# LUCI AS A FORENSIC INVESTIGATOR'S BUDDY


## Abstract

This document will show the motivation for the choices that were made during the project
The topics in this document are:
SPLAM (simultaneous planning, localization and mapping),
photorealistic mapping, and the 360 camera.



**Project members**

| | |
|---|---|
| Joost Reumer | (492079) |
| Marina Gabarda López | (520101) |
| Max Schreuder | (481547) |
| Bas Scherphof | (477771) |

**Supervisor** Kees van Teeffelen


**Date** 30 – 06 – 2022

# Contents

# SPLAM Package motivation

## Packages and choices

This chapter will explain why certain packages were chosen and how these packages collaborate to establish splamming (simultaneous, planning, localization, and mapping).

### 2D or 3D SPLAM

In this project, it is possible to SPLAM in 2D or in 3D, due to the present depth cameras on LUCI. The choice was made to use 2D splamming because LUCI is a walking robot and is only able to move in two dimensions and cannot move in three dimensions like an arial drone. So, for this project, 3D splamming does not have any benefits over 2D splamming. Furthermore, the robot will only operate in relatively small and flat areas as described within the scope of our project, walking stairs is also excluded from the project. Another reason to use 2D splamming over 3D splamming methods is the computing power 3D splamming needs in comparison to the 2D splamming methods, also, the documentation of the 2D splamming is much better than the 3D splamming documentation, which makes 2D splamming more achievable in the timeframe of this project.

### Converting 3D data to 2D data

Due to the use of a 2D splamming method, 2D data is needed. The available cameras in LUCI provide depth-images with a pointcloud as output. Most 2D splamming packages need a laserscan as input. This means that a package is required that converts the 3D pointcloud into a 2D laserscan. An important requirement for the package is that the minimum and maximum height of the pointcloud that will be considered can be selected. There are not a lot of packages which can perform the correct action that is needed for further use of the 2D data. The package that was used for converting 3D data to 2D data is called: **pointcloud_to_lasercan**.

### Mapping (SLAM)

Two different packages have been tried for the use of mapping: **gmapping** and **slam_toolbox**. **Gmapping** is the package that will be used in this project because there is a large amount of documentation available to work with this package, besides the package is directly ready to use after the installation. Trying **slam_toolbox** caused some problems, for example when walking LUCI around, the current position was lost continuously, this resulted in that the map was not usable for splamming. In general **slam_toolbox** is a better mapping package, because it is generating small pieces of the map and is replacing or making them constantly. This way of mapping is more accurate than **gmapping** because it prevents or corrects drift of the map that is caused by poor odometry of the robot. However, **gmapping** works well enough for the goals that have to be accomplished during this project. Therefore, the choice was made to continue with **gmapping**.

### Path finding, localization and planning

The **navigation stack** is the most developed and used ROS package for these topics. It is also the most well-known package. This package handles everything that is needed, like: the speed, path finding, local planning, global planning, and many other things.

### Exploring an unknown area

For the exploration of an unknown area, the **explore_lite** package is used. This package sets a navigation goal outside the existing map and that is exactly what is needed when exploring in an unknown area. This package seemed to work very well, which was expected, as the **explore_lite** documentation states that is it compatible with the navigation stack. It is also the most common used package in combinations with the navigation stack. Therefore, the choice was made to use **explore_lite** in this project.

### Recommendations

For next groups working on this project, the first recommendation is doing research into different mapping packages. The current mapping package (**gmapping**) is prone to have some drift and is known to have issues when walking in the same area multiple times. Therefore, trying out a different package like **slam_toolbox** is recommended. Furthermore, taking some time to further tune the parameters of the pathplanning and localization algorithms is recommended. The expectation is that the current parameters are not fully optimised, and further tuning can improve the performance.

## Photorealistic mapping motivation

### Packages and choices

This chapter will explain why certain packages were chosen and how these packages collaborate to establish the photorealistic mapping.

### Photorealistic mapping package

At the start of the project, the task was to make a program so that a photorealistic map of the unknown room could be created with the internal cameras of LUCI. The only requirement that was given is that the software that had to be created had to work on ROS melodic. The rest of the choices that had to be made were completely free and had to be clarified. Therefore, before the choices were made, a good reasoning was needed.

The first step that was made is that the previous work and packages that were used by the students before us was looked at. The package that was used by the students was **RTAB-map**, so that package was the first one to be used by us but at our first tries the package wasn't working and had a lot of errors. At that point it was time to look for other packages that also creates a 3D photorealistic map. Packages that were tried after this were: **ORB-slam2** and **Elastic_fusion**. **Elastic_fusion** was the best package, but the only problem was that the package had to be transferred with **Elastic_bridge** because otherwise it would not work on ROS melodic, and this caused a lot of problems.

The biggest difference between **RTAB-map** and **Elastic-fusion** is that **Elastic-fusion** doesn't need to process the image afterwards. They both have another approach to come to the photorealistic map, and that **Elastic-fusion** theoretically would be better with keeping track of the drifting. This may solve the drifting issue that appears now when you walk too fast with LUCI.

After doing research for other packages and learning more about programming with ROS another try was given for using **RTAB-map** and a solution came for the problems. The default launch file that was used for **RTAB-map** was way too long and complicated. Therefore, a new, simpler launch file was created, which made **RTAB-map** work. After this success, the choice was made to use **RTAB-map** as the photorealistic mapping package.

### Cropping

The reason for needing cropping is the difference between the size of some LUCI's published images. The depth images a had 240x424 resolution, whereas the black and white had a 480x640 resolution. To fix this, it was necessary to crop the black and white image to the size of the depth image. This was done by modifying a launch file using **crop_decimate,** this function applies decimation (software binning) and ROI to a raw camera image post-capture. The images that needed to be cropped were left camera, right camera, front-left camera and front-right camera.

### 4 cameras implementation

First, to start working and try **RTAB-map** just one camera was implemented, the left one. When this camera was working properly, the rest of the cameras were implemented by adding the proper parameters to the 4 cameras launch file. LUCI has 5 cameras, but it was decided to work only with 4 because **RTAB-map** could handle a maximum of 4 cameras. The four cameras of LUCI that were used are the left, right, front-left and front-right camera because the back camera was not going to provide added information to the information already given by the rest of the cameras, the reason of this is that LUCI normally walks forwards.

### Image rotate

When **RTAB-map** was working with 4 cameras there was a problem with the orientation of the cameras because all the cameras were orientated in a different way. This caused a problem with the **RTAB-map** because walls were upside down. That is why the choice was made to use **image_rotate** to rotate the cameras in the right direction so that **RTAB-map** works properly. When **image_rotate** worked and the cameras were rotated a new problem appeared because **image_rotate** cannot work in our launch file because it doesn't publish a *camera info* and that is needed to let RTAB-map work, so that is why our final design just has two working cameras. The left and back camera were chosen as final two cameras this because they are both rotated in the right direction by default. Now, **RTAB-map** works fine with two cameras.

### Recommendations

Our recommendations for the next group will be to look for other packages than **RTAB-map** to do the photorealistic mapping this because **RTAB-map** is not optimal. **RTAB-map** is not optimal to work with multiple cameras and has a lot of drift when you walk to fast with LUCI.

## 360 Camera

For the 360 camera, apart from the ROS package provided by the manufacturer of the camera, there were no packages available that would work with the camera. Therefore, the choice was made to use the **image_pipeline** package in order to simply save images of the 360 camera to a file. This will allow easy viewing of the images on a computer that does not have ROS or even Ubuntu.