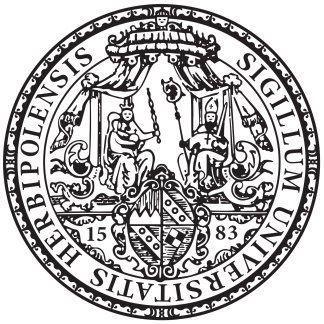
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Relative Position Estimation and Wireless Communication between two CubeSats

Master Thesis

Author:	Steffen, Christopher
Program of Study:	Luft- und Raumfahrtinformatik Master
Matriculation Number:	2450010
Supervisor:	M.Sc. Eng. Atheel Redah
Examiners:	Prof. Dr.-Ing. Sergio Montenegro
	Prof. Dr. Marco Schmidt
Submission Date:	September 2024



Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik 8
Emil-Fischer-Str. 70, 97074 Würzburg

Abstract

In the future, space applications utilising self-assembly will become of great importance as self-assembly enables a disordered system of smaller pre-existing components to form an organised structure. This will reduce launch and development costs, but is difficult due to the requirement of autonomous rendezvous and docking systems in space that are low-cost, reliable and of a small package. The TAMARIW mission goal is to develop these systems. In the TAMARIW mission, two identical 3U cubesats will attempt to dock with one another. The cubesats will use three sensor systems for guidance and docking control. First a long range radio ranging system comprised of two ultrawide band radios per satellite to estimate the relative position of each cubesat. The second sensor system is a the medium range optical system ORPE comprised of multiple LEDs and a camera on the docking surface to estimate the relative position and additionally the relative orientation. Finally, a short range LiDAR based system to estimate the relative angles. This thesis will concentrate on the integration of the ORPE snesor system and the development of a WiFi and UART based Datalink between both satellites. The communication also allows an additional control mode where one cubesat takes control over another during docking. Additionally a sensor fusion scheme to filter the estimations from the long-range radio ranging system and the ORPE system is developed. Several tests were performed to validate the developed system. Finally, conclusions and suggestions for future work were presented.

Contents

Abstract i
List of Figures iv
List of Tables v
1 Introduction 1
2 Platform 2
2.1 Raspberry Pi Zero 2w . . . 2
2.2 Raspberry Pi OS . . . 2
2.3 Raspberry Pi Camera Module V2 . . . 3
2.4 STM32F4 . . . 3
2.5 Software Dependencies . . . 3
3 ORPE 4
3.1 Image processing . . . 5
3.1.1 Reduced camera resolution and FPS . . . 5
3.1.2 Region Of Interest . . . 6
3.1.3 Adaptive Threshold . . . 7
3.1.4 Connected Components . . . 9
3.2 LED Coding and Processing . . . 10
3.2.1 Coding with Hamming codes . . . 10
3.2.2 LED position prediction . . . 11
3.2.3 Quick LED Identification . . . 12
3.3 PNP Improvements . . . 13
3.3.1 Plausibility Checks . . . 13
3.3.2 PNP Refinement . . . 14
3.4 ORPE LED Pattern . . . 15
3.5 ORPE Testing . . . 17
3.5.1 Image Processing Testing . . . 17
3.5.2 Limits Testing . . . 20
3.5.3 Failure Testing . . . 24
3.6 ORPE Conclusion . . . 25
4 Datalink 26
4.1 Architecture . . . 26
4.2 RODOS Datalink . . . 26
4.3 ORPE Datalink Integration . . . 27
4.3.1 UDP IPC . . . 27
4.3.2 Exclusive Router . . . 28
4.3.3 ORPE Topics . . . 28
4.4 Heartbeat . . . 29
4.5 Datalink Testing . . . 29
4.6 Performance . . . 30

4.6.1 Distance Limit . . . 33
4.7 Datalink Conclusion . . . 35
5 Sensor Fusion **36**
5.1 Architecture . . . 36
5.1.1 UWB Position Estimation . . . 37
5.1.2 Inputs and Outputs . . . 37
5.2 Design . . . 38
5.2.1 Start Values . . . 39
5.2.2 Weighted Average . . . 40
5.2.3 Depth and Tangent spaces . . . 40
5.2.4 Fusion Calculation Method . . . 40
5.3 Testing . . . 42
5.3.1 UWB Simulation Data . . . 42
5.3.2 Results . . . 43
5.4 Sensor Fusion Conclusion . . . 45
6 Software implementation **46**
6.1 ORPE . . . 46
6.1.1 ORPE Library . . . 46
6.1.2 ORPE Process . . . 47
6.1.3 Control via STM32 . . . 49
6.1.4 ORPE LED Control . . . 49
6.2 Datalink . . . 50
6.2.1 Translation Layer . . . 50
6.2.2 ORPE . . . 50
6.2.3 WiFi nmcli . . . 51
6.2.4 WiFi AP Negotiation . . . 52
6.3 Sensor Filter . . . 53
7 Conclusion and Future Work **54**

List of Figures

Figure 1: Flowchart of ORPE. 4
Figure 2: Camera Lateral Distance. 5
Figure 3: Example of ROI being used. Left is without ROI, right is with ROI. 6
Figure 4: Comparison of Adaptive and Global thresholding with clouds in background. 7
Figure 5: Comparison of Adaptive and Global thresholding with sun in background. 8
Figure 6: Comparison of Adaptive and Global thresholding at docking distance. 8
Figure 7: Comparison of Contour Finding and Connected components. . . . 9
Figure 8: Previously used LED coding. 10
Figure 9: Diagram of previous LED pattern. 15
Figure 10: Camera FOV issue at close range. 15
Figure 11: Diagram of new LED pattern with corresponding LED IDs. 16
Figure 12: Image of test with sun in background. 18
Figure 13: Example image of the sunlit docking surface taken with a different camera. 19
Figure 14: Diagram showing rotational limits. 22
Figure 15: Rotation limit test 1. 23
Figure 16: Rotation limit test 2. 23
Figure 17: LED Failure test with bright background at 8 meters. 24
Figure 18: Datalink Architecture. 26
Figure 19: Datalink software design. 27
Figure 20: Results of UART datalink testing. 30
Figure 21: Plot of packet interval. 31
Figure 22: Results of UDP 10ms datalink testing. 32
Figure 23: Results of UDP 5ms datalink testing. 33
Figure 24: Results of datalink limit datalink testing at 10ms interval. 34
Figure 25: Results of datalink limit datalink testing at 20ms interval. 35
Figure 26: TAMARIW docking sensor subsystem. 36
Figure 27: UWB radio based position estimation. 37
Figure 28: Varying Data Rate of filter output. 38
Figure 29: Filter settling. Left no start value, right with start values. 39
Figure 30: Simulation of UWB only at 20 meters. 43
Figure 31: Simulation with real ORPE data at 1.4 meters. 44
Figure 32: Simulation with of transition from UWB to ORPE. 45
Figure 33: Flowchart of the simulation runtime. 47
Figure 34: Flowchart of the ORPE process on TAMARIW. 48
Figure 35: State machines of the LED control. 50
Figure 36: State machines of the WiFi negotiation protocol. 52

List of Tables

Table 1: Limit testing results. 21
Table 2: Limit testing results in sunlight. 21
Table 3: LED failure test. 24

1 Introduction

Self-assembly is a process in which a disordered system of pre-existing components forms an organized structure or pattern as a consequence of specific, local interactions among the components themselves, without external direction. An interesting application of this is the formation of large space structures. Self-assembly would allow for reduced component size reducing launch cost and removing the requirement of people to construct the structure, thereby also reducing risk and cost.

For self-assembly of space structures to function, each component must be able to determine and control the relative pose to their corresponding component. For this, special sensor and actuator systems must be developed to reach the strict accuracy and reliability requirements of space applications while also being cheap and small to make self-assembly of space structures cost effective. Furthermore the systems developed in this project can be used to combat the orbital debris problem, in-orbit refueling and servicing or de-orbiting satellites that are out of commission.

TAMARIW "TeilAutonome Montage/Augbau und Rekonfiguration Im Weltraum" is a planned LEO satellite mission made of two 3U-Cubesats to test new technologies for docking in space. The satellites will be launched in a docked configuration. Once in space they will perform multiple undocking and docking operations at varying relative distances. To achieve this mission, a propulsion system of cold-gas thrusters and an Attitude Determination and Control System (ADCS) of reaction-wheels and magnetorquers to control the position and orientation of the satellites will be utilized. In addition, a Guidance and Docking Control (GDC) subsystem consists of three different guidance sensor systems and an active magnetic docking module has been developed for controlling the final docking process.

The three guidance sensor systems are designed for different relative distances. First and for long distances from 10 Meters to 100 Meters an Ultra-Wide-Band Radio ranging based system calculates the relative position of the satellites. For medium ranges from 10 Meters to 30 Centimeters, the optical based system ORPE (Optical Relative Pose Estimation) using a camera and 12 LEDs placed on the docking face, calculates the relative pose of the satellites. Finally and for very short distance range less than 100 Centimeters and up to proximity range of 1 Millimeter, a LiDAR based systems using 4 Time-Of-Flight (TOF) sensors determines relative distance, velocity, and tilt angles of the other satellite for the docking controller.

The proposed solutions of the thesis builds upon previous work of ORPE [10] by implementing new image processing, PNP-Solution and LED Coding/Processing tech-

niques with multiple critical performance improvements enabling real-time performance on performance constrained systems, while also developing and implementing a reliable WiFi based datalink for communication between satellites and finally a fusion algorithm to combine the outputs of the three sensor systems to a single best output.

2 Platform

Each TAMARIW Guidance and Control (GDC) subsystem consists of an STM32F407VG (STM32F4) micro controller and a Raspberry Pi Zero 2 W (RPI Zero 2) as cooperative guidance and docking computers. The STM32F4 runs the Real-Time Onboard Dependable Operating System (RODOS) [8] and controls most of the docking hardware (E.g. electromagnets, TOF sensors, LEDs etc.). The RPI Zero 2 runs Raspberry Pi OS and RODOS on top of it's operating system. The RPI Zero 2 is used as a powerful compute unit capable of image processing, a backup in case of failure of the STM32F4 docking controller and as a wireless data-link utilising it's built in WiFi. The Raspberry Pi is connected to the microcontroller over UART and SWD. The UART connection is used to share information between the microcontroller and the Raspberry Pi, and the SWD pins are used to program the microcontroller wirelessly. The STM32F4 microcontroller communicates with satellite main On-Board Computer (OBC) via I2C bus.

2.1 Raspberry Pi Zero 2w

The RPI Zero 2 is a small low-cost single-board computer (SBC) based off the 1GHz quad-core, 64-bit ARM Cortex-A53 CPU and has 512MB of RAM. The operating system is stored on a micro-SD card. As ORPE uses a camera as a basis for the optical system, ORPE requires a powerful CPU and plenty of RAM capable of storing and processing the images from the camera in real-time, hence the reason the RPI Zero 2 is integrated into the TAMARIW Satellites and where ORPE runs.

2.2 Raspberry Pi OS

The operating system chosen for the RPI Zero 2 is Raspberry Pi OS, the reasoning being both the RPI Zero 2 and Raspberry Pi OS were specifically designed for one another. Although other operating systems can run on the RPI Zero 2, Raspberry Pi OS is the most widely used OS and therefore is well tested. Specifically Raspberry Pi OS 64 bit lite version is used to utilize the full 64 bit processing of the RM Cortex-A53 CPU and lite due to its minimal amount of background processes running to increase performance, reliability and reduce RAM and power usage. Access to the RPI Zero 2

is provided through networking and SSH.

2.3 Raspberry Pi Camera Module V2

The Raspberry Pi Camera Module V2 was chosen for its great support, documentation, wide usage and meeting the performance requirements from ORPE. It has a maximum resolution of 3280 x 2464 for images and capable of up to 60 Frames per second at 720p resolution. The lens system is fixed focus allowing for the camera parameters to stay constant and a FOV of roughly 62 Degrees, making it ideal for the medium range application of ORPE. Although due to cropping of the image and changing of resolution, the resulting FOV calculated by the camera matrix is of maximum 40.8 Degrees.

2.4 STM32F4

The STM32F4 is a 168MHz single-core 32bit ARM Cortex-M4 based microcontroller. It offers the relevant peripherals (GPIO, UART, SPI, I2C etc.) required by the TAMARIW satellites for communication with the main OBC and the Raspberry Pi single board computer as well as the sensor and module of the GDC subsystem.

2.5 Software Dependencies

Both the RPI Zero 2 and STM32f4 use RODOS for managing applications and communication between applications and as a data-link protocol over the UART serial and UDP WiFi connections. ORPE on the secondary GDC computer makes use of OpenCV for image processing and LCCV for Camera control and binding with OpenCV.

3 ORPE

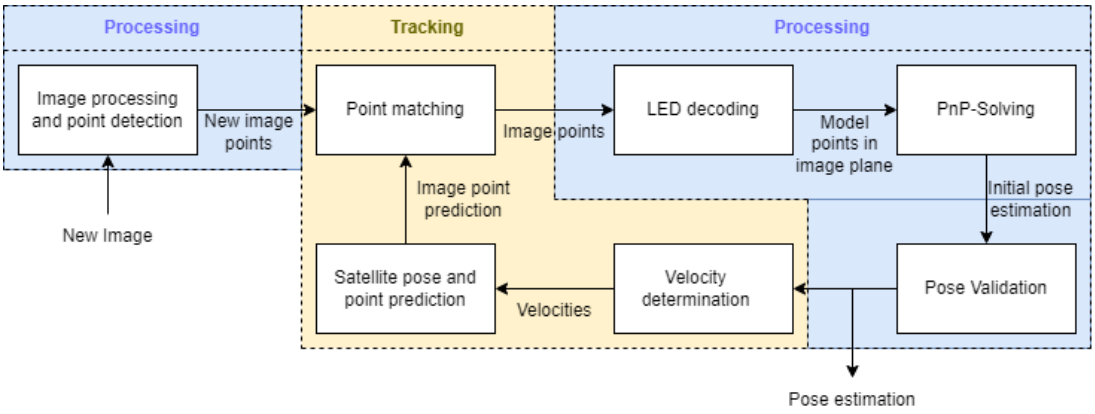


Figure 1: Flowchart of ORPE.

ORPE (Optical Relative Pose Estimation) is an optically based relative pose estimation system [10]. ORPE utilises a camera on one satellite and 12 LEDs spread around the docking surface of another satellite, to estimate the relative pose (Rotation and position) between the satellites with centimeter accuracy and low noise from distances of 15 centimeters up to 10 meters.

In fig. 1 we can see an overview on how ORPE functions. Initially image processing is used to identify bright sources of light. Each LED blinks a specialized binary code with redundancy allowing for ORPE to distinguish each LED from any unwanted external light sources (Sun, moon, earth etc.), this is called LED encoding. The LEDs are identified by the image processing and the binary code is used to individually identify each LED, which ORPE uses to match each LED to a point in a model of the LED positions to solve a Perspective-N-Point problem and obtain the relative pose. Further validation is done to check the plausibility of the pose estimation being correct. Lastly ORPE processes the information in the LED and satellites pose and velocities to track their positions to improve the next images reliability.

For TAMARIW, ORPE's reliability must be greatly improved on and performance must also be greatly improved to run in real-time on the RPI Zero 2. For optimisation, the camera frame rate and resolution were lowered to reduce the CPU requirements for the RPI Zero 2, as well as implementing a Region-Of-Interest (ROI) when satellite tracking begins. The image processing was also greatly improved on to reduce image noise causing a lower amount of points to be tracked and also improve the detection of points in multiple different lighting situations. Changes were also made to the LED coding system to decrease the chance of false identification, while also implementing Error-Code-Correction (ECC) to help improve LED detection. An algorithm was also implemented to quickly identify all LEDs once satellite tracking begins to help improve

accuracy and reliability. Lastly, PNP-Pose-Refinement was added to the algorithm as a fallback, to improve reliability. ORPE's implementation for TAMARIW was also tested thoroughly to find minimum and maximum limits in multiple environments and characterize its performance.

3.1 Image processing

Due to the image processing consuming the largest amount of processing time, optimisations in this area are of upmost priority. The image processing changes can be split into four separate parts, each improving on different aspects of ORPE. The following section will go over each change in detail.

3.1.1 Reduced camera resolution and FPS

The initial change was to reduce the performance requirements to enable real-time performance on the RPI Zero 2. ORPE was developed on a laptop computer with an I5 8250U CPU which is a powerful CPU capable of the previous implementation of 1080p at 30 FPS. For TAMARIW, ORPE's camera frame-rate was reduce to 10 FPS. This has an effect on the maximum dynamic movements that ORPE can track and the Time-To-Lock from the first LED blinking and then getting the first pose estimation. As the TAMARIW satellites will be moving slowly (<1°/s and <1cm/s), allowing for the maximum dynamic movement to be really low and Time-To-Lock can be very long. The camera resolution was also reduced to 1280x720 at the cost of long range accuracy. We can determine the theoretical spacial resolution at the maximum distance of 15 Meters. We can create a right triangle in the following diagram:

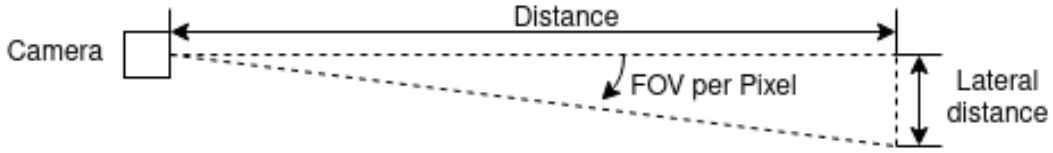


Figure 2: Camera Lateral Distance.

Using simple trigonometry we can characterize the lateral distance via the following formula

$$X_{lateral} = \tan\left(\frac{FOV}{N_{Pixel}}\right) X \tag{1}$$

With $X_{lateral}$ being the lateral distance, X being the distance, FOV the field and view of the camera and N_{Pixel} being the number of pixels make of this FOV of the camera

in fig. 2. We can now calculate and compare the pixel accuracy between 1080p and 720p resolutions for a FOV of 40.8°. With 1080P we get a lateral resolution of 5.56 millimeters and at 720P 8.34 millimeters. It is obvious that the spacial resolution is reduced, but due the 2.25 times amount of pixels required to be processed per image when using 1080p, this is a trade off that frees up performance that is made use of in image processing techniques shown later. It is also important to note that ORPE can calculate the LED position in the image at sub pixel precision using the algorithms in section 3.1.4.

3.1.2 Region Of Interest

Another approach to reduce the number of pixels to be processed, is to ignore areas of the image that are not likely to contain the LEDs from the target satellite. A simple technique is Region-Of-Interest (ROI). If the position of the LEDs is already known, the camera image can be initially cropped down to roughly where the LEDs are assumed to be. This is accomplished by using previous predictions of the satellite position, predicting where it should be and then cropping the image down to this prediction, with a small border region for errors in the prediction. This also has the benefit of reducing external light interference's and reducing therefore the number of points for processing. We can see ROI's effect in fig. 3, were in the left image many points can be seen in the background, but after tracking begins and ROI is used, only the LEDs that are inside the region are detected.

Figure 3: Example of ROI being used. Left is without ROI, right is with ROI.

It is important to note that although this can greatly increase performance, there are a few cases where ROI has little to no effect on performance. The most obvious case is during the initial searching phase where ORPE has no information of the target satellites position and must therefore process the entire image. The second case is when the satellite is close enough to fill the image, therefore ROI will not crop any part of the image. Therefore ROI will not allow the camera FPS or resolution to be increased.

3.1.3 Adaptive Threshold

ORPE uses image thresholding to remove noise and other unwanted low brightness light sources from the image. Specifically the algorithm used Otsu's Binarization [2] to determine the best threshold value to separate bright from dark values, as the LED brightness inside the image changes with distance. This technique worked optimally in the case of dark environments with a clear difference between background and foreground, but unfortunately the TAMARIW mission will be in Low-Earth-Orbit resulting the background of the image to often contain views of earth. As Earth's clouds or freshly fallen snow can have a high albedo (>0.9), the reflected sunlight during the day would absolutely make it impossible to determine a threshold to separate the unwanted light from the image without also removing the LEDs at larger distances from the target satellite. This issue can be solved by choosing a threshold depending on the images lighting condition for a given area at a given time. The solution used to improve ORPE is adaptive thresholding, which calculates a threshold value using the neighboring pixels. One can see this also as an edge filter that will detect large differences in lighting. As the direct vicinity of the LEDs will be the satellite, which is dark, the LEDs edges will create a sudden change in lighting, which is detected by adaptive thresholding, as the neighboring pixels to the LEDs are dark. This sudden change in lighting is more prominent to be detectable, regardless of LED brightness and therefore distance.

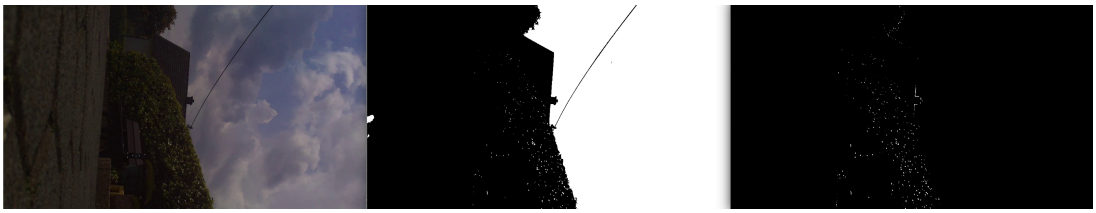


Figure 4: Comparison of Adaptive and Global thresholding with clouds in background.

In fig. 4 we can see an input image with a model of the LEDs from the TAMARIW Satellite and clouds in the background. The first image is the image from the camera, the second image after thresholding using Otsu's Binarization to determine the global threshold value and last the adaptive thresholding. We can see there is not much difference other than the removal of the sky. This is is usual environment, but often the environment can change substantially.

Figure 5: Comparison of Adaptive and Global thresholding with sun in background.

In fig. 5 we can see a worse case scenario, but very possible to occur during the docking process. In this environment, the sun is shining directly from behind the target satellite completely saturating the camera sensor. Even the optics have glare bright enough to go over the global threshold value, causing the opposing satellites LEDs to be indistinguishable. The cause of this is due to the gradient making it difficult to find an optimal threshold value, without knowing the LED brightness. But as seen in the adaptive threshold image, many LEDs can still be seen.

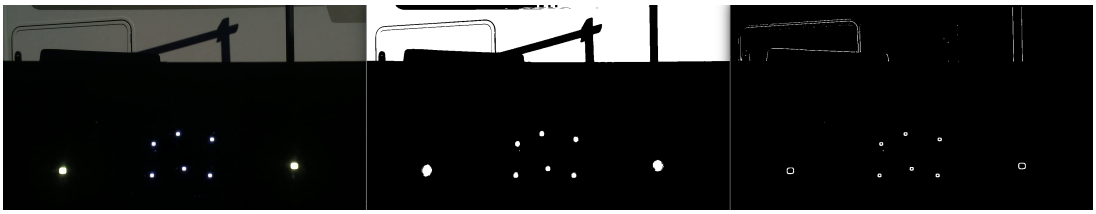


Figure 6: Comparison of Adaptive and Global thresholding at docking distance.

One issue that can be seen in fig. 6, is at close distances, due to the LEDs being large in the image, in the center of the image, neighboring pixels are also bright. This results in the center of the LEDs at close distances to be removed by the adaptive thresholding. This issue is easily mitigated, as the resulting circle's center can still be accurately calculated and used for the points position. Furthermore, another issue is edges with a sharp change in brightness that was previously not detected by global thresholding, is now detected by adaptive thresholding. These are usually very small (Noise, Stars) or large (Sun, Earth, Moon). These unwanted pixels will mostly be filtered out by the LED binary identification codes, but can also be alleviated by the improvements in the following section 3.1.4.

3.1.4 Connected Components

The final improvement to the image processing was the method used to find points in the binary image. Previously ORPE used contour finding [1] to find the contours of the bright spots and then used the resulting contours to calculate the center. The issue with this solution was many objects have holes or are generally rough, causing for only one shape to have multiple contours and therefore points, when we are only interested in the bright object as a whole. Furthermore, calculating the center of a contour sometimes fails for small points, failing to detect an LED at large distances. Lastly, contour finding can sometimes generate one contour for two very close objects. In general, contour finding is not a very deterministic method of detecting the points in the binary image. Connected components [6] is a function in OpenCV that extracts regions from a binary image that are not connected to other regions. The center and area of these regions is also calculated. Using the center of the shape also allows for sub pixel precision. The region area can be used to further filter out any points that are too small like noise and stars, while points that are too large like earth can also be removed, the center of each region is then used as the points position.

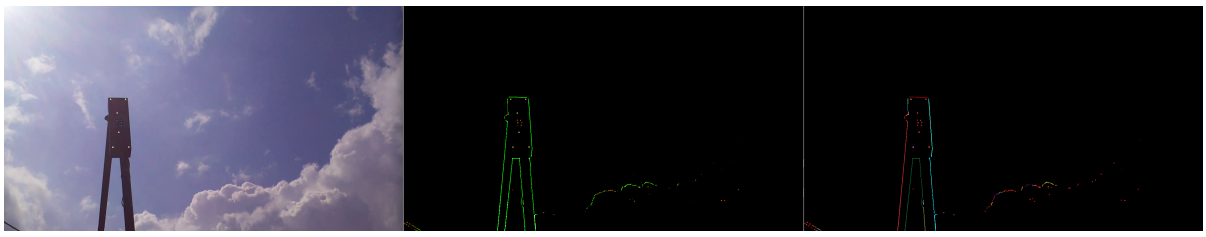


Figure 7: Comparison of Contour Finding and Connected components.

In fig. 7 we can see the input image on the left, the contours in green from contour finding in the middle, the shapes from Connected components on the right and both image have the resulting point positions marked as red circles. Both methods are able to detect all the LEDs, but the contour method fails to calculate the position of one of the LEDs on the center. This new approach results is much more deterministic in detecting points in the image and calculating their center results in less noise and higher accuracy. This also improves reliability, as the point tracking and matching systems get better information, which is critical for the next section on LED coding 3.2.

3.2 LED Coding and Processing

The LEDs used by ORPE to calculate the target satellites pose, must be reliably detected and matched to the Model describing their 3D positions relative to the target satellites body frame. For this the LEDs blink binary codes with redundancy to encode their ID's, which later is used to match them to the model and also distinguish LEDs from other bright external light sources like earth, sun and moon.

3.2.1 Coding with Hamming codes

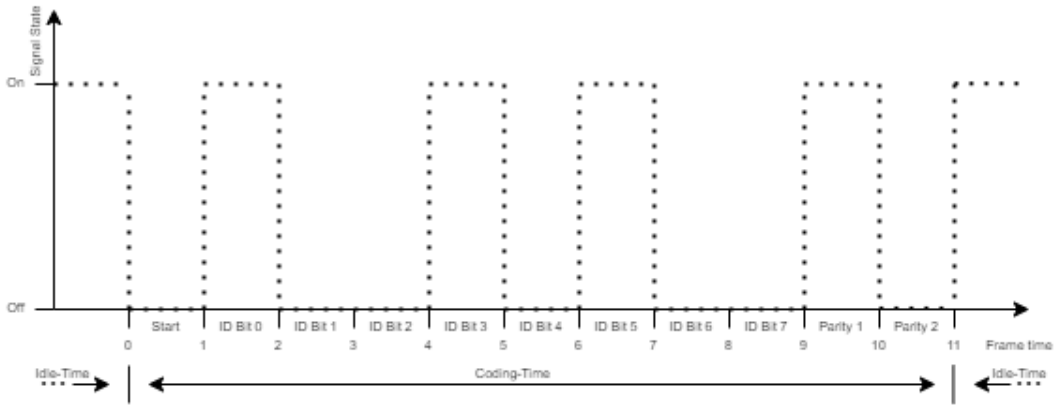


Figure 8: Previously used LED coding.

Previously the binary code used by ORPE which can be seen in fig. 8, comprised of a start bit signifying the start of the code, then the 8-Bit binary code containing the LED's ID and lastly two parity bits for redundancy. These parity bits where primarily added for filtering out any synchronisation issues between the camera frame rate and LED encoding, causing errors and worked great for environments with little noise in the image processing. For situations with the earth with clouds in the background, an order of magnitude more points could be seen, greatly increasing the chance of a random point being identified as a LED and used in the pose calculation, resulting in a failure. Also due to the reduced frame rate of the camera, the LED identification process takes substantially longer, therefore simply adding further parity bits would increase the time to lock, which is undesirable.
Moving forward, the coding for ORPE was redesigned to feature a faster and more reliable coding technique using hamming codes. The previous method using eight bits allowed for up to 256 individual LEDs to be used, but as the TAMARIW satellites will use only 12 LEDs on the docking surface, theoretically only four bits are required, allowing up to 16 LEDs to be used. This fits perfectly into a hamming coding method called Hamming(7,4), which uses seven bits, of which 4 are data and 3 are redundancy bits. Hamming(7,4) adds the ability to correct the code for single bit errors, allowing

failures in the coding to possibly be corrected, speeding up identification. A fourth parity bit is added to the code to also add the ability to detect a second bit failure, further improving reliability. This removes the need for the two previous parity bits, resulting in a reduction of two bits in the total code length, actually improving the time to identify LEDs. Unfortunately this is not enough to completely filter out the chance of a point being falsely identified. As noise will produce a constant stream of random binary values, over time a point will eventually be falsely identified. Although it is unlikely that noise will produce the specific binary code, its is very unlikely that the noise will do it repeatedly, therefore the new method requires LEDs to identify themselves twice with the same ID in a row. The new coding method almost completely removes the possibility for random points to incorrectly identify themselves as LEDs.

3.2.2 LED position prediction

In order for ORPE to decode LEDs, it must be able to track each point over time. For this, ORPE will match each point found in the image to the points currently being tracked. This point matching algorithm also takes care of creating new points and also removing old ones. Previously, ORPE would predict the tracked point positions in the image from past frames using a simple linear movement prediction. While this does improve the reliability when the target satellite is moving, it also causes points created by noise to get large estimated velocities and once the point does not show up anymore and has no new matches, the points position will continuously be predicted for each consecutive frame if the last estimated velocity is large. This results in noise creating many points that "fly" through the image. If one of these points get close to a correctly identified point, this identified point could possibly take the false points ID and resulting the correctly identified point loosing its ID. This results in situations where a noisy background directly behind the satellite to constantly remove correctly identified LEDs. The solution to this implemented for TAMARIW's ORPE was to simply not predict the position of points that have not been correctly identified yet. This has little to no effect on the TAMARIW satellites LED tracking, as the assumed rotational and translational velocities are low and within the position noise floor due to errors in the point position estimation.

3.2.3 Quick LED Identification

Finally, the last improvement to ORPE's LED processing was to increase the time it takes to identify all LEDs. ORPE only requires at least four points to estimate the relative pose. This is unreliable, as at large distances, noise in the LED positions can result in the point being identified as an outlier by the ransac algorithm in OpenCV's PNP-Solver and not making it possible to estimate the pose with the remaining three points. This also gives no room for error if a point in lost during tracking or an LED were to fail. Therefore, the more LEDs that are identified at a given time, the more reliable the pose estimation will be. Although not much can be done to decrease the time until the minimum of four LEDs are identified, we can use the pose estimation to explicitly search for the LEDs and their matching IDs inside the image within a single frame. Therefore greatly increasing tracking reliability as each frame will have the maximum possible amount of points identified and also even if a point is lost, it will be immediately be re-identified once visible in the image, without the need for the lengthy encoding. This quick LED identification is done in multiple steps and as mentioned, requires that the satellites relative pose in known (Usually the first valid pose estimation is immediately used). The initial step is to use the relative pose and camera parameters to project the model of the LEDs onto the image plane. With this it is known where the LEDs should be located in the image. Next, due to noise and inaccuracies in the entire process until now, each projected model point, probably will not directly match with any points in the image. Therefore the most probable matching point must be found. For this, first the deviation of the currently identified points in the image to their corresponding projected model point is calculated. With this it is known what a realistic match should look like and this information is used in the final matching step. Lastly, for each unidentified point in the image, the algorithm finds the closest projected LED model point that is within the deviation and use this projected LED models ID for to identify the unidentified point. If no projected model LED point is found within the deviation, we consider this point to not be an LED and will not be changed. It is important that the relative pose estimation is correct, as an incorrect pose estimation, will cause the projection of the LED model points to be incorrect and the possibly incorrectly identified points will make the next frame fail when determining image and model point correspondences or estimating the pose. Therefore quick LED identification algorithm is only used if the current pose estimation is considered valid, which is elaborated on in section 3.3.1.

3.3 PNP Improvements

ORPE's calculation of the relative pose is based off of the Perspective-N-Point (PNP) problem [5]. The PNP problem has to do with the estimation of a camera's relative pose using 3D set of points in space and their corresponding 2D set on points that are projected onto the cameras image plane. There are many solutions to this problem. ORPE previously had the issue that the estimation algorithms often generated incorrect pose estimations or none for a single frame. In this work, we incorporate further plausibility checks in section 3.3.1 for the estimations and also PNP-Refinement in section 3.3.2.

3.3.1 Plausibility Checks

ORPE does plausibility and validity checks for the pose estimations. Plausibility checks determine of the pose estimation generated by the PNP-Solver could in general be correct by checking for impossible values. Where as validity checks determine if the pose estimation over time is probable and can be considered to be reliable. ORPE does multiple plausibility checks, by evaluating the pose estimation values if a value in a NAN or the total rotation is larger than 360°, signaling that there is a calculation error in the PNP-Solver. ORPE checks the plausibility further by checking if the pose estimation is close to the prediction. Although these checks are enough to check if the pose estimation is incorrect due to a failure in the PNP-Solver, further checks were added to account to detect poses not possible due to technical limits of ORPE. The first check added, was to determine if the estimated position is within the cameras FOV. The Raspberry Pi Camera Module V2 2.3 has a maximal FOV of 62°, although due to cropping of the image to image ratios, resolution and mode, the resulting FOV calculated by the camera matrix is of maximum 40.8°. Therefore we calculate the angle between the Z-Axis and the direction of the satellites pose estimation. If this angle is larger than 50°, then we consider this pose estimation not plausible. The final check is if the distance of the satellite is within known bounds. The limits of ORPE with for TAMARIW is between 15 Centimeters and 15 Meters. If the pose estimation is outside this range, we again consider this estimation incorrect.

3.3.2 PNP Refinement

One issue that comes up when filtering out incorrect pose estimations, is of course the loss of an estimation for the given frame. Currently ORPE looses a large percent of estimations for the given frames due to incorrect or no solutions. Another issue is we require at least four LEDs to be visible for the given frame, meaning if only four LEDs are visible and one is blinks, resulting in only three visible LEDs for the frame, then a solution is not possible for this frame. Even is more LEDs are visible, then this could still be an issue if multiple LEDs blink at once.
The solution to this problem is, in the case if a solution error as a result of not enough LEDs visible or plausibility checking detecting an incorrect solution, then to attempt to use the pose prediction and refine it using PNP-Refinement [5]. PNP-Refinement takes a given pose and attempts to refine/estimate the pose so that the re-projection error of the estimated pose and current LED positions in the image is reduced to a minimum. This requires a minimum of three LEDs and a start pose, therefore can only be used if the previous pose is known.
One issue using this method is that if the pose estimation is incorrect, the next image frame is likely to also be incorrect as PNP-Refinement searches for a local minimum in the re-projection error. This is due to an ambiguity issue that results from the solutions. With three points, there are up to four solutions. PNP-Refinement will solve for the nearest solution, which is usually correct if the past solution was done by a PNP-Solver, although over time in special cases where all solutions converge to a single solution and then diverge, PNP-Refinement will choose an arbitrary solution and follow it as a local minimum of the re-projection error. A real world example of this is if the satellite at long distances points towards the camera and then rotates away again. In this case the solutions for a rotation or its mirrored opposite are extremely similar.

3.4 ORPE LED Pattern

ORPE's previous design used eight high power SMD LEDs spread around on the docking surface. This LED pattern was functional for distances from 1 meter to 10 meters.

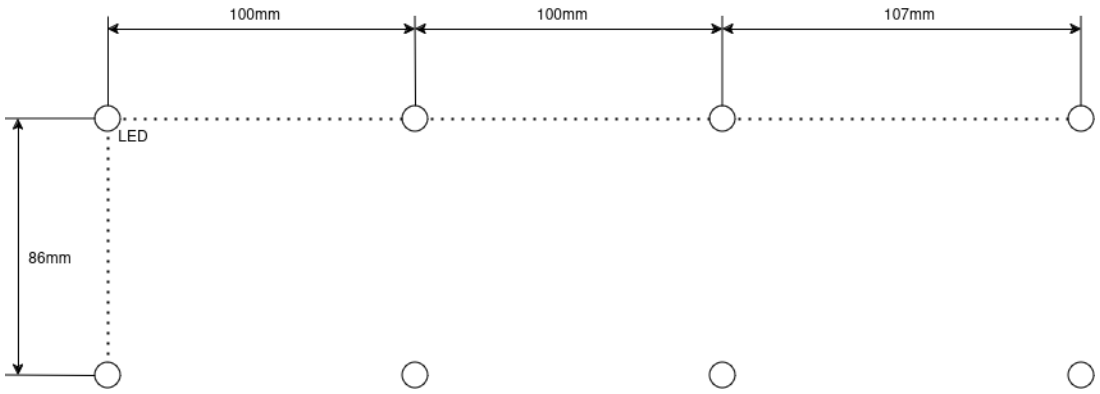


Figure 9: Diagram of previous LED pattern.

This arrangement creates an issue at close ranges which is illustrated in fig. 10. Due to the camera FOV, the outer LEDs could not be seen resulting in only four LEDs being visible at very close ranges. In this case, we have no redundancy for LED failures and the estimations become unreliable.

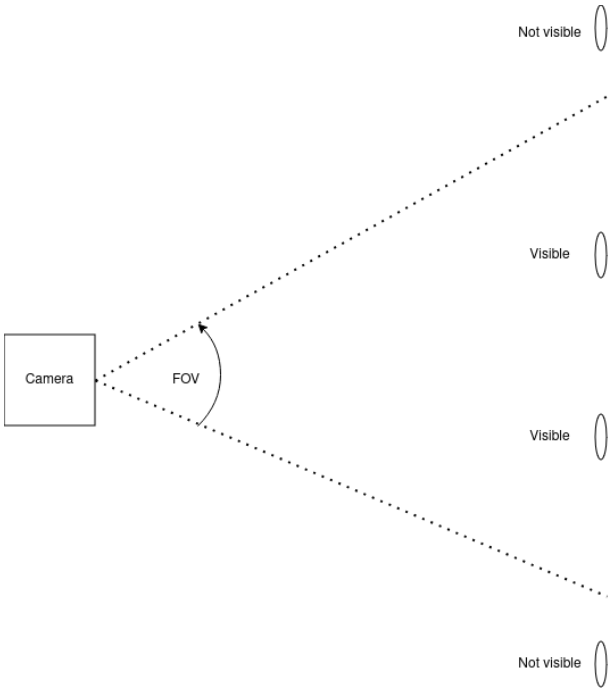


Figure 10: Camera FOV issue at close range.

We can calculate the minimum range at which even the inner LEDs would not be visible. By using same principle in 2, we can calculate the camera distance from the docking surface. This results in the formula:

$$X = 2\frac{X_{lateral}}{\tan\left(\frac{FOV}{2}\right)} \tag{2}$$

With X being the distance of the camera from the docking surface, $X_{lateral}$ being the distance between LED at 100 millimeters and FOV of the camera with 40 degrees. This results in a minimum distance of 549 millimeters. This is the absolute minimum physical distance that ORPE would work at. Therefore the LED pattern needed to be changed. We can calculate the required maximum LED distance using the previous formula for the minimum operating distance of 15 centimeters. This results in a maximum LED distance of 5.46 centimeter from the center of the docking port. Moving the LEDs this close to the docking center creates issues at long range, as it will be harder for the PNP-Solver in ORPE to effectively estimate the orientation and distance of the target satellite. We also have to contend with a second issue at close distances. Due to the LED brightness and package size at these distances, ORPE will have difficulties correctly estimating the LED's center in the image. The use of smaller LEDs with less brightness is not possible, as these will not be visible at long ranges and harder to distinguish from bright light sources. Lastly this pattern is not very redundant, as the multiple symmetries makes it difficult for PNP-Solvers to estimate the pose.

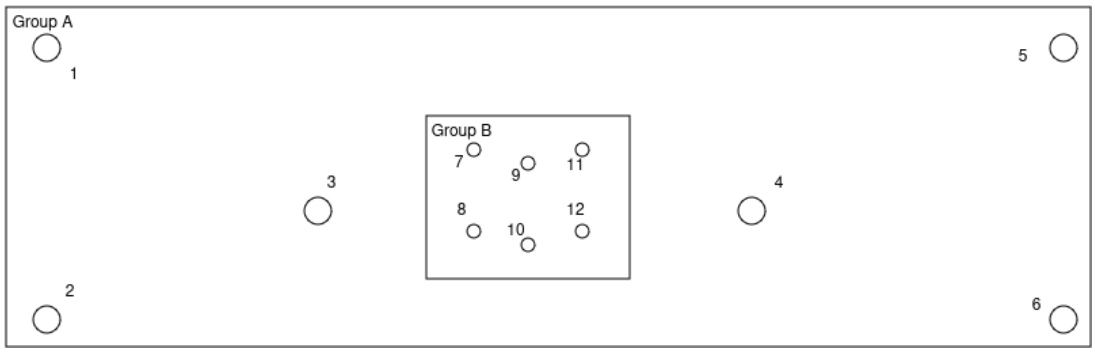


Figure 11: Diagram of new LED pattern with corresponding LED IDs.

In the above figure we can see the new LED design for the TAMARIW satellites. The biggest difference being in the idea of using two groups of LEDs. The large group A uses the same high power SMD LEDs and is for long ranges where the LEDs must be bright, large and have the maximum possible distance from each other as possible.

Also two further LEDs were added to this group to increase redundancy in case an LED fails and also to improve reliability. The inner group B uses much smaller, closer

grouped together LEDs and also has two further LEDs added for the same reason of better redundancy and reliability. This arrangement also has the LEDs placed with less symmetries, further improving reliability. This change does not require any changes to ORPE other than updating the LED model. ORPE automatically chooses the LEDs as needed.

3.5 ORPE Testing

ORPE's performance was tested to characterize the absolute limits under different environments. As the TAMARIW mission will be in LEO, the most interesting environments are with a large and bright background made of clouds, as this will be a difficult situation to differentiate the LEDs from external light sources.
Two main groups of tests were made to observe different aspects of ORPE: image processing and workspace performance. The Image processing tests are used to analyse the image processing performance. The criteria for this this test is to make sure the image processing can reliably detect the LEDs in the image, regardless of the environments external light sources. The second set of tests are to analyse ORPE in it's entirety and test the limits in position, orientation and performance. Together these two groups of tests give a picture on how well ORPE performs in any lighting condition and at any relative pose of the target.
Finally a third test was made to analyse ORPE's performance in the case where two LEDs fail.
Due to the slow movement of the TAMARIW mission, dynamic testing was not performed as the satellite can be considered stationary as the measurement noise is larger than the movement.

3.5.1 Image Processing Testing

Testing the Image processing allows us to see how well ORPE is able to detect the LEDs in the image in different environments. These tests were performed by setting ORPE to record the camera images and save them to the SD-Card, this allows the results of the image processing to be displayed. This process unfortunately requires the recorded video to be compressed as the SD-Card is write performance is unable to write at the speeds required for raw uncompressed video. The compression results in a reduction of image sharpness and colour fidelity. This actually makes if harder for ORPE to distinguish LED and the real in situ performance will be better than these tests.

The most difficult environment is when the Sun is directly behind the opposing satellite. In this case the sun could possibly saturate the camera and wash-out the LEDs.

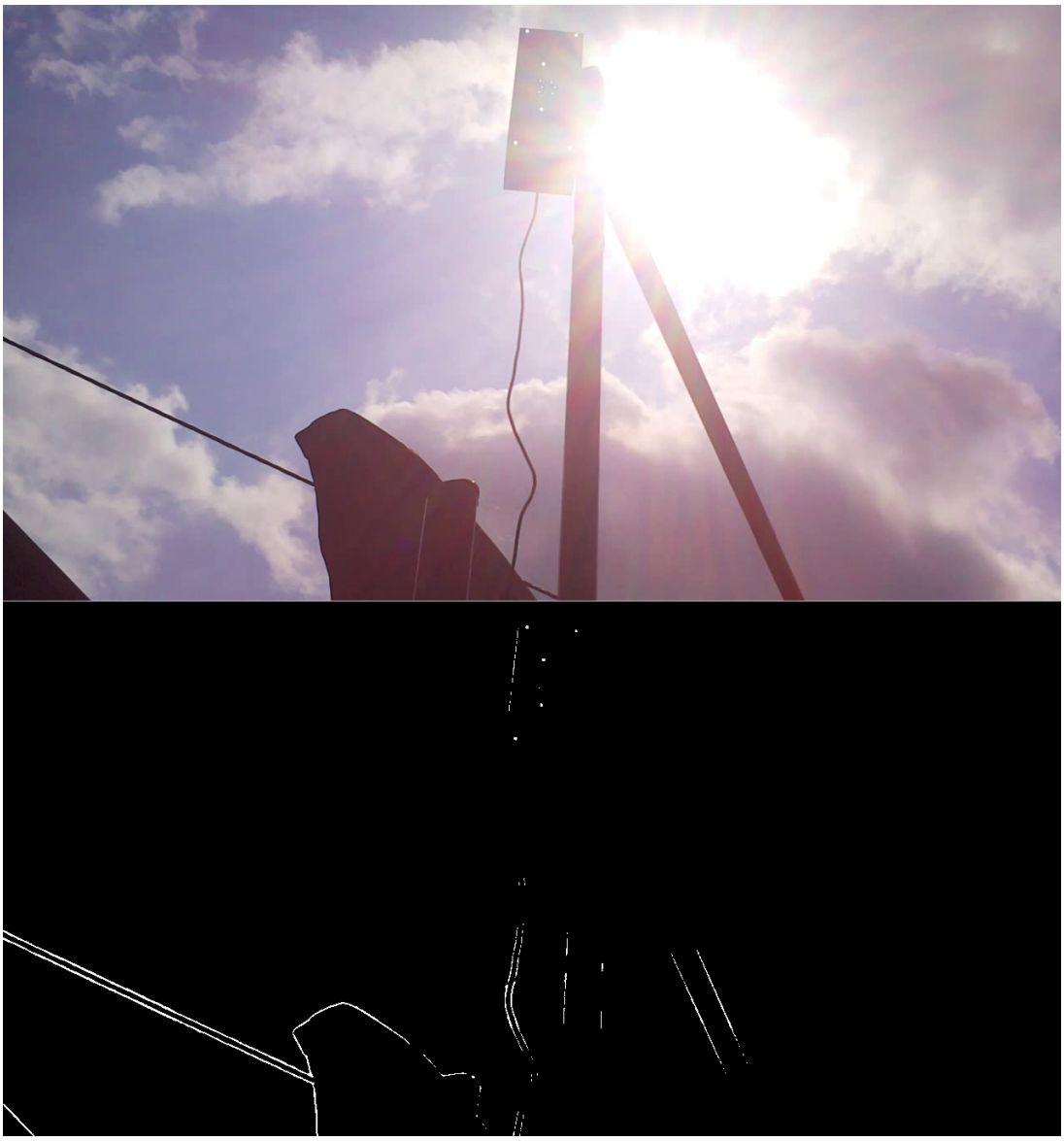


Figure 12: Image of test with sun in background.

Here we can clearly see the sun almost washing out the bottom right LED and making it difficult to detect the LEDs. In the image processing step we see that the improvements made in 3.1.1 have made ORPE capable of detecting five out of the six main external LEDs in extremely bad conditions. It is important to note that due to the previously mentioned issue with compression, the sixth LED would possibly be detected if ORPE was ran on the RPI Zero 2 and using the raw camera images.

In the same environment, the opposing satellite will also have difficulties as the satellites docking surface will be strongly illuminated by the sun. For this test the mock-up was place directly in sunlight and the camera pointed towards the sunlit satellites docking surface.

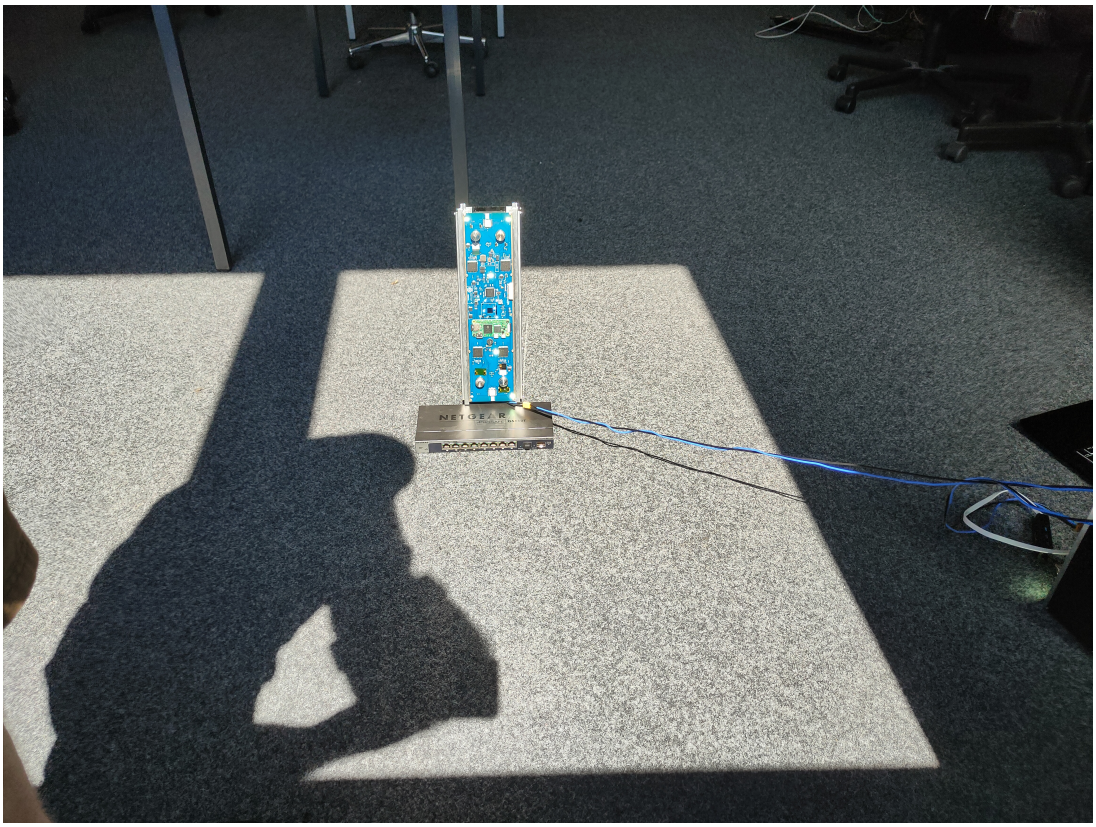


Figure 13: Example image of the sunlit docking surface taken with a different camera.

This test also was difficult for ORPE as there were many highly reflective object on the docking surface. Here ORPE was ran in situ on the RPI Zero 2 for best performance. Here interestingly, only the center six docking LEDs were detected, probably due to their higher contrast to the surroundings because of their smaller package size. This is no issue at closer ranges and at longer ranges the reflections should be much dimmer than the LEDs. From this test, it is highly recommended that the TAMARIW satellites use an anti-reflective coating to keep reflections from being visible.

3.5.2 Limits Testing

The second tests were made to determine ORPE's resulting operational limits in distance and orientation. For these tests the satellite was placed at increasing distances until ORPE was unable to estimate the position. The satellite was also placed near the recommended operation limits with a bright background to confirm the operational limits in a worse-case scenario. Finally a test was performed were the satellite was slowly rotated to find the maximum rotation limit. For all tests, ORPE's data was logged and later analysed to determine the noise characteristics and reliability.

The initial test was to determine the maximum distance at which ORPE could estimate the opposite satellites relative pose. As previously mentioned, the satellite was placed at varying distances. To compare the results, the standard error will be calculated using the following formulas:

$$\overline{x} = \frac{\sum_{i=0}^{N} x_i}{N} \tag{3}$$

$$S_{\overline{x}} = \left[\frac{1}{N-1} \sum_{i=0}^{N} (x_i - \overline{x})^2 \right]^{\frac{1}{2}} \tag{4}$$

$$\Delta\overline{x} = \frac{S_x}{\sqrt{N}} \tag{5}$$

We will only be comparing the error, because this contains the information that defines the reliability at a given distance. The position accuracy from the average is not of interest as in the previous work it was shown that the accuracy is a result of the camera calibration accuracy.
Once ORPE reaches its limit, the estimation will become noisy and the percentage of valid frames will greatly decrease. We will also be looking into the TTL (Time-To-Lock) to compare the length of time before ORPE will begin producing pose estimations. For these tests, the satellite mock-up will already be powered on, so once ORPE it started, the LED are already encoding their IDs.

Distance	ΔP_X [mm]	ΔP_Y [mm]	ΔP_Z [mm]	ΔR_x [deg]	ΔR_y [deg]	ΔR_z [deg]	Valid [%] (TTL [s])	Notes
11cm	0.01	0.01	0.12	0.6	0.4	0.04	37.0 (12.8)	Incorrect position
15cm	0.002	0.001	0.016	0.03	0.03	0.004	89.7 (12.3)	Lower limit
140cm	0.04	0.02	0.31	0.05	0.06	0.01	89.9 (12.0)	
1000cm	0.06	0.26	1.17	0.48	0.13	0.02	80.2 (23.7)	
1400cm	0.08	0.13	1.97	0.43	0.14	0.01	89.2 (13.0)	Upper limit
2800cm	0.47	3.11	60.68	0.40	0.35	0.06	9.1 (109.2)	

Table 1: Limit testing results.

In table 1, we can see the results from the distance testing. We can see the initial test at just 11 centimeters distance is not reliable due to its low amount of produced estimations and when comparing the rotation estimation to other tests, the standard errors are much larger signifying the the rotation estimation at this distance is very unreliable. Once moving the satellite to 15 centimeters distance and restarting the test, we get the first reliable estimations. Here the number of valid estimations percentage was very high. Also the position and rotation estimations show very little noise. Therefore we can determine that ORPE's lower operational limit for the distance is 15 centimeters. Each subsequent test up to 28 Meters worked similarly to the 15 centimeter test. Therefore we can also determine that the upper operational limit of ORPE is at 15 meters. Although it is important to note that these tests were done in darkness. A brighter environment only has the effect that the LEDs could be more difficult to distinguish from background. Therefore further tests were made to verify the upper operational limit of 10 Meters. The tests were repeated but the satellite placed outside in the sunlight and a bright background with clouds.

Distance	ΔP_X [mm]	ΔP_Y [mm]	ΔP_Z [mm]	ΔR_x [deg]	ΔR_y [deg]	ΔR_z [deg]	Valid [%] (TTL [s])	Notes
140cm	0.02	0.03	0.29	0.14	0.43	0.02	82.0 (11.9)	Sunlit Surface
600cm	0.08	0.09	2.97	0.55	0.17	0.03	90.7 (11.1)	
860cm	0.66	0.28	6.56	0.44	0.13	0.04	88.2 (12.7)	

Table 2: Limit testing results in sunlight.

From the above table we can conclude that ORPE's operational limits in any environment is from 15 centimeters up to 10 meters. With a possible upper limit of 15 meters in ideal environmental conditions. We can also see that the Z-Axis rotation (The camera was pointed towards the satellite) has an order of magnitude lower noise than the X, or Y-Axis'.

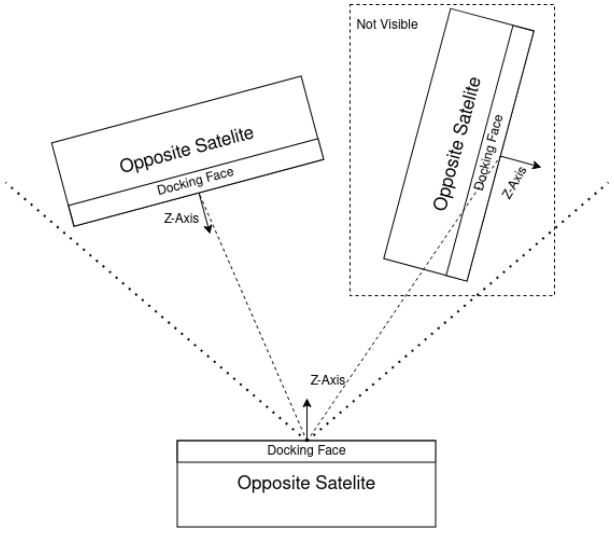


Figure 14: Diagram showing rotational limits.

The rotational limit is mainly a limit determined by the camera accuracy and the PNP-Solvers ability to find a pose estimation. Clearly, if the docking port surface normal is pointed further from the camera than 90 degrees, then LEDs will not be visible at all. This is illustrated in fig. 14. But the rotation around Z-Axis has no limit. To test the rotational limit around the other axis', the satellite was placed at the upper operational limit of 10 meters and then slowly rotated from one side to the other. This will give ORPE time to identify the LEDs even during movement and once the lock is lost, at this angle ORPE is unable to produce a pose estimation and is therefore the rotational limit.

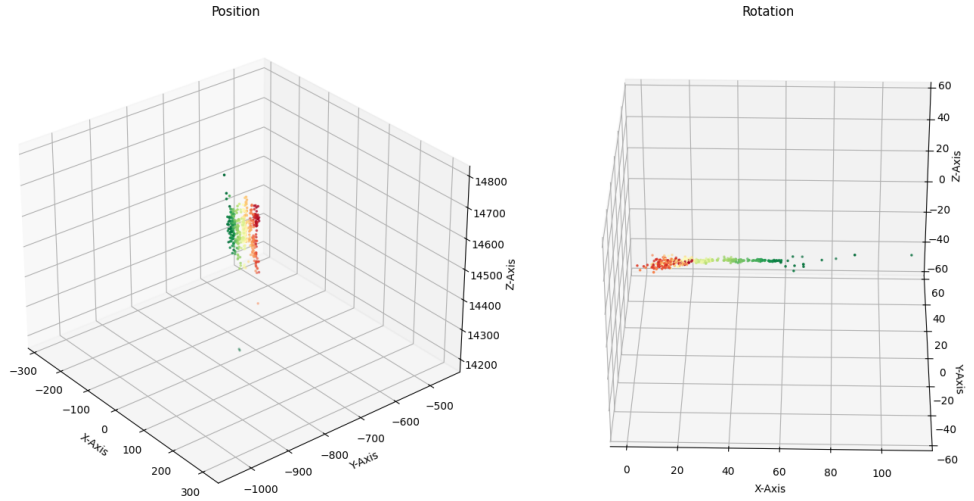


Figure 15: Rotation limit test 1.

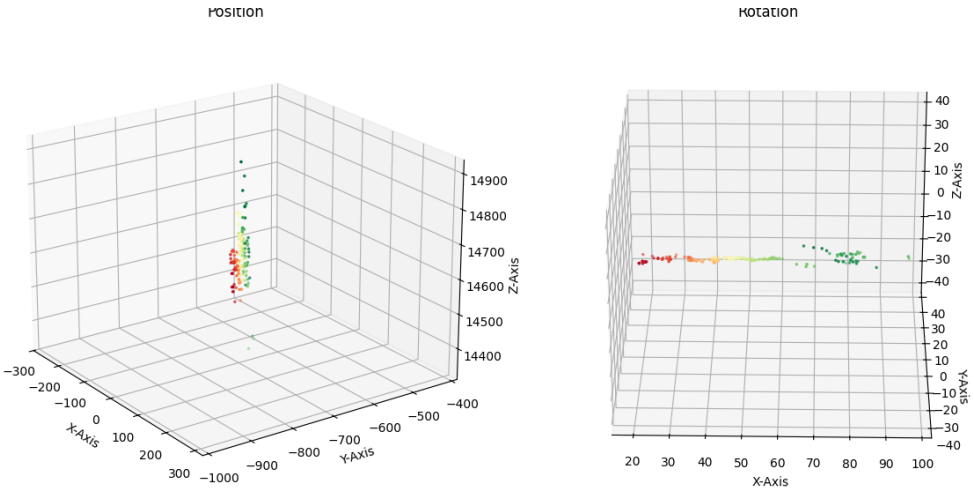


Figure 16: Rotation limit test 2.

The data seen in fig 15 and 16 shows the pose estimations split up into two plots, left showing the position and right the rotation. Each dot being an estimation, its colour showing its time with red being the first and green being the last.
In the first test in fig 15 we can see ORPE began estimating the pose at roughly 0 Degrees (Satellite pointing straight to camera) and failing at 60 Degrees. We can see some points scattered past 60 degrees, these are most likely from the PNP-Refinement from section 3.3.2 producing a few more inaccurate estimations when the main PNP-Solver failed. The position spread in the Z-Axis (Depth) is due to noise and the Y-Axis is due to the rotation movement also having a position change component. The X-Axis has very little change (<1cm). In the second test in fig 16 we can see very similar results. The main difference being the much larger spread near 80 Degrees, possibly due to ORPE incorrectly matching LEDs, causing the position estimation to

be incorrect. One thing to note during these rotation tests is that they were made at absolute upper operational distance limit. This simulates the worst case environment and better performance will be seen at the recommended upper limit of 10 meters. Therefore the recommended operational rotation limit is 60 degrees around the X, and Y-Axis. As previously mentioned, the Z-Axis has no rotation limit.

3.5.3 Failure Testing

In this final testing section we will take a look at what effects LED failures have on ORPE. For this test, two of the satellite's LEDs are covered so they are not visible to ORPE's camera. Specifically the two outer LEDs in fig 11, 5 and 6 were covered. These were chosen, as they have the largest distance from the center and other LEDs and therefore the greatest effect on ORPE's PNP-Solver to estimation the pose. The previous tests results can be carried over to the Failure testing results, as the number of visible LEDs only has an effect on the reliability and time-to-lock, the operational limits are dependant on the image processing, visibility and differentiation of the LEDs. This is due to the number of LEDs not having an effect on the LED visibility or image processing, but rather only the PNP-Solvers ability to estimate the pose. Therefore the test is mainly to verify that ORPE can function with up to two LED failures and how reliable the position estimations are.

Distance	ΔP_X [mm]	ΔP_Y [mm]	ΔP_Z [mm]	ΔR_x [deg]	ΔR_y [deg]	ΔR_z [deg]	Valid [%] (TTL [s])
860cm	0.71	1.25	8.11	0.66	0.25	0.07	49.5 (14.7)

Table 3: LED failure test.

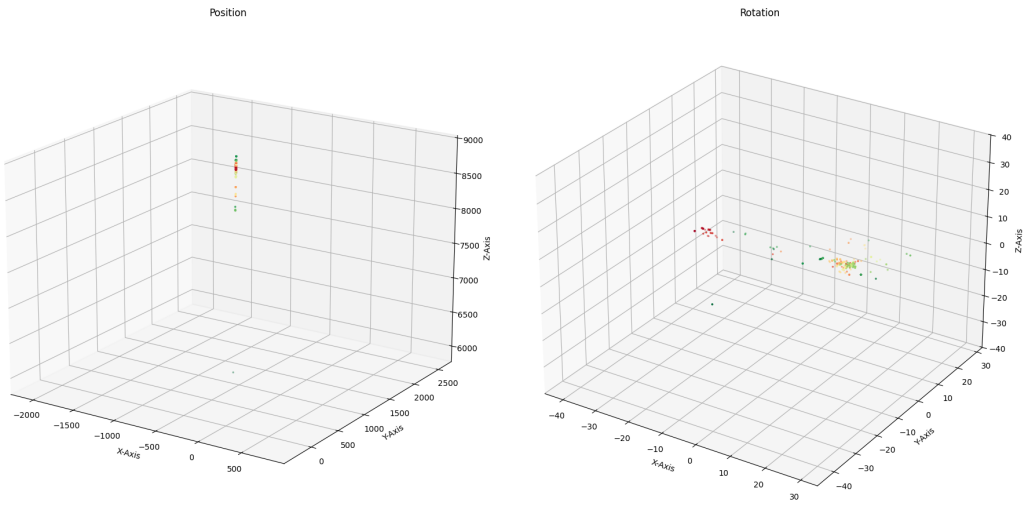


Figure 17: LED Failure test with bright background at 8 meters.

Comparing the results in table 3 to the results from table 1 test at 10 meters, we can see a slight increase in noise, but more notably a great decrease in the percentage of estimations from 80% to just 50%. This can be attributed to ORPE loosing lock suddenly and having to wait for the LEDs to encode. This is due to an incorrect estimation causing the quick LED identification from section 3.2.3 to fail in projecting the LED model and loosing all LED tracking and therefore also IDs. We can also see in fig 17 that the rotation estimations can be incorrect as seen by the initial red points.

3.6 ORPE Conclusion

Through the new changes made to improve ORPE's reliability and the testing made, we can conclude that ORPE can reliably estimate the relative pose between two cubesats at distances from 15 centimeters up to 10 meters and tilt angles of up to 60 Degrees in any environment and up to 20 meters in ideal conditions.

4 Datalink

As the TAMARIW satellite mission is comprised of two cubesats that require close coordination, a direct datalink connection between each satellite is of upmost importance. Although each satellite can dock independently, a datalink allows the possibility to also take control of its opposing satellite if a fault were to occur by initiating a Recovery and Takeover Protocol [9].

4.1 Architecture

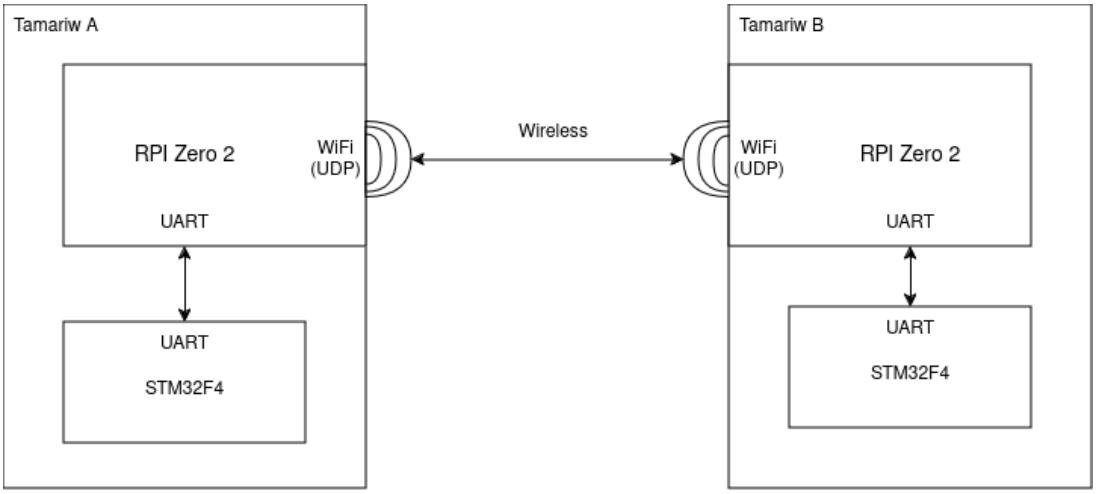


Figure 18: Datalink Architecture.

As the RPI Zero 2 used on both satellites already has WiFi capabilities integrated, it was chosen as the basis for the wireless direct communication between both satellites. One satellite is setup as an Access Point (AP) and the other as a client. For communication between the STM32F4 and RPI Zero 2, a serial UART connection was chosen using the UART ports integrated into the STM32 and RPI Zero 2. Furthermore, as ORPE is also running on the RPI Zero 2 as a separate process, the datalink is used for communication between the STM32F4 and ORPE. This entire network of RPI Zero 2 and STM32F4 for both TAMARIW satellites will be referred to by the TAMARIW Network.

4.2 RODOS Datalink

The RODOS real-time operating system also provides a communication system based on the Publish-Subscriber pattern. RODOS also adds gateways which provide a link between local and external networks. By connecting topics to a gateway and therefore to an external network, we can send and receive data to other networks. The TAMARIW datalink is fully based of the RODOS topic and gateway system. Using the gateway provided, the STM32 directly communicates with any other node on the

TAMARIW network.

4.3 ORPE Datalink Integration

As the ORPE system does not have RODOS integrated, the RODOS topic and gateway system cannot be used to communicate with ORPE. Therefore a translation system is integrated into the Datalink that transforms and forwards the messages, allowing for the STM32F4 to seamlessly communicate with ORPE via Topics. The datalink also has a special inter-communication system to allow the control and communication of the opposing satellites ORPE system. Furthermore, the datalink offers an API to control the WiFi of the RPI Zero 2.

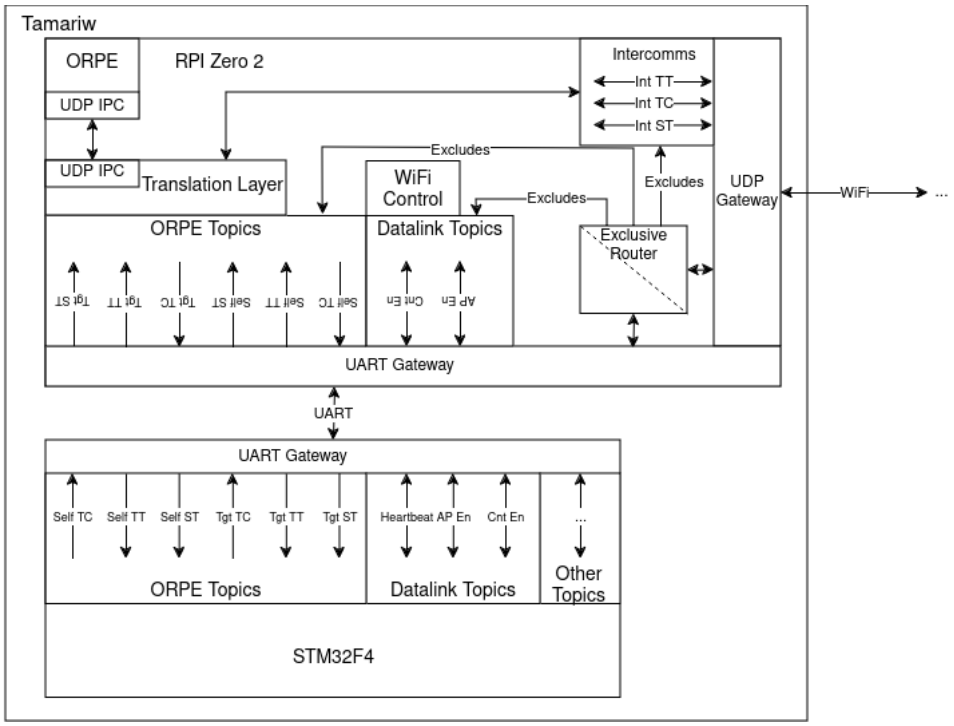


Figure 19: Datalink software design.

4.3.1 UDP IPC

Since both the Datalink and ORPE run on separate processes, there is no direct data communication between them. For this an Inter-Process-Communication (IPC) was created to allow communication between processes based on local UDP connections. The UDP-IPC creates a socket and port for each process and allows for fast communication over UDP. As the connection is UDP based, there is no guarantee that the packets sent will arrive at the other end. Although this is unlikely as the communication in over the local machine. Also there is no buffering, therefore packets waiting to be received are dropped as new one arrives. We can see in fig 19 how the translation

layer and UDP IPC interact.

4.3.2 Exclusive Router

RODOS offers a router that can connect multiple gateways to each other, allowing messages to be forwarded from one gateway to another. This allows seamless communication between one satellite's STM32F4 and the opposing satellites STM32F4. The RPI Zero 2 makes use of this to forward the messages, although this creates an issue when transferring messages from the satellites OPRE system to the STM32, the router will also forward these messages to the opposing satellite over the same topics used to communicate with the other ORPE process. Therefore the Exclusive Router was created, to which specific topic IDs can be given that will not be forwarded. This allows ORPE to also utilise the datalink while isolating itself from the opposing satellite. The Exclusive router works by simply checking each messages topic ID that it supposed to be forwarded, to a database of topic IDs that are to be excluded. If a match is found, then the message is not forwarded. We can see in fig. 19 that the exclusive router in the datalink does not forward any messages from ORPE or from the WiFi control topics.

4.3.3 ORPE Topics

Since ORPE will be using the Datalink system to communicate with the STM32F4 while also communicating with the opposing satellites STM32F4, a topic structure was developed. We can mainly see the topics from fig. 19, were TC, TT, ST denote Telecommand, Telemetry and State, each having two topics, one for communication with ORPE on the same satellite and another for communication with the opposite satellite. We can also the three topics denoted with "Int". These are the Intercomm topics used to transfer the ORPE messages between the satellites. In the ORPE Intercomm, the Datalink explicitly receives messages from the intercomm topics and forwards them to the STM32F4 via the Opposing ORPE data topics and vice versa for messages that come from the STM32F4 topics that are to be forwarded to the opposing satellites ORPE system. Similarly the ORPE messages to and from the satellites ORPE are forwarded to the ORPE Intercomm topics and also the STM32F4 own ORPE Topics. Thereby giving each satellite control and data from both ORPE systems of itself and also the opposing satellite. This is useful, as each ORPE system will generate the same relative pose estimation, allowing for redundancy and also fusion for improved noise and performance characteristics.

4.4 Heartbeat

It is important to know if both TAMARIW satellites are connected, therefore a heartbeat system is implemented to test if the Datalink as a whole is correctly functioning. In this system, each satellite transmits a boolean value that reflects the connection status with the other satellite. A value of `true` indicates that the satellite has successfully received a heartbeat from its counterpart, while a `false` value signals that the heartbeat has not been received, suggesting a potential communication failure. By continuously exchanging these boolean values, each satellite can independently assess whether its heartbeats are being received by the other, enabling real-time monitoring of the link's status. This minimalistic approach ensures efficient and reliable communication without the overhead of more complex protocols.
The interval at which the heartbeat messages are sent is important, as an interval too long will take a long time to detect when connection is established. An interval too short will result in unneeded bandwidth being taken up, for simply testing the connection. Furthermore, if the same constant interval is chosen, and both satellites send their heartbeat synchronised, due to the implementation using a single topic, the heartbeats could collide, resulting in one or both satellites not receiving the opposite satellites heartbeat. Therefore, after sending a heartbeat, a random interval of (1 ± 0.5) seconds is chosen for the next heartbeat send. Although simply receiving the heartbeat message shows the connection functioning, the STM32F4 must wait for a given timeout, in order for it to consider the connection lost. A timeout of four seconds was chosen as a trade-off between disconnection detection time and sensitivity. A timeout of 2 seconds would theoretically work, but if a single packet is lost, this could result in the timeout falsely detecting a disconnection.

4.5 Datalink Testing

Multiple tests were made to find the limits of the datalink and characterize it's performance. For these tests, a data packet containing the send time of the packet and its ID and the total the size of bytes to be sent over the datalink at a constant packet rate. The opposing side then simply receives this packet and immediately resends it back over a return topic. Once the packet returns to the sending system, the time is recorded and the latency is calculated with the following formula:

$$\Delta\overline{T} = \frac{T_2 - T_1}{2} \tag{6}$$

With $\Delta\overline{T}$ being the average latency between the send and return times, T_2 the absolute

receive time and T_1 absolute send time. Every sent package latency was recorded and the average, variance, minimum and maximum latency times are used to characterise the datalink performance. For testing the reliability of the datalink, the packet IDs and known information put into the remaining bytes in each packet can be used to determine the packet loss, wrong packet order and corrupt packets. Although, the UDP section of the datalink already has error detection integrated and therefore packet corruption is not of much interest. Finally we will also be looking into the maximum distance of the WiFi datalink section and checking the effects on the performance.

4.6 Performance

Initially the performance of the UART connection between the RPI Zero 2 and STM32F4 was tested. For this, the RPI Zero 2 acted as the sender and the STM32F4 as the returner. The packets were sent at a 5ms interval and had a total size of 100 Bytes.

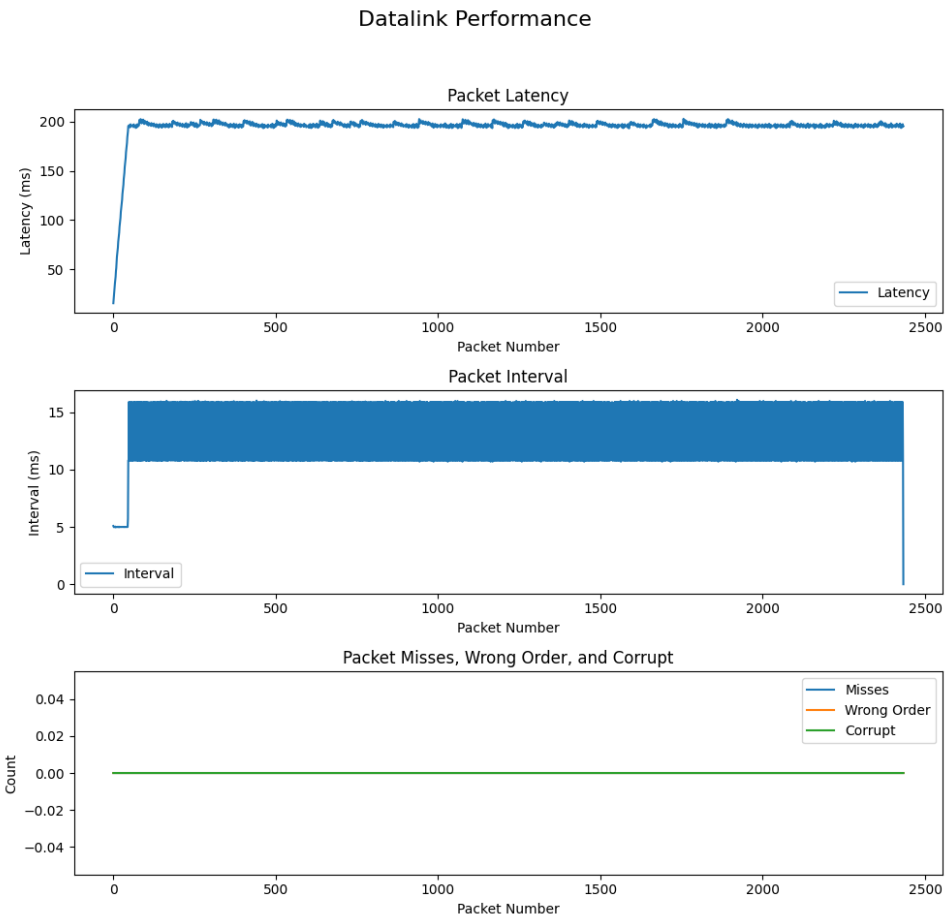


Figure 20: Results of UART datalink testing.

In fig. 20 we can see the test results from the datalink performance testing. In the top Plot we can see the Packet latency, initially starting at roughly 16 milliseconds, but

rising up to roughly 195 milliseconds. If we take a look at the second plot of the packet interval we can see that it starts at the set 5 milliseconds interval, but then suddenly jumps to a larger value of 16 milliseconds. This indicates a buffer used in the UART sending protocol inside RODOS is running full, resulting in the sending thread to get paused when publishing the package, until room in the buffer is available. This also is the cause of the high latency, as the packets must wait inside the buffer to be sent.

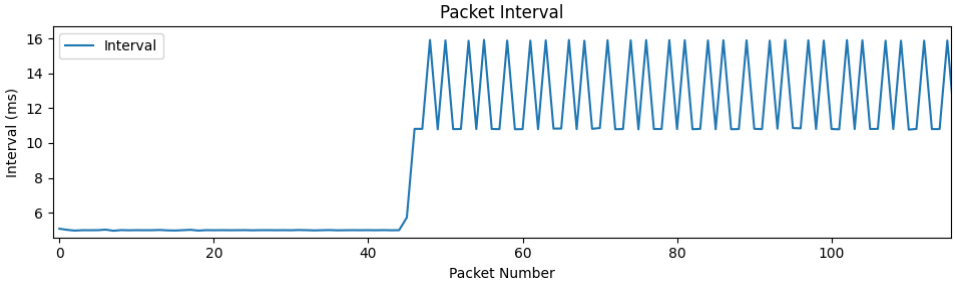


Figure 21: Plot of packet interval.

Taking a closer look at packet interval issue in 21 we can see that the interval switches between 11 and 16 milliseconds. We can conclude from this that the packet interval limit is roughly 16 milliseconds. Choosing a safe limit of 20 milliseconds and accounting for a packet size of 100 bytes, we can determine the maximum datarate of the UART section of the datalink to be $5000\frac{Byte}{s}$. We can also assume the initial latency of 15 milliseconds to remain constant if no buffers are saturated UART and there is no interferences. We can also see that there are no packet errors, showing as expected the UART section to be very reliable.

For the UDP section of the datalink, the satellite prototypes were placed at a distance of 20 centimeters facing towards each other similar to docking. Initially a packet interval of 10 milliseconds and 100 bytes data packet size was used.

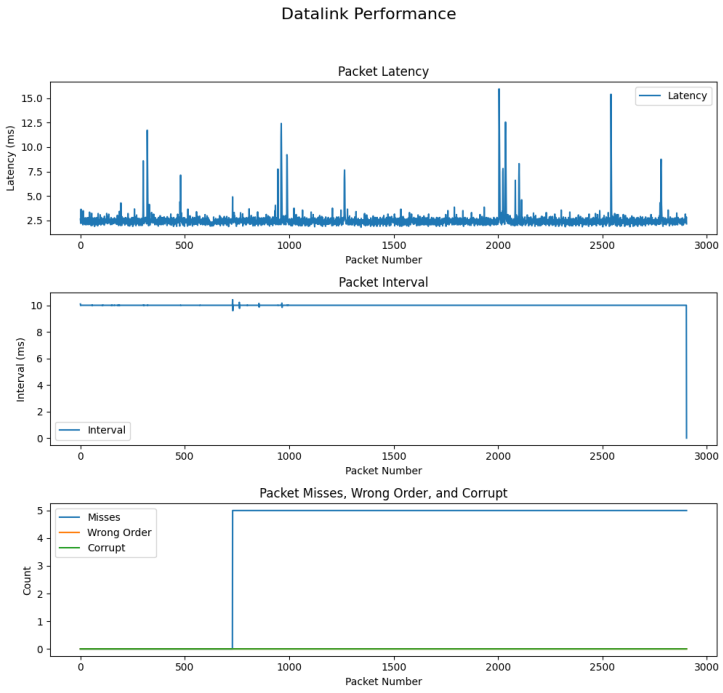


Figure 22: Results of UDP 10ms datalink testing.

We can see in fig. 22 that the WiFi link is greatly more susceptible to interference than the UART based link. In the top plot of the Latency we can observe the average latency of 2.5 milliseconds with a minimum of 1.8 milliseconds and a maximum of 16 milliseconds around the 2000 packet mark. We can also see multiple sudden spikes in latency and also around the 700 packet mark five packets are lost of in total 2904 packets, but no corrupt packets and none in the wrong order. We can determine that the packet loss rate is roughly 0.2%. These packets were probably lost due to the surrounding WiFi interference and shows a worst case.

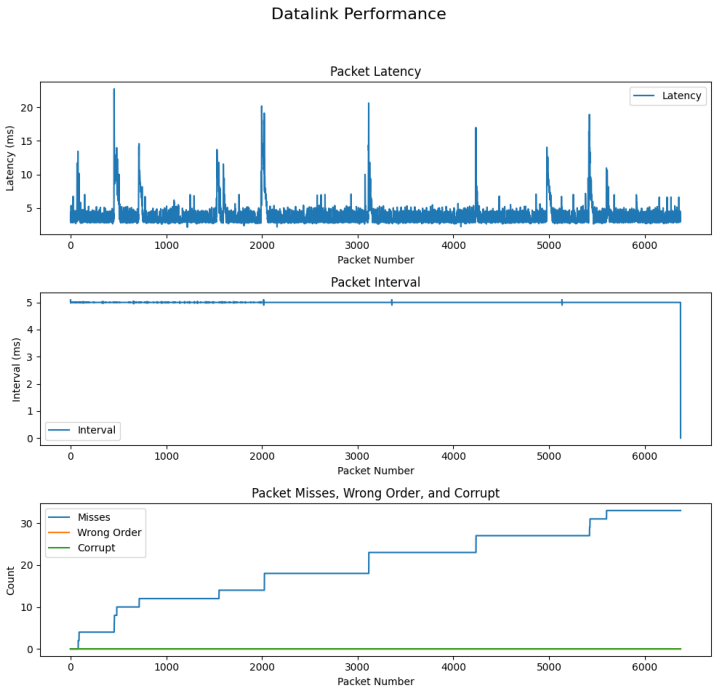


Figure 23: Results of UDP 5ms datalink testing.

Next the interval was increased to 5 milliseconds to find the packet rate limit. We can see in fig. 23 that at this packet rate the rate of packet loss is much greater. Taking a look also at the packet latency we can see the spikes in latency correspond to the loss of packets. Together we can determine that the higher packet rate makes the WiFi based link more susceptible to interference. Concluding that the link would possibly be capable of a 5 millisecond interval, because between interference spikes the datalink has no issues running at the interval of 5 milliseconds. Once again there are no packet order errors or corrupt packets. These results show us that with a minimum interval of 10 milliseconds and 100 Byte packets results in a maximum datarate of $10000\frac{Byte}{s}$ for the UDP datalink section.

4.6.1 Distance Limit

Finally we will be taking at look at the maximum distance of the Datalink and its performance at this limit. For this test we will not be looking at the UART link as the distance between satellites has no effect on the UART link.
For this test the satellites were place increasingly far apart until the WiFi between the satellite was unable to regain connection. This limit resulted in being roughly 20 meters apart for reliable connection. It is important to note that this is in absolute worst conditions as these tests were preformed with line-of-sight inside a building with many other WiFi signals. These interference's are not expected in orbit.

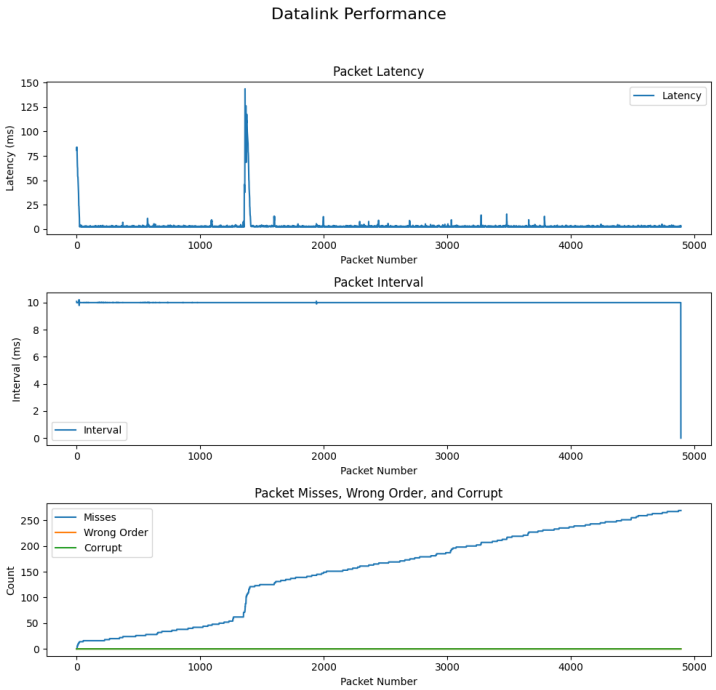


Figure 24: Results of datalink limit datalink testing at 10ms interval.

In fig. 24 we can see the test results of the datalink at 20 meters distance and a packet interval of 10 ms and size of 100 Bytes. The most important difference to 23 being the much larger amount of packets lost and a very large latency spike of 144 milliseconds at 1400 packets. The average latency being roughly 3.7 milliseconds with a minimum of 1.9 milliseconds and 269 packets of 4890 lost. From this we can determine the packet loss rate to be roughly 5.5%.

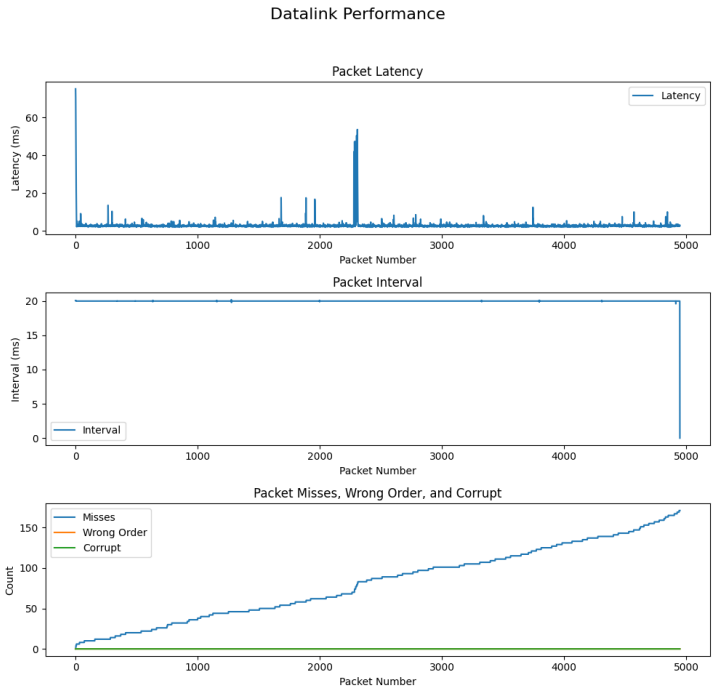


Figure 25: Results of datalink limit datalink testing at 20ms interval.

To check if a reduced packet rate will improve the reliability, the packet interval was increased to 20 milliseconds. In this test 25 we can see that reducing the packet rate reduced the packet loss to 3.4% or 171 packet of 4949. This shows that the packet loss at the datalink limit is correlated to the packet rate. The latency characteristics remained roughly the same with an average of 3 milliseconds, minumum of 2 milliseconds and maximum of 75 milliseconds.

4.7 Datalink Conclusion

From the datalink distance testing we can conclude the maximum datalink working range to be in the worst case at 20 meters. At this range the STM32F4 to STM32F4 datalink characteristics show a datarate limit of $5000\frac{Byte}{s}$. We can calculate the total latency by simply summing up each links latency. Since a packet travelling from one STM32F4 to another traverses the WiFi once and UART twice, the latency is simply $\Delta T = T_{WiFi} + 2T_{UART}$ and results in an average latency of 33 milliseconds, but at times can be up to 200 milliseconds if there is interference for the WiFi. Since the UART link has no packet loss, only packets going over the WiFi link travelling between satellites will have a packet loss rate of up to 5.5%.

5 Sensor Fusion

The goal of the Sensor Fusion, is to take the data from multiple sensor systems and produce a single reliable estimation. A further goal it also to improve the accuracy and redundancy when multiple sensor systems are producing estimations at the same time.

5.1 Architecture

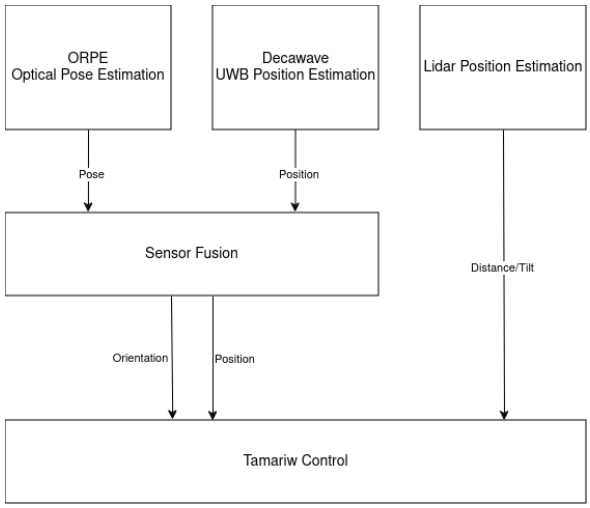


Figure 26: TAMARIW docking sensor subsystem.

The TAMARIW satellites will use three sensor systems. An Ultra-Wide-Band (UWB) radio based position estimation system. The UWB system estimates the relative position of the satellites at long ranges of 10 meters up to 100 meters. This system only produces position estimations and at a low absolute accuracy and data rate. This system is only used for the initial rendezvous phase and therefore does not require high accuracy or speed. The second system as previously in section 3 discussed is the ORPE optical based system. It is used for medium ranges from 15 cm to 10 meters and has a good relative accuracy and high data rate. Finally, the LiDAR based system is used for estimation under 40 centimeters. The LiDAR based system can estimate the relative distance between the docking surfaces at one millimeter accuracy. The 3D relative position data gathered from stage one and two of the sensor systems will be filtered and then forwarded to the satellites ADCS to be utilized by the rendezvous and attitude control system. Meanwhile, the relative distance, velocity, and tilt angles estimation from the third system will be used directly by the Guidance and Docking Control subsystem to ensure a smooth docking process. Furthermore, the opposing satellite's ORPE system could also be used to improve the noise characteristics and redundancy at medium ranges.

5.1.1 UWB Position Estimation

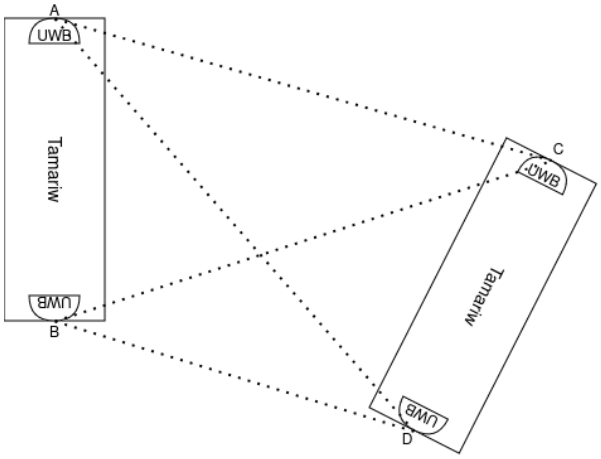


Figure 27: UWB radio based position estimation.

The radio based position system estimates the relative 3D position between the TAMARIW satellites. The TAMARIW mission will use four DW1000 Ultra-Wide-Band radio modules, two per satellite to estimate the relative positions of the satellites by measuring the distances between each module. The radio base position estimator can accurately measure the satellite distance, but the tangential position is substantially more inaccurate. This makes it a great complementary system for the ORPE estimator, as the ORPE system accuracy is great tangentially but worse at distance.

5.1.2 Inputs and Outputs

The sensor fusion filter will have two inputs, one for each sensor system and also two outputs, one for the relative position and another for the relative orientation. The reason for separating both position and orientation outputs is due to the different in rate at which the position will be estimated by ORPE and UWB. UWB estimates the position at 2Hz and ORPE both position and orientation at 10 Hz.
As ORPE runs also on the opposite satellite and the estimations are only inverted, we can also use the opposite satellites ORPE estimations for higher accuracy and redundancy. This can be simply implemented by inverting the estimations from the opposite ORPE and integrating it the same way into the filter as ORPE is already.
The resulting data rate of the filter will depend on its input sources. As the filter runs on an event basis, only once a new input source produces information, a new output will be produced. We can see a visual diagram in fig 28. This results in the following characteristics: If only the UWB estimator is running, then the position output will run in sync with the UWB estimator. If both ORPE and UWB are running, then position will be produced from 2-12 Hz. Noting that the interval between each output will not

be constant. The orientation will be estimated at 10Hz in sync with ORPE. Finally, if both ORPE and the opposite ORPE are running with also UWB, then this results in a rate from 2-22Hz for the position and 2-20Hz for the orientation. In this case both the position and orientation are not guaranteed to have a constant interval.

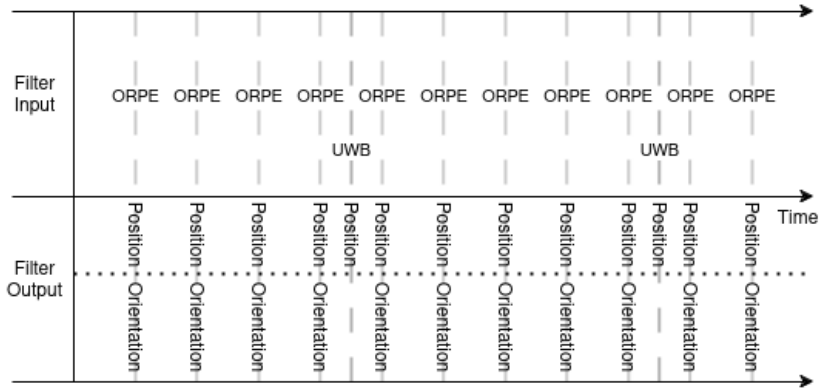


Figure 28: Varying Data Rate of filter output.

5.2 Design

As previously mentioned, the design goal of the filter is to fuse the multiple inputs from the different sensor systems, improve the reliability and accuracy of inputs. A further design goal is to keep the filter calculation efficient, as the CPU and RAM resources of the STM32F4 are limited. Normally, for similar use-cases a Kalman filter is chosen, due to its optimal estimation results and great characteristics. One issue is the non linear nature of the orientation model. This is generally solved by use of the Extended Kalman filter. Although the extended Kalman filter would certainly be capable of this use case, the large amount of complex matrix calculations will result in the filter using large amounts of the already limited resources provided by the STM32F4. Therefore the proposed filter is based on a complementary system of weighted averages.
For the following sections, we define the covariances with the following vectors.

$$\vec{cov}_p = \begin{bmatrix} cov_{px} \\ cov_{py} \\ cov_{pz} \end{bmatrix} \qquad \vec{cov}_s = \begin{bmatrix} cov_{sx} \\ cov_{sy} \\ cov_{sz} \end{bmatrix} \tag{7}$$

With $\vec{cov}_p$ being the covariance of the process and $\vec{cov}_s$ of a sensor. Here we can define each sensors influence of the sensor in different axis'.

Its important that the chosen values reflect the sensors characteristics. With ORPE $cov_{sx} = cov_{sy} < cov_{sz}$ and UWB $cov_{sx} = cov_{sy} > cov_{sz}$. Since ORPE's noise and accuracy characteristics are dependent on the distance between the satellites, the covariance for ORPE is given as a relative percentage and the actual covariance is recalculated based on this percentage for each new sensor value. For this the following formula is used:

$$\vec{cov}_s = |x| \begin{bmatrix} cov_{sx} \\ cov_{sy} \\ cov_{sz} \end{bmatrix} \tag{8}$$

With again $\vec{cov}_s$ being the sensor covariance used in the filter, and cov_{sx}, cov_{sy} and cov_{sz} being given in percentages. $|x|$ being the measured distance between the satellites, which can be calculated directly from ORPE's position estimation.

5.2.1 Start Values

When the filter is started, it uses the default values of zero for position and orientation. This does not cause stability issues for the filter when running, but causes a long settling time. This effect can be seen in fig. 29 below, were the covariance valuers were chosen to illustrate this effect. We can see in the left plot that the values start incorrect and converge over time to the correct output. To combat this issue, the filter will set the internal output value explicitly to the initial value it receives from the sensors. The resulting effect can be seen in fig. 29 on the right plot that implements this starting value. If receiving a position estimation from the UWB radio estimator, the position estimation is set to the sensor output and once the initial estimation from ORPE is received, the orientation is set too. If the initial sensor data is received from ORPE, then both the position and orientation estimations are set. If receiving both ORPE and UWB sensor data, then ORPE is preferred due to its higher accuracy.

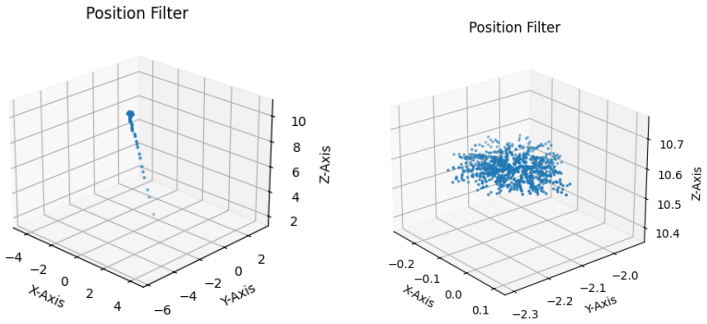


Figure 29: Filter settling. Left no start value, right with start values.

5.2.2 Weighted Average

The developed sensor fusion algorithm is based on weighted averages. Each sensor system has a covariance matrix describing its accuracy/influence on the resulting output. Basing the filter on weighted averages has the advantage of fast execution times and substantially lower memory footprint. The weighted average can be calculated via the following formula:

$$Avg_{xy} = \frac{xa_x + ya_y}{a_x + a_y} \tag{9}$$

With the values to be averaged being x and y and their respective weights being a_x and a_y. Here we can see, that for larger weights, the input values will have more influence on the average. This relationship will be inverted in 5.2.4.

5.2.3 Depth and Tangent spaces

The complementary part of the fusion problem was simplify it into two problems in order to find much simpler solutions and have the sensor systems complement each other. As the UWB position estimator is accurate with distance but not tangentially and the ORPE is the opposite by being accurate tangentially but not in depth, the position fusion problem was split into two parts for the Z-Axis (depth) and XY-Axis (tangent). Using the covariance matrix, we calculate the depth and tangent covariances:

$$cov_d = cov_z \tag{10}$$

$$cov_t = \sqrt{cov_x cov_y} \tag{11}$$

With COV_d being the covariance value used for the depth and COV_t use for tangential.

5.2.4 Fusion Calculation Method

Building upon the past sections design principles, we can derive the fusion algorithm. Taking a look at the weighted average formula 9, we can see that a higher weight results in more influence on the filter output, but on the other side, higher covariance is correlated with higher inaccuracy and reliability. Therefore it is more desirable that a sensor with larger covariance should have less influence on the filter algorithms output. We can take the inverse of the covariance to reverse this relationship and have a larger covariance correspond to less influence on the output.

$$cov^{-1} = \frac{1}{cov} \tag{12}$$

Finally we combine each sensor into the weighted average formula 9 and inverse 12, for the position estimation results in:

$$\vec{x_n} = \begin{bmatrix} \frac{x_{xn-1}cov_{ts}+u_x cov_{tp}}{cov_{ts}+cov_{tp}} \\ \frac{x_{yn-1}cov_{ts}+u_y cov_{tp}}{cov_{ts}+cov_{tp}} \\ \frac{x_{zn-1}cov_{ds}+u_z cov_{dp}}{cov_{ds}+cov_{dp}} \end{bmatrix} \tag{13}$$

With $\vec{x_n}$ being the filter output, $\vec{x_{n-1}}$ the previous filter output and $\vec{u}$ being the new sensor input, cov_{ts} the tangential (or d for depth) covariance calculated with 11 (or 10 for depth) for the sensor input (or p for process covariance). Although rotations are non-linear, assuming small rotations between sensor reading, with Quaternions we can still use the weighted average to get a good approximation. As only the ORPE sensor can estimate the relative orientation, the sensor fusion filter acts only as a smoothing low pass filter to help reduce high frequency noise and dampen any incorrect estimations. The orientation part of the algorithm also calculates the covariance as a single value with the following formula:

$$cov_r = \sqrt{cov_x cov_y cov_z} \tag{14}$$

With cov_r as the covariance value used for the rotation fusion from the covariance vector $\vec{cov}$ of the input. We then simply use the rotation quaternion as an input to the weighted average 9 and 12. This results in the following formula:

$$x_n = \frac{x_{n-1}cov_{rs} + u_s cov_{rp}}{cov_{rs} + cov_{rp}} \tag{15}$$

With once again x_n being the filter output but for rotation as a quaternion, x_{n-1} the previous filter output and u_s as sensor input. cov_{rs} as the rotation covariance for the sensor and cov_{ps} for the process rotation covariance. After the averaging we must normalize the quaternion. This is simply done with the following formula:

$$\hat{q} = \frac{q}{\sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2}} \tag{16}$$

5.3 Testing

The sensor fusion was tested to analyse its effects on the estimations of the sensor systems and how well its capable of combining their outputs. For this, sensor data from the ORPE testing in section 3.5 was given to the sensor fusion algorithm in a simulation and its outputs were recorded for analysis. Each simulation is ran with the filter process covariance set to 20 centimeters for all axis.

5.3.1 UWB Simulation Data

Unfortunately the UWB radio based estimation system was not ready for in the required time frame for testing to be able to record data for these tests. Therefore simulation data was produced for the testing. It is assumed that the noise characteristics of the UWB radio based estimator can be approximated by Gaussian noise. Therefore the simulation data will produce estimations with a given Gaussian noise, and for the tangential and depth position differently. The simulation will test ORPE alone at close distances, UWB radio estimation alone at large distances and ORPE and radio based estimation together at medium ranges.
For the UWB simulation, it is assumed that the standard deviation of the depth is roughly 10 centimeters and the tangential standard deviation is 1 meter.

5.3.2 Results

The initial simulation was to test how well the sensor fusion works for only UWB radio based sensor values. For this, the simulated sensor data was set to 20 meters distance and remains stationary.

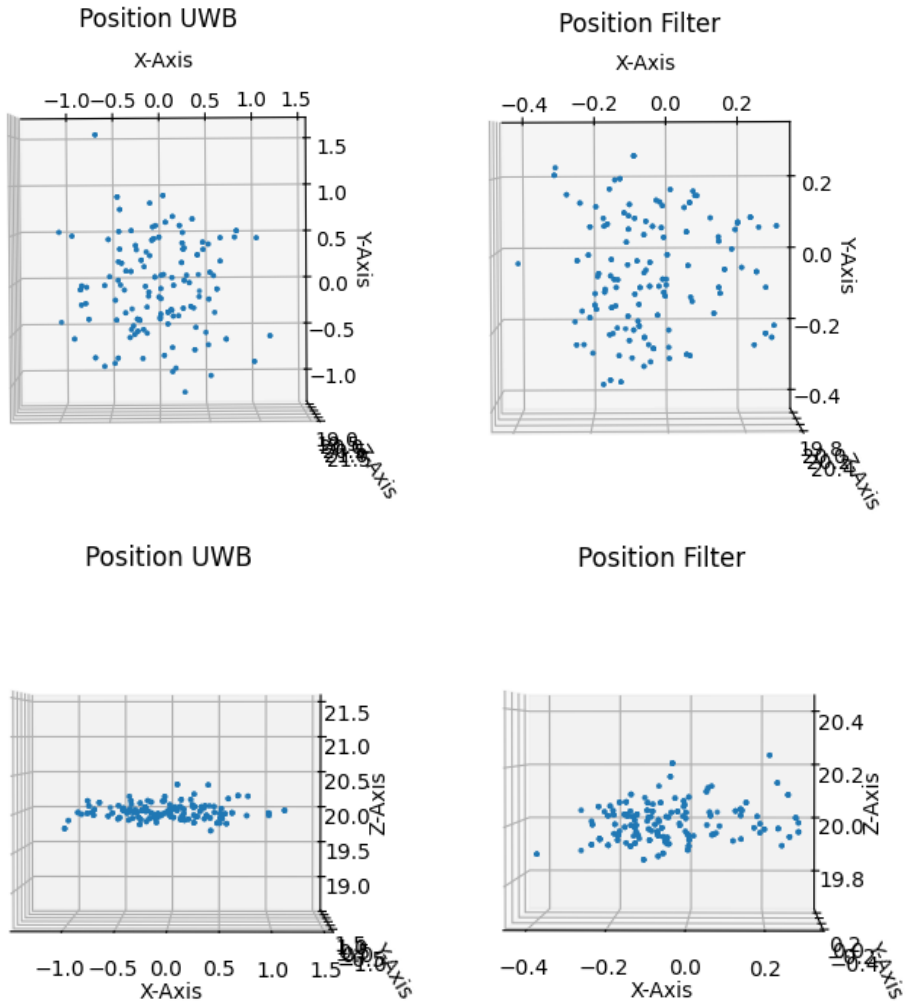


Figure 30: Simulation of UWB only at 20 meters.

As we can see in fig 30, mostly the tangential axis' have a reduced noise from the initial noise of 1 meter in X- and Y-Axis' to roughly 0.3. This is the expected result, as the weighted average acts as a simple low pass filter damping higher frequency noise.

The simulation was repeated with real test data from ORPE at a distance of 1.4 meters. The ORPE covariance in the filter was set to 0.1% percent for X- and Y-Axis and 2% for the Z-Axis.

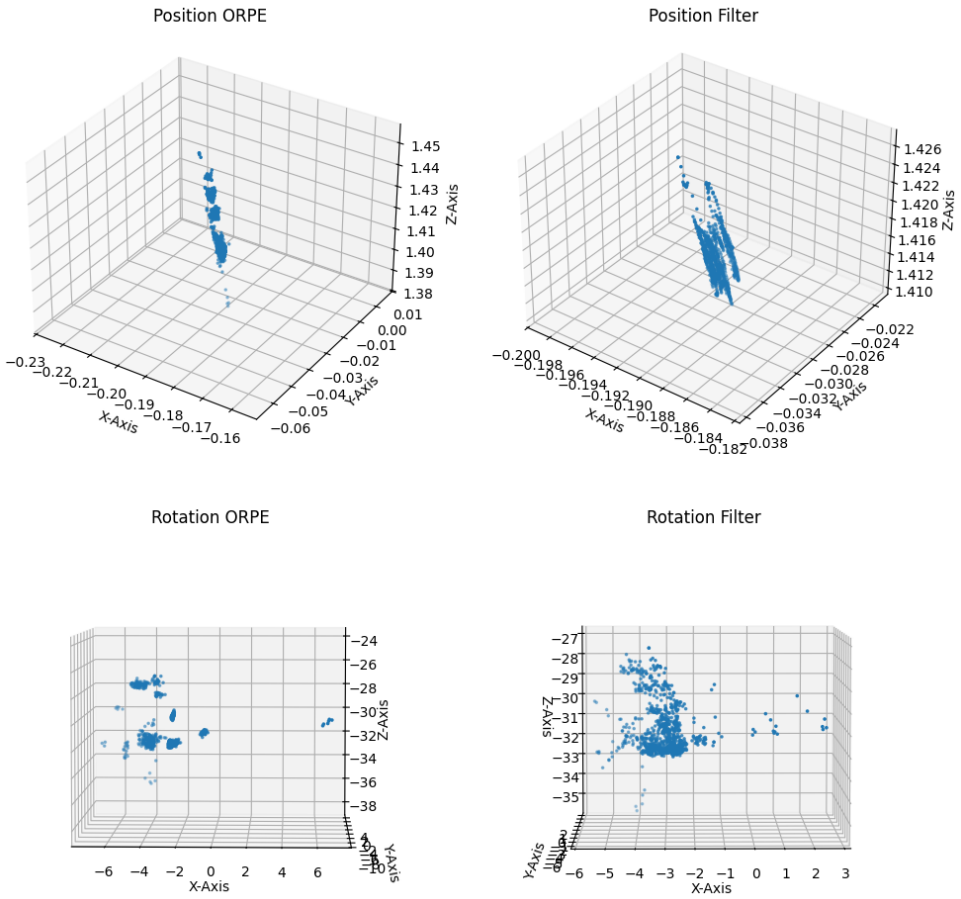


Figure 31: Simulation with real ORPE data at 1.4 meters.

As seen in fig 31, the filter reduces the spread of the position and rotation estimations. Taking a closer look at the rotation estimation plot from ORPE, we can see a large outlier of the X-Axis at 7 Degrees. These outliers are clearly dampened in the filter rotation plot.

More interesting will be at the transition from the UWB position estimator to the ORPE estimator. For this the simulation was repeated with real ORPE data, but also UWB radio based estimation as the same position as the ORPE position estimation. The real time ORPE data was taken from a test at 14 meters distance.

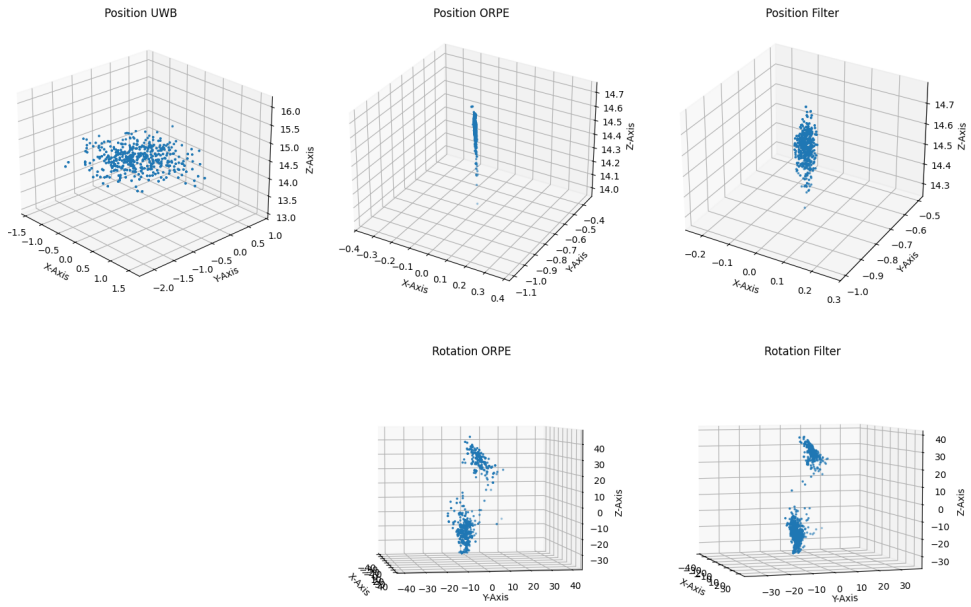


Figure 32: Simulation with of transition from UWB to ORPE.

In fig. 32 comparing the position plots of UWB and ORPE we can clearly see the expected tangential and depth noise characteristics and in the filter clearly combining the two into a single output. In the rotation plot from ORPE we can see two clouds of points that are denser in the filter output, showing the noise damping due to the low pass filter characteristics of the rotational part fo the filter. The possible reason for the two separate clouds of points in the ORPE rotation estimation is an example of ambiguity issues of the rotation at long distances. This could possibly be filtered and solved in the future by utilising the opposite satellite ORPE estimations.

5.4 Sensor Fusion Conclusion

We can conclude from the sensor fusion testing that the filter is capable of reducing estimation noise and combine the sensor data from both the UWB radio based sensor system with the ORPE optical based sensor system. The sensor fusion is capable of running with only the UWB radio based estimator or ORPE or both and has therefore no issues during the transition from the UWB to the ORPE estimators.

6 Software implementation

In this section we will go over the implementation specifics for the TAMARIW mission.

6.1 ORPE

ORPEs implementation has gone over many large changes since the initial implementation, in order to run as its own process on the Linux operating system, and also allow for testing using pre-recorded videos from tests for simulation and evaluation.

6.1.1 ORPE Library

Due to the much different runtime performance requirements and platform between running ORPE on the TAMARIW satellite and running simulations for testing and evaluation, ORPE's codebase was restructured to let ORPE be used as a library. This allows for easy version control via the use of git and also allows for the exact same code base to be used for the simulation and TAMARIW mission.
There are two telemetry data types that are created by ORPE at different times. The first is the ORPE state, that simply indicates is ORPE is active but idle (not estimating), running (estimating), timeout meaning the timeout limit was reached (This is a debug feature and should never be enabled during the mission), Stopped meaning ORPE was stopped by a telecommand, camfailed meaning some camera failure occurred and must be restarted and finally shutdown meaning that the ORPE process was triggered to be shutdown by a telecommand. Also ORPE telemetry data is sent when running for every new image frame. This data encompasses the estimation pose, current frame count, the timestamp of the image in milliseconds in reference to the start of ORPE estimation, a boolean signifying if ORPE considers the current estimation to be valid. Two more arrays of length 16 for information of the identified LEDs found inside the image. The index of the array corresponds to the ID, meaning the information to ID 4 is found inside array index 4. The first array contains the number of LEDs identified with the ID. The second array is of 2D and contains the respective LEDs X and Y coordinates in pixels in the image frame. This is of course only valid if the corresponding index in the first array LED count is exactly one, otherwise the LED is either not yet found or could be ambiguous.

6.1.2 ORPE Process

The change of ORPE to a library also required the structure of the previous ORPE process to be changed. This was split into two versions, one for running simulations for testing new improvements, tuning the parameters and is optimized for debugging. The second for running on the Tamariw satellites optimized for performance.

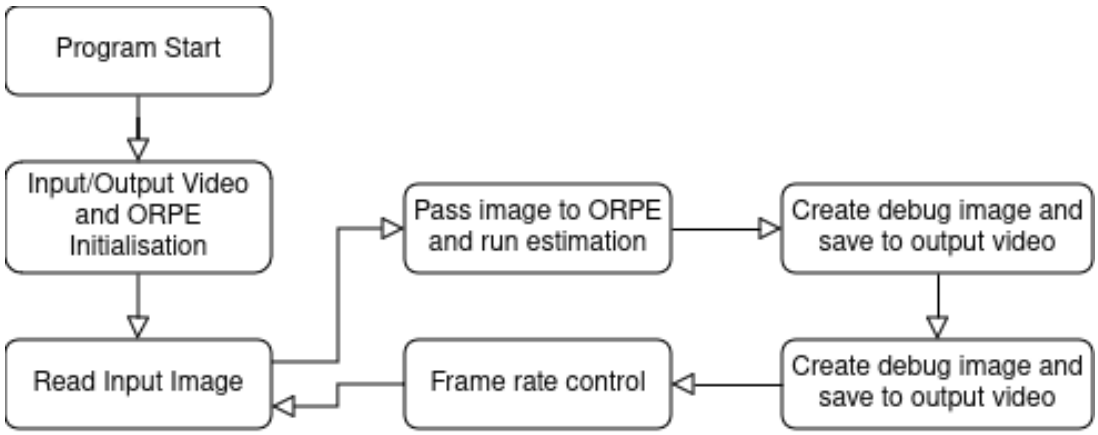


Figure 33: Flowchart of the simulation runtime.

The simulation runtime uses videos recorded by the TAMARIW camera as an input for ORPE and allows for visual previews of the different image processing steps and allows for the simulation to be slowed down, paused or fast forwarded. Everything runs in a single thread. Initially an image is extracted from the input video, then the image is given to ORPE, ORPE's estimation loop is called, the result is retrieved and then visually displayed inside the original image. This allows for improvements to be easily evaluated and debugged and performance between algorithms and techniques to be compared.

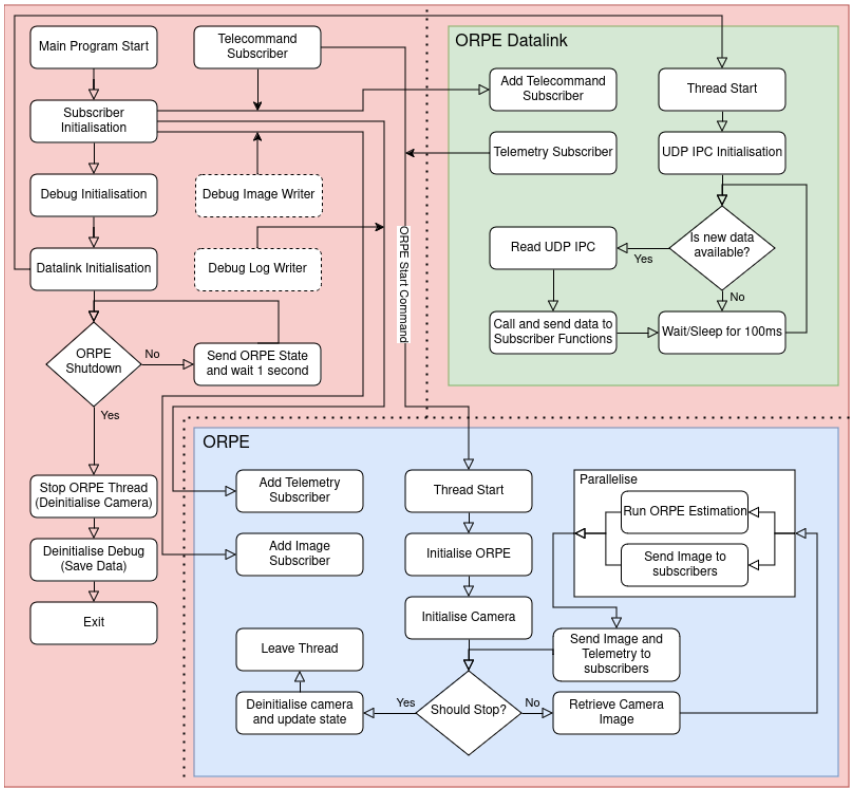


Figure 34: Flowchart of the ORPE process on TAMARIW.

The TAMARIW version is much more complex, due to its optimizations that attempt to parallel many tasks and utilise the multiple cores of the RPI Zero 2 and simplify the need to also run the communication between the ORPE and Datalink processes. The ORPE process is split up into three different sections: Main, ORPE and Datalink. The communication between the different threads works by the publish and subscribe design pattern. The main thread is the initial main function that is called at process start. It takes care of the initialisation, subscribing and starting of the datalink and ORPE threads and shutdown. The main thread also periodically transmits the current state of ORPE to the datalink, so to make sure the entire system knows the current ORPE state. The ORPE thread takes care of retrieving images from the camera, giving them to ORPE, running the ORPE processing and finally sending the telemetry via the Datalink thread. The ORPE processing was parallelised with the sending of the new camera image to the subscribers, as the copying of images takes a substantial amount of time and some modules like the debug video writing have very long processing times and should be done on a separate core. Theoretically the process of retrieving the image from the camera could also be parallelized as this process take a substantial amount of time, but the implementation in LCCV [3] already parallelised this process. Finally the third is the Datalink thread. This should not be confused with the Datalink process, which is the main Datalink system from 4. The Datalink thread

in ORPE's context simply facilitates the data transfer between ORPE and the Datalink process. ORPE's Datalink was parallelized due to the necessity to constantly to check the UDP-IPC based connection 4.3.1 to the main Datalink process. The Datalink thread simply checks the UDP-IPC channels for new data at a rate of 10Hz. If new data is received from the IPC, then its forwarded to all subscribed functions. The Datalink sends telemetry and ORPE state data simply by sending it immediately over the IPC once the subscriber function is called.
For debugging, a subscriber takes the published images and telemetry from the ORPE thread and saves them to a video or logging file that later can be used for testing and debugging in the ORPE simulator. This process should not be running during the TAMARIW mission, as the saving to SD-Card uses large amounts of resources and creates unnecessary wear on the SD-Card.

6.1.3 Control via STM32

Since the datalink system 4 allows for a connection from any point in the TAMARIW network, the main control and communication point for ORPE is via the STM32F4 of each Satellite, but ORPE can be controlled by any point inside inside the TAMARIW network including the RPI Zero 2. For simplicity a translation layer was implemented only on the STM32F4 called the ORPE manager. The ORPE manager takes care of setting the correct ORPE state and making sure ORPE is running. It also translates the telemetry and pose estimations into 4x4 Transform matrices, for use by the rest of the system.

6.1.4 ORPE LED Control

As the LEDs used by the opposite satellites ORPE are connected to the GPIO of the STM32F4, the control of the LEDs is done by a single thread. This LED control thread initialises the GPIO and runs at the same rate as ORPE. The LED control can change in real time the LED control modes. Each LED has four control modes, one where the LED is powered off, another where the LED is constantly illuminated, one with the LED blinking its its identification code only once and then staying otherwise illuminated and finally with constantly blinking its identification code. Therefore, using the telemetry sent by the opposite satellites ORPE, we could only blink the LEDs that have not yet been identified, and keeping all others illuminated, improving the reliability of the ORPE tracking. The LED control thread works by having two state machines for controlling the LED blinking. The first state machine controls the actual LED coding, starting with the initial off bit, then the LED hamming code and finally the idle state which is controlled by the second state machine. The second state machine controls if

the first state machine should start coding, and if the idle state should be illuminated of powered off, depending on the control mode of the LED. The second state machine also automatically switched from single coding mode into illuminated once the coding has finished.

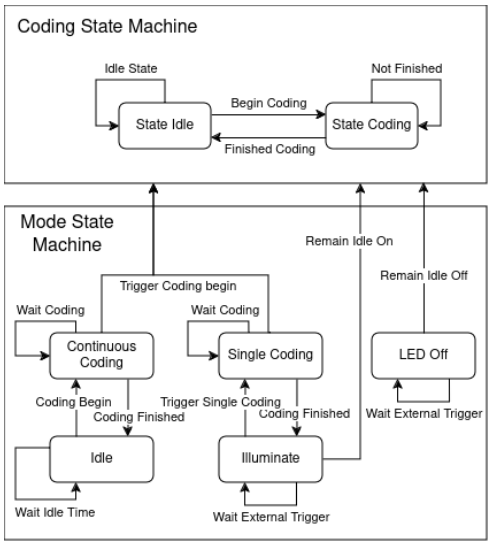


Figure 35: State machines of the LED control.

6.2 Datalink

As previously mentioned, the Datalink system for the TAMARIW network, runs using the RODOS operating system on top of the Linux operating system in its own process.

6.2.1 Translation Layer

In the Datalink implementation, a RODOS thread implementing the translation layer simply checks all subscribers if new data is available, and forwards it to the corresponding topic for the ORPE intercomms and similarly checks the UDP-IPC to ORPE if new data is available and repackages the data and forwards it to the correct topics.

6.2.2 ORPE

The Datalink furthermore takes care of starting the ORPE process. A RODOS thread called the ORPE startup thread simply waits and periodically polls for new commands from the ORPE telecommand topic. It this point in time the ORPE process is assumed to be currently not running. Once the ORPE startup telecommand is received by the ORPE startup thread, it makes a shell call to startup the ORPE process. This is a blocking call that waits until the ORPE process finishes and closes. During this time the telecommmand subscriber for the ORPE startup thread is disabled. The reason is that during the time the ORPE process is running, if another startup telecommand were to

be sent, the subscriber would hold on to this and once the ORPE process closes the ORPE startup thread could receive the old startup telecommand and start the ORPE process again.

6.2.3 WiFi nmcli

NetworkManager is a program for providing detection and configuration for systems to automatically connect to networks [4]. The command line interface is called nmcli and is the primary interface used by the Datalink and used to setup the initial networking interface for the RPI Zero 2's. The RPI Zero 2 WiFi network was setup to connect to a LAN connection for internet and an Access Point (AP) for connection between the TAMARIW satellites. This allows for an internet connection at the satellite connected to LAN and this also allow internet access to the other satellite. This is vital during the development and testing of the TAMARIW satellites. The networking is setup with the following commands from the Raspberry Pi Documentation [7]:
First a network bridge interface to connect ethernet to WiFi Access point is setup.
`sudo nmcli connection add type bridge con-name 'Bridge' ifname bridge0`
Then an ethernet connection is created with the network bridge as a master.
`sudo nmcli connection add type ethernet slave-type bridge con-name 'Ethernet' ifname eth0 master bridge0`
Finally the Access Point is created with the bridge also as a master
`sudo nmcli connection add con-name 'Hotspot' ifname wlan0 type wifi slave-type bridge master bridge0 wifi.mode ap wifi.ssid TMWNetwork wifi-sec.key-mgmt wpa-psk wifi-sec.proto rsn wifi-sec.pairwise ccmp wifi-sec.psk TMWNetwork`
The resulting network name is `TMWNetwork` and the password is also `TMWNetwork`.
The bridge can now be activated with `sudo nmcli connection up Bridge`.

6.2.4 WiFi AP Negotiation

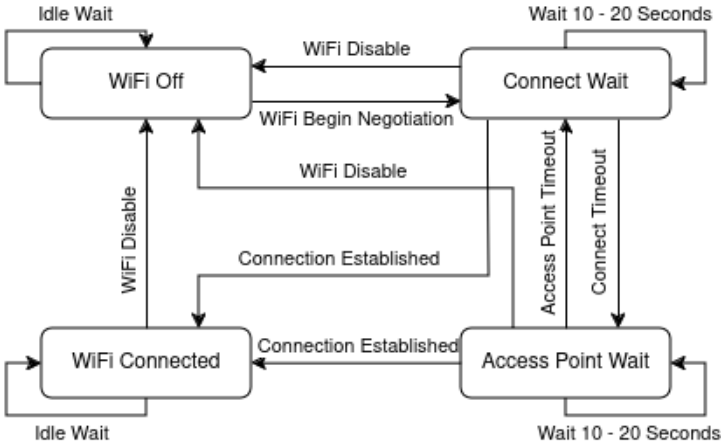


Figure 36: State machines of the WiFi negotiation protocol.

As the infrastructure mode in WiFi requires one device to be setup as the AP and another as the client, both the Tamariw satellites must negotiate which one should be the AP and which the client. This also requires that the Datalink is able to change between client and AP mode. For this, a few simple topics are implemented between the STM32F4 and RPI Zero 2 with boolean values. One is to enable or disable the AP and the client mode. These topics are excluded from the router on the RPI Zero 2. This allows for in total three states: AP, Client and WiFi powered off. The logic for connecting negotiating the AP or client modes, is implemented on the STM32F4. The negotiation protocol relies on the Satellites enabling either the AP or client mode and waiting a random time for a connection to be established. The implementation is based on a state machine with four modes: `Off` the WiFi is powered off and waiting for the command to begin negotiating, `Connected` the WiFi is connected and Datalink is working, `ConnectWait` client mode is activated and waiting for confirmation and `AccesspointWait` with AP enabled and waiting for confirmation. For connection confirmation the datalink heartbeat 4.4 is used to check if the connection to the opposing satellite is working and the WiFi has successfully connected. Once achieved, the state machine will stop switching and keep the current mode enabled. If the connection suddenly fails, the state machine will repeat the connection process. Furthermore the state machine also allows the WiFi modes to be manually set. A fourth mode being Automatic that implements the WiFi negotiation protocol when connection is lost. Also, the WiFi negotiation will only begin at relative distances less than 20 meters, where WiFi connection is feasible. This will be determined using the sensor fusion position output.

6.3 Sensor Filter

The sensor fusion filter is implemented as a RODOS thread on the STM32F4. The thread is permanently suspended until new sensor data arrives. Two functions receive new sensor data from ORPE or the UWB radio based position estimation. Each Function simply copies the sensor data and then sets a flag to indicate new data is available and then resumes the sensor fusion thread. Depending on the new sensor data, the sensor fusion will publish the new fused and filtered estimation data to the corresponding topic. New UWB position data results in only new filter position data being published, new ORPE data results in both position and orientation filter data being published. This way the sensor fusion thread only runs at the same time as new sensor data is produced, but does not block the publish functions with long calculation algorithms. After the new data is processed, the filter thread suspends until is it resumed by new sensor data once again.

7 Conclusion and Future Work

We can conclude from section 3 that the optical based pose estimation is ready for the Tamariw mission due to the changes and improvements and capable of estimating the relative position and orientation between two satellites for distances from 15 centimeters up to 10 meters in any environment with a 0.2% relative accuracy and angles of up to 60 Degrees. Although it is highly recommended that the docking surface have some type of non reflective cover or coating to improve reliability in cases where the sun shines onto the docking port of the opposite satellite. In section 4 we can conclude that the datalink datarate from one STM32F4 to the opposite satellite STM32F4 is capable of up to $5000\frac{Byte}{s}$ with a latency on average of 33 milliseconds and up to 200 milliseconds with packet loss of up to 5.5%. Where as the datalink between the STM32F4 and RPI Zero 2 within one satellite has a 0% packet loss. The WiFi based section of the datalink between satellites in worse case has a maximum range of 20 meters, but should be tested in the future to find the range in a best or at least more realistic case. Finally we can conclude from section 5 that the sensor fusion is capable of fusing the two input sensor systems from the UWB radio based position estimation and optically based position and orientation estimation system into a single estimation for position and orientation that features reduced noise and improved characteristics as the two sensor systems are combined in a complementary process. However, it is recommended as discussed in section 5.1.2 to use the opposing satellite optically based estimator as a third input for the filter. This will improve redundancy especially in cases where one satellite has the Earth or sun in view or the docking surface of the opposite satellite in sunlight and the optical system has difficulty estimating the pose. Finally, the filter should be tested with real world data from the UWB radio based sensor system.

References

[1] Contours: Getting started. https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html, 2024.

[2] Image thresholding. https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html, 2024.

[3] Libcamera bindings for opencv. https://github.com/kbarni/LCCV, 2024.

[4] Networkmanager. https://wiki.archlinux.org/title/NetworkManager, 2024.

[5] Perspective-n-point (pnp) pose computation. https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html, 2024.

[6] Structural analysis and shape descriptors. https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#gac2718a64ade63475425558aa669a943a, 2024.

[7] Use your raspberry pi as a network bridge. https://www.raspberrypi.com/documentation/computers/configuration.html#use-your-raspberry-pi-as-a-network-bridge, 2024.

[8] Sergio Montenegro and Frank Dannemann. Rodos - real time kernel design for dependability. *DASIA 2009 - Data Systems in Aerospace*, 669:66, 2009.

[9] Atheel Redah, Saurav Paudel, Milan Pathak, Chaitanya Athawle, Om Sai Swaroop Mopidevi, Felix Sittner, and Sergio Montenegro. Design and Development of an Active Magnetic Docking System for Small Satellites. Small Satellite Conference, Utah, United States, August 2024.

[10] Christopher Steffen. LED Markers and Cameras for Satellite Optical Relative Pose Estimation. Bachelor thesis, Julius-Maximilians-Universitat Würzburg, 2022.