Project no. VI20172020068

Tools and Methods for Video and Image Processing to Improve Effectivity of Rescue and Security Services Operations (VRASSEO)

Drone sensory data processing for advanced drone control for augmented reality.

Technical report

Alfredo Chávez Plascencia, Vítězslav Beran, Kamil Sedlmajer

Brno University of Technology

Faculty of Information Technology

Božetěchova 1

Brno, 612 66, Czech Republic

December 2019

Abstract

In order to improve the navigation and control abilities of a drone's operator, sensory data information from a StereoLabs ZED camera is collected over an outdoor environment to implement the Simultaneous Localization and Mapping method. Moreover, a PointCloud and/or octree is used to represent the generated 3D map. Also, transmission of the data from a JetsonTX2 companion computer mounted on a drone to a ground control station is analyzed. Furthermore, a methodology of the data collection-transmission is presented.

1 Introduction

During the last years, a great development in robotics has been achieved specially in unnamed aerial vehicles (UAV). This emerged field has impacted the field of science and also has influenced human life. However, due to achievements in more sophisticated sensors and development of more reliable and robust algorithms, these robotic systems are able to do specific tasks. For instance, [1] uses a brain computer interface (BCI) to control a drone. Moreover, work to control a drone with a gaze is carried out in [2]. On the other hand, full AUV outdoor autonomy [3] is a more sophisticated way to control a drone, because the drone has to deal with the interaction and coordination of the different modules of autonomy, like: sensor fusion, control, path planning, obstacle avoidance, 3D mapping, among others. In this taxonomy, for instance, 3D dynamic maps are crucial for a quad-rotor 3D path planning in an free space [4]. As a practical application we have avoidance[1] which is a Robot Operating System (ROS) package that allows the drone to work as global planner mode. In this mode, a 3D octree map for path planning and obstacle avoidance is used. An alternate way of controlling an AUV is by means of augmented reality (AR) [5]. The work done in [6] provides the operator to use stereo vision system with first person view (FPV) trough AR.

Some work using AR-FPV and third person view (TPV) in UAV has been done in [7] where the user is able to switch from FPV to TPV. This work fuses topography, elevation and also 3D building maps with a video stream from a stereo vision source to allow the user to see and control the UAV remotely from the ground control station (GCS). In TPV mode, the user overcomes the limitations of the camera field of view (FoV) meaning that the user has better visibility of the UAV's surroundings. However, the work is lacking the use of 3D stereo vision PointCloud and/or octree data that could strength the visibility of the UAV. In other words, it would be worth to explore the fused of 3D PointClouds and/or octree into the AR-FPV-TPV.

To this end, one of the duties of the on-board is to process complex algorithms like the Simultaneous Localization and Mapping (SLAM) which is an algorithm for localization and map making of 2D and 3D indoor and outdoor environments respectively [8]. In the literature, there are algorithms to solve the SLAM chicken-egg problem. For instance; RTAB-Map [9], large scale direct monocular SLAM (LSD-SLAM) [10] and ORB-SLAM [11]. A comparison method between LSD-SLAM and LSD-SLAM is carried out in [12] showing that LSD-SLAM is more suitable for robot navigation. However the RTAB-Map has better support for ROS[2] and for this reason there is an ease in the integration of the current application.

When the processing power of UAVs is limited, the data, specially images, is processed offline in the GCS. On the other hand, when the UAV has enough processing power the data can be processed on-board. This ensures that the data can be analyzed in real time. Since one of the duties of the on-board is to process SLAM, a question arises, is the on-board device good enough for the task at hand? According to [3] and [13], the Nvidia Jetson TX2[4] (NJTX2) is useful for deploying computer vision algorithms and deep learning.

[1] https://github.com/PX4/avoidance
[2] `http://wiki.ros.org/rtabmap_ros`
[3] `https://elinux.org/Jetson_TX2`
[4] `https://developer.nvidia.com/embedded/jetson-tx2`

The goal of the present work is to implement a 3D PointCloud and/or octree SLAM-rtabmap_ros[5] map of some outdoor environment that could be fused with AR-FPV-TPV to increase the abilities of an operator when flying and controlling a drone . For that, the collected stereo vision system sensory data information is processed on the on-board companion computer placed on a drone. Moreover, the data shall be effectively transmitted over the wireless network to a GCS to be used by the AR-FPV-TPV work done in [7]. In the GCS side, Unity[6] has been used as a develop tool to show the AR-FPV-TPV application. For that purpose a ROS-point_cloud/octree-Unity application is also considered to be implemented

The equipped black sheep-discovery (TBSD) experimental flying platform[7] has been used to carry out the experiments.

2 System Hardware

During the functioning of the TBSD, the PixHawk® mini[8] failed, in other words it stopped working completely, causing the RC communication transmitter-receiver fail. The previous issue brought the team to replace the FCU.

2.1 FCU replacement

The PixHawk® 4 mini[9] was the chosen FCU to be replaced due to that PixHawk® mini was already discontinued. The kit set mainly comes with the following devices.

- PixHawk® 4 mini.
- Power magnament board (PDB)
- GPS
- Cables.

2.2 PixHawk® 4 Mini

PixHawk® is an independent hardware for open source autopilots that provides a rapid implementation for a hiqh-quality and low-cost autopilot hardware designs for the academic, hobby and developer communities. PixHawk® supports multiple flight stacks, like PX4® & ArduPilot®[10].

The flight stack is a collection of guidance, navigation and control algorithms for autonomous drones. It includes controllers for fixed wing, multirotor as well as estimators for attitude and position[11].

Figure 1 shows the PixHawk® 4 mini board whereas the Table 1 shows the technical specifications.

[5] http://wiki.ros.org/rtabmap_ros
[6] https://unity.com/
[7] https://github.com/NVIDIA-AI-IOT/redtail/wiki/Skypad-TBS-Discovery-Setup#assembly-2-jetson--j120-connections
[8] https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk_mini.html
[9] https://docs.px4.io/en/flight_controller/pixhawk4_mini.html
[10] http://pixhawk.org/
[11] https://dev.px4.io/en/concept/architecture

Figure 1: The PixHawk® 4 mini.

Manufacturer	Holybro® and Auterion®
Model	PixHawk® 4 mini
Main Processor	STM32F765 (32 Bit Arm® Cortex®-M7, 216MHz, 2MB memory, 512KB RAM)
Accel/Gyro/Mag	ICM20689
Barometer	MS5611
USB Power Input	4.75~5.25VDC
Weight	27.2 g
Size	38 × 55 × 15.5 mm
Interfaces	∗ 8 PWM servo outputs ∗ 4 dedicated PWM/Capture outputs ∗ Dedicated R/C input for CPPM ∗ 2 I2C ports ∗ 1 CANBuses for CAN ESC ∗ 3 SPI buses ∗ Micro USB Port

Table 1: Technical specifications.

2.2.1 GPS-Compass

The GPS-compass module as depicted in Figure 2 comes with the u-blox M8N, which is a set of positioning modules that can singly receive and track the following systems:

1. Global position system (GPS) which is owned by the United States government and operated by the United States Air Force.
2. Global navigation satellite system (GLONASS) which is operated by Russia and has a global coverage with a comparable precision.

Table 2 shows the technical specifications.

Figure 2: GPS-compass.

Manufacturer	Holybro®
Model	Pixhawk® 4 GPS
Module	u-blox M8N
Weight	32g
Diameter	50mm

Table 2: Technical specifications.

2.2.2 PDB Board

The PDB is a regulator board that supplies VDC to the PixHawk® 4 mini as shown in Figure 3 and the table 3 shows the technical specifications.

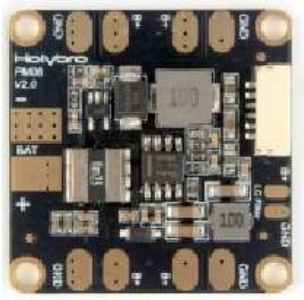


Figure 3: PDB-board.

Manufacturer	Holybro®
Model	PM06 V2 Power Module
PCB Current	120A
UBEC Current	3A Max
Input voltage	7 to 42 VDC
Output voltage	5.1 to 5.3 VDC
Weight	7g
Size	35 × 35 × 5 mm

Table 3: Technical specifications.

2.2.3 TBSD

Figure 4 shows the TBSD with the PixHawk® 4 mini kit set.

Figure 4: The TBSD with PixHawk® 4 mini set.

3 System Architecture

The system architecture with the new FCU is depicted in Figure 5.

1. The FCU unit which is an independent hardware for open source autopilots that provides a rapid implementation for a high-quality and low-cost autopilot hardware designs for the academic, hobby and developer communities. PixHawk® supports the flight command control software stacks PX4.

2. PDB is a regulator board that supplies VDC to the PixHawk® 4 mini.

3. The GPS-compass module that comes with the u-blox M8N, which is a set of positioning modules that can singly receive and track the following systems:

   (a) Global position system (GPS) which is owned by the United States government and operated by the United States Air Force.

   (b) Global navigation satellite system (GLONASS) which is operated by Russia and has a global coverage with a comparable precision.

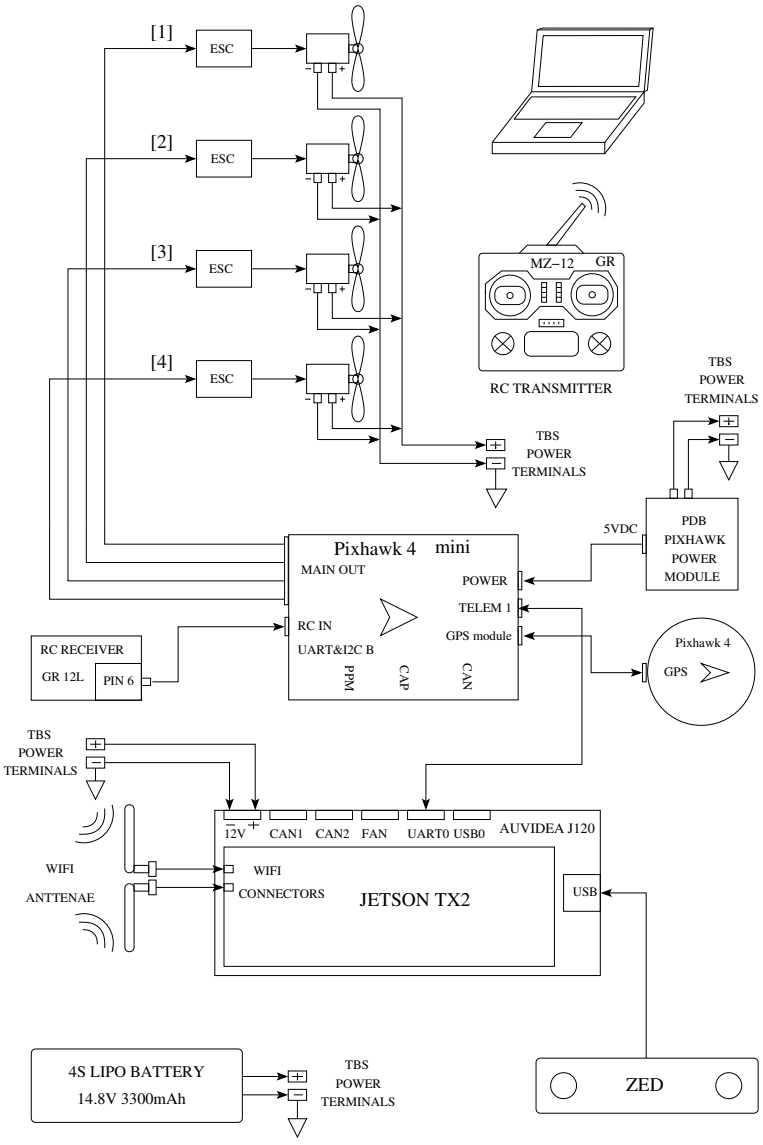


Figure 5: Devices of the architecture.

The architecture shown in Figure 6 consists of three levels:

1. The low level architecture (A_{LL}) is in charge of receiving the speed reference motor signals from the PixHawk® and make the motors to follow them. The A_{LL} includes the motors over the ESCs and the interface with PixHawk®.

2. The middle level architecture (A_{LM}) has the task to interpret the control commands and transform them to velocity references. The A_{LM} consists of accelerometer, gyroscope, IMU, GPS, NJTX2 and the interface with the PixHawk®.

3. The high level architecture (A_{LH}) deals with the generation control commands for different flight modes by means of QGCS. Also, the A_{LH} consists of the Unity software that communicates with the drone over a rosbridge.

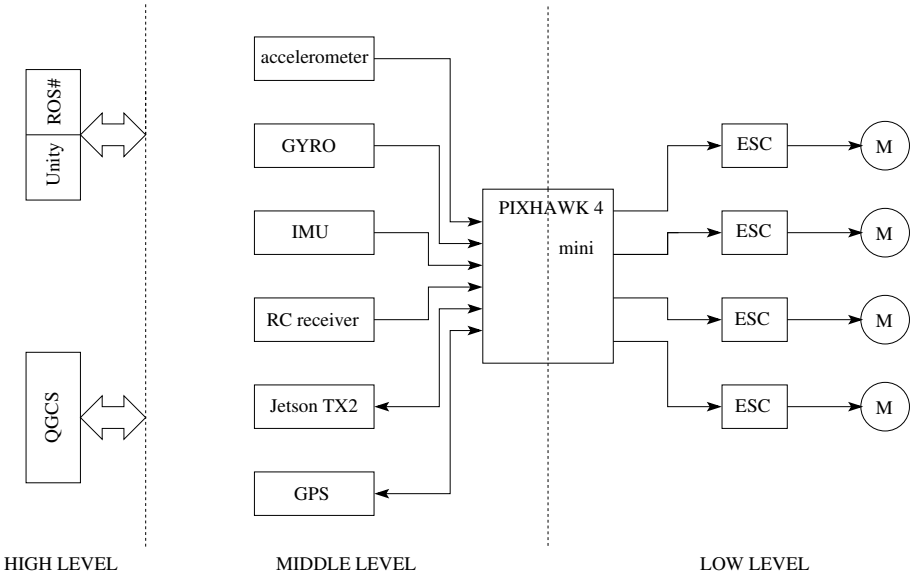


Figure 6: Levels of the architecture.

4 System Software

The communication between the different devices is done using the PX4 software flight stack[12] and the mavros ROS package[13]. In other words, the communication and the actions between QGCS and TBSD are done over the PixHawk® that uses the PX4 software which in turn mainly contains:

1. The flight control software, e.g. position controller, attitude estimator, autonomous flight, output driver for PWM.

2. Drivers, e.g. camera control, GPS, IMU, RC input and gimbal drivers.

More specifically, on the drone side, the PX4 uses the micro air vehicle link (MavLink)[14] which is a very lightweight message marshalling protocol, which is used to communicate the drone with the QGCS over the IEEE-802.11 wireless local area network (WLAN) at a frequency of 2.4 GHz and a bit rate of 144.4 Mb/s. Moreover, a rosbridge is used in order to be able to communicate the drone with Unity which is a non ROS-program. The SDK-ZED from StereoLabs is the official driver for the ZED camera, nonetheless a wrapper called rtabmap_ros[15] is used to communicate the ZED camera with ROS. Also, The RTAB-Map RGBD SLAM algorithm as mentioned previously has been chosen due to it has a full ROS support over the rtabmap_ros[16]. Figure 7 shows the software system.

[12] `https://dev.px4.io/en/concept/architecture.html`
[13] `http://wiki.ros.org/mavros`
[14] `https://mavlink.io/en/`
[15] `http://wiki.ros.org/rtabmap_ros`
[16] `http://wiki.ros.org/rtabmap_ros`

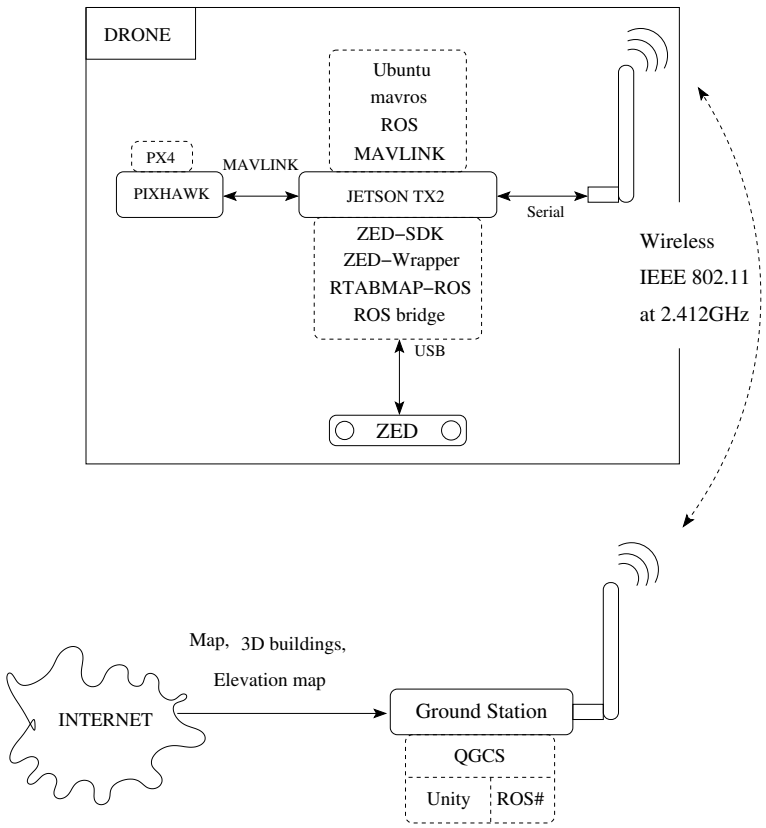


Figure 7: Software communication.

5 Methodology

In the following the main parts of the methodology are addressed:

- **Environment** is where the robot has to navigate to collect data.
- **Data** is where sensors are used in order for the robot to interact with the environment.
- **RTAB-Map-SLAM** is the algorithm that uses the data collected from vision sensor to construct a map of the environment.
- **PointCloud/octree** is the type of data the map is represented.
- **Data transmission** is the transmission of the data over the wireless network.
- **GCS** is where the data is acquire from the transmission.
- **QGCS** is a software that can handle different flight modes on the drone over a wireless network.
- **Unity** is program where an application can be created to handle the AR-FPV-TPV modes.

Figure 8 shows the methodology of the architecture.

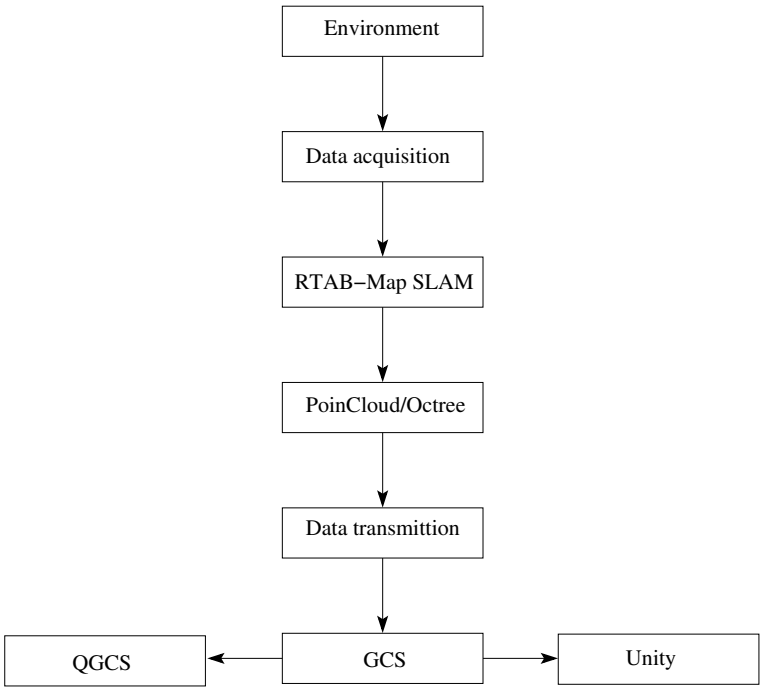


Figure 8: Methodology of the architecture.

5.1 SLAM

As mentioned previously in section 1, ORB-SLAM and LSD-SLAM have been evaluated in [12] using data set. The result of the evaluation have shown that ORB-SLAM is more accurate in rate frame performance than LSD-SLAM. In the other hand, the evaluation had shown that LSD-SLAM is much more suitable as an input for robotic navigation and path planning because the maps are more dense. Also, in that work the experiments are lacking the use of more realistic scenarios with a ground truth included.

Moreover, the RTAB-Map, as a suite, works with a myriad of sensors, has excellent ros integration and a UI application that will help to learn about how mapping works and all of the different SLAM components work together[17].

According to [9], RTAB-Map is a graph based SLAM approach that has been evolved into a cross-platform stand alone C++ library and also integrated in ROS as the rtabmap_ros package and driven by practical requirements such as:

- Online processing.
- Robust and low-drift odometry.
- Robust localization.
- Practical map generation and exploitation.
- Multi-session mapping (a.k.a. kidnapped robot problem or initial state problem).

The following list provides a summary of the open-source ROS-compatible SLAM approaches in relation to their inputs and outputs.

- Stereo
- RGB-D
- Multi
- 2D Lidar
- 3D Lidar
- Odometry
- Pose
- Occupancy
- PointCloud

The odometry is an external input to RTAB-Map, which means that SLAM can also be done using any kind of odometry to use what is appropriate for a given application and robot. The odometry node can implement any kind of odometry approaches from simpler ones derived from wheel encoders and IMU to more complex ones using camera and lidar. Also, RTAB-Map has evolved to handle the following practical requirements; Visual SLAM approaches integrated in ROS and proper transform (tf) library.

5.2 Augmented Reality

Augmented reality[18] (AR) is a technology that allows to add digital elements into the actual environment. Moreover, AR is an interactive experience between real world environment and a world enhance by computer generated perceptual information such as visual, auditory among others. Applications of AR in drones can be found in the literature [5, 6, 14, 7]. Broadly speaking, the work done in [7] mainly consists of the user be able to change from FPV to TPV and vice versa interactively in which video stream and the flight data are integrated into a 3D model map that consists of external sources like topography, elevation and 3D buildings.

[17] https://www.reddit.com/r/computervision/comments/62aofs/which_slam_project_to_use/
[18] https://en.wikipedia.org/wiki/Augmented_reality

The aim of the work done in [7] is to improve the orientation of the pilot and also to reduce his mental load during the drone remote control. Figure 9 shows the drone AR in which by scrolling up or down the mouse wheel the user can zoom in or out the drone FoV or change from FPV to TPV modes respectively.

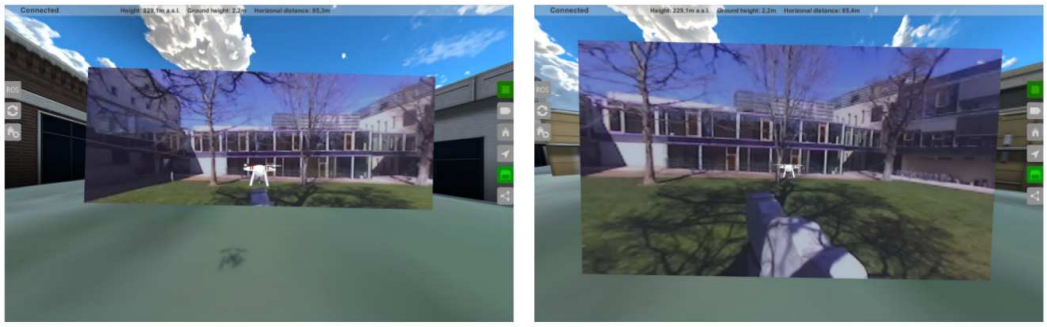


Figure 9: The user can zoom in or out the FoV of the drone by scrolling up or down the mouse wheel.

With regards to AR in the present work, the aim is to enhance the work carried out in [7] by fusing 3D PointCloud/octree map into the FoV of the drone.

5.3 Data Transmittion

A reliable communication between the UAV and the GCS is necessary. Thereby, there are wireless technologies for communication between UAV and GSC such as IEEE 802.15.4, IEEE 802.11x, 3G/LTE and infrared. Some real world tests experiments have been carried out in [15] where it is concluded that IEEE 802.11 technologies are commonly used for small-scale UAVs to enable connectivity due to their wide availability in current networking devices, high performance links, and their suitability. For instance, a data-driven 3D augmented reality approach has been propose in [16]. The main part of this architecture is that it uses two data-links for transmission, the first link is a wireless data transmission and uses the protocol MAVLink. The second data link works at the frequency of 5.8GHz and uses one TS832 transmitter for each stereo vision camera.

5.4 ROS communication

Moreover, in our case, ROS offers a communication between server and client called Master-Slave. In this setup, the NJTX2 has been configured as the master and the GCS as the slave[19]. The setting up of the master-slave consists of the following steps.

1. In the NJTX2 (Master);
   -In a command line open the host file; sudo gedit /etc/hosts
   -Write the following in the host file; `10.42.0.253 acp` ;
   where; 10.42.0.253 is the IP and acp is the hostname of the slave.

2. In the GCS (Slave);
   -In a command line open the host file; sudo gedit /etc/hosts
   -Write the following in the host file; `10.42.0.1 tegra-ubuntu`
   where; 10.42.0.1 is the IP and tegra-ubuntu is the hostname of the master.

[19] `http://wiki.ros.org/ROS/Tutorials/MultipleMachines`

3. The command `ROS_MASTER_URI="http://tegra-ubuntu:11311"` shall be inserted in the last line of the file `gedit ~/.bashrc` in the slave.
4. In a command terminal on the slave side; `ssh nvidia@10.42.0.1`
5. The TBSD is ready to start scanning the environment.

To stablish the comunication over a rosbridge_websocket between the TBSD and the GCS, a roslaunch file called `zed_filter_depth_unity.launch` has been configured. The file mainly sets up and launch the `rosbridge_websocket` `zed_wrapper`, `rtabmap_ros`, `mavros` and `octomap_server` node packages respectively. The current state of the GCS unity-application set up as proposed in [7] only requires the following topics from the previous mentioned nodes: `/zed/right/image_rect_color/compressed` `/mavros/local_position/pose`, `/mavros/imu/data`, `/mavros/local_position/compass_hdg`, `/mavros/global_position/global` as they are depicted in Figure 10.

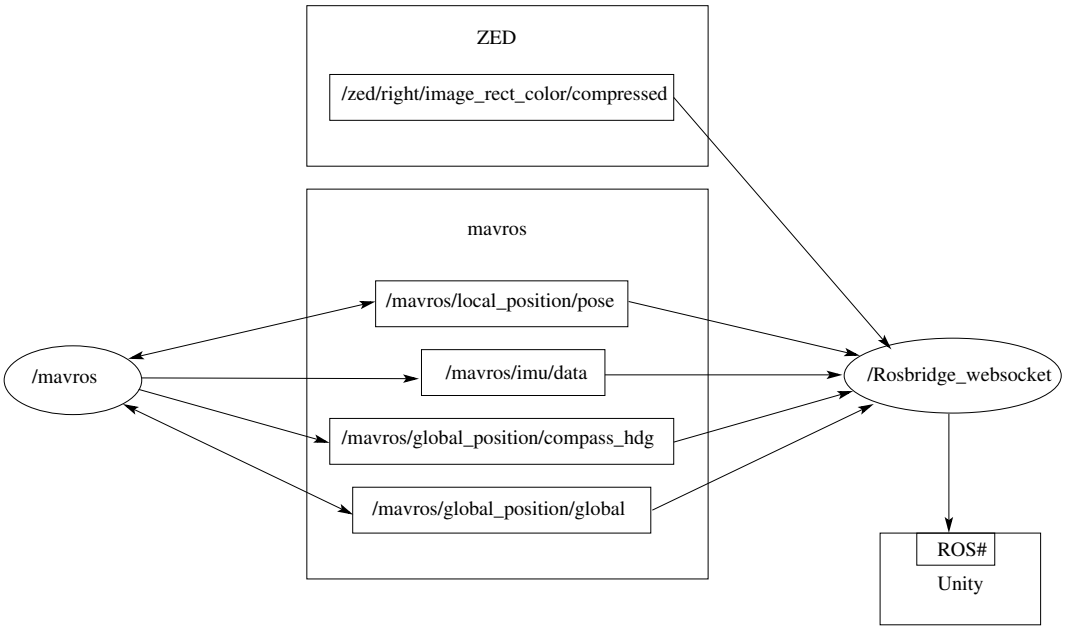


Figure 10: Shows the topics and nodes that the GCS unity-application need.

In the other hand, Nimbro_network[20] (NN) is a ROS stack which offers a robust transport of ROS topics and services for high-latency, low quality networks. Some of the advantages of using NN over the ROS-network[21] (ROSN) is stated as follows:

1. NN offers the same functions as the ROS-network transparency layer.
2. The NN overcomes the lengthy TCP handshake that is needed for a ROSN subscription.
3. ROSN has no compression while NN has.
4. ROSN needs several handshakes for every call while NN does not.
5. ROSN messages are transmitted at the rate which they are published where us NN is not.

All the previous features make NN a a good alternative to ROSN.

[20] `https://github.com/AIS-Bonn/nimbro_network`
[21] `http://wiki.ros.org/ROS/Tutorials/MultipleMachines`

6 Results and Experiments

In order to test the methodology stated in section 5. Firstly, a ros master-slave has been configured at the drone-GCS respectively. Secondly, the roslaunch file `zed_filter_depth_unity.launch` is launched.

Then, the drone is placed outdoor at the inner yard of the FIT faculty, where the flight has been simulated by taking the drone by hand and moved it around the yard mainly following the walls of the building of the faculty. During the moving, the drone is collecting the video data where the walls, the trees and the floor around them have been well detected. However, when trying to build up the 3D map of the area, two problems showed up:

1. The algorithm detects multiple walls.
2. There is a bending of the map.

6.0.1 Multiple wall problem

According to StereoLabs[22], the ZED camera uses stereo vision to estimate the depth. This technique detects details of objects in the left and right images to compare their position to estimate their depth. Homogeneous surfaces are texture-less so the ZED driver depth algorithm tries to estimate them based on nearby objects. As this is less precise, the depth can vary in these areas which result in duplicated walls in the scan. The ZED SDK has no option to filter this out. However, to solve this issue a filter may be created on the depth map/point_cloud to remove the pixels in the homogeneous zones. In other words, a ROS node can be written that takes the depth and the left RGB image from the ZED ROS Wrapper and outputs a filtered depth. This node invalidates the fake walls on the depth image that are sent to RTABmap, that's how the results shall be improved. The node is a kind of filter between the ZED-wrapper and the RTABmap nodes. The RTABmap ROS node must be configured to receive the new filtered depth images published by the node instead of subscribing to the depth topic of the ZED node.

The basic idea to implement the node will be to use a very sensible edge detection, so most of the color variations in the image are considered as an edge. Then, a dilate/erode is applied to make the edges bigger in the mask, covering the major part of it. The remaining black areas in the image are the ones without any details, the homogeneous ones. The next step is to take the depth image and modify it so that all the black pixels in the edge-mask are set to NAN in the depth map. Then the depth map is re-publish on another topic and having a filtered depth ready to be used in ratabmap_ros. Figure 11 shows the previous implementation.

[22] `https://support.stereolabs.com/hc/en-us`

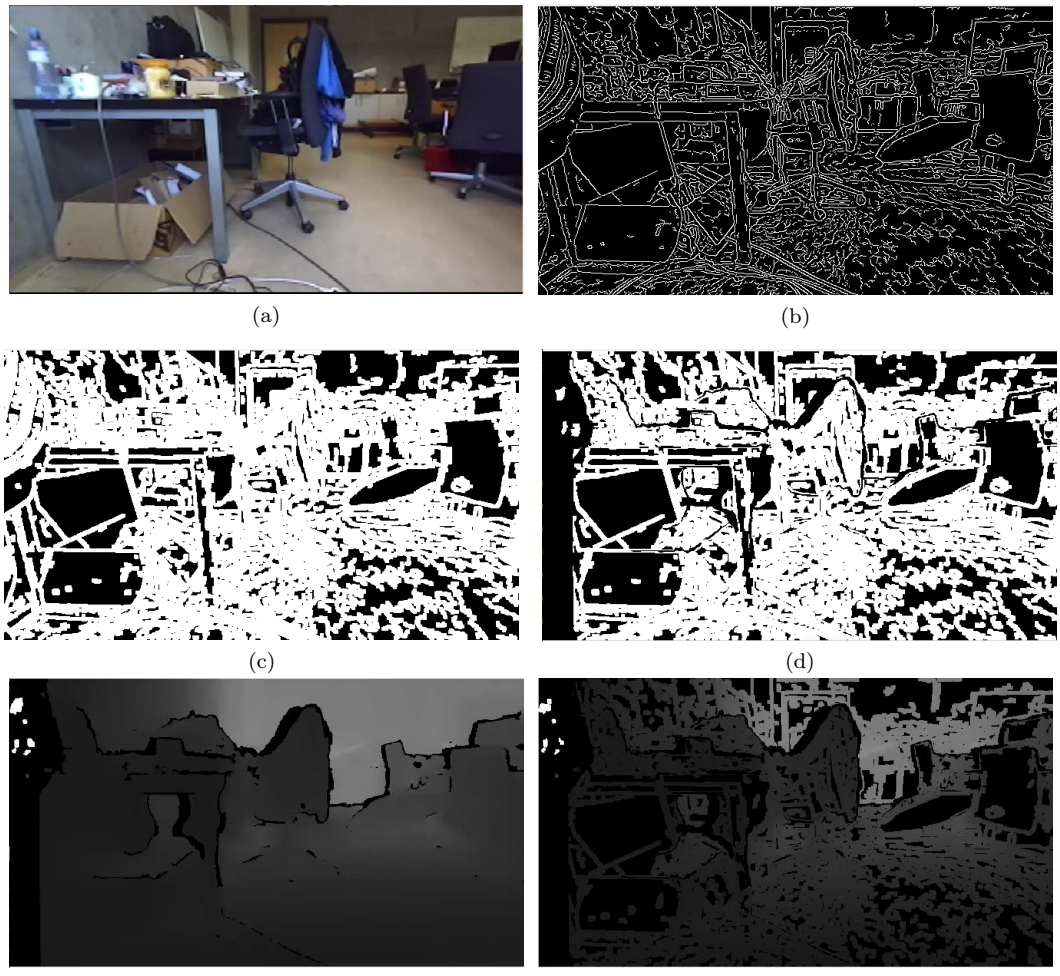
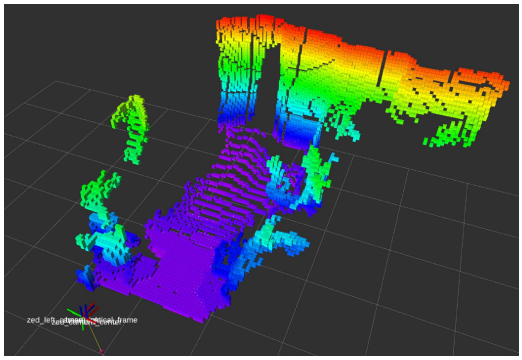


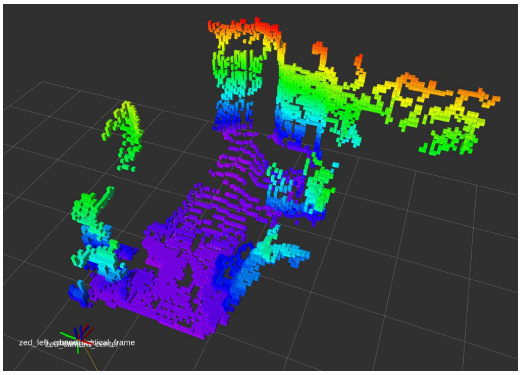
Figure 11: Figure 11a shows the RGB image whereas Figure 11b shows the canny edge detector, Figure 11c depicts the dilate image, Figure 11d presents the filtered image with NaN values, 11e is the ZED-wrapper depth image and the Figure 11f is the depth filtered image that rtabmap_ros shall subscribe.

Figure 12 shows two 3D occupied-empty octree images of a laboratory environment. Figure 12b corresponds to the no filtered image whereas Figure 12c correspond to the filtered one. The filtered image clearly shows how the texture-less areas have been removed from the rtabmap_ros algorithm.

(a)



(b)



(c)

Figure 12: a) shows the rgb image of a laboratory indoor office. b) shows a 3D octree representation of the non-filtered cloud. c) shows the filtered one.

Figure 13a shows the 3D comparison of the filtered and the non-filtered PointClouds where the texture-less areas have been disappeared and can easily be seen. As an another perspective view of the previous comparison, the Figures 13b and 13c show the euclidean distance from the camera to each single point in the clouds. The two Figures look quite similar just the filtered one shows 3526 points that are less when comparing with the non-filtered one which has 4700 points.

Also, table 4 shows the mean and variance of the euclidean PointClouds of the filtered and non-filtered images. The mean value of the non-filtered PointCloud is bigger than the mean of the filtered PointCloud, the fact of this is that the non-filtered PointCloud has more points.

Comparison		
	non-filtered	filtered
mean	5.8739	5.3978
variance	7.4583	6.9808

Table 4

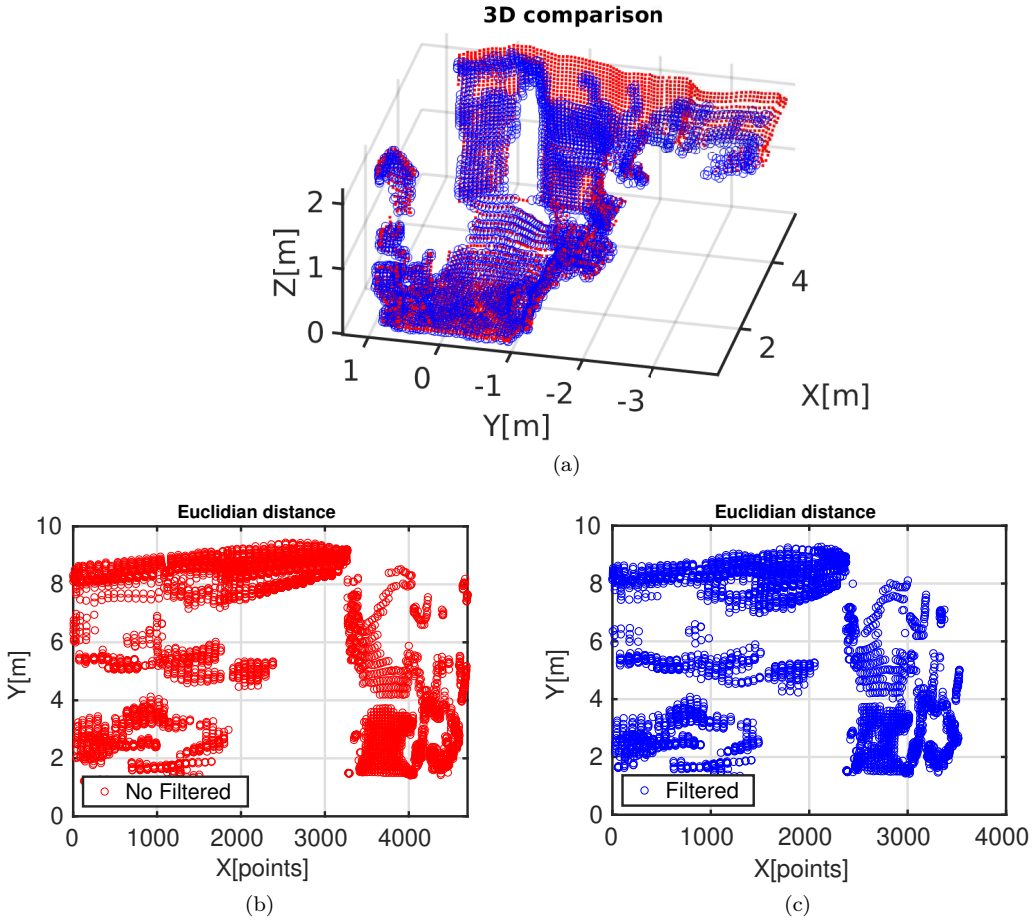


Figure 13: a) shows the 3D comparison of the filtered and the non-filtered PointClouds. b) shows the non-filtered cloud with 4700 euclidean distance points. c) shows the filtered one with 3526 points.

6.0.2 Bending problem

The bending is caused by roll/pitch errors in odometry. According to RTABmap forum[23], to reduce the bending in 6DoF mapping, injection of gravity constraints to graph are needed. This could be done in two steps. First, by inserting in rtabmap node the drone IMU's topic. Second, by making the TF between ZED and IMU accurate at least for orientation.

1. rtabmap node drone IMU's topic
The insertion of the topic is done as the following script:

```
<include file="$(find rtabmap_ros)/launch/rtabmap.launch">
   <arg name="imu_topic" value="/mavros/imu/data"/>
</include>
```

2. ZED-IMU TF
To get the TF between the IMU and the ZED frames, the following steps are carried out:

- Kalibr[24] has been used to calibrate the IMU topic `/mavros/imu/data` from the PixHawk® 4 mini with the `/stereo_camera/left/image_mono` and `/stereo_camera/right/image_mono` from the ZED camera. Then, two rotational matrices are obtained: `T_cam_imu_left` and `T_cam_imu_right`.
- Next, the `3D Rotation Converter`[25] is used to convert the `cam_imu_left` rotational matrix to the [xyzw] quaternion angles. Afterwards, the `getRPY()-ROS` function is used to obtain the roll, pitch, yaw angles: `Roll: -3.00242`, `Pitch: -1.34878`, `Yaw: -1.58442`. Subsequently, these angles are inserted into the `zed.urdf` file as it can be seen in the following. |

  ```
  <joint name="zed_base_link_camera_center_joint" type="fixed">
     <parent link="zed_left_camera_optical_frame"/>
     <child link="base_link"/>
     <origin xyz="0 0 0" rpy="-3.00242 -1.34878 -1.58442 " />
  </joint>
  ```

The Figure 14 shows the transformation between the base_link and camera_center links.

Figures 15 and 16 show the multiple wall and the bending respectively.

Figure 17 shows the 3D PointCloud map where the bending and multiple wall has been reduced.

[23] http://official-rtab-map-forum.67519.x6.nabble.com/
[24] https://github.com/ethz-asl/kalibr/wiki
[25] https://www.andre-gaschler.com/rotationconverter/

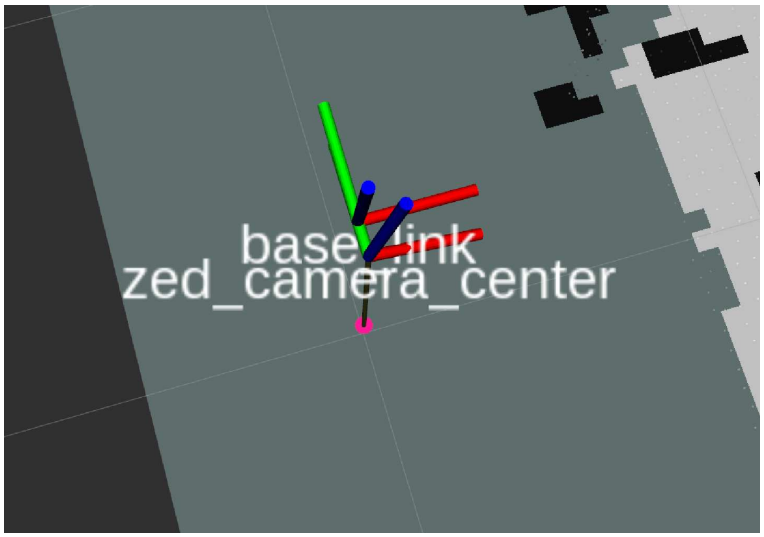


Figure 14: The figure shows two frames: the IMU base_link and the camera_center link.

Figure 15: The blue line shows the traveled path, where multiple walls are detected.

Figure 16: The map is bent

Figure 17: 3D PointCloud of the Yard.

7 Conclusions and Future Work

An experimental flying platform has been equipped with sensors and software to carry out manual and autonomous modes. The manual mode was handled by using a radio control (RC) transmitter where the drone responded satisfactorily to the commands of take off, land, throttle, roll and pitch.

In this work two simulations were carried out:

- The rtabmap_ros SLAM package was tested by creating first a roslaunch file with the default values. The algorithm behaved acceptable just two issues were found, multiple walls and bending of the floor. The multiple walls issue was tackled by filtering out the texture-less surfaces. And, the bending issue was tackled by calibrating the PixHawk® 4 mini IMU topic with the ZED camera. The results showed an improvement in the bending and multiple walls.
- The experimental platform has used to test the AR in the modes of FPV and TPV respectively. The matter wont continue here just to say that for further information the reader may refer to [7].

The future work is to implement a rtabmap_ros-Unity application with the purpose of integrating the PointCloud/octree 3D map into the AR work done in [7].

References

[1] S. Rosca., M. Leba., A. Ionica., and O. Gamulescu., "Quadcopter control using a BCI," *IOP Conference*, vol. 294, p. 012048, January 2018.

[2] J. P. Hansen, A. Alapetite, S. MacKenzie, and E. Moellenbach, "The use of gaze to control drones," *ETRA, Symposium on Eye Tracking Research & Applications*, March 2014.

[3] S. Kawabata, K. Nohara, J. H. Lee, H. Suzuki, T. Takiguchi, O. S. Park, and S. Okamoto, "Autonomous flight drone with depth camera for inspection taskof infra structure," *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 2, March 2018.

[4] J. L. S. Lopez, M. Wang, M. O. Mendez, M. Molina, and H. Voos, "A real time 3d path planning solution for collision-free navigation of multirotor aerial robots in dynamic environments," *Journal of Intelligent & Robotic Systems*, pp. 33–53, 2019.

[5] A. Hitchcock and K. Sung, "Multi-view augmented reality with a drone," *The 24th ACM Symposium*, November 2018.

[6] N. Smolyanskiy and M. G. Franco, "Stereoscopic first person view system for drone navigation," *Frontiers in Robotics and AI*, vol. 4, March 2017.

[7] K. Sedlmajer, D. Bambušek, and V. Beran, "Effective remote drone control using augmented virtuality," *Third International Conference on Computer-Human Interaction Research and Applications*, September 2019.

[8] H. D. Whyte and T. Bailey, "Simultaneous localization and mapping: Part i," in *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, 2006, pp. 07–10.

[9] M. Labbé and F. Michaud, "RTABMAP as an open-source lidar and visual SLAM library for large scale and long term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[10] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *pringer International Publishing*, 2014, pp. 834–849.

[11] R. M. Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular slam system," vol. 31, no. 5, pp. 1147–1163, 2015.

[12] V. Beran, M. Šolony, A. Herout, V. Bartl, P. Zemčík, S. Nosko, J. Zendulka, V. Bartík, and V. Fröml, "Selected Video-Processing Methods and System Architecture Design," Brno University of Technology, Tech. Rep., 12 2017.

[13] D. Hulens, J. Verbeke, and T. Goedemé, "How to choose the best embedded processing platform for on-board uav image processing?" *Computer Science*, 2015.

[14] G. L. Calhoun, M. H. Draper, M. F. Abernathy, F. Delgado, and M. Patzek, "Synthetic vision system for improving unmanned aerial vehicle operator situation awareness," *In Proceedings of SPIE - The International Society for Optical Engineering*, vol. 5802, pp. 219–230, May 2005.

[15] S. Hayat, E. Yanmaz, and R. Muzaffar, "Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2624–2661, April 2016.

[16] X. Ji, X. Xiang, and T. Hu, "Data-driven augmented reality display and operations for UAV ground stations," *IEEE 6th Data Driven Control and Learning Systems Conference*, May 2017.