Robotic mapping is a discipline related to computer vision and cartography which has the goal to construct (and use) a map and to localize itself and its recharging bases. [1] Mapping itself refers to the action to model the environment.

Robot mapping is related to state estimation, localization, SLAM, navigation and motion planning. [2]

* *State estimation* of a robot is a set of quantities (position, orientation, velocity) that fully describe that robot motion’s over time. [3]
* *Localization* is the process of determining where a mobile robot is located. [4]
* *SLAM* is an acronym from “simultaneous localization and mapping” and it refers to the simultaneously construct or updating a map of an unknown environment while keeping track of an robot’s location within it. [5]

One project regarding this topic is made by Matthew W. from University of Pittsburgh named: “ROOM MAPPING Arduino Robot with Unity 3D”. The project has three scenarios and can be found at the following link[7]:

1. The first scenario it allows the user to drive around the robot with a camera feed going back to the user computer. This can be done by using arrows from the mobile application.
2. The second scenario allows the robot to track anything that is put in front of the camera.
3. The last scenario it allows the user to drive the robot with the computer using the arrow keys. The app uses an augmented reality SDK to find the walls and ceiling which is send back to the computer, giving the user a digital representation of the environment.

For this project, the following components were used: Arduino, ESP-8266 Node MCU, I298N motor drive, mobile phone, Arduino wires, robot chassis, wheels, battery and the Unity application.

Arduino is used for controlling the motors of the robot, based on the commands received through wireless network.

ESP-8266 is a Wi-Fi microchip with full TCP/IP stack and microcontroller capability, which is used for the wireless communication between robot (Arduino), computer and mobile phone.

l298N is a dual H bridge which allows the connection between the motors and Arduino board.

Mobile phone is running an application made in the video game engine Unity 3D. The application is used for image processing and creating an AR (augmented reality). The augmented reality is used to give a better digital representation of the current environment, based on the ability to distinguish the walls and ceiling from the rest of the elements, since they have a big role in every room mapping system.

For the implementation of the first scenario, the unity application is used for controlling the robot which will make a TCP connection to the computer. The application is splitted in two parts: the server and the client part. The server part is being implemented on the computer and it’s using image processing algorithms, and the client part is being implemented on the telephone and it is using the camera to fetch the visual information from the current environment.

The second scenario uses Vuforia, which “*is a software platform for creating augmented reality apps*”. [6] The main advantages of Vuforia is that “*developers can easily add advanced computer vision functionality to any application, allowing it to recognize images and objects, and interact with spaces in the real world.”[6].* The scene is named the follow scene, because over here it is created a trackable object which will be used by the robot in order to follow it.

The last scenario is the mapping scenario, and uses Apple’s ARkit, to detect vertical and horizontal planes. This uses a script to send the name, position, rotation and scale of the generated planes to the computer, so the ARkit could instantiate them.

Regarding robot’s navigation, there are two main strategies: *map-based navigation* and *map-less navigation*.

The research article “Robot Navigation Control Based on Monocular Images: An Image Processing Algorithm for Obstacle Avoidance Decisions”, written by William Benn and Stanislao Lauria from Brunel University, is focussed on map-less navigation and obstacle avoidance using images. [8]

The paper focuses on “how effective vision alone can be used as a tool for navigation and collision avoidance”. [8] Here, the computation is focused on the floor and will be used to calculate the “values for the parameters which are necessary to extract the required information from each image”.[8]

For the obstacle avoidance process, two main strategies were used: *image segmentation* and *edge detection.* “Each of the steps above is based on one of two basic proprieties of intensity values: discontinuity and similarity” [8].

The image segmentation is using the color segmentation process, which is based “on partitioning and image into regions that are similar according to a predefined criteria” [8]. The edge detection is based on “partition an image based on abrupt changes in intensity” [8].

The algorithm implementation was made in C++ and the image processing was completed using the OpenCV library. The main idea of the algorithm is to split the image in rectangular areas, in order to calculate the thresholds for the image color segmentation. The image will be splitted in multiple pixels and each one of them will be iterated, in order to update the threshold only when it is less than or bigger that the current threshold. “If a pixel’s RBG value is between the desired threshold, then the pixel can be marked as white otherwise as black” [8].

For image processing, there are a lot of research articles and projects made until now, and it is a really vast domain, which can be used in multiple industries: automotive, medicinal, machine/robot vision, patter recognition, video processing, etc.

In robotics, one of the projects is made by Aswinth Raj and is called “Raspberry Pi Ball Tracking Robot using Processing”. [9]

The idea of this project is to build a robot that could track a ball and follow it, just like a robot that plays football. The algorithm is based on image processing in the Processing framework. The real time video is a set of continuous frames of pictures. Each image is split into pixels, and each one of them is compared with the color of the ball. If the position of the pixel which is matching the ball color is far left, the robot will be moved to the right, and if the position is far right, the robot will be moved to the left, until the pixel position will be in the center.

For this project, the main controller will be a *Raspberry Pi*, which will receive information from camera module, will control the robot motors and will process the images. The *camera* used is a camera module special develop for Raspberry Pi, with ribbon cable – for connection with Raspberry. The camera will be connected to the CSI (Camera Serial Interface) port of the Raspberry. The *robot chassis* supports all components, including raspberry, camera, motors and batteries. The robot *motors* are controlled through L293D motor driver, and the batteries are used to power all components.

For the algorithm, there were imported two main libraries: “GL Video” – for accessing the Raspberry Pi camera and “Hardware I/O” library - for accessing the GPIO pins on the raspberry directly from the IDE.

Another project proposed by Heramb Nadkishor Joshi from STES’s Sinhdan Academy, India, has the main idea on finding a path for autonomous robots in an unknown environment using image processing. The project named “*An Image Based Path Planning And Motion Planning for Autonomous Robot*” is based on finding the shortest path for a robot, while avoiding obstacles in a complete unknown environment. “The proposed method is based on stereo images processing which can be captured by the camera set equipped on Base station system (Computer)”. [10]

The image detection is based on captured images, which are processed as grayscale images. The grayscale images are used because “the information quantity can be reduced to one third of color image”. [10] The obstacle detection methods is based on two fundamental image processing techniques: edge extraction and thresholding.

The outline of the obstacle, which is the edge of it, it is calculated based on the difference between the brightness of the pixels: if the pixels are brighter or darker, it means that there is something different from the background. Each pixel is compared with a threshold, which is decided according to the captured image. This process is made in order to detected in which “category” will the pixel distributed, brighter pixels or darker pixels. [10]

For path planning, the previous image (from the obstacle detection step) is divided into a square grid “which is called as grid-based approach”. This approach reduces the search area to a simple two dimensional array. From start configuration “S”, until goal configuration “G”, it is calculated the shortest path based on figuring out which node of the grid should be taken.

Pagina 2

**AUTOMATIC VEHICLE PARKING SYSTEM**

[1] - <https://en.wikipedia.org/wiki/Robotic_mapping>

[2] - <http://ais.informatik.uni-freiburg.de/teaching/ws12/mapping/pdf/slam01-intro.pdf>

[3] – State estimation for robotis – Timothy D. Barfoot <http://asrl.utias.utoronto.ca/~tdb/bib/barfoot_ser17.pdf>

[4] - <https://onlinelibrary.wiley.com/doi/full/10.1002/047134608X.W8318>

[5] - <https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping>

[6] - <https://library.vuforia.com/articles/Training/getting-started-with-vuforia-in-unity.html>

[7] - <https://www.instructables.com/id/ROOM-MAPPING-Arduino-Robot-With-Unity-3D/>

[8] - [file:///C:/Users/uib00020/Downloads/Robot\_Navigation\_Control\_Based\_on\_Monocular\_Images.pdf](file:///C:\Users\uib00020\Downloads\Robot_Navigation_Control_Based_on_Monocular_Images.pdf)

[9] - <https://circuitdigest.com/microcontroller-projects/raspberry-pi-ball-tracking-robot-using-processing>

[10] - <http://ijcsit.com/docs/Volume%205/vol5issue04/ijcsit2014050405.pdf>

<https://www.youtube.com/watch?v=SeNLUW79_-c>

<https://www.youtube.com/watch?v=4X_EjUZp2c0>

READ THIS: <https://www.geeksforgeeks.org/digital-image-processing-basics/>

Projects: <https://www.elprocus.com/image-processing-projects-for-engineering-students/>

Ovierview: <https://www.intorobotics.com/overview-of-robotic-vision-object-tracking-and-image-processing-software/>