ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

V4RL VISION FOR ROBOTICS LAB
Vision for Robotics Lab
Prof. Margarita Chli

Semester Thesis

Initialization Test Algorithm For EKF Based Pose Estimation Algorithms

Spring Term 2018

Supervised by:
Prof. Margarita Chli
Patrik Schmuck
Lucas Pinto Teixeira

Author:
Armando Amoros

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Initialization Test Algorithm For Visual Inertial Odometry Based Flying Systems

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

Author(s)

Armando Daniel	Amoros Lozano

Student supervisor(s)

Patrik Lucas Pinto	Schmuk Teixeira

Supervising lecturer

Margarita	Chli

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

The above written work may be tested electronically for plagiarism.

Place and date	Signature

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

Contents

Abstract

Autonomous flight of UAVs used for mapping, exploration, data collection, security monitoring, and so on, has become more common and important in recent years. The present work provides a solution to automatically test the initialization of pose estimation algorithms based on Kalman filters (e.g. ROVIO and MSF), combining its pose measurements with external IMU acceleration data measurements.

Symbols

Acronyms and Abbreviations

ETH	Eidgenössische Technische Hochschule
EKF	Extended Kalman Filter
IMU	Inertial Measurement Unit
UAV	Unmanned Aerial Vehicle
ROVIO	Robust Visual Inertial Odometry
MSF	Multi Sensor Fusion
FFT	Fourier Fast Transform
VI	Visual Inertial
ROS	Robot Operating System
MPC	Model Predictive Control
PID	Proportional, Integral, Derivative
RMSE	Root Mean Squared Error
MAV	Micro Aerial Vehicle
rqt	rqt is a software framework of ROS
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

Chapter 1

Introduction

1.1 Motivation

When thinking about autonomous flight of UAVs many concepts and topics to explore and improve may come to mind, such as sensors, pose estimation methods, energy consumption, control, and so on. But in daily life we are rarely concerned with how easily all these factors are tested when getting integrated into a real system. The amount of resources spent on setting up and initializing tests before implementation of an actual innovation can be high. This waste in time and human effort, as well as UAVs damage, etc. can be reduced with the work presented in the following chapters.

To understand the problem better, imagine in particular a scenario where multiple UAVs, using a visual inertial odometry framework as a pose estimator combined with IMU sensors (e.g. ROVIO in cascade with MSF) are used to explore a certain area. In order to achieve this, every UAV has to take off manually, a person must initialize the pose estimation algorithms and make sure they initialize correctly (in an empirical way), to finally give the control to the UAV autonomous control feedback loop. Under this scenario, it is very difficult and dangerous for one person to carry out all these tasks himself.

1.2 Objective

The objective of this project was to develop the (*initialization test algorithm*) that allows a single person to work the scenario mentioned above, testing the automatic initialization of the whole pipeline of pose estimator algorithms (ROVIO and MSF) embedded with the control loop, while flying after manual take off. Once the automatic initialization is positively tested, the pilot can safely hand in control directly to the autonomous control feedback loop, eliminating the need for a human to decide whether the pose estimator algorithms initialized correctly or not.

1.3 Framework

The following points were essential in the development of this project, since they were taken as a base for the simulation system in which all the data was gathered

and the *initialization test algorithm* was tested.

1.3.1 ROVIO

ROVIO stands for Robust Visual Inertial Odometry. This framework was developed by Michael Bloesch et al. at the Autonomous System Lab of ETH Zurich[1]. It is a visual inertial odometry algorithm, which calculates and directly integrates the pixel intensity errors of image patches using one or more cameras and fuses inertial measurements by means of an iterated extended Kalman filter.

The state of the filter is formulated in a fully robocentric fashion. Therefore, the algorithm estimates the location of 3D landmarks with respect to the actual camera pose. Through this approach no time is expended in the initialization procedure, leading to a real plug and play pose estimator framework.

It is clear now that the framework takes images and inertial measurements as input, this information is given by the VI Sensor directly. It outputs velocity and position, the last one relative to the initial value established in the framework initialization.

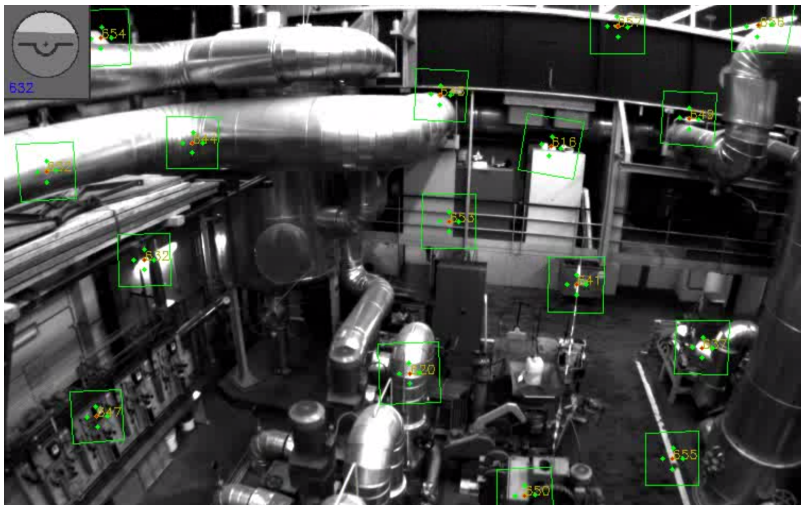

Figure 1.1: ROVIO interface showing the tracked features

1.3.2 MSF

MSF stands for Multi Sensor Fusion. This framework was developed by Simon Lynen et al. at the Autonomous System Lab of ETH Zurich[2]. It was created to solve the challenge of integrating and calibrating different sensors before deployment in the field, as well as the problems generated for the different measurement rates and delays. All of this is required to get higher accuracy and robustness in the system, but most often left aside for the simplicity of not using all the sensor information present.

MSF is a framework able to process delayed, relative and absolute measurements coming from more than one sensor of the same or different types. The frameworks allows a self calibration of the sensors while running. All of this is done through

the implementation of an iterated extended Kalman filter. In case a sensor signal is lost, the filter uses a state buffer to allow for a re-linearization of the prediction.

In this project the MSF takes as input the pose output from the ROVIO framework and combines it with the inertial measurements from the IMU mounted directly in the UAV itself. As outputs the MSF retrieves a pose and velocities, which will be used after for the control loop.

1.3.3 VI Sensor

The VI Sensor (stands for Visual Inertial Sensor) was developed at the Autonomous Systems Lab of ETH Zurich and Skybotix[3]. The sensor counts with a stereo camera data streams and a full time synchronized and calibrated IMU.

This sensor is mounted directly on the UAV and is used to feed the information required for the ROVIO framework (images, inertial measurements).

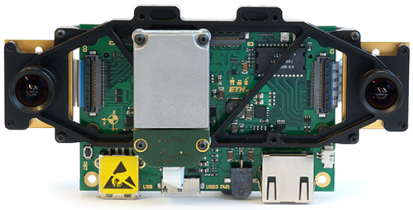

Figure 1.2: VI Sensor

1.3.4 V4RL MAV Ground Control

The v4rl_mav_ground_control framework is an interface, developed in ROS environment, for planning and executing trajectories for one or more UAVs through pre-established waypoints that avoid collisions between UAVs.

This framework works using ROVIO and MSF as its main pose estimators, avoiding any reliance on external measurement; thus, it is able to operate fully autonomously.

This framework was used to generate all the data used for the experiments. The *initialization test algorithm* depicted in this report is intended to be implemented as a ROS node into this framework itself.

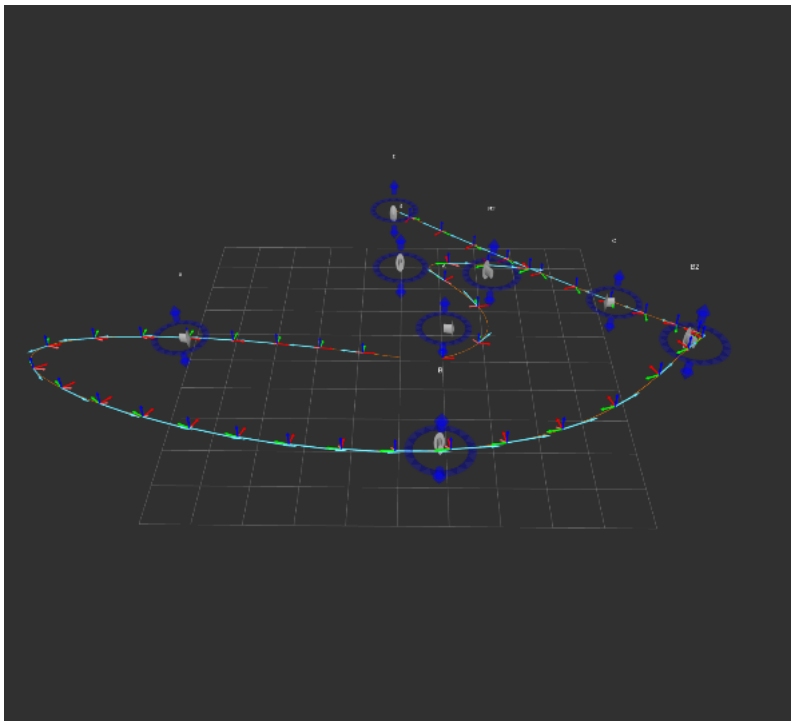

Figure 1.3: Example of a trajectory generated in the v4rl_mav_ground_control framework

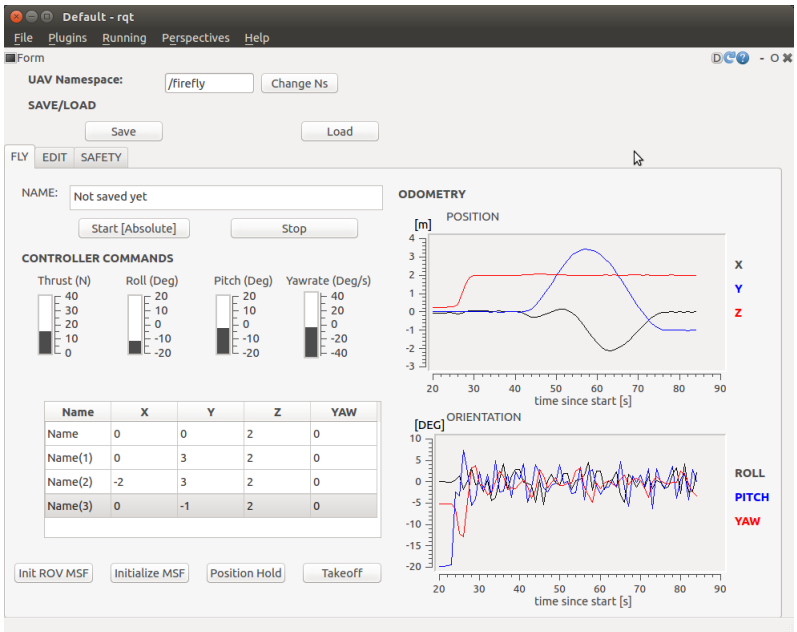

Figure 1.4: rqt interface of the v4rl_mav_ground_control framework

Chapter 2

Implementation Method

2.1 Data Available

Pose estimation frameworks such as ROVIO and MSF give back as output: pose estimates (relative to the initial position from where the algorithm started) and velocity estimates. Besides that, the information from the camera and IMU from the VI sensor is also available, as well as the information from an external IMU mounted on the UAV itself.

2.2 UAV Autonomous Control Pipeline

Before this project, the control pipeline for each one of the UAVs within the v4rl_mav_ground_control framework had a control pipeline with pose feedback provided by external sensors (most commonly VICON systems). Part of this project was to integrate the visual inertial sensor (VI Sensor) and the visual inertial odometry framework (ROVIO) to close the autonomous control pipeline. In this way, the whole system is automated without any necessity for external sensors to provide measurements.

After completion of this project, the autonomous pipeline starts with the VI sensor providing images and inertial measurements to ROVIO. After ROVIO processes the information, it outputs pose estimates that later feeds the MSF. On the other hand, the MSF is also receiving inertial measurements from the IMU mounted in the UAV itself. Then, the MSF outputs pose estimates that feed the core controller of the UAV (conformed by the trajectory planner, nonlinear MPC[4, 5] and PID attitude controller), which will finally give the commands to the motors. This flow is shown in figure 2.1

2.3 Initialization Test Algorithm Pipeline

Once the autonomous control pipeline was complete, the main focus of this project was to implement the *initialization test algorithm* which tests the automatic initialization of the control pipeline, guaranteeing that any of the EKF involved in the pipeline did not diverge, giving confidence that the UAV is capable of safely flying in an autonomous mode.

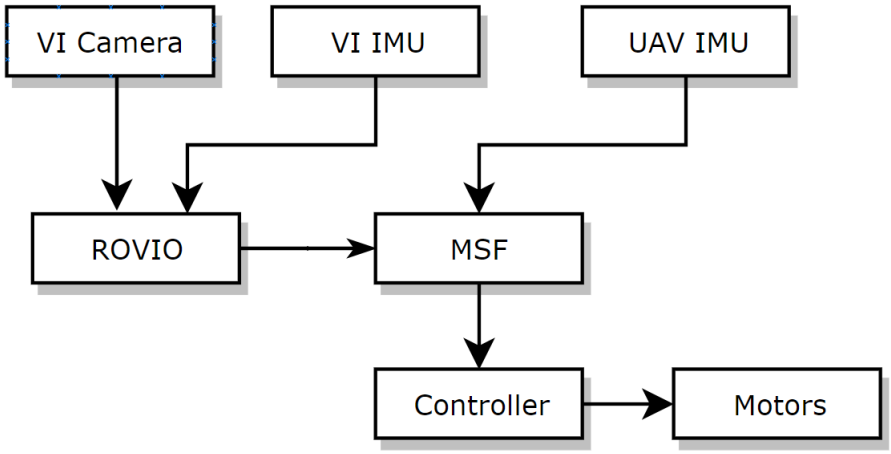

Figure 2.1: UAV control pipeline applied in the v4rl_mav_ground_control framework

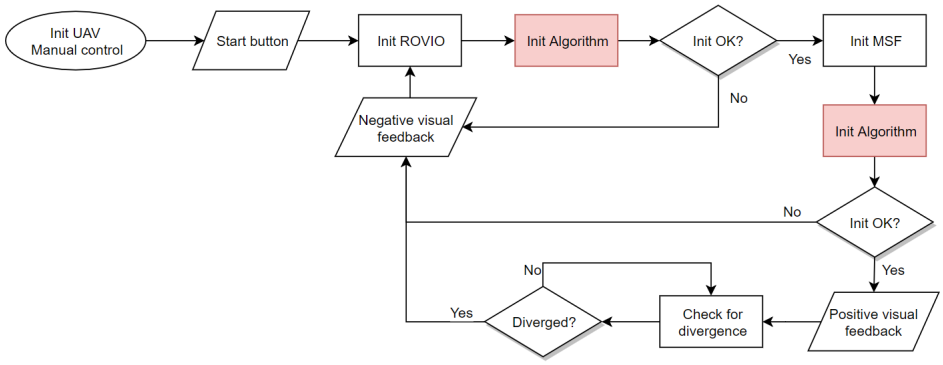

Figure 2.2: Automatic initialization pipeline, showing the integration of the *initialization test algorithm* in red

This *initialization test algorithm* is made to run after the initialization of the pose estimation frameworks, to ensure their convergence and give a feedback (positive or negative) to the pilot. In case of a correct initialization, the test algorithm should keep running in the background, to keep ensuring the convergence of the EKF based estimators. This flow is shown in figure 2.2

2.4 First Approach: IMU Acceleration and Angular Velocities FFT

When doing the integration of ROVIO to the v4rl_mav_ground_control framework, it was evident that the correct initialization of ROVIO was somehow dependant on the first movements of the UAV and their frequency, just after initializing the framework, with high frequencies being the major cause of divergence in the framework.

This approach was intended to try to identify some pattern in velocities and acceleration frequencies that would favor the EKF convergence or, on the other hand, promote its divergence. This way it would be possible to identify possible good or bad initializations.

After gathering data with different random initialization movements of the UAV, leading to convergence or divergence, it was difficult to recognize a pattern in the angular velocities or accelerations that would promote divergence or convergence specifically. Since the EKF acts as a low pass filter, all the high frequencies were masqued, preventing the data from showing a useful pattern and making the divergence and convergence graphs look practically the same, as it is possible to see in figure 2.3.

Moreover, the fact that the presence of the high frequencies would promote the EKF to diverge, doesn't imply that the absence of this would promote the algorithm to converge, since it was observed that a variety of initialization movements with no high frequencies diverged in the same manner. The above mentioned reasons were major deterrents in continuing this approach.

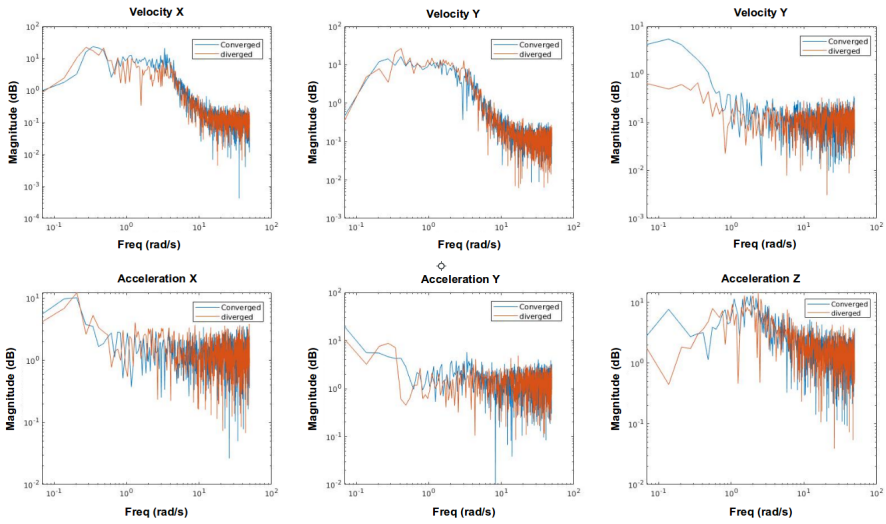

Figure 2.3: Comparison of velocity and acceleration FFT magnitude graph, between a divergence test (orange) and a convergence test (blue)

2.5 Second Approach: Acceleration RMSE and Final Pose

The second and final approach with which the *initialization test algorithm* was developed is based on the Acceleration RMSE and the final pose of the UAV. The combination of these two measurements (as a logical AND after comparing with predefined thresholds) would result in the output of the *initialization test algorithm* defining if the EKF based pose prediction algorithm converged or diverged.

2.6 Assumptions

The *initialization test algorithm* makes the following assumptions.

When asked, right after initiating the EKF based pose prediction algorithm, the pilot is capable of performing an initialization maneuver with normal random movements enclosed in a final position threshold square (specified in the parameters section 2.9) relative to the initial position of the UAV (when the EKF based prediction algorithm is initiated). The pilot should be capable of maintaining and stopping the UAV inside the square at any time during the initialization.

2.7 Acceleration RMSE

The acceleration RMSE is used to measure the accuracy of the estimated acceleration values compared with the predicted acceleration values.

The estimated accelerations are obtained deriving the velocities given by the EKF based pose estimation algorithm, and in order for the *initialization test algorithm* to declare a good initialization it is expected that these estimates are consistent with the prediction accelerations given by the IMU data. For the mathematical comparison to be possible, a preprocessing of both measurements (captured over the initialization time window) has to be performed in the following way.

First the IMU acceleration measurements have to be compensated for bias. The bias of the IMU is calculated as the average of the whole captured data set, which will later be subtracted from the same measurements. After, it is necessary to apply a low pass filter to the data set, since as it was mentioned in the first approach, the EKF from the pose estimation framework acts like low pass filter itself, getting rid off the high frequencies.

Besides, it is needed to derive the velocity estimate received as an output from the EKF based prediction algorithm. This way it is possible obtain the acceleration prediction, which later should be scaled accordingly to a rate between the sampling time from the EKF based pose estimation algorithm and the IMU measurements.

Once both data sets are pre-processed correctly, a RMSE value for this time window is obtained as shown in equation 2.1. This value is expected to be smaller than a predefined threshold stated in the parameters section 2.9. If the value is outside the threshold the *initialization test algorithm* would state that the initialization of the EKF based pose estimation framework has diverged.

$$AccelerationRMSE = \sqrt{\sum_{n=1}^{N} \frac{(EkfAlgorithmAccel_n - ImuAccel_n)^2}{N}} \tag{2.1}$$

2.8 Final Pose

A complementary measure is used by the *initialization test algorithm* to ensure the non divergence of the EKF based pose estimation framework while initializing (and

under the assumptions previously mentioned). This measure is the final position of the UAV relative to the position of the UAV when the EKF based pose estimation framework was initialized. The measurement is taken within the initialization time window after the initialization starts, just when the data acquisition for the acceleration RMSE is finished.

This final position value is expected to be inside the final position threshold square defined in the parameters section 2.9. If the final position is localized outside the square, the *initialization test algorithm* would state the divergence of the EKF based pose estimation framework. An example of divergence acceleration data is shown in figure 2.5 and convergence acceleration data is shown in figure 2.4,it is easy to observe the difference in the correlation between IMU and ROVIO in each case.

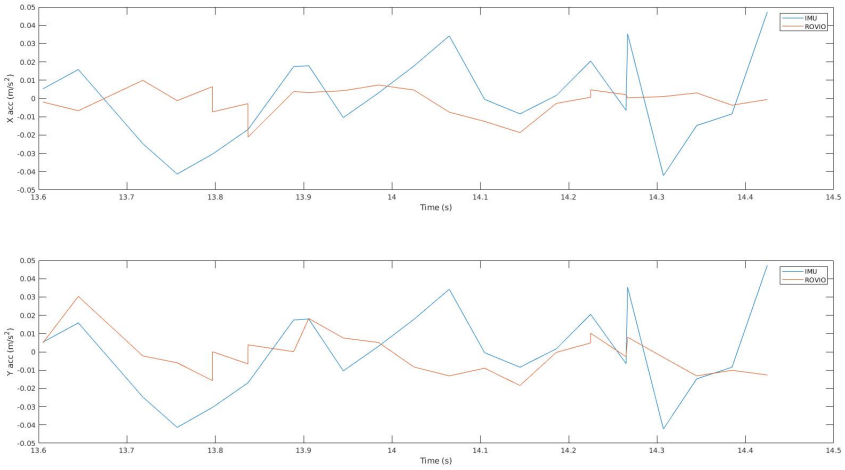

Figure 2.4: Comparison of acceleration in X and Y axis, between ROVIO (orange) and IMU (blue), this data was classified by the *initialization test algorithm* as convergence

2.9 Parameters

The resume of the configurable parameters of the *initialization test algorithm*, and the values used for the latests experiments shown in this report are the following ones.

Time window: This parameter refers to the time lapse after the initialization of the EKF based pose estimation framework in which we store the IMU and the EKF based pose estimation framework data for later processing. This time window is set to $3 seconds$.

IMU bias compensation: This parameters is calculated online, which means we do not have a fix value set for it, and it may change between every initialization. The IMU bias compensation is calculated taking the mean of all the data gathered during the initialization time window and then subtracting it to the data itself.

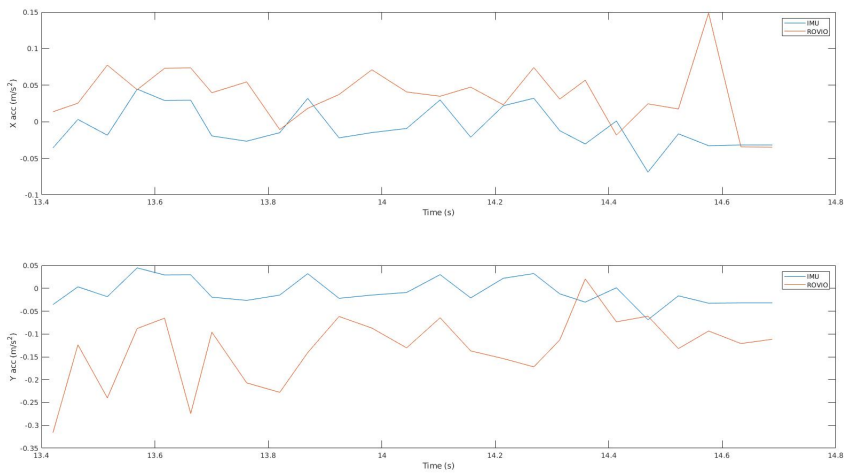

Figure 2.5: Comparison of acceleration in X and Y axis, between ROVIO (orange) and IMU (blue), this data was classified by the *initialization test algorithm* as divergence

Low Pass filter frequency: The normalized pass band frequency used in the low pass filter, used for the IMU data preprocessing, is $0.5\pi * rad/sample$.

Scale factor for velocity derivative: After deriving the velocity of the EKF based pose estimation framework, it is necessary to multiply the result by a factor between the sample time of the IMU and the EKF based pose estimation framework, such factor is calculated as shown in the equation 2.2. Thus, this factor changes accordingly with the samples times of the systems used (e.g. ROVIO sample time is $0.4seconds$, the VI-IMU sample time is $0.1seconds$, which leads to a multiplication factor of 4 for the ROVIO acceleration prediction data).

$$ScaleFactor = \frac{EkfAlgorithmSampleTime}{ImuSampleTime} \tag{2.2}$$

Acceleration RMSE Threshold: After getting the acceleration data from the IMU and the EKF based pose estimation framework during the initialization time window, the acceleration RMSE is calculated and the result is compared with this parameter threshold. This threshold is set to be $0.5m/s^2$. If the acceleration RMSE is smaller than the acceleration RMSE threshold, the *initialization test algorithm* will declare, for this parameter only, the convergence of the EKF based pose estimation framework (the *initialization test algorithm* still needs to declare convergence for the final position threshold, to declare the general convergence).

Final Position Threshold: After the initialization time windows is finished, the *initialization test algorithm* is going to compare this parameter against the last position given as an output from the EKF based pose estimation framework. The last position is relative to the position of the UAV when the EKF based pose estimation framework was initialized. For the experiments shown in the following section, this threshold is set to be the $\pm 30cm$ square around the initial position of the UAV (figure 2.6) when the EKF based pose estimation framework was initialized.

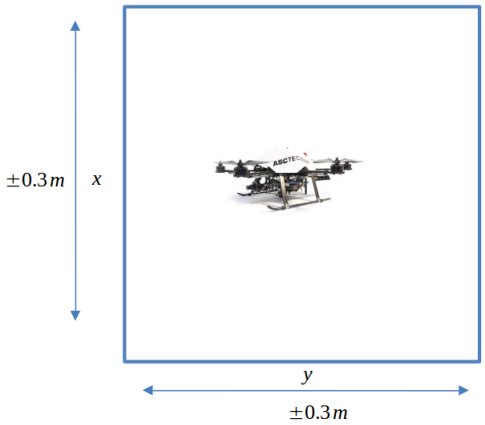

Figure 2.6: Final position threshold square. If the UAV is detected outside the square the *initialization test algorithm* would state divergence

Chapter 3

Experiments

3.1 Data Gathering (Creating Test Sets)

The test sets were created in ROS virtual simulation using the v4rl_mav_ground_control framework with ROVIO integration. Using one firefly UAV, the procedure to create the test sets used in the experiments is described next. This procedure was the same for each one of the experiments, although the specific conditions and initialization were different (world, movements, repetitions, etc); thus, these will be described in the specific section of each experiment.

Each test set was created in a predefined world with specific visual characteristics starting with the UAV in hovering position at a $2m$ altitude. A uniformly, randomly distributed initialization trajectory within the $\pm 30cm$ square was generated, and these trajectories had an approximate duration of $5seconds$ each, so it was possible to cover flawlessly the $3seconds$ needed in the time window of the *initialization test algorithm*, as specified in the parameters section 2.9. The EKF based pose estimation framework was then initialized, and the initialization trajectory movements were executed right after.

The collection data for each one of the test sets includes the images and inertial measurements from the VI sensor, the pose and velocity output from ROVIO and MSF, the inertial measurements from the IMU mounted in the UAV itself, and the ground-truth pose of the UAV given by the simulation environment.

Although the whole purpose of this project is to get rid off the ground-truth measurements, so the UAV is able to be fully autonomous, at this point of the validation of the *initialization test algorithm*, the ground-truth measurements were only used to label the initialization of the test sets (labeled as convergence or divergence) to later compare the output of the *initialization test algorithm* against those labels. The labeling process is described in the following chapter.

3.2 Labeling of the Test Sets

Once all the test cases were generated (as explained in section 3.3, 3.4 and 3.5), it was necessary to label the output of ROVIO and MSF in each one of the test cases as a convergence or divergence, so we could compare these labels with the output of the *initialization test algorithm*. Then, it was possible to prove the quality of the

initialization test algorithm results.

In order to label the tests sets, two measurements were obtained from the data. The first one was obtained performing a comparison between the UAV ground-truth physical position and the estimated position from the EKF based pose estimation framework during the whole trajectory. The second measurement considered the difference between the ground-truth final position and the EKF based pose estimation framework.

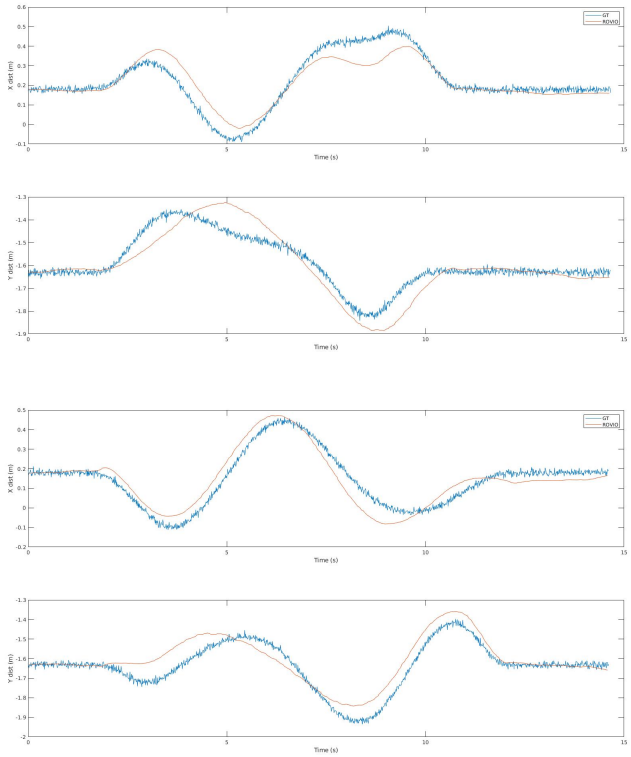

Figure 3.1: Comparative position plot in x and y axis of two test sets labelled as convergence, ROVIO in red and ground-truth in blue

Similar to the Acceleration RMSE used for the *initialization test algorithm*, the first measurement, comparing the ground-truth physical position and the estimated position from the EKF based pose estimation framework, was calculated as a RMSE, this time in position instead of acceleration (equation 3.1). This measurement captures the accuracy of the estimated position value compared against the real value. The threshold for this measurement was set to $0.1m$, which means, the position RMSE had to be lower than $0.1m$ to be marked as a convergence.

$$PsotionRMSE = \sqrt{\sum_{n=1}^{N} \frac{(EkfAlgorithmFinalPos_n - GroudTruthFinalPos_n)^2}{N}} \tag{3.1}$$

The second measurement helps to test the quality of the convergence, capturing the difference of the final position of the EKF based pose estimation framework against the ground-truth final position (equation 3.2). This threshold was set to $0.15m$, which means any test set in which the EKF based pose estimation framework finished $0.15m$ or more away from the ground-truth position, in any direction, was marked as a divergence (figure 3.2.

$$FinalPositionError = |EkfAlgorithmFinalPos - GroudTruthFinalPos| \quad (3.2)$$

The two measurements mentioned above, work as a logic AND operation to get the label of the test set, divergence is equivalent to 0 and convergence is equivalent to 1, which means the only way for the label to converge is if the two measurements converge (as shown in the figure 3.1).

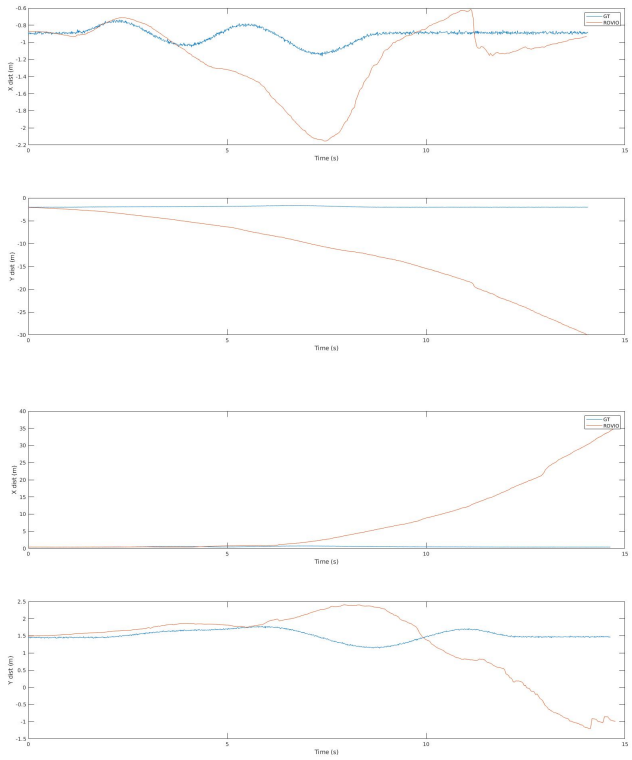

Figure 3.2: Comparative position plot in x and y axis of two test sets labelled as divergence, ROVIO in red and ground-truth in blue

3.3 Trying Different Initialization Paths (Experiment 1)

For this experiment 5 different worlds (figure 3.3) with specific visual features in each (lines, corners and noise) were used. 20 different initialization trajectories were

used in each one of the 5 worlds, resulting in 100 test sets.

The purpose of using specific and different features in each world was to force the EKF based pose estimation framework to the limits, making it diverge part of the time. It was important to have similar number of converging and diverging initializations in the test sets, in order to prove that the *initialization test algorithm* works in both cases.

After labeling the 100 test sets, the rate of labels with convergence was 84%.

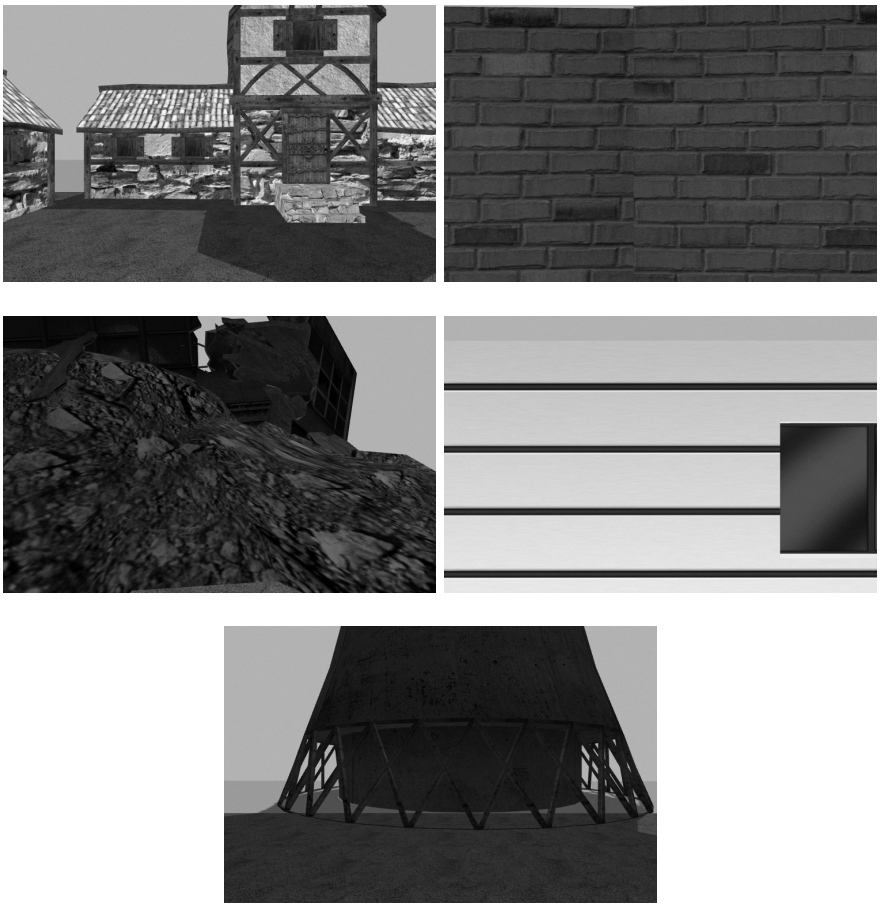

Figure 3.3: 5 Worlds used for experiment 1 in section 3.3 and experiment 2 in section 3.4

3.4 Repeating Difficult Paths (Experiment 2)

For this experiment 5 different initialization paths were taken from experiment 1. These were the paths with higher number of divergences. The reason for choosing these paths is that it nearly impossible to know what the pilot is going to do when asked to do the initialization maneuver; thus, nice initialization movements are not expected and covering the most difficult ones gives more certainty that the algorithm is going to perform well under difficult situations.

These 5 differente initialization paths were tested in the same 5 different worlds as experiment 1 (figure 3.3), but this time each path was tested 5 times, resulting in 125 test sets.

After labeling the 125 test sets, the rate of labels with convergence was 88%. This rate also shows that the initialization may not be dependant on the initialization path, but possibly on the exact initialization conditions of the EKF, since the rate (88%) is greater than the rate in the experiment 1 (84%).

3.5 Random Respawn in a 360 World (Experiment 3)

This experiment was conducted in 2 worlds, which were different from the previous experiments. The previous experiments had 1 position only, having always the same scene. For this experiment the worlds are a 360 degrees compositions (figure 3.4). This compositions has many different features mixed in every direction.

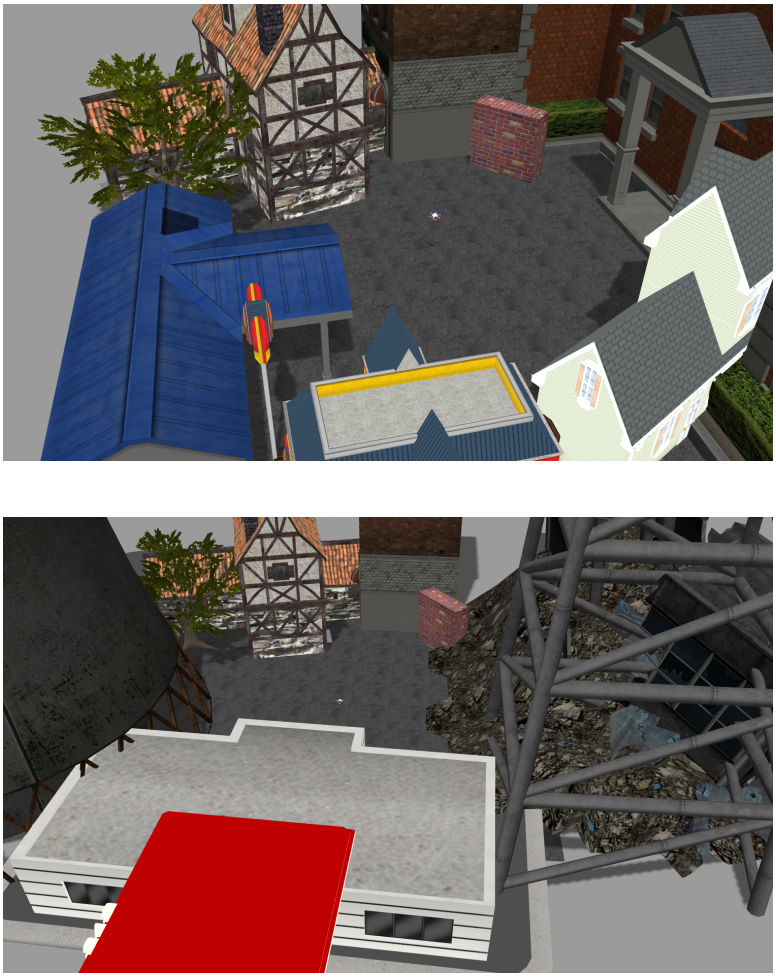

Figure 3.4: Aerial scene of the 2 Worlds used for experiment 3 in section 3.5

This experiment spawned the UAV in a random position with a random orientation, taking off up to a $2m$ altitud (as in the previous experiments) and executing the random initialization path. Doing the experiment in 2 worlds, 30 different spawn positions (some examples shown in fig3.5) and 10 different initialization paths per spawn, the result was 600 different test sets.

The experiment was conducted in this way to explore all the possible situations one may encounter in the real world, including a large variety of different mixed visual features, different random depth distances to the initialization scene and different initialization paths. Forcing this way the EKF based pose estimation frameworks to much harder situations.

After labeling the 600 test sets, the rate of labels with convergence was 37%. The drop in the convergence rate in this experiment was observable. This was attributed to the random spawns, which will position the UAV too far away from the scene itself (around $\sim 1m$ for experiment 1 and 2, and $\sim 6m$ for this experiment), making the EKF based pose estimation frameworks diverge more easily.

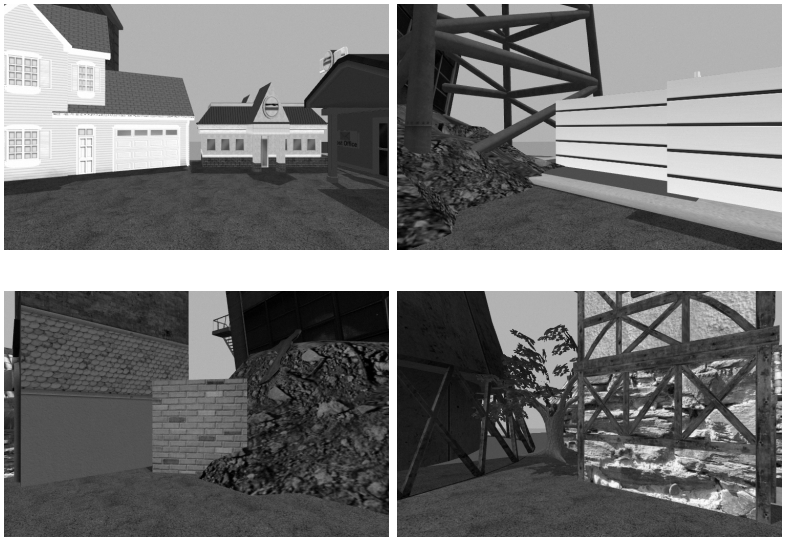

Figure 3.5: Random spawns used for experiment 3 in section 3.5

Chapter 4

Results

4.1 Accuracy and Precision

To evaluate the quality of the *initialization test algorithm*, two metrics were chosen, accuracy (equation 4.1) and precision (equation 4.2). The importance of the precision value is really high due to the nature of this project, which is an algorithm that defines between the action of handing in or not the control of the UAV to an algorithm that may be diverging. It is easy to see that the most worrying case is the presence of false positives, since this would be the most expensive error, with high probability of crashing the UAV and causing an accident. The precision value captures the information about these false positives.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

The accuracy captures the overall performance of the *initialization test algorithm*, capturing also information about the false negatives, although less critical also important.

4.2 Experiments Results

The results of the three previously mentioned experiments are captured in the following table 4.1. The values of accuracy and precision reported were obtained comparing the output of the *initialization test algorithm* applied to the test sets compared with the test sets labels previously obtained with the method mentioned in section 3.2.

In addition, the coherence in the pipeline of the v4rl_mav_ground_control framework with ROVIO integrated was tested, checking for the accuracy and precision between the output of the *initialization test algorithm* when applied to the ROVIO and MSF in the same test set (expecting them to be equal).

	Experiment 1		Experiment 2		Experiment 3	
	Accuracy	Precision	Accuracy	Precision	Accuracy	Precision
Classification -> Rovio Prediction	100%	100%	100%	100%	96.2%	97.1%
Classification -> MSF Prediction	100%	100%	100%	100%	95.5%	96.3%
Rovio Prediction -> MSF Prediction	100%	100%	100%	100%	100%	100%

Figure 4.1: Accuracy and precision results of the experiments

4.3 False Positives

As discussed in the previous sections, the false positive results given by the *initialization test algorithm* are the most concerning results to prevent from happening due to the high risk/cost involved with giving a diverged control to a UAV.

When investigating the false positive cases and the causes for the *initialization test algorithm* to fail this way, it was found that in all the cases where a false positive was reported, the EKF based pose estimation framework didn't diverge in a regular way, instead it suffered what seems to be an inversion (figure 4.2). The reason for this is subject of further discussion.

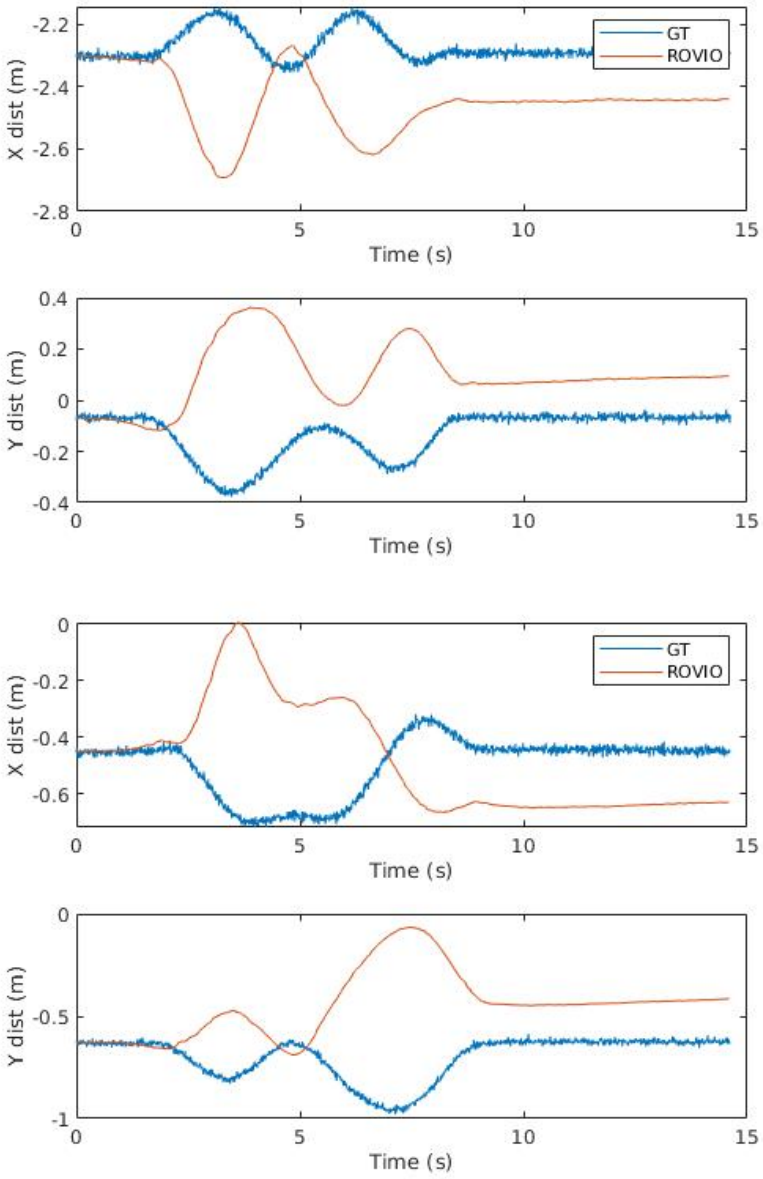

Figure 4.2: Comparative position plot in x and y axis of two test sets, labelled as divergence, but marked as convergence by the *initialization test algorithm*, ROVIO in red and ground-truth in blue

Chapter 5

Discussion

5.1 Achievements

The presented algorithm achieved prominent results when applied to the different test sets created for the experimentation of this project. Positioning itself as an algorithm capable to identify, with high certainty rate, the divergence or convergence of EKF based pose estimation frameworks, providing the autonomous flight world with a possibility to reduce risks and costs when initializing its autonomous control loops.

5.2 Future Work

So far the *initialization test algorithm* is implemented in MATLAB only, the test sets were exported from the ROS environment in BAG files and later processed in MATLAB to test the algorithm itself. As a future work it would be desirable to deploy the *initialization test algorithm* in a ROS node for testing in a real platform.

Find a way to detect ROVIO inversions described in the false positive section, would be helpful to prevent the pilot from handing in the control to the autonomous control pipeline while diverging.

Cross validate a series of different parameters to ensure the optimal performance of the *initialization test algorithm*.

Bibliography

[1] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach," in *Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2015.

[2] S. Lynen, M. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013.

[3] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. Furgale, and R. Siegwart, "A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam," in *IEEE International Conference on Robotics and Automation (ICRA), Hongkong, China*.

[4] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, "Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system," in *Robot Operating System (ROS) The Complete Reference, Volume 2*, A. Koubaa, Ed. Springer.

[5] M. Kamel, M. Burri, and R. Siegwart, "Linear vs Nonlinear MPC for Trajectory Tracking Applied to Rotary Wing Micro Aerial Vehicles," *ArXiv e-prints*, Nov. 2016.