Base to post - $T_{base\_to\_post}$
- Post to arm - $T_{post\_to\_arm}$
- Arm to camera - $T_{arm\_to\_cam}$
- Camera to optical - $T_{cam\_to\_opt}$

for which $T_{base\_to\_post}$ was unique for each camera while the following were identical. To obtain a transformation $T_{s,i}$ from base to optical, calculation of

$$T_{s,i} = T_{base\_to\_post,i} T_{post\_to\_arm} T_{arm\_to\_cam} T_{cam\_to\_opt} \, . \tag{3.5}$$

was made. Another necessary transformation was T_d^k from the simulator world coordinates to the current base of the vehicle. Let the scenario with all the transforms of the simulator by described by Fig. 3.3.

From Fig. 3.3, it could be deduced that the desired transformation T_i^k corresponded to

$$T_i^k = T_{s,i}^{-1} T_d^{0,-1} T_d^k T_{s,i} \, , \tag{3.6}$$

where $T_{s,i}^{-1}$ was calculated by applying Eq. (3.7) on Eq. (3.5) and $T_d^{0,-1}$ by applying Eq. (3.7) on T_d^1. $T_d^{0,-1}$ denoted the inverse transformation from world to vehicle base coordinates for the first set of frames.

$$T^{-1} = \begin{bmatrix} R^T & -R^T t \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{3.7}$$

3.6 Finding extrinsic parameters

For each camera, a sequence of transformations T_i^k from a reference pose to the current pose was obtained from ground truth. The reference pose of a camera was defined by setting the pose of the 0^{th} frame to and identity matrix. The calibration required transformation data for at least two cameras, a master camera 0 and slave cameras i. To recover the relative pose ΔT_i between two cameras set up according to Fig. 3.4 two major steps were taken, solving for the relative rotation ΔR_i and solving for the relative translation Δt_i given the estimated ΔR_i.

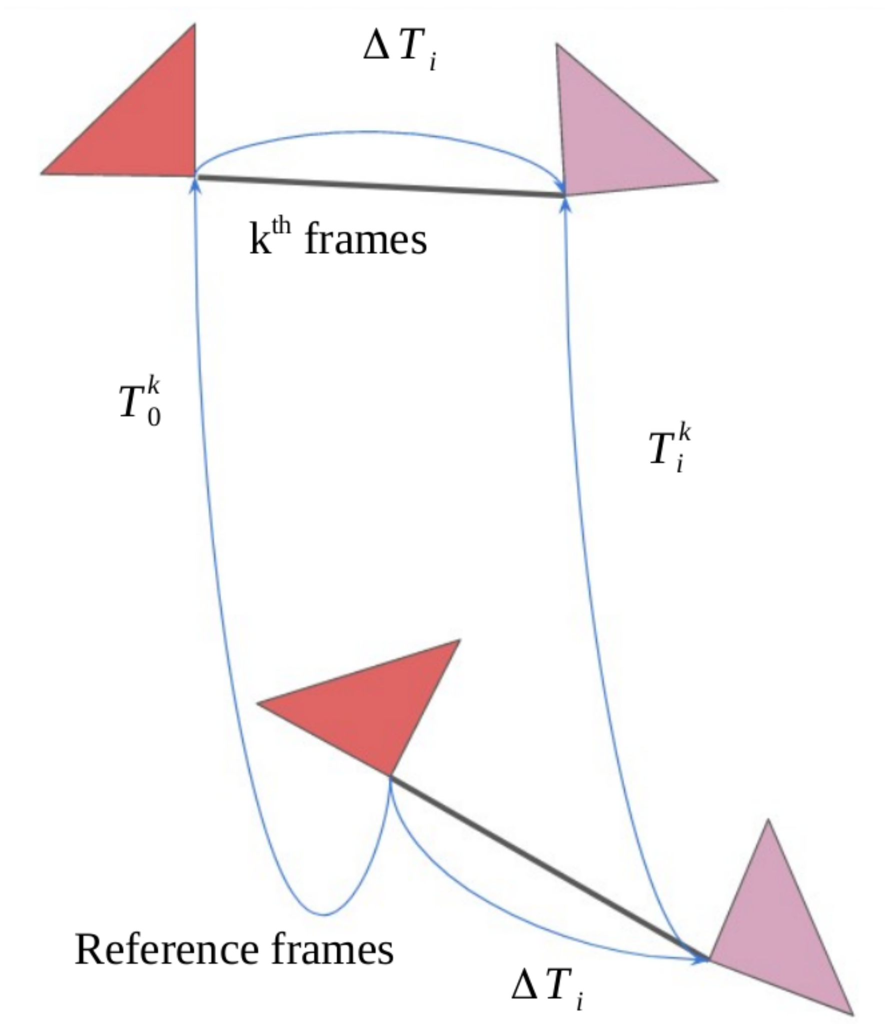


Figure 3.4: Rig transforms, rigid link between the two cameras.

From Fig. 3.4 in combination with an assumption of rigidity Eq. (3.8) was derived.

$$T_0^k \Delta T_i = \Delta T_i T_i^k \tag{3.8}$$

The construction of a pose T where R is a rotation matrix and t a translation vector is shown in Eq. (3.9).

$$T = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \tag{3.9}$$

Given Eq. (3.9), Eq. (3.8) could be divided into two constraints where Eq. (3.10) was for rotation and Eq. (3.11) was for translation.

$$R_0^k \Delta R_i = \Delta R_i R_i^k \tag{3.10}$$

$$R_0^k \Delta t_i + t_0^k = \Delta R_i t_i^k + \Delta t_i \tag{3.11}$$

Examples of vectorization of a matrix and the kronecker product of matrices are shown in Eq. (3.12) and Eq. (3.13) respectively.

$$vec\left(\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\right) = \begin{bmatrix} a_{11} \\ a_{21} \\ a_{12} \\ a_{22} \end{bmatrix} \tag{3.12}$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \otimes \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{bmatrix} \tag{3.13}$$

The use of vectorization and kronecker product came in handy for the property

$$vec(M_1 M_2 M_3) = (M_1 \otimes M_3^T) vec(M_2)\,, \tag{3.14}$$

which allowed for rewriting Eq. (3.10) and Eq. (3.11) into

$$(I_9 - R_0^k \otimes R_i^k) vec(\Delta R_i) = \mathbf{0} \tag{3.15}$$

and

$$(I_3 - R_0^k)\Delta t_i = t_0^k - \Delta R_i t_i^k \tag{3.16}$$

respectively. For k_{max} motions a resulting systems of linear equations to be solved became

$$\underbrace{\begin{pmatrix} I_9 - R_0^1 \otimes R_i^1 \\ \vdots \\ I_9 - R_0^k \otimes R_i^k \\ \vdots \\ I_9 - R_0^{k_{max}} \otimes R_i^{k_{max}} \end{pmatrix}}_{M_i} vec(\Delta R_i) = \mathbf{0}\,. \tag{3.17}$$

A vector $w_i = vec(W_i)$ in the null space of M_i in Eq. (3.17) was calculated via singular value decomposition, see Eq. (3.18). Numpy syntax was applied. E.g. MATLAB the matrix named V_i is transposed.

$$\begin{gathered} U_i S_i V_i = M_i \\ \Rightarrow w_i = V_i[-1,:] \end{gathered} \tag{3.18}$$

Rotation estimate is described in Eq. (3.19)

$$\Delta R_i = W_i sign(det(W_i))|det(W_i)|^{-1/3} \tag{3.19}$$

From Eq. (3.16), iterating motions from the sequence of frames Eq. (3.20) was obtained.

$$\underbrace{\begin{pmatrix} I_3 - R_0^1 \\ \vdots \\ I_3 - R_0^k \\ \vdots \\ I_3 - R_0^{k_{max}} \end{pmatrix}}_{N_i} \Delta t_i = \underbrace{\begin{pmatrix} t_0^1 - \Delta R_i t_i^1 \\ \vdots \\ t_0^k - \Delta R_i t_i^k \\ \vdots \\ t_0^{k_{max}} - \Delta R_i t_i^{k_{max}} \end{pmatrix}}_{O_i} \tag{3.20}$$

Translation Δt_i was estimated by multiplication of pseudo-inverse of N_i and both sides of Eq. (3.20).

3.7 Addition of noise

The nature of the noise coming from a generated track is cumulative, so that the noisy rotation would affect the whole trajectory coming after. Disturbances were created following zero mean Gaussian distributions with independently selected amplitudes for rotation noise and translation noise. Individual samples were generated for each element of R and t shown in Eq. (3.21), (3.22).

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} + \begin{bmatrix} n_{11} & n_{12} & n_{13} \\ n_{21} & n_{22} & n_{23} \\ n_{31} & n_{32} & n_{33} \end{bmatrix} \tag{3.21}$$

$$t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} \tag{3.22}$$

Two methods of adding noise were tried. The first was to simply generate noise and add it to each ground truth pose. However, it was reasoned that in reality the effects on a trajectory would be permanent if it had diverged from the ground truth. Therefore, a cumulative noise was also tried. Here the noise was generated in the same fashion but the list of poses were instead divided up into transformations between subsequent poses and by multiplying each transformation up until a frame index k, the noise would be included and saved in those transformations.

The distribution of noise was chosen as Gaussian because of the unpredictability of waves and wind disturbances. Why were the additive and cumulative models chosen? Additive because it was desired to try the calibration algorithm on the trajectory that was "more or less" correct, in that way the constraint on fixed links between cameras easier to enforce correctly. The cumulative model was chosen as reality-based situation, that whenever a transformation between two frames was badly estimated, the error would stack to the previous errors.

3.8 Optimization

After the initial estimate was found one could apply a nonlinear refinement method to further improve the result. The goal was to minimize the error function in Eq. (3.23).

$$L(\Delta q_i, \Delta t_i) = \sum_{n=1}^{N} \|A_i^n \Delta q_i\| + \|O_i^n \Delta t_i - t_0^n\| \tag{3.23}$$

The subscript represents the slave camera number, and the superscript is the frame number. Quaternions were used rather than rotation matrices so it became 7 parameters to estimate rather than 12. There were different optimization algorithms that could minimize the error function and the most common for visual applications is Levenberg Marquardt, a damped least squares approach. In this work however, Trust Region Reflective was used since it allowed bounds. Typically it would yield only a slightly better solution if the initial guess was close and not already perfect.

4
Results

This chapter contains results from the stages towards extrinsic calibration, and results on how well the trajectory based calibration handles noise.

4.1 Visual odometry

The first test was to investigate if visual odometry would be sufficient for calibration of the cameras. The algorithm was tested on KITTI Odometry sequence 9. The KITTI data sets are very popular, feature rich, rectified, provided with ground truth and are commonly used as benchmarks. The features and descriptors were generated using FAST and BRIEF and a brute force matcher.

The first test conducted was for how long to track features. The tracking was done by matching the features in the latest frame with the tracked features until the number of matches were below a certain threshold. If the threshold was crossed, all matches from the latest frame pair would be used for feature tracking. In Fig. 4.1 this was done with FAST+BRIEF with the limits [100,400,700,2000].

In Fig. 4.1 the biggest difference was between 100 and 400, while there was an improvement between 400, 700 and 2000, it was not as significant.

A similar test was done using SIFT. With SIFT, Lowe's test [32] ratio determined how good the matches had to be, which also determined the amount of features found. Fig. 4.2 displays two ratios commonly used. Higher ratio, more key points.

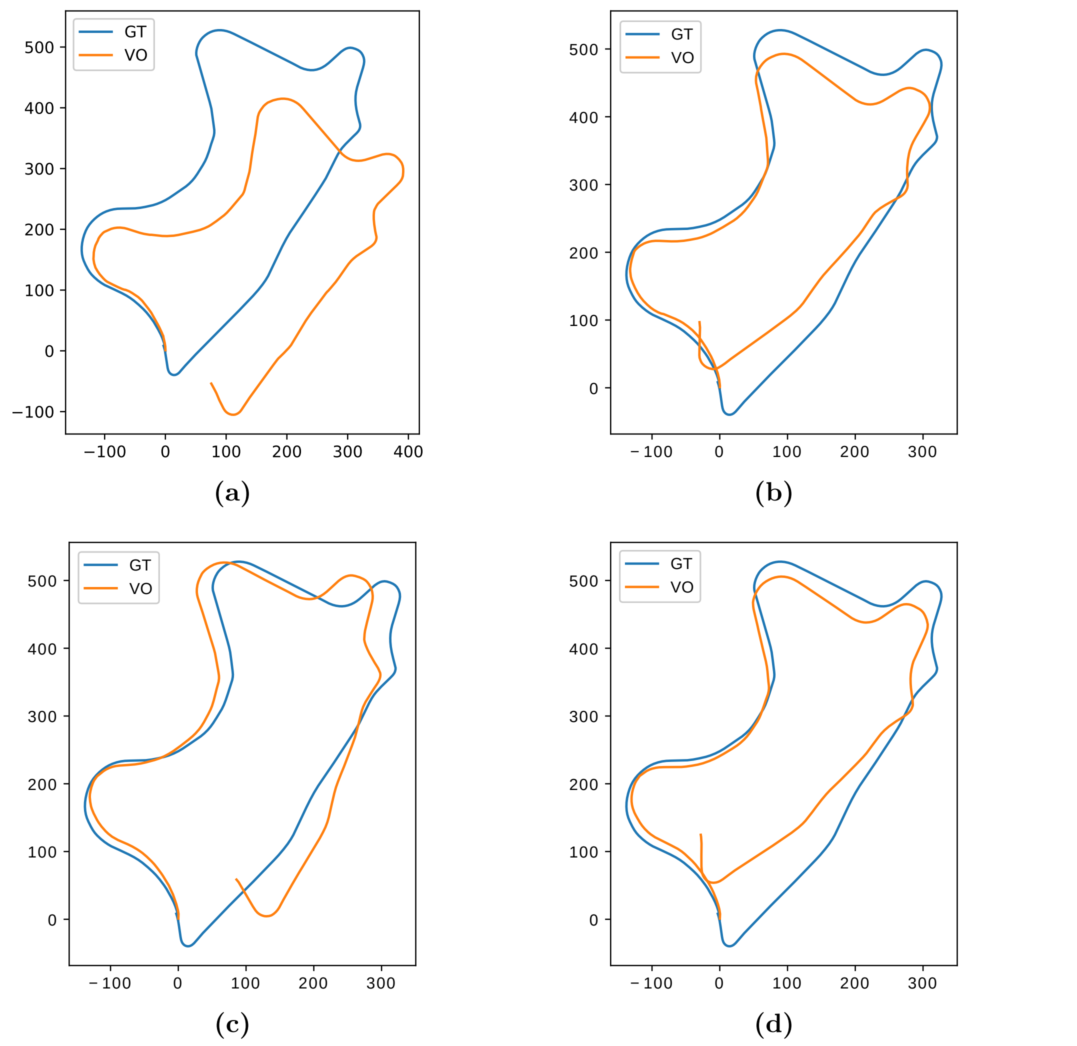


Figure 4.1: KITTI Odometry sequence 9. Different thresholds for detecting new features [100, 400, 700, 2000] for [a, b, c, d].

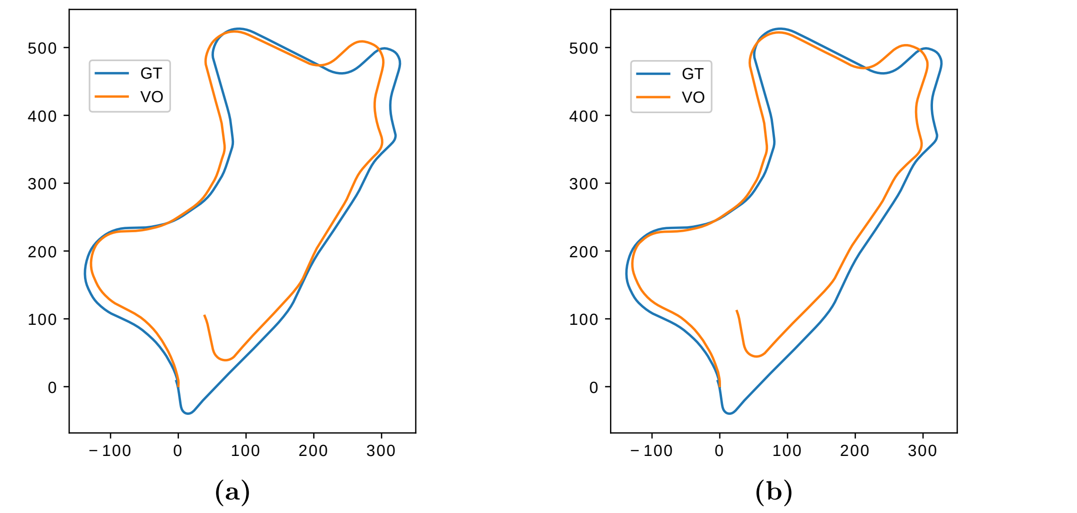
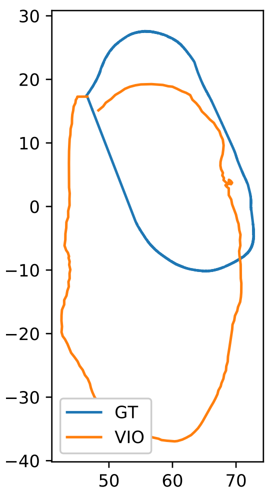


Figure 4.2: KITTI sequence 9, SIFT detector, Flann matcher, Lowe's ratio 0.75 and 0.85.

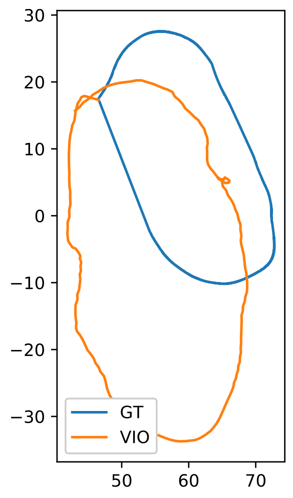
SIFT outperformed FAST and BRIEF, in terms of distance from ground truth for the KITTI data set. In the test, having 0.75 Lowe's ratio performed better than 0.85. The result means that neglecting the lower quality matches that were allowed for 0.85 ratio was preferable. In terms of time consumption SIFT took 10 times longer than FAST and BRIEF on the same sequence.

4.2 Visual inertial odometry

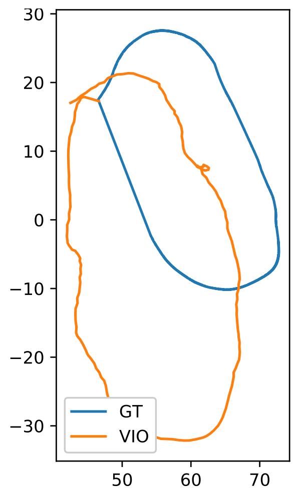
First test, see Fig. 4.3, was to see what performed best on the simulator, with no IMU influence, $\gamma = 1$ using SIFT which proved to be the best for the KITTI data set.



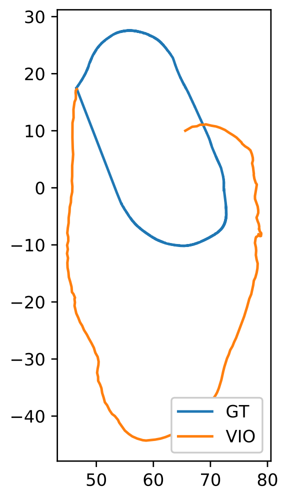
(a)



(b)



(c)



(d)

Figure 4.3: VIO Algorithm, SIFT, Lowe's ratio [0.75, 0.75, 0.85, 0.95], [70, 100, 150, 200] update threshold, values of γ: [1.0, 1,0, 1.0, 1.0], meaning no IMU influence.

Since the gyroscopes in IMUs do not drift significantly over the course of a minute, the heading of the boat could be accurately found by integrating the angular velocity of the z axis of the IMU. In Fig. 4.4 the values of $\gamma = [0, 0.3, 0.7, 1]$ are tested. With a γ value of 1, this is a VO algorithm with FAST and BRIEF feature extractor since most of the features available here are corners.

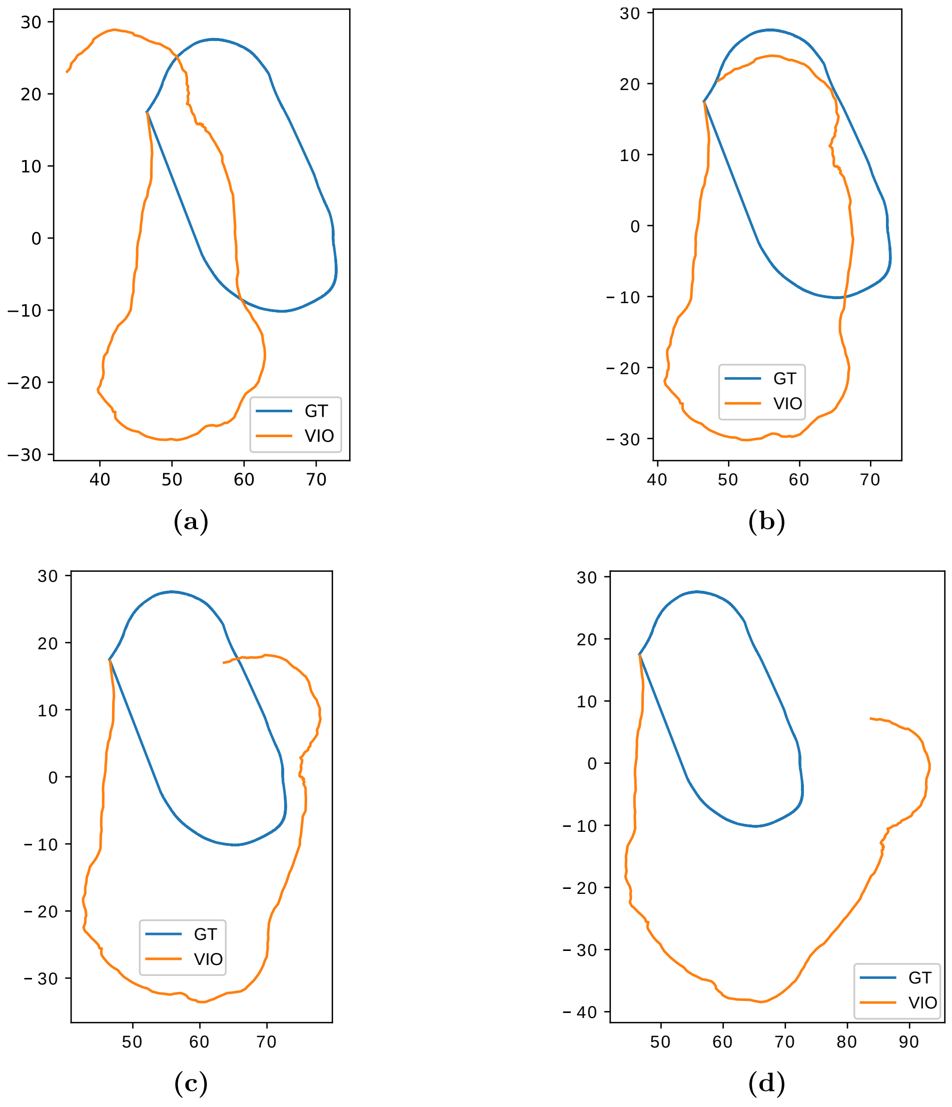


Figure 4.4: VIO Algorithm, FAST-BRIEF with the values of γ: [0, 0.3, 0.7, 1.0].

Even if a gamma of 0 gave the correct rotations, it was not perfect with respect to the estimated translations and scales. The best result was from $\gamma = 0.3$.

For calibration, a trajectory from all of the cameras was needed. The test was conducted with SIFT, 0.75 Lowe's ratio, $\gamma = 0.5$, 100 update threshold. The result is shown in Fig. 4.5.

The trajectories in Fig. 4.5 are behaving in the way you would expect from four cameras 90° upon each other.

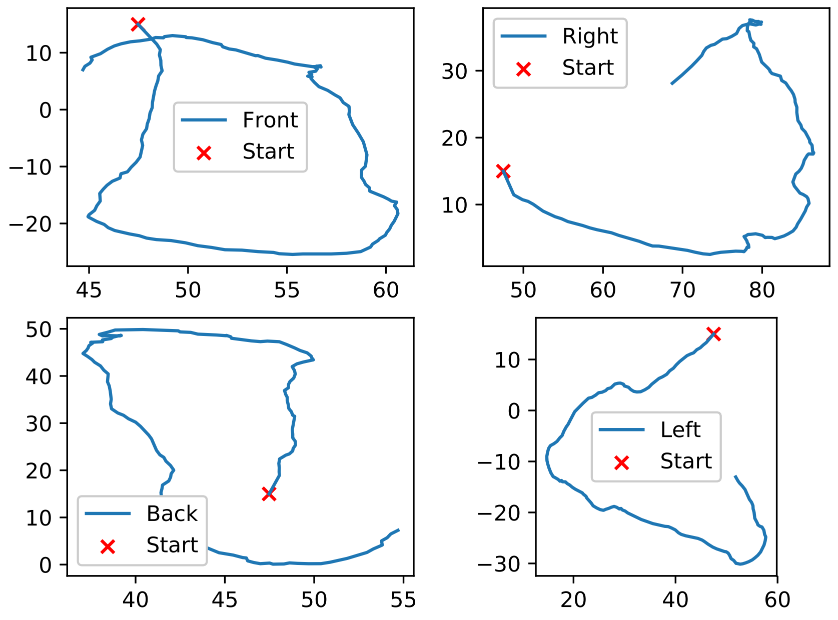


Figure 4.5: The cameras own trajectory viewed from their coordinate system. SIFT, *Lowe* = 0.75, $\gamma = 0.3$.

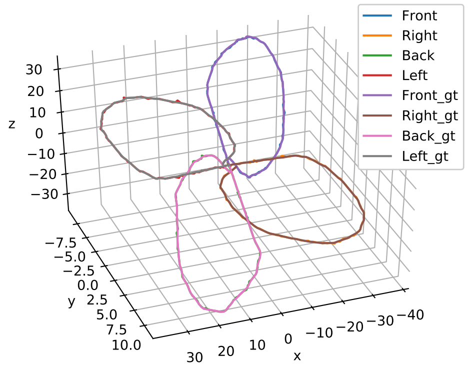
4.3 Extrinsic calibration on ground truth trajectories

In the section below, results for clean ground truth trajectories are presented, along with results for the same trajectories with subsequently added noise.

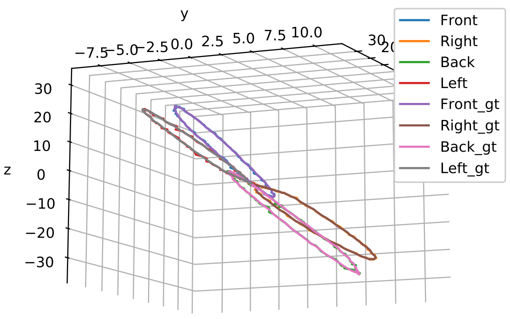
4.3.1 Trajectories with added noise

Four plots are shown in Fig. 4.6. The upper row illustrates the effect of additive noise applied to the translation vectors. The ground truth, plotted for comparison in each subfigure, was similar since the errors only affect the trajectory locally. The bottom row of Fig. 4.6 features the cumulative noise, where the effect on the translation vectors was permanent. As a result, the trajectories differed much more from the ground truth.

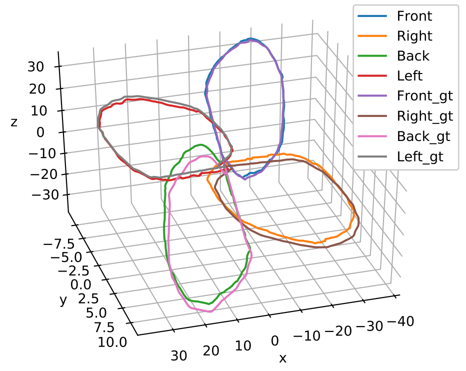
For additive rotation noise, $\mathcal{N}(0, 0.0001)$ started to result in noticeable errors. Further increasing the standard deviation with an order of amplitude, increased the highest translation error by about 5 times. In general, the optimized transform improved slightly, but not significantly. The same standard deviation increase for cumulative rotation noise, instead caused an increased error of two orders of magnitude. For 176 poses per camera the algorithm took about 0.4 sec to perform an extrinsic calibration between a pair. The increase in translation errors appeared



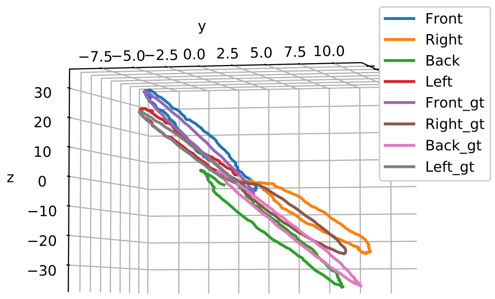
(a)



(b)



(c)



(d)

Figure 4.6: VRX simulation run. Top row, ground truth with additive translation noise. Bottom row, ground truth with cumulative translation noise. Noise level $\mathcal{N}(0, 0.1)$.

more or less proportional to the increase in translation noise standard deviation, both for additive and cumulative. However, the error values were orders of magnitude higher when applying cumulative noise.

Table 4.1: The angular and translation errors for the transformation between front and left camera, for different values of additive noise.

		NOT OPTIMIZED					
		Rotation error in deg			**Translation error in cm**		
stdR	**stdT**	**x**	**y**	**z**	**x**	**y**	**z**
0	0	0	0	0	0	1.11438E-03	-1.75659E-04
0.0001	0	0.1360499	0.00264001	0.12934821	-0.64888966	**2.108529787**	-4.12838181
0.001	0	-0.0659454	0.0063285	-0.05965763	0.3074497067	-9.793094693	**0.40649455**
0	0.001	0	0	0	0.002163605	**0.12289717**	0.083389795
0	0.01	0	0	0	0.051626345	2.924271455	-0.64143827
0	0.1	0	0	0	-0.5982221	-21.42732076	-6.327840835
0.0001	0.0001	0.15780312	-0.00081437	0.15404562	-0.750350035	4.30544334	-1.95124992
0.001	0.01	2.2095363	0.0096517	2.13640167	**-10.6353266**	-24.81991883	-22.5635101
		OPTIMIZED					
		Rotation error in deg			**Translation error in cm**		
stdR	**stdT**	x	y	z	x	y	z
0	0	0	0	0	0	1.11438E-03	-1.75659E-04
0.0001	0	0.1501814	0.00152543	0.051467363	**-0.64888895**	2.108552713	**-1.70618995**
0.001	0	-0.9304554	0.00107273	-0.898786696	**0.30736692**	-9.438791183	0.49425971
0	0.001	0	0	0	0.002163605	-0.30704108	0.083389795
0	0.01	0	0	0	0.051626345	2.924271455	-0.64143827
0	0.1	0	0	0	-0.5982221	-21.42732076	-6.327840835
0.0001	0.0001	0.25382848	0.00016301	0.2477051	**-0.75034958**	**4.305438495**	**-1.9512479**
0.001	0.01	0.48233448	-0.01627355	0.4633280	-10.63597427	**-23.968070**	**-22.350672**

Table 4.2: The angular and translational error for the transformation between front and left camera, for different values of Cumulative noise.

		NOT OPTIMIZED					
		Rotation error in deg			**Translation error in cm**		
stdR	**stdT**	**x**	**y**	**z**	**x**	**y**	**z**
0	0	0	0	0	**3.14078E-03**	-1.44161E-01	**3.90152E-02**
0.0001	0	-2.36737E-01	-3.55691E-02	-1.84910E-01	**-6.40138E-01**	-3.91255E+01	-6.60384E+00
0.001	0	-1.00882E+00	1.10114E-02	-8.40895E-01	**8.42317E+00**	-3.00070E+02	-5.01448E+01
0	0.001	0	0	0	4.52670E-01	-2.73476E+01	4.23833E+00
0	0.01	0	0	0	-5.84884E+00	-2.24266E+02	-3.06379E+01
0	0.1	0	0	0	3.50068E+01	-4.03357E+03	-1.01745E+03
0.0001	0.0001	-1.34492E-03	1.88126E-02	4.66248E-02	-2.15189E-01	-1.48278E+01	**2.98691E+00**
0.001	0.01	9.05291E-01	3.79198E-01	4.07749E-01	3.44968E+00	-1.20469E+02	**2.49594E-02**
		OPTIMIZED					
		Rotation error in deg			**Translation error in cm**		
stdR	**stdT**	**x**	**y**	**z**	**x**	**y**	**z**
0	0	0	0	0	3.14107E-03	-1.44161E-01	3.90154E-02
0.0001	0	-1.72583E-01	-2.97113E-02	-1.44507E-01	-6.40140E-01	**-3.91204E+01**	**-6.60257E+00**
0.001	0	-9.24552E-01	8.84481E-02	-7.36161E-01	8.42684E+00	**-2.95187E+02**	**-4.89000E+01**
0	0.001	0	0	0	4.52670E-01	-2.73476E+01	4.23833E+00
0	0.01	0	0	0	-5.84884E+00	-2.24266E+02	-3.06379E+01
0	0.1	0	0	0	3.50068E+01	-4.03357E+03	-1.01745E+03
0.0001	0.0001	-3.96211E-02	4.19596E-03	7.23770E-03	**-2.15188E-01**	**-1.48277E+01**	2.98693E+00
0.001	0.01	1.06713E+00	4.99078E-01	6.93456E-01	**3.44967E+00**	**-1.20464E+02**	2.62176E-02

5 Discussion

The discussion chapter primarily contains interpretations and implications of the Results chapter.

5.1 Evaluation of visual odometry

The proposed algorithm showed promising results for KITTI Odometry, other than being some degrees off in the corners and a bit too small scale estimate it did well on the 1590 long image sequence KITTI 9. The algorithm did so without the addition of loop closure and bundle adjustment. The level of accuracy was still not good enough for the extrinsic calibration algorithm. Loop closure and bundle adjustment might have lifted it to acceptable levels.

5.2 Evaluation of importance of more features

The availability of high quality features proved to be the most important factor. When applying the VO algorithm to the simulator, it found at most 300 features for FAST+BRIEF and 200 for SIFT. In those matches there were also a few wrongfully matched due to matching an equal object from a different location. The threshold for updating matches needed to be lowered for it to have some continuity and this gave less jittery movement in this case.

In Fig. 4.1 it is clear how important it was for the scale estimate to have plenty of keypoints. Evenly distributed features were also an important factor for a good motion estimate, this was also an issue on water which cannot be interpreted as stationary in any way. As a result, sections of the image became featureless. The previously mentioned factors motivates enriching the simulator world with a lot more and varied features. To run that simulation with a decent frame rate would require a better computer or have it recording offline. Another option from enhancing simulations is to set up a camera rig on a boat and record data in a real harbor or natural harbor. The objects on land would be more varied there than in the simulator, and provide better and more features.

5.3 Evaluation of feature extraction algorithms

This thesis primarily used SIFT and FAST+BRIEF for compatibility with ROS. There were big differences in how they found features, where FAST+BRIEF is a corner detector SIFT is more general. The strict conditions of the matches found with SIFT ensured the matches found were really good and it had the upper edge on the KITTI Data set, while for the simulator FAST+BRIEF seemed to have the upper edge. Another positive aspect of the corner detector and binary descriptors is that they were a lot faster to compute, ten times faster than SIFT from our experience.

5.4 Evaluation of masking

It was often unavoidable to have a part of the vehicle present in the image. The vehicle present in the image caused problems if there were few other features present, namely that the features detected on the vessel were considered to be the inliers. To address this problem one could either apply a mask to the image or have a check that discards matches that were stationary over two frames. A euclidean distance-check is very general and can be applied to all cameras, whereas a mask needs to be tailor made to the specific camera, and it is also possible to add an maximum distance to exclude false matches. The distance check had the problem of not handling a standstill situation where all matches were generally stationary.

Since the simulator had a low frame rate and the boat was always moving the distance-check was selected. SIFT detected the most features of the boat, which was very problematic. The effect of masking is shown in Fig. 5.1.

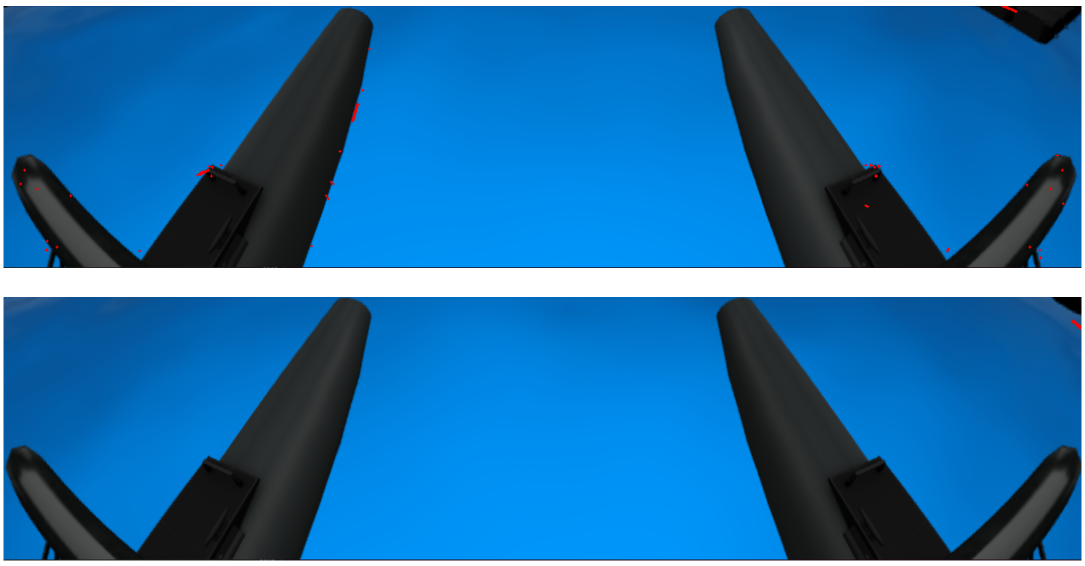


Figure 5.1: Screenshot of vessel with and without a minimum euclidean distance condition for SIFT. In the top picture there is approximately 40 matches, seen as the red dots.

The resulting trajectory when the matches on the vessel were not discarded can be seen in Fig. 5.2. Therefore, naturally, a mask was always applied.

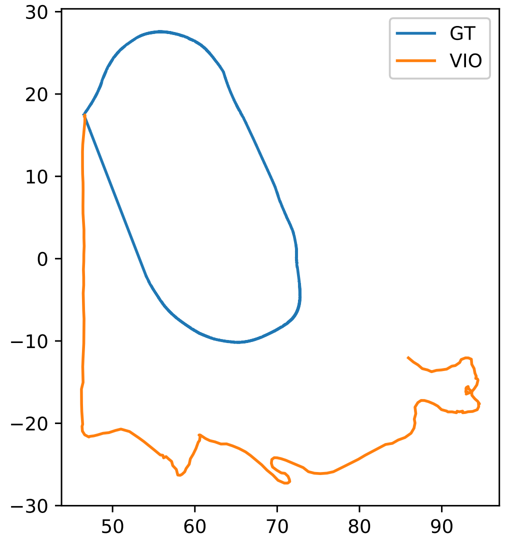


Figure 5.2: Resulting trajectory without max/min euclidean distance constraint between matches.

In Fig. 5.2 the trajectory estimation fails in the corner when the features on the boat were considered to be inliers.

5.5 Evaluation of the addition of gyro

With just visual odometry the algorithm tended to underestimate the rotation, as a result the loops were not closed. Compared to linear acceleration from the accelometer, the angular velocity around an axis is the same for the entire body, which meant that the heading estimated from the gyro could be applied to all of the cameras. To have the majority of the rotation influence taken from the gyroscope improved the estimated trajectory.

5.6 Quality evaluation of trajectories

The trajectories from all of the cameras were behaving as expected, but the trajectories were simply not good enough for the extrinsic calibration. The worst part of it was usually the scale estimate, not being concise enough. If it would be the same for all cameras it would be defined up to scale but, this is not the case. Correct scale estimate would help a lot.

5.7 Evaluation of results with added noise

In Fig. 4.6, observe how the cumulative noise trajectories "derailed", such that the relative positions and orientations were no longer fixed relative to each other. Since the goal behind the extrinsic calibration method we employ was to find the best solution to a fixed distance between trajectories, and the trajectories changing differently impacts the fixed transform, the quality of the result deteriorated significantly with large amounts of this type of noise.

It became apparent that high cumulative noise was not handled well by the estimation of translation, cumulative noise was also the most realistic scenario. In Table 4.1 and 4.2 it is clear that the translational error in Y axis was generally the greatest. The error in Y axis was due to the planar motion of the vessel on still water which made it sensitive to noise in the Y direction of the camera. At the time of writing, the top tier KITTI odometry results were at 0.53 % translation error and 0.0009 [deg/m] rotation error, this would be well acceptable results to do calibration on.

5.8 Determination of section of trajectory

It was discovered that selecting different sections of the trajectory, to estimate extrinsics from, had a big impact on the result. Looking at the trajectory in advance of the calibration could provide benefits. To avoid deterioration of the trajectory caused by cumulative noise one should take the earlier poses rather than the later. The section of trajectory chosen to calibrate with was very important. When having singular motions (no turns), the pseudo-inverse of N_i from Eq. (3.20) diverges. When this happened, the translation would diverge equally much. Therefore, choosing sections of the trajectory when constantly turning the vessel proved to give a much better calibration. In Fig. 4.6, one can see that the first part is dead straight, and one should select a portion of the trajectory from the first corner for best results.

6
Conclusion

The main research question was how automatic extrinsic calibration of a multiple fisheye camera rig can be achieved in a marine environment, a question which proved to be more challenging than first expected. In the simulator, the absence of good features spread across the image was an issue when it came to automatically calibrating the cameras. There are still improvements needed for creating the trajectories enabling an accurate calibration. Another research question was whether we would achieve better results with VO or VIO. It was seen that VIO gave more accurate trajectories than VO when there were not sufficiently good features to track. The computation of trajectories with VO and VIO were both done offline (not while recording the images). Both the VO and VIO algorithm had real-time performance given binary features (FAST+BRIEF), but SIFT was far from that. The trajectory based calibration was very fast, and a pair of cameras with 176 poses each were calibrated in less than half a second. Another question was how to determine whether or not the calibration was accurate enough. In the simulator the ground truth poses were available to compare with. To determine if results were good enough, checks on if features had subpixel accuracy for reasonable distances from the cameras could be performed, or a comparison to ground truth.

6.1 Future work

For future development of this work, the focus should be on developing a better VIO algorithm with the possibility of adding a GNSS sensor, because they are regularly available in boats. With the addition of an accurate GNSS, the drift in position would be eliminated. New data sets should also be made on an actual boat, in a harbor surrounded rich in features. The data recorded should be from several cameras, IMU and GNSS. The new data sets should satisfy the conditions of top tier odometry algorithms (15+ FPS, IMU 100 Hz) to see how well they can do on water applications.

Bibliography

[1] Gustaver, M. (2020) A Chalmers University of Technology Master's thesis template for LaTeX. Unpublished.

[2] C. Mei and P. Rives, Single view point omnidirectional camera calibration from planar grids. in ICRA 2007.

[3] Davide Scaramuzza, Agostino Martinelli, Roland Siegwart. A Toolbox for Easily Calibrating Omni-directional Cameras. Iros, 2006, Benjing, China. inria-00359941

[4] L. Heng, B. Li and M. Pollefeys, "CamOdoCal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry," 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 2013, pp. 1793-1800, doi: 10.1109/IROS.2013.6696592.

[5] L. Heng, M. Bürki, G. H. Lee, P. Furgale, R. Siegwart and M. Pollefeys, "Infrastructure-based calibration of a multi-camera rig," 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 2014, pp. 4912-4919, doi: 10.1109/ICRA.2014.6907579.

[6] M. Ferrera, A. Eudes, J. Moras, M. Sanfourche and G. Le Besnerais, "OV^{2}SLAM : A Fully Online and Versatile Visual SLAM for Real-Time Applications," in IEEE Robotics and Automation Letters, IEEE 2021. arXiv:2102.04060 [cs.CV]

[7] Richard Szeliski, Computer Vision: Algorithms and Applications, draft: Springer September 3, 2010, pp. 58-60

[8] Azzam, R., Taha, T., Huang, S. et al. Feature-based visual simultaneous localization and mapping: a survey. SN Appl. Sci. 2, 224 (2020). https://doi.org/10.1007/s42452-020-2001-3

[9] Yang Liu and H. Zhang, "Indexing visual features: Real-time loop closure detection using a tree structure," 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, 2012, pp. 3613-3618, doi: 10.1109/ICRA.2012.6224741.

[10] Bill Triggs, Philip Mclauchlan, Richard Hartley, Andrew Fitzgibbon. Bundle Adjustment – A Modern Synthesis. International Workshop on Vision Algorithms, Sep 2000, Corfu, Greece. pp.298–372, 10.1007/3-540-44480_21 . inria-00548290

[11] D. Galvez-López and J. D. Tardos, "Bags of Binary Words for Fast Place Recognition in Image Sequences," in IEEE Transactions on Robotics, vol. 28, no. 5, pp. 1188-1197, Oct. 2012, doi: 10.1109/TRO.2012.2197158.

[12] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley, Reading, Massachusetts, 1993.

[13] Ferrera, Maxime et al. "OV^{2}SLAM : A Fully Online and Versatile Visual SLAM for Real-Time Applications." (2021).

[14] T. Qin, P. Li and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," in IEEE Transactions on Robotics, vol. 34, no. 4, pp. 1004-1020, Aug. 2018, doi: 10.1109/TRO.2018.2853729.

[15] Mukherjee, D., Jonathan Wu, Q.M. & Wang, G. A comparative experimental study of image feature detectors and descriptors. Machine Vision and Applications 26, 443–466 (2015). `https://doi.org/10.1007/s00138-015-0679-9`

[16] Knuth: Computers and Typesetting,
`http://www-cs-faculty.stanford.edu/~{}uno/abcde.html`

[17] MSW motor, 2020, 15 Types of Automotive Sensors Your Vehicle is Likely to Have
`https://mzwmotor.com/automotive-sensor-types/`

[18] Author(s) Initial(s). Surname(s), Title of the Book, xth ed. City of Publisher, (U.S. State or Country if the City is not 'well known'): Publisher, Year of Publication, pp. xxx–xxx

[19] G. Carrera, A. Angeli and A. J. Davison, "SLAM-based automatic extrinsic calibration of a multi-camera rig," 2011 IEEE International Conference on Robotics and Automation, Shanghai, 2011, pp. 2652-2659, doi: 10.1109/ICRA.2011.5980294.

[20] Hernandez-Juarez, D., Schneider, L., Cebrian, P. et al. Slanted Stixels: A Way to Represent Steep Streets. Int J Comput Vis 127, 1643–1658 (2019). `https://doi.org/10.1007/s11263-019-01226-9`

[21] D. Nister, "An efficient solution to the five-point relative pose problem," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 6, pp. 756-770, June 2004, doi: 10.1109/TPAMI.2004.17.

[22] P. Liu, L. Heng, T. Sattler, A. Geiger and M. Pollefeys, "Direct visual odometry for a fisheye-stereo camera," 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 2017, pp. 1746-1752, doi: 10.1109/IROS.2017.8205988.

[23] E. Fernández-Moral, J. González-Jiménez, P. Rives and V. Arévalo, "Extrinsic calibration of a set of range cameras in 5 seconds without pattern," 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 2014, pp. 429-435, doi: 10.1109/IROS.2014.6942595

[24] F. Zhao, T. Tamaki, T. Kurita, B. Raytchev and K. Kaneda, "Marker-based non-overlapping camera calibration methods with additional support camera views," 2018 Image and Vision Computing, Vol. 70, pp. 46-54, doi: https://doi.org/10.1016/j.imavis.2017.12.006.

[25] Ariane Spaenlehauer, Vincent Frémont, Ahmet Sekercioglu, Isabelle Fantoni. A Loosely-CoupledApproach for Metric Scale Estimation in Monocular Vision-Inertial Systems. IEEE InternationalConference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2017), Nov 2017,Daegu, South Korea. pp.137-143. hal-01678915

[26] Brian Bingham, Carlos Aguero, Michael McCarrin, Joseph Klamo, Joshua Malia, Kevin Allen, Tyler Lum, Marshall Rawson, & Rumman Waqar (2019).

Toward Maritime Robotic Simulation in Gazebo. In Proceedings of MTS/IEEE OCEANS Conference.

[27] Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981

[28] Andreas Geiger, Philip Lenz, & Raquel Urtasun (2012). Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In Conference on Computer Vision and Pattern Recognition (CVPR). 4

[29] Kannala, Juho & Brandt, Sami. (2006). A Generic Camera Model and Calibration Method for Conventional, Wide-Angle, and Fish-Eye Lenses. IEEE transactions on pattern analysis and machine intelligence. 28. 1335-40. 10.1109/TPAMI.2006.153.

[30] Flir, `https://www.flir.eu/globalassets/support/iis/knowledge-base/modifiedstereoaccuracy.xls`

[31] C. Forster, L. Carlone, F. Dellaert, D. Scaramuzza. (2015). IMU Preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation. Robotics: Science and Systems (RSS) conference.

[32] `https://en.wikipedia.org/wiki/Scale-invariant_feature_transform#Feature_matching_and_indexing`

[33] Figure of epipolar geometry: `https://en.wikipedia.org/wiki/Epipolar_geometry#/media/File:Epipolar_geometry.svg`

[34] D. G. Lowe. (2004). Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision.

[35] E. Rosten and T. Drummond. (2006) Machine Learning for High-Speed Corner Detection. ECCV, 9th, Vol. 1. pp. 433-436.

[36] M. Calonder, V. Lepetit, C. Strecha and P. Fua. (2010) BRIEF: Binary Robust Independent Elementary Features. ECCV, 11th. pp. 778-792.

[37] M. Muja and D. G. Lowe. (2009) Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration, International Conference on Computer Vision Theory and Applications

[38] G. Falco, M. Pini and G. Marucco. (2017) Loose and Tight GNSS/INS Integrations: Comparison of Performance Assessed in Real Urban Scenarios. 17(2):255