BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Project no. VI20172020068

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

# HDR and 3D Occupancy map implementation on Embedded HW

# Technical Report

Alfredo Chávez Plascencia, Svetozár Nosko, Vítězslav Beran, Pavel Zemčík

Brno University of Technology

Faculty of Information Technology

Božetěchova 1

Brno, 612 66, Czech Republic

December 2020

## Abstract

This technical report describes the implementation of selected methods for sensory data processing on embedded HW. Outdoor 3D environment modeling provides volumetric space information that is important in solving some robotic tasks, including unmanned areal vehicles with remote control but with limited visual connection. To this end, accurate 3D occupancy maps and reliable flight control are key to the task. This report represents the generation of a 3D occupancy map from a 3D point cloud and its realization on embedded HW. In addition, the ZED2 stereoscopic camera is set up and tested to generate a 3D point clouds. Finally, we expanded the HDR camera (High Dynamic Range) with a new HDH deghosting algorithm, which enabled the integration of HDR shooting directly on the drone. This algorithm reduces the adverse effects that can occur during the flight.

# Contents

## 1 Introduction

During the last years, robotic applications that require 3D models of the environment have been increasing. These include, airbone underwater, mobile robots, Unnamed Aerial Vehicles (UAV), among others. Also, several approaches for modelling $3D$ environments have been proposed [1], namely PointClouds, elevation, multi-level surface and octree maps. In this context, PointClouds have been generated using different Simultaneous Localization and Mapping (SLAM) approaches [2]. Thus, one of the drawbacks of the PointClouds is the increase of memory consumption in as much as the number of measurements increases, making this approach not suitable for systems with limited memory. On the other hand, 3D OctoMap models of the environment based on octrees [1] have the advantage of having efficient and probabilistic updates while keeping the memory consumption at a minimum. Further more, OctoMap models the representation taking into account the free, occupied and unknown areas in a probabilistic manner.

One of the middlware that has been commonly used lately for robotics is the Robot Operating System (ROS) [3] that provides a broad collection of packages for sensor visualization, control, mobility, mapping, networking among others. So then, there exists a big variety of packages that with a proper set up of parameters according to the application are ready to be used. Thus, there are plenty of open source SLAM approaches that can be used in ROS, e.g. GMapping, TinySLAM, Hector SLAM, ORB-SLAM2, RGBiD-SLAM, Real Time Appearance Based on Mapping (RTABMap) and others. According to the relation of their inputs: camera (stereo, RGB-D, multi, Inertial Measurement Unit (IMU), lidar (2D, 3D), odom and their online outputs: pose, occupancy (2D, 3D), PointCloud. RTABMap, which is a graph-based SLAM, performs better [4]. Then, in order to use RTABMap into ROS context, it has been integrated as a ROS package called rtabmap_ros[1] which gives support for both visual and lidar SLAM approaches. Similarly, OctoMap has been integrated into a octomap_server[2] ROS package.

In brief, Augmented Reality (AR) overlaps virtual objects over real images of objects when seen from a smart device like a mobile phone, a tablet and a ground station. Moreover, UAV is a constant developing field where applications are used more and more in many areas, including schools, public, medicine, rescue and others. More specifically, these can go from, for instance, having a multi-view AR system that should support predefined views, e.g. first and third person, and full manipulation of the drone/camera position and viewing directions [5]. Or, future new buildings can be visualized and displayed how would they fit in the existing urban environment by means of web based on AR [3]. Also, some work using augmented reality - First Person View - Third Person View (AR-FPV-TPV) in UAV has been carried out in [6], in which the user has the possibility to have multiple-view, e.g. switching from FPV to TPV and back. What characterize this work is the fusion of different maps, such as 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 a Ground Control Station (GCS).

One of the common problems found when using 3D-OpenStreetMap[4] in AR is that they may not be precise. In other words, they may not match exactly the real buildings, houses, tress and other objects that are not present in them. Since the 3D-OpenStreetMap used in AR can be seen as an approximation to the reality, the suggestion is to insert and fuse 3D octree maps into the AR scenario. To this end, in order to strengthen the fidelity, the sight of view and the abilities of the drone operator, the goal of this report is to generate 3D octree maps that potentially can be inserted and fused into the AR-FPV-TPV.

At the time of purchasing and assembling the experimental flight platform, the ZED2 did not exist in the market. Then, Stereolabs[5] released the ZED2 camera. In brief, the ZED2 has improved the Field of View (FoV), the optics and also some sensors have been integrated to enhance the estimation of the camera, e.g. IMU, barometer and magnetometer. The ZED2 camera has been tested under rtabmap_ros to create a 3D

[1] http://wiki.ros.org/rtabmap_ros
[2] http://wiki.ros.org/octomap_server
[3] https://itechcraft.com/expertise/drone-software/drones-augmented-reality-future/
[4] https://wiki.openstreetmap.org/wiki/3D
[5] https://www.stereolabs.com/

OctoMap map.

Moreover, a High Dynamic Range Camera (HDRC) that can capture images in greater detail in dark and bright environments has been placed on the drone and connected to the NJTK2 over an Ethernet connector. An executable file is available that can run the HDRC to record the surroundings, also a ROS node that can start and stop the executable when the drone is armed and disarmed respectively has been created. The camera has been tested in a real flight showing good results. We exploited the possibilities of using HDR (High Dynamic Range) in a challenging drone environment (vibrations and a lot of movement). We have improved the HDR merging algorithm (Ghost-free HDR merging), which has improved the quality of recorded HDR video from the drone. This algorithm was published as a journal paper and also integrated into HDRC.

Additionally, the rostopic of interest, e.g. camera, battery, drone status, Global Positioning System (GPS) estimation among others have been converted into a Java Script Object Notation[6] (JSON) format which are then forward to a ViAn server over a RabbitMQ[7] (RMQ) which is an open source message broker.

## 2 System Architecture

### 2.1 Embedded Hardware Specification

This section presents the hardware components of the Team Black Sheep Discovery[8] (TBSD) which has been chosen, assembled and equipped at the FIT laboratory.

Figure 1 shows the drone's schematic hardware as well as the ground station where exteroceptive sensor measurements of the environment are interpreted by a stereo vision ZED-camera[9]. Moreover, the ZED-camera has been configured to use a Video Graphics Array (VGA)_resolution of $672 \times 376$ @30fps. Furthermore, the images from the ZED camera are forward to the Nvidia JetsonTX2[10] (NJTX2) compound computer which in turn runs the algorithms for computer vision, localization, control, 3D mapping and others. Also, a HDRC have been placed as an extra camera to capture in more detail images in dark or illuminated places. Additionally, the companion computer communicates bidirectionally with the PixHawk® 4 mini Flight Control Unit (FCU)[11]. Alike, the GPS, Wireless Fidelity (WiFi) and the Radio Controller (RC) receiver modules communicate with the FCU that contains the respective drivers for camera control, GPS, IMU, RC input and gimbal drivers. Also, the FCU software, e.g. position controller, attitude estimator, autonomous flight, output driver for Pulse Width Modulation (PWM), controls the $4\times$ motors over the Electronic Speed Control (ESC). Besides, a Lenovo ThinkPad L540 + Intel(R) Core(TM) i7-4712MQ CPU @ 2.30GHz is used as a GCS. Figure 2 shows the equipped TBSD.

### 2.2 Software Framework

This section deals with the software used by the companion computer to establish communication with all the devices mounted on the TBSD.

[6] `https://www.json.org/json-en.html`
[7] `MQhttps://www.rabbitmq.com/`
[8] `https://github.com/NVIDIA-AI-IOT/redtail/wiki/Skypad-TBS-Discovery-Setup`
[9] `https://www.stereolabs.com/zed/`
[10] `https://developer.nvidia.com/embedded/jetson-tx2`
[11] `https://docs.px4.io/v1.9.0/en/assembly/quick_start_PixHawk4_mini.html`

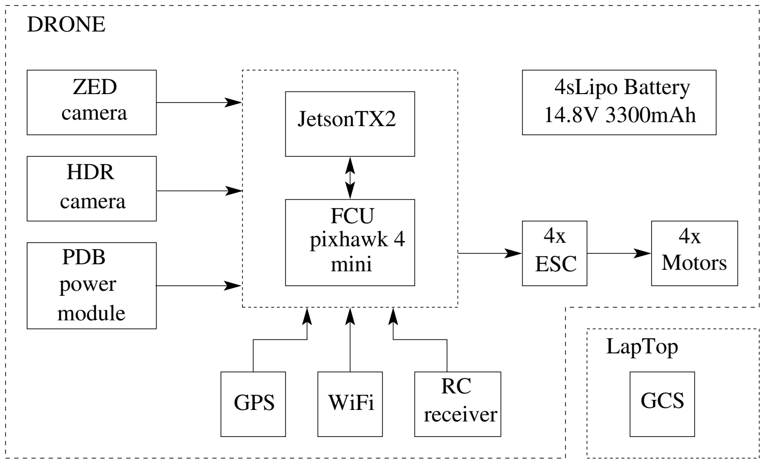


Figure 1: System harware.

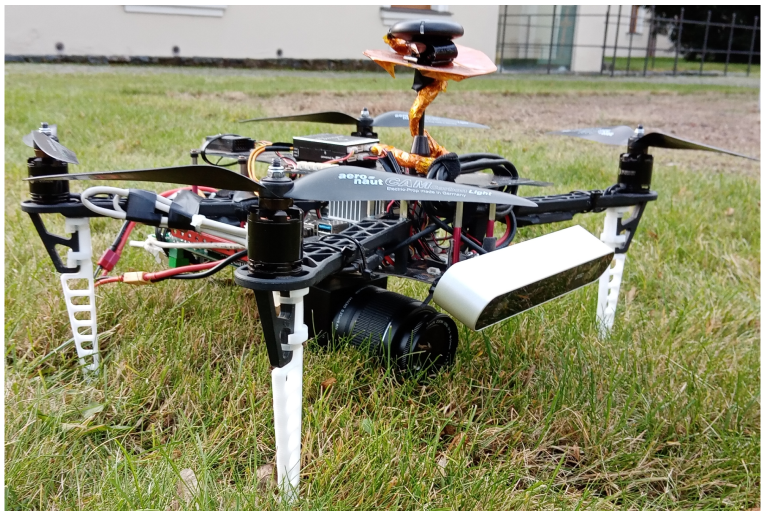
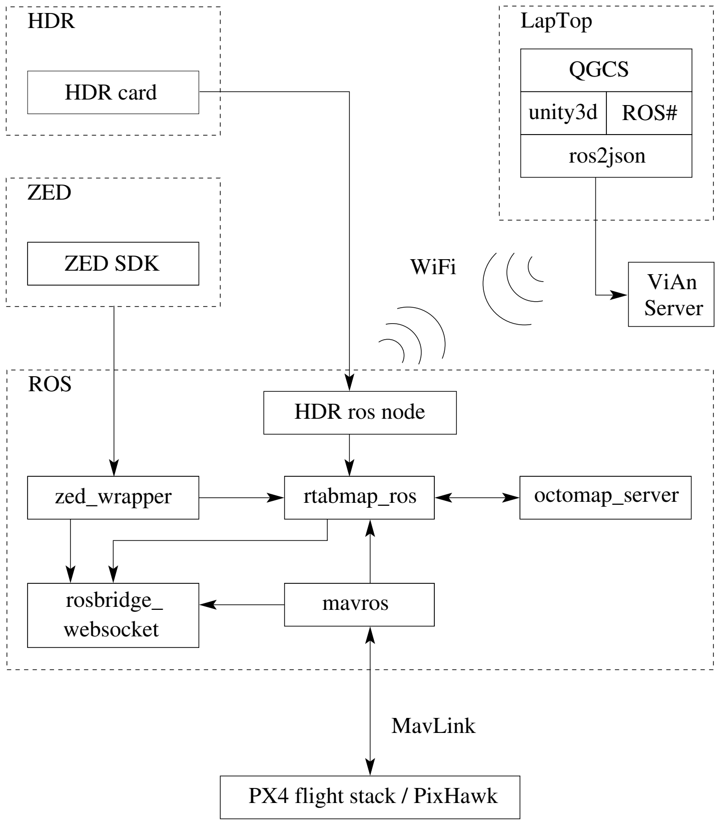


Figure 2: The equipped TBSD.

Figure 3 shows the diagram of the software architecture. In which, the NJTX2 runs Ubuntu Linux4Tegra JetPack-L4T-3.2.1 with audivea-kernel-J90-J120-V1.6 firmware and the GCS has been adopted with Ubuntu 16.04.5 LTS where ROS-Kinetic run in both of them. The PixHawk® uses the PX4 software flight stack[12] firmware to communicate with the NJTX2 via the micro air vehicle link (MavLink)[13] protocol. Moreover, communication between the drone with the GCS is handle over the IEEE-802.11 Wireless Local Area Network (WLAN) at a frequency of 5.0 GHz and a bit rate of 144.4 Mb/s. Stereo vision camera processing is handle by ZED_SDK v2.8.5[14] and CUDA 09[15]. Also, the Figure shows that ROS middlware uses the following nodes: zed_wrapper for receiving data from the ZED_SDK and forward it to another nodes, rtabmap_ros for wrapping the RTABMap and to generate 3D PointClouds of the environment, mavros for communicating with PX4, octomap_server for building volumetric 3D occupancy maps based on octrees and rosbridge_websocket for establishing bidirectional communication with Unity3d[16] that is running on the GCS.

Additionally, the laptop runs the QGroundControl station[17] (QGCS) v1.9.0 which is a Graphical User Interface (GUI) application that serves for controlling and configuring the PX4. Besides, Unity3d for Linux v.2018.3.0f2 is used to visualize the drone in AR-FPV-TPV mode and ROS#[18] which purpose is to communicate with ROS from .NET applications.



cel

Figure 3: Software architecture.

[12]https://dev.px4.io/en/concept/architecture.html
[13]https://mavlink.io/en/
[14]https://www.stereolabs.com/developers/release/2.8/
[15]https://developer.nvidia.com/cuda-90-download-archive
[16]https://unity.com/
[17]https://dev.px4.io/v1.9.0/en/qgc/
[18]https://github.com/siemens/ros-sharp

## 2.3 Sensors

### 2.3.1 ZED2

Figure 4 shows the ZED2[19] stereo camera that provides high definition 3D video and depth perception of the environment. ZED2 is an improvement of the ZED[20] camera in that, it has better optics and FoV. Furthermore, it has a precise IMU which is an electronic device that mainly contains an accelerometer and a gyroscope. The accelerometer provides information of the acceleration of the camera, in other words whether the camera is getting faster or slower in any directions with a precise value in meter per second squared (m/s$^2$). In the other hand, the gyroscope gives the angular velocity of the camera in any direction in degrees per second (deg/s). When both sensors are combined they can provide an estimation of the camera's orientation at high frequency. Moreover, the ZED camera uses the IMU information to calculate odometry estimation.

The magnetic field of the earth that surrounds the ZED2 camera is measured by the magnetometer in microteslas ($\mu$T). Thus, giving absolute orientation of the camera according to the north magnetic pole bringing a good estimation of the yaw angle. The previous sensor and other ones could be integrated by the use of kalman filter, for example the integration of the magnetometer can provide a very good estimation of the yaw of the camera.

Figure 4: ZED2 camera

| | |
|---|---|
| Manufacturer | StereoLabs |
| Model | ZED2 |
| Resolution | 4416×1242 @15fps<br>3840×1080 @30/15fps<br>2560×720 @60/30/15fps<br>1344×376 @100/60/30/15fps |
| Depth range | 0.2 to 20 m |
| Dimensions | 175 × 30 × 33 mm |
| Baseline | 120 mm |
| Interface | USB 3.0 |
| Weight | 124 g |
| Field of View | 110$^o$(H) × 70$^o$(V) × 120$^o$(D) max |
| Power | 380 mA / 5V (USB Powered) |
| Pixel size | 2 $\mu$m |
| Temperature | 0$^o$C to +45$^o$C |
| Sensors | Accelerometer Gyroscope Barometer Magnetometer |
| OS | Windows 7,8,10, Linux |

Table 1: Technical specifications.

### 2.3.2 High Dynamic Range Camera

As mentioned in section 2 a HDRC have been placed as an extra camera to capture video slightly different than normal, this camera can capture in greater detail bright and dark areas. The HDR is a custom camera

[19]https://www.stereolabs.com/assets/datasheets/zed2-camera-datasheet.pdf
[20]https://www.stereolabs.com/zed/

platform based on SoC Xilinx Zynq XC7Z020[21]. The Zynq SoC contains dual-core ARM Cortex-A9 and FPGA on the same chip. Overall power consumption of camera is 8W. Also, the platform is equipped by low noise global shutter CMOS sensor Python2000 from ON Semiconductor, connected directly to FPGA through high-speed low voltage differential lines (LVDS) interface. The CMOS has a resolution of 1920 × 1280 pixels and is capable of capturing up to 255 fps - this is the maximum speed of LVDS interface.

## 2.4 Integration to ViAn Server

The communication process between the TBSD and the ViAn server is shown in the schematic Figure 5. To ease the communication task, a rosbag was recorded with the topics of interest. During the running of the rosbag, rostopics are sent to the ros2json node which forwards the specific received topic to the sender node that is in charge of converting the ROS topics into a (JSON) format. Then, the ViAn server receives the JSON formatted topics over the RMQ listener which ensures correct data delivery to the server. RMQ is a messenger broker that implements Advance Message Queuing Protocol[22] (AMQP), it also standardizes messaging using producers, broker and consumer. This process can be thought as a post office when some one wants to deliver a mail to a recipient, then the mail is put a post box so after a period of time the mail will be delivered to the recipient. In this analogy, RMQ is a post box, a post office and a postman. In the other hand, the receiver node just checks if the transmission was successful and also save the entire formatted JSON topics into memory.

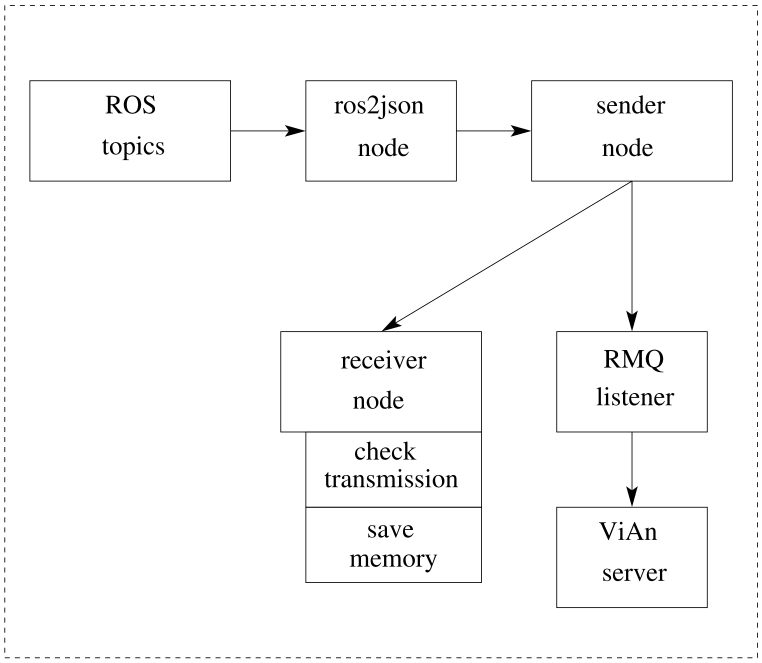


Figure 5: ViAn server schematic diagram.

[21]https://www.onsemi.com/products/sensors/image-sensors-processors/image-sensors/python2000fbclid=IwAR2Sqj_4Rpbbl8JsAg1IDsCQOn6bbX-475VNd7sewvaSIsKB5aBQQUKLM5s

[22]https://www.amqp.org/ (AMQP)

# 3 Occupancy Map Recontruction

## 3.1 OctoMap

OctoMap is a framework based on octrees which are the three-dimensional generalization of quadtrees [7]. In other words, an octree is a hierarchical data structure for spatial subdivision in 3D. They have been successfully used to represent 3D maps [1, 8, 9, 10, 11]. It mainly consists of recursively subdividing the cube into eight octans. Each octan in every division represents a node. The process ends when a minimum voxel size is reached. Figure 6 shows a single occupied voxel and its octree representation.

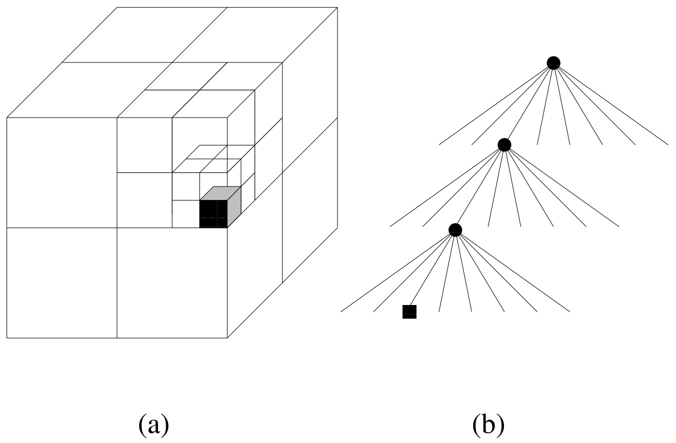


Figure 6: (a) The cube has been subdivided into tree depths, where the black cube represents an occupied voxel. (b) Octree representation.

## 3.2 Sensor Fusion

Probabilistic sensor fusion is used to update each node the occupancy grid tree [1, 10, 12, 13]. Equation 1 represents the probabilistic sensor fusion model.

In this equation, $P(n|z_{1:t})$ represents the probability of an occupied voxel or a leaf node $n$ given a sensor measurement from $z_1$ to $z_t$, $P(n)$ is the prior probability, $z_t$ is the current sensor measurement, $P(n|z_{1:t-1})$ is the previous estimate, and $P(n|z_t)$ is the inverse sensor model which gets probability of a voxel being occupied $n$ given the measurement $z_t$. According to [1], a common uniform prior probability value of $P(n) = 0.5$.

$$\frac{P(n|z_{1:t})}{1 - P(n|z_{1:t})} = \frac{P(n|z_t)}{1 - P(n|z_t)} \frac{P(n|z_{1:t-1})}{1 - P(n|z_{1:t-1})} \frac{1 - P(n)}{P(n)} \tag{1}$$

*Proof.* Let $P(n \mid z_{1:t})$ and $P_c(n \mid z_{1:t})$ the probability of an occupied and free voxel $n$ respectively. So,

$$P(n \mid z_{1:t}) = P(n \mid z_{1:t-1}, z_t) = \frac{P(n \mid z_{1:t-1})P(z_t \mid z_{1:t-1}, n)}{P(z_t \mid z_{1:t-1})} \qquad \text{(applying Bayes rule)}$$

$$P_c(n \mid z_{1:t}) = P_c(n \mid z_{1:t-1}, z_t) = \frac{P_c(n \mid z_{1:t-1})P_c(z_t \mid z_{1:t-1}, n)}{P(z_t \mid z_{1:t-1})} \qquad \text{(applying Bayes rule)}$$

$$\frac{P(n \mid z_{1:t})}{P_c(n \mid z_{1:t})} = \frac{P(n \mid z_{1:t-1})P(z_t \mid z_{1:t-1}, n)}{P_c(n \mid z_{1:t-1})P_c(z_t \mid z_{1:t-1}, n)} \qquad \text{(dividing)}$$

$$\frac{P(n \mid z_{1:t})}{P_c(n \mid z_{1:t})} = \frac{P(n \mid z_{1:t-1})}{P_c(n \mid z_{1:t-1})}\frac{P(z_t \mid n)}{P_c(z_t \mid n)} \qquad \text{($P(z_t)$ depends only of n)}$$

$$\frac{P(n \mid z_{1:t})}{P_c(n \mid z_{1:t})} = \frac{P(n \mid z_{1:t-1})}{P_c(n \mid z_{1:t-1})}\frac{P(n \mid z_t)}{P_c(n \mid z_t)}\frac{P_c(n)}{P(n)} \qquad \text{(applying Bayes rule to the last term)}$$

$$\frac{P(n \mid z_{1:t})}{1 - P(n \mid z_{1:t})} = \frac{P(n \mid z_t)}{1 - P(n \mid z_t)}\frac{P(n \mid z_{1:t-1})}{1 - P(n \mid z_{1:t-1})}\frac{1 - P(n)}{P(n)}$$

□

Some multiplications are needed in order to evaluate Equation 1, however it is computational easier to do summation instead. Therefore. a logarithm function multiplication can be turned into an addition. To this end, Equation 2 represents the Odds function which is defined as the ratio of the probability of an event divided by the probability of its complement. Whereas, Equation 3 shows the logOdds function which is the logarithm of the Odds function.

$$\frac{P(x)}{P_c(x)} = \frac{P(x)}{1 - P(x)} \tag{2}$$

$$L(x) = Ln\frac{P(x)}{1 - P(x)} \tag{3}$$

Equation 4 represents the logOdd $L$ radio of Equation 1.

$$L(n|z_{1:t}) = L(n|z_t) + L(n|z_{1:t-1}) \tag{4}$$

*Proof.* Let $L$ and $Ln$ be the Odds and OddsLog functions respectively. So,

$$Ln\left[\frac{P(n \mid z_{1:t})}{1 - P(n \mid z_{1:t})}\right] = Ln\left[\frac{P(n \mid z_t)}{1 - P(n \mid z_t)}\frac{P(n \mid z_{1:t-1})}{1 - P(n \mid z_{1:t-1})}\frac{1 - P(n)}{P(n)}\right] \qquad \text{(applying logarithm)}$$

$$L\big(n \mid z_{1:t}\big) = Ln\left[\frac{P(n \mid z_t)}{1 - P(n \mid z_t)}\right] + Ln\left[\frac{P(n \mid z_{1:t-1})}{1 - P(n \mid z_{1:t-1})}\right] + Ln\left[\frac{1 - P(n)}{P(n)}\right] \qquad \text{(logarithm property)}$$

$$L\big(n \mid z_{1:t}\big) = L\big(n \mid z_t\big) + Ln\big(n \mid z_{1:t-1}\big) + Ln\big(1\big) \qquad \text{(applying Oddslog)}$$

$$L\big(n \mid z_{1:t}\big) = L\big(n \mid z_t\big) + Ln\big(n \mid z_{1:t-1}\big)$$

□

A new sensor reading, introduces additional information about the state of the node $n$ . This information is done by the inverse sensor model $P(n|z_t)$ and it is combined with the most recent probability estimate stored in the node. It is worth noting that when initializing the map an equal probability to each node must be assigned. In other words, the initial node prior probabilities are $P(n) = 0.5$.

## 3.3 Mapping

RTABMap is a graph base SLAM approach. Graph SLAM means that the algorithm recovers the entire path, so then the full SLAM problem can be solved, allowing to work with the whole data to find an optimal solution. Moreover, appearance base means that data vision is used by the algorithm to localize and to build the map of the robot in the environment. Also, an important feature of the algorithm is the use of loop closure to determine whether the robot has been at a certain location already [14]. In as much as the complexity of the map increases the longer could take for the optimization, fortunately the algorithm has some strategies for optimization for large scale and long term SLAM, e.g TORO, g2o and GTSAM [4]. To this end, the aim of the graph SLAM is to create a graph with the whole poses and features and the most probable map and path.

Besides, rtabmap_ros is the RTABMap ROS integration that tackles the mapping issue. Figure 7 shows in more detail the rtabmap_ros node. The require inputs are: TF for the position of the sensors relative to the base of the robot, odometry for estimation of the position change over time, one or multiple RGB-D images, or a stereo image with its corresponding calibration file. The outputs are: Map Data that contains the latest compressed sensor data and the graph, Map Graph with no data, OctoMap for octree representation, a dense PointCloud and 2D Occupancy Grid. Figure 8 shows the output of the rtabmap_ros node as well as the output of the octomap_server. Figure 8(a) depicts the Map Graph, Figure 8(b) presents the rtabmap_ros_octomap, whereas the PointCloud can be seen in Figure 8(c) and finally the octomap_server can be depicted in Figure 8(d).

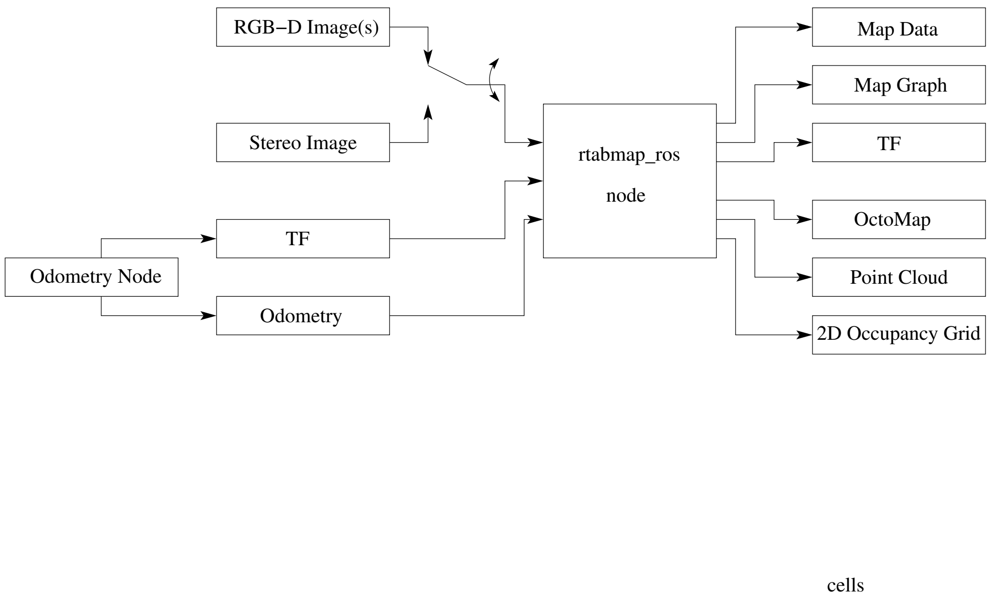
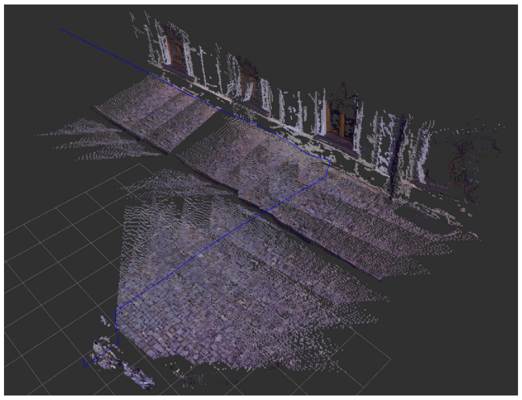
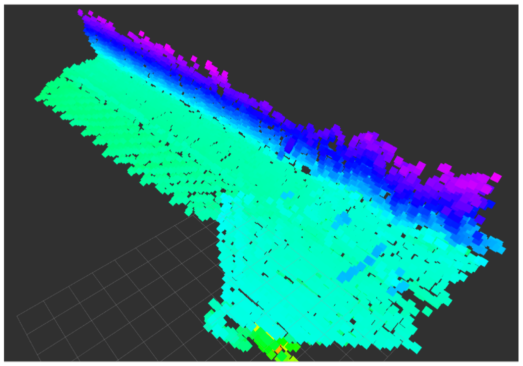


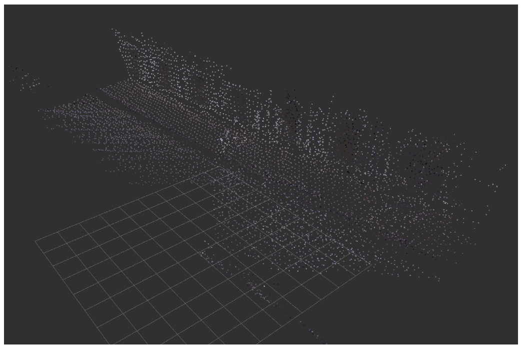
Figure 7: Scheme of the rtabmap_ros node.



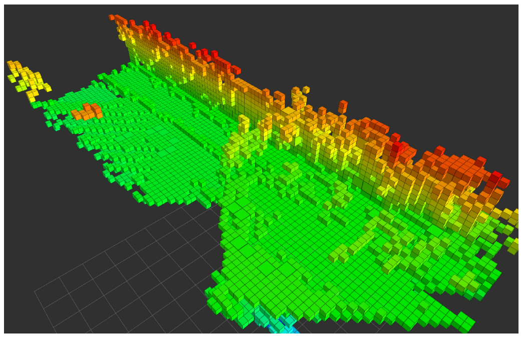
(a)



(b)



(c)



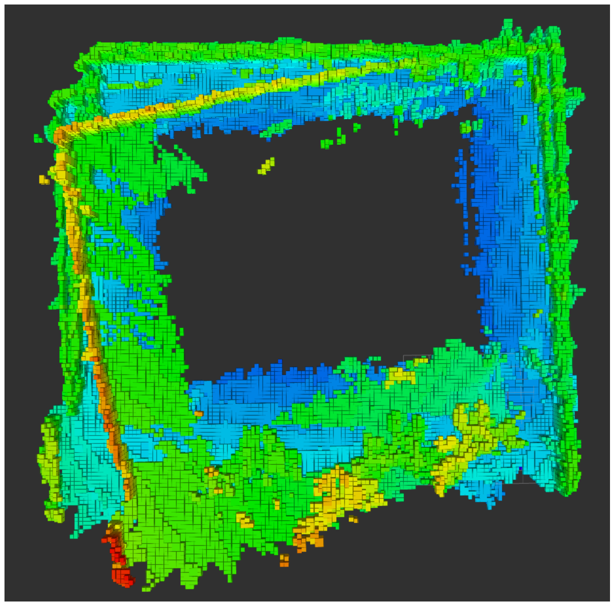
(d)

Figure 8: (a) Shows the Map Graph. (b) Shows the rtabmap_ros_octomap (c) Shows the PointCloud. (d) Shows octomap_server.

The octomap_sever node can incrementally build 3D OctoMaps, for that it needs as an input a 3D PointCloud. Also, rtabmap_ros can build an OctoMap of the occupied space for that it needs to be build up with OctoMap installed. Therefore, the 3D map based on octrees has been achieved in two ways.

- $Mode_1$, the octomap_server node subscribes to the PointCloud rtabmap_ros cloud_map topic and publishes the occupied_cell topic.
- $Mode_2$, the rtabmap_ros node publishes the octomap_occupied space topic.

In the first mode, the octomap_server node does not update accordingly to the cloud_map updating rate, this can be seen in Figure 9 where the OctoMap kept appending new occupied cells to the existing ones despite the cloud_map updates. In order to solve this issue, octomap_server node has to be manually cleared after loop closure to clear all obstacles. Then, in the new update the OctoMap map shall be more clean without the unnecessary appended voxels.


(a)

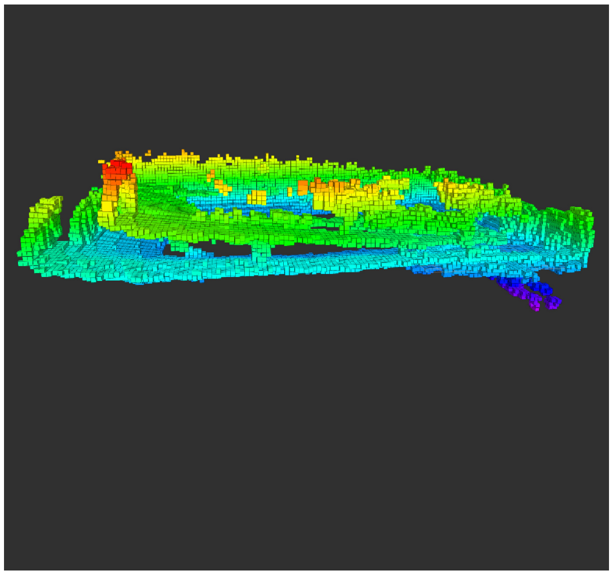

(b)

Figure 9: (a) Shows the top view of the 3D octomap_server map, (b) Shows the transversal view of the 3D octomap_server map.

Figure 10 shows the implementation of the $mode_1$. The zed_wrapper node publishes two topics; depth_image which produces the depth PointCloud of the environment. And, the zed_odom that gives absolute 3D position an orientation relative to the odometry frame based on pure visual odometry algorithm.

The depth_filter node publishes the depth_filter topic which eliminates textureless areas due to ZED_SDK has no option to filter them out. The depth can vary in the textureless areas that result in duplicated walls in the PointCloud, improvement of the reduction of duplicated walls is achieved by filtering the depth image cloud.

The rtabmap_node publishes two topics: cloud_map which contains the 3D PointCloud map of the outdoor environment and it is constantly saved in memory. And, the loopClosure topic that has to do with the ID loop closure detection that can be used to reset the OctoMap.

The turn OFF block reads the latest saved cloud_map when there is a Ctrl +C kill node action. This map is the result after it has been updated by all loopClosures, meaning that the map is clean of append voxels that the octomap_server did clean out after loopClosure. This action solves the problem of the octomap_server not clearing the map after loopClosure. So, the previous step allows to load the latest map into the octomap_server node which in turn publishes two topics: point_cloud_centers and octomap_cells.

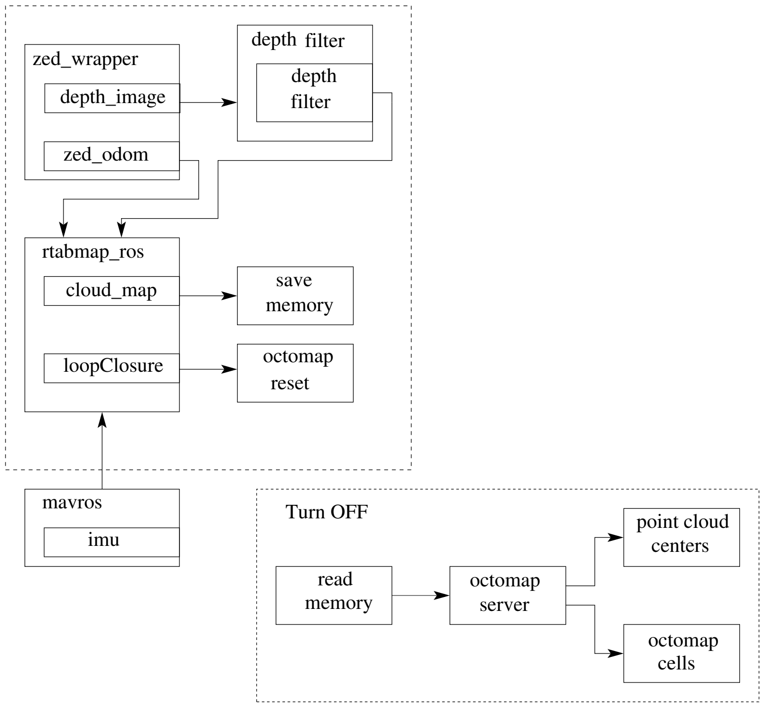


Figure 10: Schematics of $mode_1$.

$Mode_2$ uses the OctoMap output of the rtabmap_ros node directly, i.e. the /rtabmap_ros/octomap_occupied_space topic. In the process, the rtabmap subscribes to the camera depth_filter topic and publishes two topics; the /rtabmap/cloud_map and /rtabmap/octomap_occupied_space, also the OctoMap map is updating according to the frequency of the cloud map, moreover rtabmap will automatically update the full OctoMap on loop closure. The block diagram of the process can be depicted in Figure 11.

## 3.4 Experiment Results

### 3.4.1 Octomap

A rosbag data set with a size of 1.8 GB and 60744 messages has been recorded at the inner yard of the Faculty Information Technology (FIT) situated in Brno, Czech Republic which consists of the following ROS topics:

$\ast$ gps_global_position $\ast$ global_position_compass_hdg $\ast$ imu_data $\ast$ zed_right_image_rect_color_compressed $\ast$ occupied_cells_vis_array $\ast$ rtabmap_cloud_map $\ast$ rtabmap_mapData $\ast$ rtabmap_mapGraph $\ast$ rtabmap_octomap_occupied_space $\ast$ octomap_point_cloud_centers $\ast$ zed_odom

During the play of the rosbag, the data set is forward to the mode$_1$ and mode$_2$ respectively for evaluation as shown in Figure 12. The mode$_1$ and the mode$_2$ delivers 68 and 404 cloud scans respectively, both covering an approximate area of $1211m^2$ along a trajectory of 433m. Also, the mode$_1$ and the mode$_2$ delivers a total of 162668 and 46193 end points respectively where the table 2 summarizes the data. Figure 13

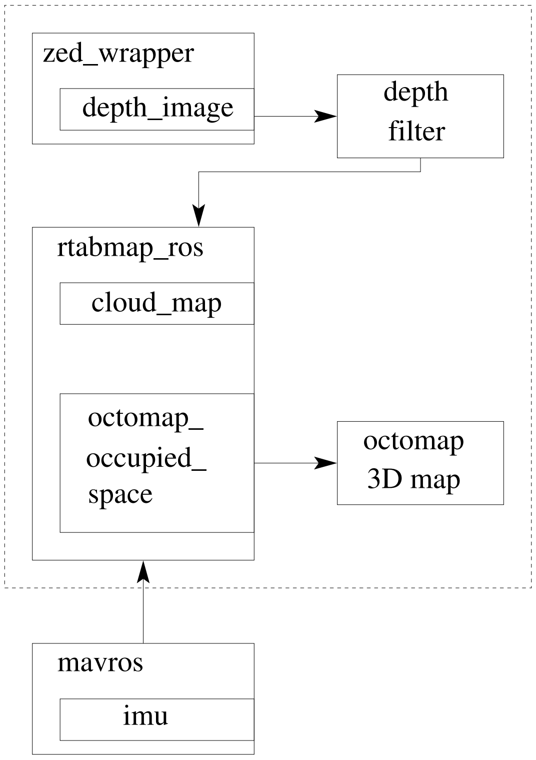


Figure 11: Schematics of $mode_2$.

shows the outcome of the $mode_1$ which is the result of the octomap_server after it has been updated every loopClosureID. Whereas, Figure 14 shows the outcome of the $mode_2$ which is the resulting OctoMap 3D map that correspond to the rtabmap_node.

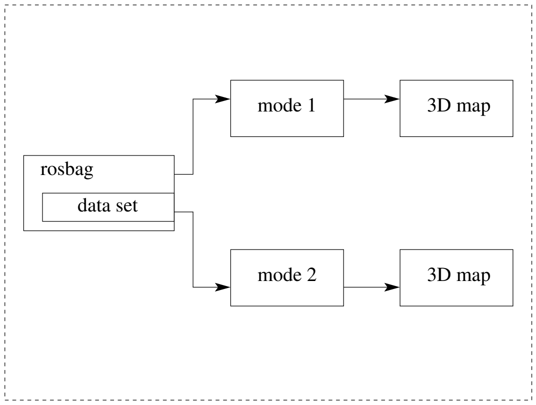


Figure 12: Shows the $mode_1$ and $mode_2$ with the rosbag data set as input and the octree 3D map as output.

| Comparison | | | | |
|---|---|---|---|---|
| | cloud scans | area$[m^2]$ | trajectory$[m]$ | points |
| $mode_1$ | 68 | 1211 | 433 | 162668 |
| $mode_2$ | 404 | 1211 | 433 | 46193 |

Table 2: Comparison between $mode_1$ and $mode_2$.

2

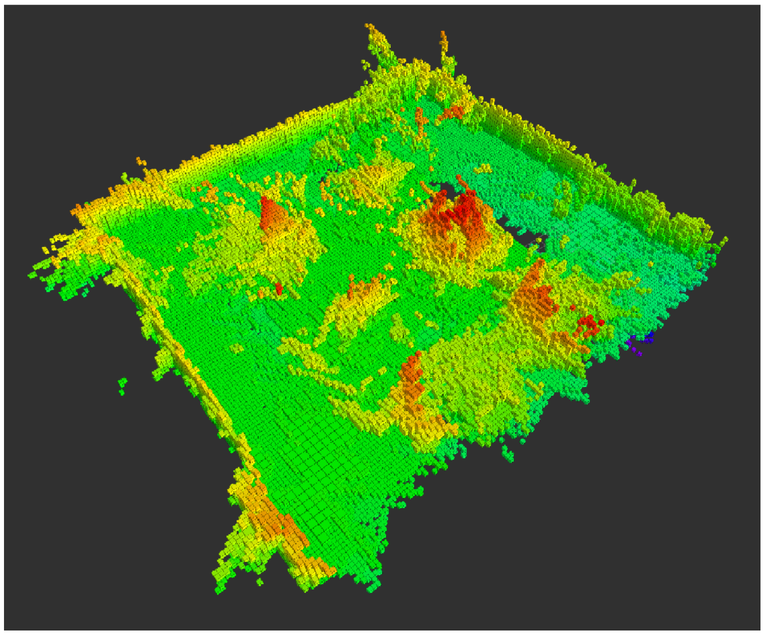


Figure 13: Shows the entire OctoMap after resetting manually every loopClosureID which correspond to $mode_1$

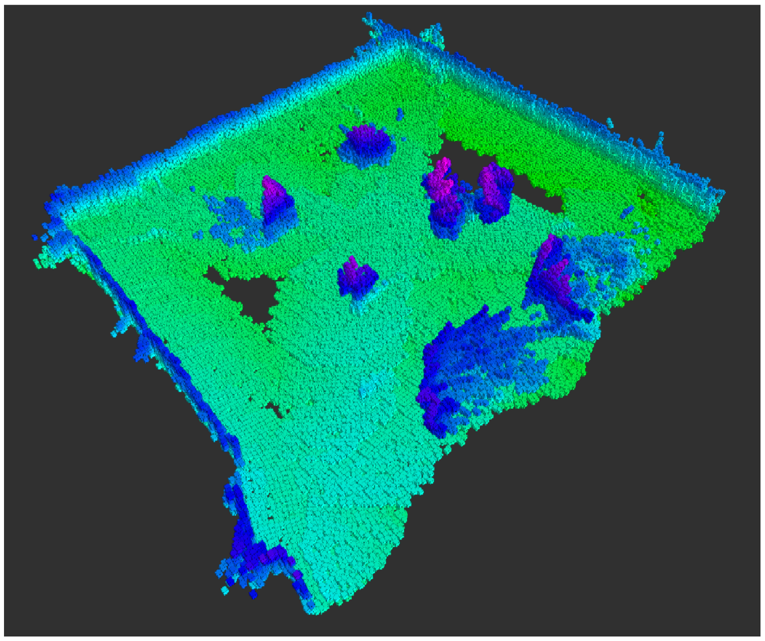
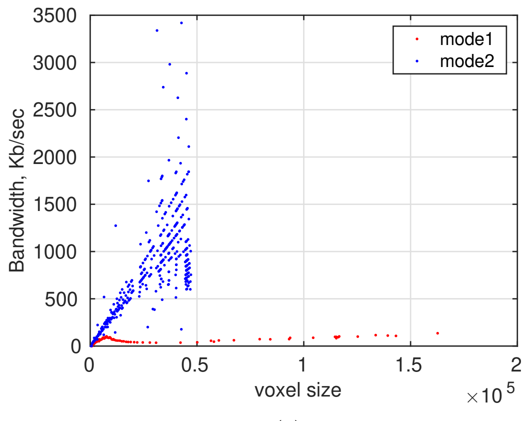


Figure 14: The resulting rtabmap_octomap 3D map which correspond to $mode_2$.

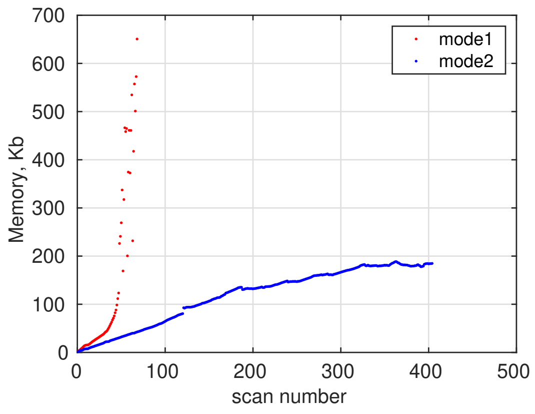
The computed maps by the two modes are compared each other by two different ways: bandwidth and memory consumption.

The observed data in Figure 15(a) shows the bandwidth average in Kb/s of the topics /occupied_cells_vis_array and /rtabmap/octomap_occupied_space with 400 and 68 scans that correspond to the output of the $mode_1$ and the $mode_2$ respectively versus the voxel cloud size. One can see that $mode_1$ has bigger cloud voxel size than the $mode_2$. But, the loading time of $mode_2$ is faster than the $mode_1$. This comparison shows that $mode_1$ loads faster with less voxel cloud size than $mode_2$.

Then, Figure 15(b) shows the evolution of memory consumption over the number of scans. It can be seen that the memory grows rapidly in the $mode_1$ whereas the memory consumption of the $mode_2$ grows less and more smooth with bigger number of scans.



(a)



(b)

Figure 15: a) Shows the bandwidth of the $mode_2$ is higher than the $mode_1$. b) Shows the memory consumption of the $mode_1$ is higher than than the $mode_2$.

### 3.4.2 ZED2 Camera

Figure 16 shows a schematic software diagram where the ZED2 camera is connected over a USB3 to a laptop acer Nitro5 + Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz running Ubuntu 18.04.1. The following software was installed on the laptop:

- ZED_SDK v3.2.2[23] for Ubuntu 18.04.
- RtabMap v0.20.4[24]
- rtabmap_ros[25]
- Cuda v11.0 [26]
- zed_wrapper[27]

[23]https://www.stereolabs.com/developers/release/3.2/
[24]http://introlab.github.io/rtabmap/
[25]http://wiki.ros.org/rtabmap_ros
[26]https://developer.nvidia.com/cuda-11.0-download-archive?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1804&target_type=deblocal
[27]https://github.com/stereolabs/zed-ros-wrapper

- octomap[28]

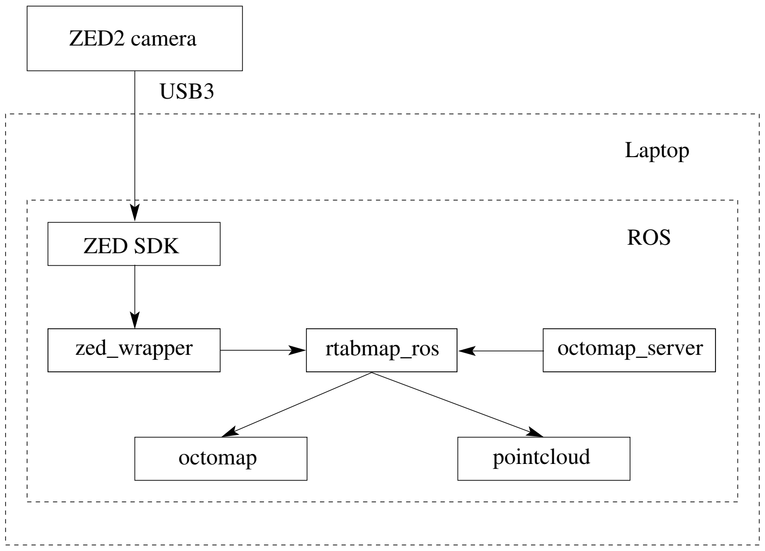


Figure 16: .

The roslaunch used to test the ZED camera which is based on RtabMap v0.20.0 was used to test the recorded ZED2fityard3.bag rosbag. During the running, a loop closure issue is found, e.g. the loop closure happens some time after the camera has been at previous location, other times did not happen or just happens almost at the and of the trajectory in contrast with the ZED loop closure that happens as soon the camera has visited a previous location and the camera has the same point of view (looking at the same direction).

Since the ZED camera runs on Ubuntu 16.04 and the ZED2 runs on Ubuntu 18.04, one difference is the version of OpenCV libraries linked to 16.04 that have access to all OpenCV's features (even nonfree) because ROS provides its own OpenCV version. In the other hand, on 18.04, ROS relies on the system version of OpenCV, which has very limited features (not built with opencv_contrib or nonfree modules). Moreover, with 16.04, GFTT/BRIEF is used by default, while on 18.04, GFTT/ORB is used. So then, GFTT is a feature detector and BRIEF, ORB are feature descriptors [15]. Also, in RtabMap v0.20.4, a new parameter has been added to change the way quantization of binary features is done (to use less RAM and processing time). To use the older approach, Kp/ByteToFloat [29] shall be set to false, having the previous parameter to true may affect the matching. Also, the Rtabmap/LoopThr parameter has to do with loop closing threshold.

In order to verify the previous mentioned statement or hypothesis, the database rtabmap.db, that correspond to the rosbag ZED2fityard3.bag, is run with different combinations of feature detectors under RtabMap v0.20.4 where Kp/DetectorStrategy corresponds to "6=GFTT/BRIEF and 8=GFTT/ORB".

Table 3 shows the combination GFTT/ORB and the parameters set up for Figures 17(a) and 17(b). It can be seen in Figure 17 that a change from true to false in the parameter Kp/ByteToFloat has improved significantly the loop closure which can be seen by the lines connecting the path.

Table 4 shows the combination GFTT/BRIEF and the parameters set up for Figures 18(a) and 18(b). Figure 18(a) shows an improvement of loop closure compared with Figure 17(a) also Figure 18(b) shows an improvement of loop closure compared with Figure 17(b).

[28]`http://wiki.ros.org/octomap`
[29]`https://github.com/introlab/rtabmap/blob/master/corelib/include/rtabmap/core/Parameters.h`

| GFTT/ORB combination | | | |
|---|---|---|---|
| | DetectorStrategy | ByteToFloat | LoopThr |
| Figure (a) | 8 | true | 0.11 |
| Figure (b) | 8 | false | 0.11 |

Table 3

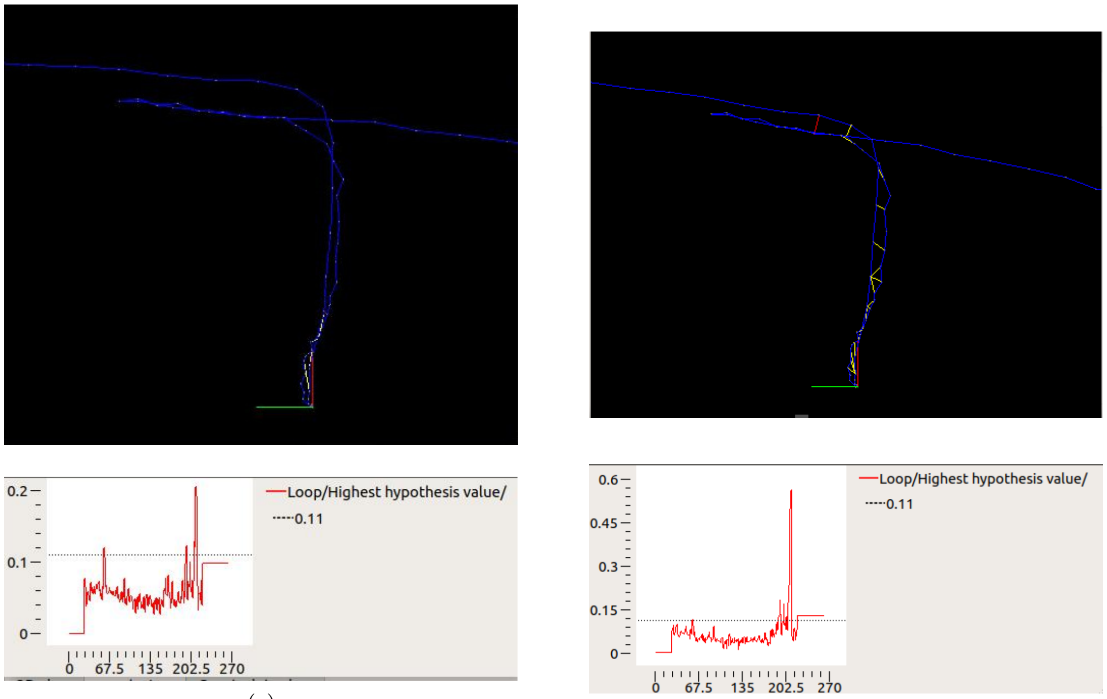


Figure 17: (a) Data base with combination; Kp/DetectorStrategy=8, Kp/ByteToFloat=true and Rtabmap/LoopThr = 0.11. (b) Data base with combination; Kp/DetectorStrategy=8, Kp/ByteToFloat=false and Rtabmap/LoopThr = 0.11.

| GFTT/BRIEF combination | | | |
|---|---|---|---|
| | DetectorStrategy | ByteToFloat | LoopThr |
| Figure (a) | 6 | true | 0.11 |
| Figure (b) | 6 | false | 0.11 |

Table 4

Table 5 shows the combination GFTT/ORB and the parameters set up for Figure 19. Figure 19 shows by lowering the parameter Rtabmap/LoopThr to 0.06 an improvement of the loop closure has been carried out.

| GFTT/ORB combination | | | |
|---|---|---|---|
| | DetectorStrategy | ByteToFloat | LoopThr |
| Figure (b) | 8 | false | 0.06 |

Table 5

Based on the previous combinations the conclusion is that by setting Kp/ByteToFloat to false, a better loop closure hypotheses can be gotten with binary descriptors. Also, Rtabmap/LoopThr can be lowered to test more hypotheses with Kp/DetectorStrategy=8. Besides, rtabmap node receives IMU-ZED2 data with Optimizer/GravitySigma to 0.3, so the map can be optimized with gravity and be aligned with the ground. Since good map optimization and rejection of bad loop closures are wanted, a good zed's odometry covariance is needed, this can be fixed on rtabmap node with odom_tf_angular_variance=0.001 and odom_tf_linear_variance=0.001.

Table 6 shows the final chosen parameters values and Figure 20(a) shows the 3D PointCloud map, whereas Figure 20(b) shows the occupied probabilistic voxel space with a size of 0.06m.

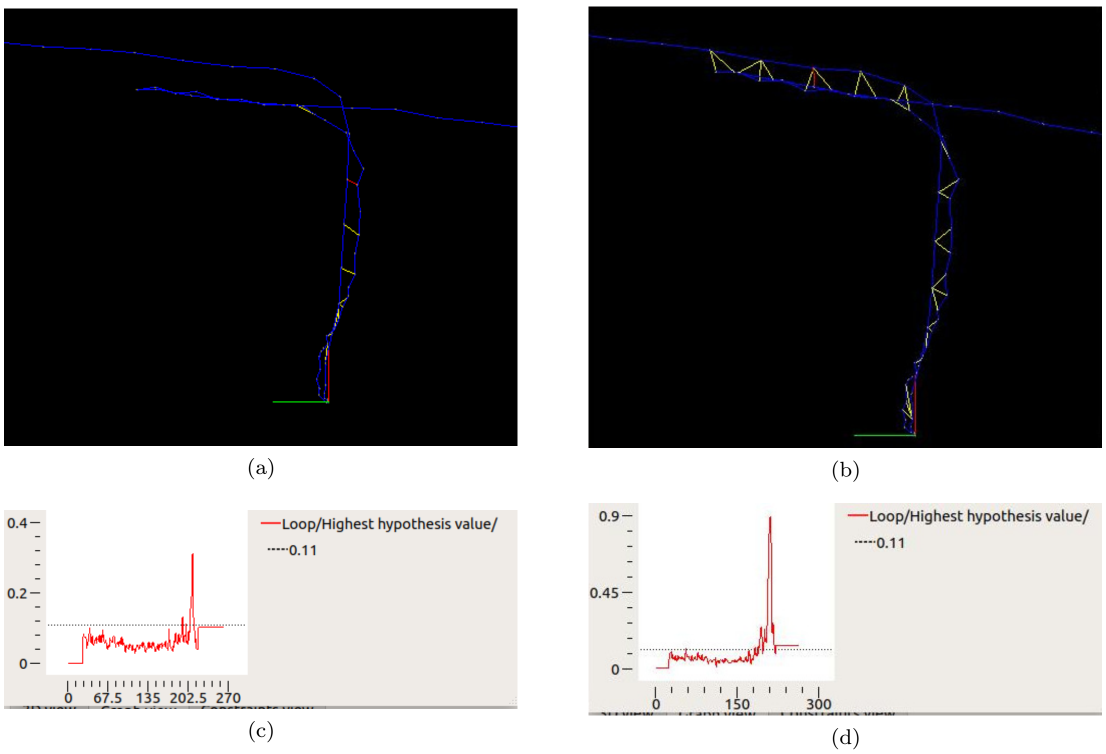

(a)
(b)
0.4
0.2
0
Loop/Highest hypothesis value/
0.11
0
67.5
135
202.5
270
(c)
0.9
0.45
0
Loop/Highest hypothesis value/
0.11
0
150
300
(d)

Figure 18: (a) Data base with combination; Kp/DetectorStrategy=6, Kp/ByteToFloat=true and Rtabmap/LoopThr = 0.11. (b) Data base with combination; Kp/DetectorStrategy=6, Kp/ByteToFloat=false and Rtabmap/LoopThr = 0.11.

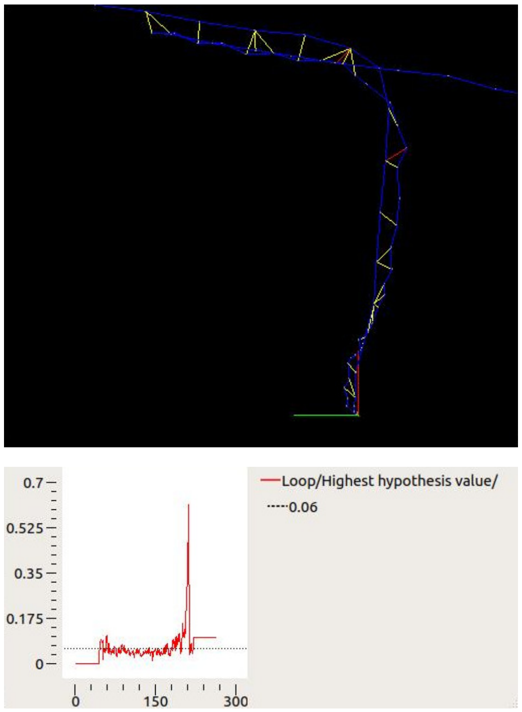
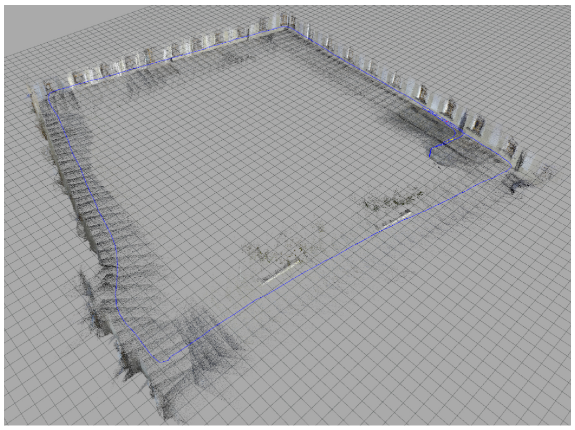
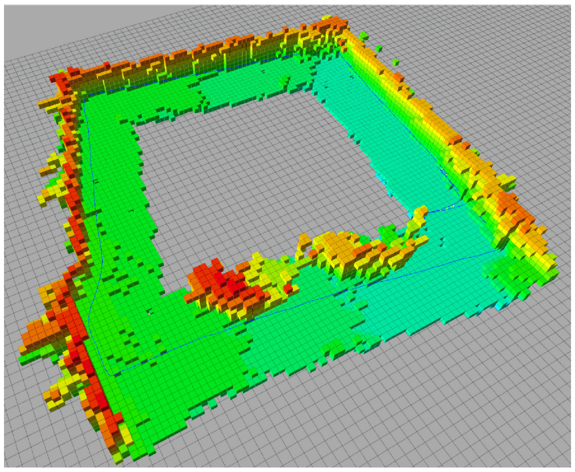


Figure 19: Data base with combination; Kp/DetectorStrategy=8, Kp/ByteToFloat=false and Rtabmap/LoopThr = 0.06.



(a)



(b)

Figure 20: a) 3D PointCloud map. b) 3D OctoMap map.

| parameter combination | | | | |
|---|---|---|---|---|
| GravitySigma | DetectorStrategy | ByteToFloat | LoopThr | Strategy |
| 0.3 | 8 | false | 0.06 | 2 |

Table 6

## 4 Ghost-free HDR merging

HDR videos are typically acquired through multi-exposure by sensors with a limited (standard) dynamic range [16, 17, 18], since this is both technologically and economically feasible as compared to other possible alternatives. Unfortunately, such videos typically suffer from "ghosts" caused by individual exposures of the motion objects taken at different times (and thus also capturing the objects in different positions) or by camera motion. Alternatives include theoretically ghost-free approaches, such as systems using beam-splitters with several CCD/CMOS sensors [19] or expensive and technologically demanding specific HDR sensors [20, 21].

We proposed a novel ghost-free HDR acquisition method that is powerful yet well implementable even in embedded systems (like HDR Camera) in real-time with low resource requirements. While de-ghosting has been researched for a long time, the state-of-the-art methods with good performance are computationally very demanding and so they are impossible to implement in smart cameras and/or embedded systems attached to cameras. In [22] we proposed algorithm which was designed with respect to real-time processing in embedded hardware.

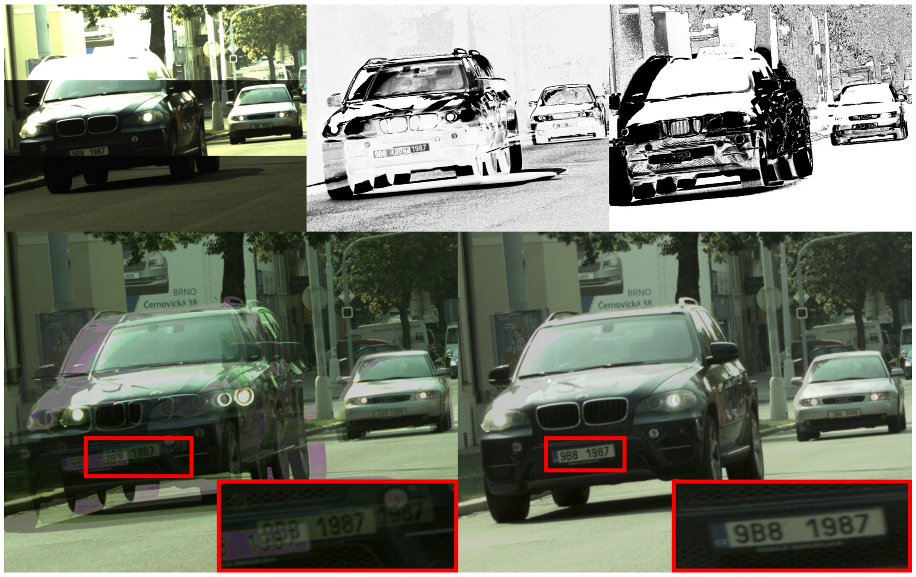


Figure 21: Example of ghost artefacts and result of the proposed ghost free merging. Top left - stripes of original images with a significant car motion. Top middle and top right - Images representing coefficients used for the HDR merging (*certainty maps*, see Section 4.2). Bottom left - ghosted HDR image. Bottom right - HDR image merged using the proposed method.

### 4.1 Novel ghost-free HDR merging algorithnm

The idea being similar to the solutions proposed by Grosch [23], Wu [24], and Wang [25] but with quite different and improved processing. The exposure time of each image is known; therefore, it is possible to estimate and match pixel values in the adjacent images, except for the over or under-exposed patches where the pixel values will obviously not match. Such estimation is not very precise, the captured image data is affected by factors such as noise, sensor quantization errors, CRF, etc. The reviewed methods generally use fixed or user-guided thresholds which must be employed in order to introduce user-defined tolerance to these factors. These fixed or user-defined thresholds often cause adverse effects in the final HDR images, such as visible transitions between static and motion areas etc. We propose a method to overcome such problems.

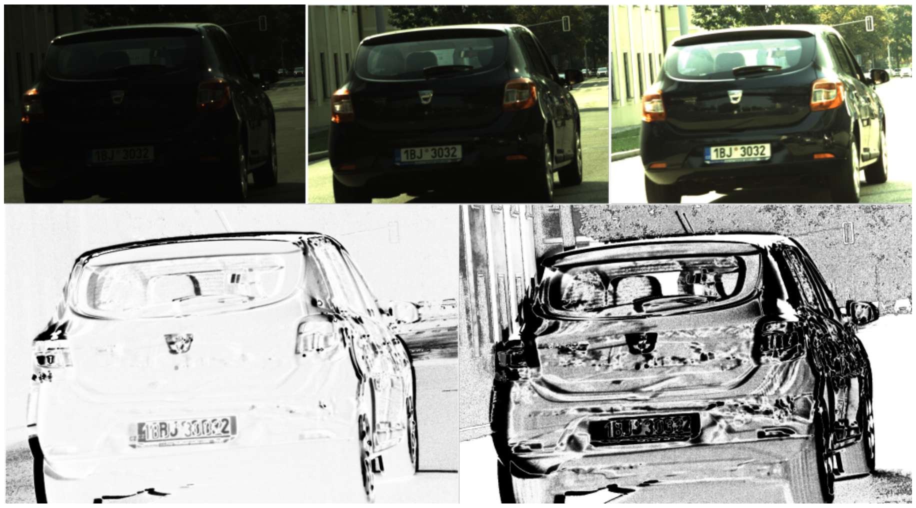


Figure 22: Two *Certainty maps* (bottom) obtained from the sequence on the top. The *Certainty map* on the left was obtained from top left and top middle (reference) image, the *Certainty map* on the right was obtained from top middle (reference) and top right image.

## 4.2 Certainty map computation

In our approach, every image $L_i$ is assigned a *Certainty map* $C_i$ related to the reference image $L_{ref}$, which is generally considered to be the middle (exposure) image in the sequence. The *Certainty map* $C$ contains values representing the estimated level of certainty that the individual pixels contain the same patch of the scene as the reference pixel, but obtained under a different exposure. Unlike ghostmaps, *Certainty maps* hold not only the patches containing motion, but rather all patches inappropriate for merging - such as under and over-exposed pixels.

The probability distribution of low level value pixels is Poisson [26] due to the discrete nature of the incoming photons. With higher intensities, the distribution transforms into Normal (Gaussian). Therefore, we use the Gaussian function to derive the certainty (estimated probability) that the two luminance levels, estimated and measured, match. The *Certainty map* $C_i$ (see Figure 22) replaces the binary *ghostmap* with soft assigned values, obtained using the information from the reference image $L_{ref}$, the estimated image $\overline{L_i}$, the exposure times $t_i$ and $t_{ref}$, as well as the CRF. Note, please, that in this paper the inverse CRF was implicitly applied to all images $L_i$. Image $\overline{L_i}$ is estimated by the following equation:

$$\overline{L_i} = L_{ref} \cdot \left( \frac{t_i}{t_{ref}} \right) \tag{5}$$

Consequently, the estimated value for image $i$ is processed along with the actual value of $L_i$ to get the probability based *Certainty map* $C_i$ as:

$$C_i = e^{-\frac{\left(L_i - \overline{L_i}\right)^2}{2\sigma^2}} \tag{6}$$

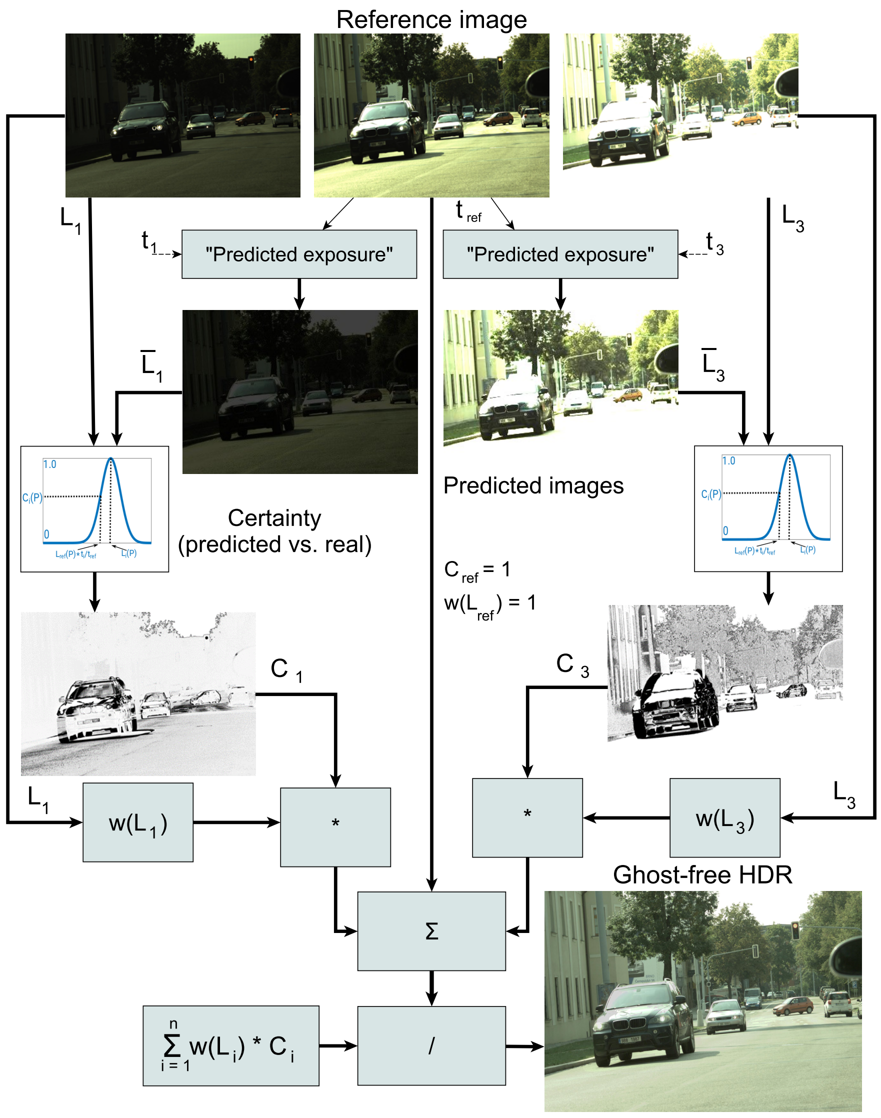


Figure 23: A scheme illustrating the proposed ghost-free merging of according to Equation 7 on a sequence of three images.

where $\sigma$ reflects the standard deviation of the pixel measurement (affecting the "softness" weight). The lower $\sigma$ is, the sharper or more strict the *Certainty* maps are, which results mainly in the dynamic range reduction. On the other hand, a high $\sigma$ causes "softer" *Certainty* maps, which may start to be ghosted. Ghost detection generally, and indeed inherently, cannot work well for the over and under-exposed spots of an image; thus the *Certainty* map algorithm contains a boundary condition: If the estimated value lies beyond the point of saturation, the Certainty is assigned at maximum value.

## 4.3 Multi-exposure merging algorithm

Our modification of Debevec's [16] merging algorithm incorporates the weights from the *Certainty map*, obtained through Equation 6. The HDR image $H$ is calculated as the weighted sum of pixels from $n$ images using the following equation:

$$H = \frac{C_i \cdot w(L_i) \cdot L_i \cdot \frac{t_i}{t_{min}}}{\sum_{i=1}^{n}(C_i \cdot w(L_i))} \tag{7}$$

The $C_i$ for reference image certainty is considered to be 1. The $w(L_{ref})$ is considered to be 1, as the reference image is a "pattern" with the desired object layout; it is not desirable to weight out the pixels, even if poorly exposed. A scheme illustrating the Equation 6 is shown in Figure 23.

## 4.4 Dataset evaluation and comparison

We performed the evaluation on datasets [27, 28, 29], containing sequences of images of various scenes and different types of motion. The results provide a comparison of the proposed method with generally more precise and computationally demanding methods, commonly based on optical flow, which were not even included into the related work due to their complexity and high computational demands.

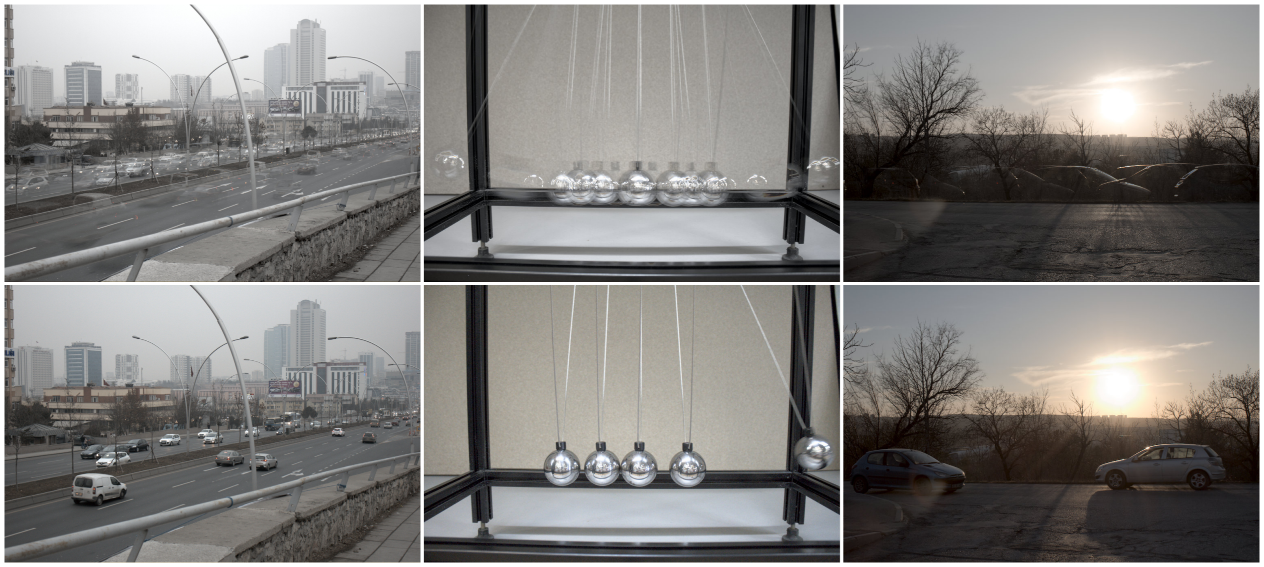


Figure 24: Ghosted HDRs (top line) and HDRs merged using proposed ghost-free method (bottom line) on sequences "Fast cars" [28] (left), "105" [29] (middle) and "117" [29] (right). Datasets contains 9 LDR (Low Dynamic Range) images.

Tursun et al. [28, 29] published two datasets and proposed metrics for evaluation of HDR de-ghosting quality. The evaluated samples from the datasets are shown in Figure 24 and the HDR quality metric [28] is evaluated in Table 7. The metric evaluates the dynamic range achieved inside the motion regions, considering also the correctness of the de-ghosting. The image sets, in which we got worse results than other algorithms, were successfully de-ghosted anyway; however, the worse results were probably caused by losses in the dynamic range. Evaluation of the proposed method on these datasets also proves that the proposed method is generally usable for sequences larger than two/three images, commonly used in cameras. However, the proposed method and also many HDR de-ghosting methods may yield artifacts in regions where the moving objects in the reference image are poorly-exposed, as Tursun et al. concluded [28].

## 4.5 Experiments

HDR camera was initially designed as a static camera (e.g., traffic camera) but the integration of advanced ghost free merging is key enabling technology for the recording (or even processing) of HDR video on the

Table 7: Results of the "Dynamic Region Dynamic Range" metric proposed by Tursun [28] and evaluated on their dataset. The metric evaluates the resulting dynamic range within regions containing movement; the higher the value, the better.

| Metric "DR" | [23] | [30] | [31] | none | **This work** |
|---|---|---|---|---|---|
| Cafe | **2.63** | 2.61 | 2.60 | 2.47 | 2.42 |
| FastCars | 1.12 | 1.18 | 1.10 | 1.10 | **1.38** |
| Flag | 1.40 | 1.50 | 1.49 | 1.45 | **1.59** |
| Gallery1 | 1.59 | 1.59 | 1.56 | 1.55 | **1.70** |
| Gallery2 | 2.41 | **2.56** | 2.14 | 2.29 | 2.05 |
| LibrarySide | 1.78 | 1.93 | 1.60 | 1.76 | **3.20** |
| Shop1 | 2.20 | 2.39 | 2.00 | 2.10 | **2.42** |
| Shop2 | 2.68 | 2.72 | **2.89** | 2.55 | 2.42 |
| WalkingP. | 1.94 | **2.07** | 1.83 | 2.05 | 1.58 |

drones. We performed several test flights with an installed HDR camera (see section 2.3.2). Example scenes are shown in Figure 25.

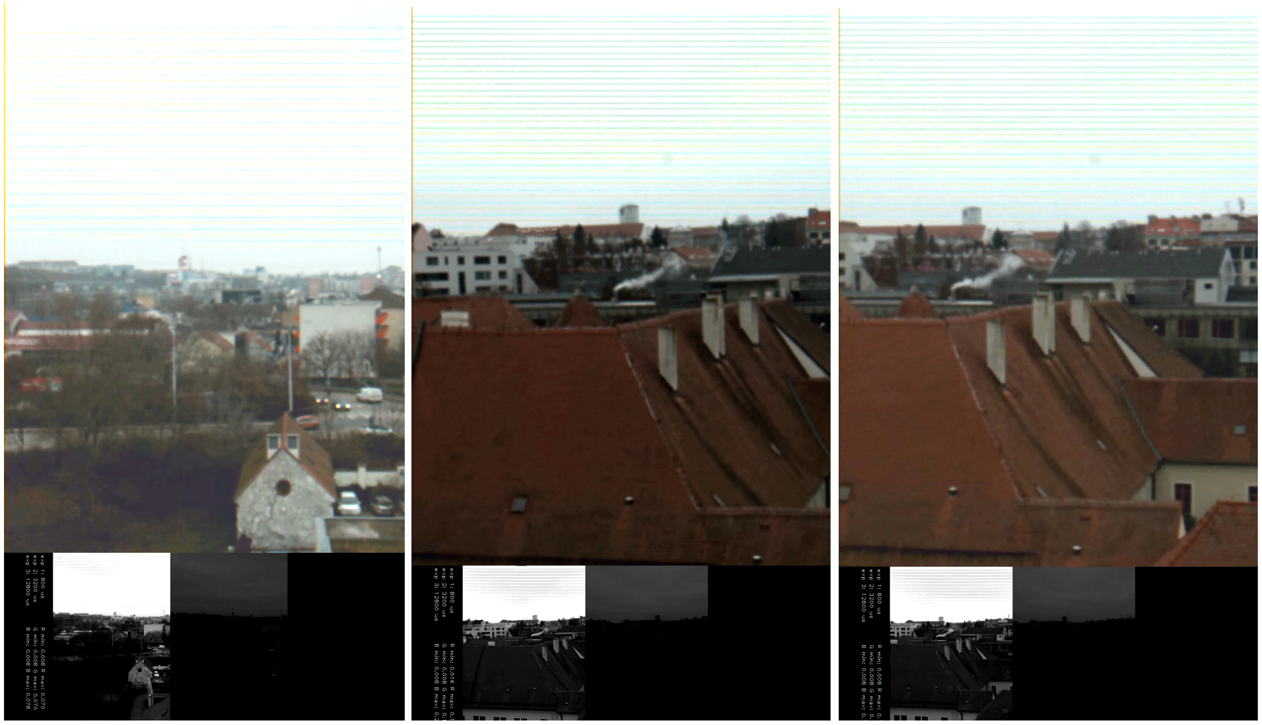


Figure 25: Ghosted HDRs (top line) and HDRs merged using proposed ghost-free method (bottom line) on sequences "Fast cars" [28] (left), "105" [29] (middle) and "117" [29] (right). Datasets contains 9 LDR (Low Dynamic Range) images.

## 5 Conclusions and Future Work

In this report, an application of the OctoMap open source for three-dimensional mapping has been presented. The approach is based on an efficient data structure base on octress where volumetric 3D models are represented in a probabilistic estimation that include occupied, free and unknown spaces. The octomap_server and the rtabmap_ros which are the integrated ROS wrappers have been used to handle the mapping process.

The evaluation is carry out by recording a rosbag data set that consists of many rostopics. Two modes were established; $mode_1$ is based on octomap_server whereas $mode_2$ is based on rtabmap_ros. The results had demonstrated that $mode_2$ has less memory consumption, scan numbers and faster data transmission over $mode_1$.

Also, the communication process between the TBSD and the ViAn server has been carried out. The data transfer process is handled by a Rabbit MQ (RMQ) listener which ensures correct data delivery to the server.

Additionally, the ZED2 camera has been set up. During the testing, an issue of loop closure was detected. To solve the issue, the GFTT feature detector and the BRIEF, ORB feature descriptors were tuned. The results gave a reliable 3D octomap map.

We proposed and published a novel ghost-free HDR merging algorithm suitable for real-time implementation in embedded devices. We intagrated this algorithm into HDR camera in order to improve the output quality of HDR video captured directly from the drone. Finally, HDR Camera with ghost-free HDR merging has been placed on the drone as an extra camera to capture video slightly different than normal, a flying test was made at the yard of the faculty, the video showed acceptable result. This experiment showed additional utilization of HDR processing directly on drones, where HDR can take place in difficult light conditions which may occur during a drone flight.

## References

[1] M. W. Kai, H. Armin, B. Maren, S. Cyrill, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, AK, USA, May 2010, software available at http://octomap.sf.net/.

[2] M. C. David and M. N. Paul, "Using laser range data for 3d slam in outdoor environments," *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pp. 1556–1563, May 2006.

[3] Q. Morgan, C. Ken, P. G. Brian, F. Josh, F. Tully, L. Jeremy, W. Rob, and Y. N. Andrew, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[4] 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.

[5] H. Aaron and S. Kelvin, "Multi-view augmented reality with a drone," *the Conference on the 24th ACM Symposium*, November 2018.

[6] 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.

[7] A. Max, *Computer Graphics and Geometric Modelling Implementation and Algorithms.* Springer, 2005.

[8] P. Pierre, H. Patrick, L. Denis, and G. Clement, "Probabilistic octree modeling of a 3D dynamic environment," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, 1997, pp. 1289–1296.

[9] W. Jane and V. G. Allen, "Octrees for faster isosurface generation," *IEEE TRANSACTIONS ON MEDICAL IMAGING*, vol. 19, pp. 739–758, 2000.

[10] M. W. Kai, H. Daniel, B. R. Dirk, Holz Radu, S. Cyrill, K. Kurt, and B. Wolfram, "Hierarchies of octrees for efficient 3d mapping," San Francisco, CA, USA, 2011.

[11] D. Meagher, "Geometric modeling using octree encoding," *Computer Graphics and Image Processing*, vol. 19, no. 2, pp. 129–147, Jun. 1982.

[12] t. Sebastian, B. Wolfram, and F. Dieter, *Probabilistic Robotics.*

[13] P. M. Hans, "Sensor fusion in certainty grids for mobile robots," *AI Mag.*, vol. 9, pp. 61–74, July 1988.

[14] D. Sagarnil, "Simultaneous localization and mapping (SLAM) using RTAB-MAP," *nternational Journal of Scientific and Engineering Research*, vol. 09, 2018.

[15] G. Oguzhan and B. C. Ahmet, "A comparison of feature detectors and descriptors in rgb-d slam methods," in *ICIAR, International Conference Image Analysis and Recognition*, 2015.

[16] P. E. Debevec and J. Malik, "Recovering high dynamic range radiance maps from photographs," in *ACM Trans. Graph.*, ser. SIGGRAPH '97, 1997.

[17] T. Mitsunaga and S. K. Nayar, "Radiometric self calibration," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, vol. 1, 1999, p. 380 Vol. 1.

[18] M. A. Robertson, S. Borman, and R. L. Stevenson, "Estimation-theoretic approach to dynamic range enhancement using multiple exposures." *J. Electronic Imaging*, vol. 12, no. 2, pp. 219–228, 2003.

[19] M. D. Tocci, C. Kiser, N. Tocci, and P. Sen, "A versatile hdr video production system," in *ACM SIGGRAPH 2011 Papers*, ser. SIGGRAPH '11, USA, 2011.

[20] M. Sakakibara, S. Kawahito, D. Handoko, N. Nakamura, H. Satoh, M. Higashi, K. Mabuchi, and H. Sumi, "A high-sensitivity cmos image sensor with gain-adaptive column amplifiers," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 5, pp. 1147–1156, 2005.

[21] H. Zhao, B. Shi, C. Fernandez-Cull, S.-K. Yeung, and R. Raskar, "Unbounded high dynamic range photography using a modulo camera," in *ICCP*, 2015.

[22] M. Musil, S. Nosko, and P. Zemcik, "De-ghosted hdr video acquisition for embedded systems," *Journal of Real-Time Image Processing*, pp. 1–10, 2020.

[23] T. Grosch, "Fast and robust high dynamic range image generation with camera and object movement," 2006.

[24] S. Wu, S. Xie, S. Rahardja, and Z. Li, "A robust and fast anti-ghosting algorithm for high dynamic range imaging," in *2010 IEEE International Conference on Image Processing*, Sept 2010, pp. 397–400.

[25] C. Wang and C. Tu, "An exposure fusion approach without ghost for dynamic scenes," in *2013 6th International Congress on Image and Signal Processing (CISP)*, vol. 2, Dec 2013, pp. 904–909.

[26] L. Mandel, "Fluctuations of photon beams: the distribution of the photo-electrons," *Proceedings of the Physical Society*, vol. 74, no. 3, p. 233, 1959.

[27] K. Karaduzovic-Hadziabdic, T. J. Hasic, and R. K. Mantiuk, "Multi-exposure image stacks for testing hdr deghosting methods," 2017.

[28] O. T. Tursun, A. O. Akyüz, A. Erdem, and E. Erdem, "The state of the art in hdr deghosting: A survey and evaluation," *Computer Graphics Forum*, vol. 34, no. 2, 2015.

[29] ——, "An objective deghosting quality metric for hdr images," *Comput. Graph. Forum*, vol. 35, no. 2, pp. 139–152, May 2016. [Online]. Available: https://doi.org/10.1111/cgf.12818

[30] P. Sen, N. K. Kalantari, M. Yaesoubi, S. Darabi, D. B. Goldman, and E. Shechtman, "Robust Patch-Based HDR Reconstruction of Dynamic Scenes," *ACM SIGGRAPH Asia*, 2012.

[31] S. Silk and J. Lang, "Fast high dynamic range image deghosting for arbitrary scene motion," in *Proceedings of Graphics Interface 2012*, ser. GI '12. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2012, pp. 85–92. [Online]. Available: http://dl.acm.org/citation.cfm?id=2305276.2305291