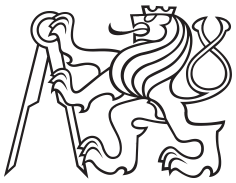
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Robotic Lawn Mower

Lukáš Bauer

Supervisor: Ing. Jan Drchal, Ph.D.
Field of study: Cybernetics and Robotics
Subfield: Cybernetics and Robotics
May 2020

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Bauer Lukáš** Personal ID number: **456970**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Robotic Lawn Mower

Master's thesis title in Czech:

Robotická sekačka

Guidelines:

The task is to integrate state-of-the-art AI techniques to a robotic lawnmower platform:
1) Research current state-of-the-art robotic lawnmowers, and what techniques they use, to navigate the environment.
2) Research alternative methods for using perimeter wire around the target area. Consider localization solutions based on GPS/Galileo.
3) Propose a robotic platform, computing platform as well as sensors needed for the security of operation and additional features. Aim for low-cost solutions.
4) Design and implement the controller. Focus mainly on localization and visual recognition of lawn grass.
5) Perform and evaluate experiments with the platform.

Bibliography / sources:

[1] Codol, Jean-Marie, et al. "Safety robotic lawnmower with precise and low-cost L1-only RTK-GPS positioning." Proceedings of IROS Workshop on Perception and Navigation for Autonomous Vehicles in Human Environment, San Francisco, California, USA. 2011.
[2] Schepelmann, Alexander, et al. "Vision-based obstacle detection and avoidance for the CWRU cutter autonomous lawnmower." 2009 IEEE International Conference on Technologies for Practical Robot Applications. IEEE, 2009.
[3] Schepelmann, Alexander, et al. "Visual segmentation of lawn grass for a mobile robotic lawnmower." 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2010.

Name and workplace of master's thesis supervisor:

Ing. Jan Drchal, Ph.D., Artificial Intelligence Center, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **23.01.2020** Deadline for master's thesis submission: _____

Assignment valid until:
by the end of summer semester 2020/2021

Ing. Jan Drchal, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student's signature

Acknowledgements

I would like to thank my family and colleagues for the support provided to me during my studies and to my supervisor for all the help and advice given to me.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date 20. May 2020

Abstract

Robotic lawn mowers are ever increasing in popularity. Unfortunately, lower cost robotic lawn mowers usually lack ”intelligence” and move in the designated mowing area randomly. In this thesis, multiple low cost localization options, together with a terrain classification method are presented. These methods are implemented as Robot Operating System (ROS) nodes and integrated to a DIY open source, open hardware robotic lawn mower platform Ardumower. The proposed filtered odometry and SLAM methods are then evaluated during multiple experiments and their error with respect to the ground truth, available from Whycon external localization method is measured. Both methods are able to achieve relative error with respect to the ground truth of less than 10%. After further corrections of the SLAM method, this method is able to achieve relative error with respect to the ground truth of less than 5%. The proposed terrain classification method classifies images to one of two classes. These classes are grass and non grass. The classifier achieved 98% accuracy on the evaluation dataset.

Keywords: robotic lawn mower, localization, visual classification, ROS, robot operating system

Supervisor: Ing. Jan Drchal, Ph.D.
Artificial Intelligence Center, FEE

Abstrakt

Robotické sekačky se těší stále větší popularitě. Bohužel levnější robotické sekačky obvykle postrádají ”inteligenci” a pohybují se vyhrazeným prostorem náhodně. V rámci této práce je představeno několik levných lokalizačních method spolu s klasifikátorem terénu. Tyto metody jsou implementovány v rámci Robot Operating System (ROS) a integrovány do DIY otevřené robotické platformy Ardumower. Navrhnuté metody lokalizace pomocí filtrované odometrie a SLAM jsou poté vyhodnoceny během několika experimentů a je změřena jejich chyba, vůči ground truth, která je dostupná z externí lokalizace pomocí Whycon. Obě metody dosahují relativní chyby, vzhledem k ground truth menší než 10%. Po dalších úpravách metody SLAM, tato metoda dosahuje relativní chyby vůči ground truth menší než 5%. Navrhnutý klasifikátor terénu klasifikuje obrázky z kamery do jedné ze dvou tříd. Tyto třídy jsou grass a not grass. Klasifikátor dosahuje 98% přesnosti na testovacích datech.

Klíčová slova: robotická sekačka, lokalizace, vizuální klasifikace, ROS, robot operating system

Překlad názvu: Robotická sekačka

Contents

Figures

Tables

Chapter 1

Introduction

Robotic lawn mowers are ever increasing in popularity. Unfortunately, lower cost robotic lawn mowers usually lack ”intelligence” and move in the designated mowing area randomly. To avoid obstacles, ultrasonic sensors are used. Furthermore, most of the currently available robotic lawn mowers require wire buried around the mowing area. This wire is then detected by the robot and marks the boundary of the area[2]. Most of the robotic lawn mowers also cannot differentiate between grass and flower garden and such non mowing areas cannot be inside the mowing area[3].

In this thesis, DIY robotic lawn mower platform Ardumower is presented, for this platform, multiple low cost localization options are implemented, that could be used as a substitution for a buried wire. These options are implemented using Robot Operating System (ROS) and tested on the chosen robotic platform during multiple experiments. Secondly, a simple, fast visual terrain classifier is presented, that can differentiate between mowing and non mowing areas.

1.1 Thesis overview

In the first part of this thesis, currently used localization and obstacle detection methods used on robotic lawn mowers are researched. Secondly, multiple localization options, that could be used instead of the currently used methods are researched. Finally, terrain classification methods are listed.

In the second part of this thesis, a DIY robotic lawn mower platform Ardumower is presented, this platform is then used, together with a Robot Operating System as a base robotic platform for all experiments. The chosen localization methods are implemented as ROS nodes and integrated together with Ardumower driver node. Finally, an implementation of a chosen terrain classification method is described. All localization methods and terrain classification method are then tested on multiple experiments and their performance is evaluated.

In the final chapter of this thesis, future work options, that are needed for a fully functional robotic lawn mower are proposed.

Part I

Related work

Chapter 2

Available low-cost robotic mowers

Currently available low-cost autonomous lawn mowers unfortunatelly have only very basic reactive behaviour with no currently available state-of-the art mapping and planning algorithms used. Lawn mowers use GPS for position estimation. For dead reckoning, sonar, laser scanners, cameras or differential GPS is used [2]. Some of these techniques will be described in the next section of this thesis.

Most of the available lawn mowers use buried perimeter wire for the detection of the area boundary. This techinque works by sending signal to the wire, which is then received by the robot if it approaches the area boundary. Same method can also be used when there are multiple robots in the same mowing area[2]. Some mowers use humidity sensors for boundary detection. This method works by detecting the change in the humidity of the terrain when mower reaches lawn boundary. This method unfortunately works only when the area is completely bounded by differently humid terrain (i. e. a sidewalk) and the robot will not avoid gardens or other similiary humid areas. One of such mower is for example the Ambrogio L60[3]. Other option is to use already mentioned GPS.

2.1 Ardumower

Ardumower is an open source, open hardware robotic lawn mower project. The idea of this project is to allow everyone to add in their own ideas for

autonomous lawn mowers. The core of this platform is an ardumower PCB with an Arduino. This board supports input for many different sensors either via I2C bus, or TX/RX. Adittionally, it also supports attachable bluetooth and wifi modules for wireless controll and configuration. The ardumower firmware also already has Robot Operating System (ROS) integration, where most of the sensor data is sent over USB to a connected computer and from which the robot accepts input to the motors. [4]

The original ardumower uses perimeter wire for boundary marking, the wire also serves as guidance to the charging station. For collision detection, sonars or pressure sensors are used.

Chapter 3

Available low-cost localization methods

3.1 Odometry

Very low-cost localization option is to use wheel encoders to obtain an absolute position of robot's wheels. The change in the position of a wheel can be obtained as

$$\Delta x = \frac{2\pi \Delta n r}{N}, \tag{3.1}$$

where Δn is the difference between initial and end reading from a wheel encoder, r is the radius of a wheel and N is the number of generated signals from the encoder for one wheel revolution. The average distance travelled by both wheels is then used for estimation of the new position of a robot. It's orientation difference is then obtained as

$$\Delta \varphi = \text{atan} \frac{\Delta x_R - \Delta x_L}{l}, \tag{3.2}$$

where Δx_R is the distance travelled by the right wheel in the last step, Δx_L is the distance travelled by the left wheel in the last step and l is the distance between the two wheels. The absolute pose can then be obtained as the sum of all previous steps. The major drawback of this approach is that irregularities in surface can cause errors and even small changes in the direction of the robot can lead to large differences between estimated and real pose due to the accumulation of errors over time. Odometry errors can be reduced by comparing the results with an absolute position, that becomes new initial position for further computation. [5]

3.1.1 Visual odometry

Robot position can also be estimated from a camera, mounted on the platform. Both monocular [6][7] and stereo cameras can be used for motion estimation. Approaches are based on either dense optical flow or sparse feature tracks. Using stereo images for motion estimation tends to be more stable and well behaved than when using monocular camera, but feature correspondences need to be found. These feature points are triangulated and then used to estimate 3D motion. [8] [9]

In [9], a couple of stereo cameras were used for robot localization. Corner feature points were extracted from the left image of each stereo camera. These features were then triangulated based on stereo correspondences. Three of these points are then chosen for motion estimation. Absolute pose is obtained by chaining relative motions together. The initial pose is taken from wheel odometry, IMU and GPS data. The IMU and wheel encoders were also used when visual odometry failed. This method eliminates the problem of wheel slippage and outperforms wheel odometry by factor of two.

3.2 Global Navigation Satellite System

The greatest advantage of using a Global Navigation Satellite System (GNSS) for localization of a mobile robot is an availability and cost of GNSS modules. Cheaper modules can be obtained for less than $100, which is very important, when looking for low-cost localization options. One of the disadvantages of these cheaper modules is their accuracy. For our task, we need less than 1m accuracy, to be able to follow reliably the boundary of a designated area.

3.2.1 Differential GPS

Some of the inaccuracy of GNSS localization comes from the transition of GNSS signals through the atmospere, or from slight error in satellite's orbit. This then results in slightly longer time, that it takes the signal to reach a GNSS satellite and therefore in erroneous position estimation. Some of these errors can be corrected by having a second GNSS module, that sends correction data to cancel out GNSS errors. This can result in centimeter-level positional accuracy. The greatest disadvantage of this approach is much

greater cost, which is caused by the need for the second module. Also, to reach centimeter-level accuracy, dual-frequency GNSS modules are usually needed, which further increases the cost beyond what is affordable for this case.

There have already been attempts to use Real-time Kinematic GPS (RTK-GPS) for localization of lawn mowers. In [10], a pair of cheaper L1-only receivers was used. Using statistical approach to L1-only RTK-GPS, the robot was able to successfully mow the area without the need for buried wire. The robot was also able to return to the charging station, where accuracy of $\pm$ 5cm was needed to reach an electrical plug.

3.3 Simultaneous localization and mapping

3.3.1 Robotic mapping

All of the methods described to this point, assumed some prior knowledge of the environment, where the robot is operating (position of obstacles, boundary of the area). Generally, that is not the case and some method of mapping is needed. Robotic mapping comes with several challenges. A key challenge arises from the nature of the measurement noise, because the noise of the measurement is statistically dependent. This is because errors in controll accumulate over time and they affect the way future sensor measurements are interpreted. Another problem is the correspondence problem. This means determining if sensor measurements taken at different points in time correspond to the same physical object in the world. Another problems are high dimensionality and dynamic environment. [11]

Very popular approach to the environment mapping is using occupancy grid maps. Using this method, maps are usually represented as two-dimensional grids, where each cell holds the probability of an obstacle present on the location. Standard occupancy grid mapping algorithm is a version of Bayes filters, just like any other major mapping alogrithm. [11][12]

3.3.2 Simultaneous localization and mapping

The mapping problem is often in conjunction with the localization problem, because we need information about robot position for building the map. Simultaneos localization and mapping (SLAM) tries to simultaneously build a map of the environment and localize the robot in this map [11]. There are many different methods for SLAM. One available option is to use sonar measurements for building a map of the environments. Sonars are very cheap, but the obtained measurement data are very sparse. In [13], a ring of sonars was used to build a feature based map of the environment. The feature based approach to map building allows to make characterization and relative positioning of features with uncertainty near the level of a measurement noise.

3.4 Whycon

Evaluation of the performace of tested methods needs another, more precise localization system, that will be used as ground truth. One of such precise, external localization systems is Whycon. Whycon is able to localize and track black and white circular pattern in an image[14]. Using given camera intrinsic parameters and known size of the pattern, it is then able to determine its 3D position and orientation. This position can be transformed to user defined 2D or 3D coordinate system after performing simple calibration. This tracking algorithm was proven to achieve sub-pixel precision. Another advantage is that its computational complexity is not dependent of the image size, because the search for the pattern can be initiated from any position within the image. This means that if a prior pattern position is known, the search is initiated from this position. [14][15]

Chapter 4

Terrain classification

In previous chapter, various methods of mobile robot localization were presented. Most of these methods do not carry any information about the environment, in which the robot is located. In this chapter, some possible methods for terrain classification are presented. These methods could be used to implement more complex behaviour for the final robot, for example avoiding flower gardens, ponds or recognizing already cut grass.

Visual based approach is the most common one used as cameras are very cheap to buy. The biggest issue is the processing power needed, to process images from the camera in real time. There has already been an attempt to use camera on board of an off-the-shelf autonomous lawn mower. This camera was used for image segmentation to grass and non grass areas. Pixel was classified as grass, if the pixel hue was in range $[h_{min}, h_{max}]$ and saturation exceeded s_{min}. This method was available to run onboard on an added low cost compute board. To tackle the issue of different lighting conditions and changing weather, closed loop between the camera control and color-based grass segmentation algorithm was used. This allowed the optimization of the visibility of the grass by taking into account the grass segmentation output. This method was able to achieve 88% accuracy in the performed small scale test [16].

In [17] a visual feature-based terrain classification is presented. This method uses bag of visual words (BOVW) created from features extracted by speeded up robust features (SURF) extractor and clustered together by k-means algorithm. These features are then classified using support vector machine (SVM).

4.1 Neural networks

Most of the current state-of-the-art image classificators are based on a Convolutional Neural Networks (CNN). CNN is a special type of multilayer feedforward neural network [18]. These are composed of multiple stages. The input and output of each stage is called the feature map [19]. There are two main modules of a CNN, the feature learning module and the classification module [18]. The feature learning module can contain multiple 3-layer stages, composed of a filter bank layer, a non-linearity layer and a feature pooling layer [19][20].

Filter bank layer extracts visual features from an input image. The convolution $conv(\bullet)$ is a dot product of the input image, I and the convolution kernel, K. The convolution kernel slides over the input image to produce a convolved feature map as output [18][20].

First the convolutional layer detects low level features from an image. The output from a convolutional layer is called the activation map. The next convolutional layer applies its filters on the activation map from the current layer. This results in the detection of higher level features that are an abstraction of several lower level features found by the previous layer. Non-linearity layer traditionally consits of a sigmoid function, such as the hyperbolic tangent[18][19][20].

The feature pooling layer subsamples the feature map to reduce its spatial dimensionality, thus producing a more compact feature representation [18][20].

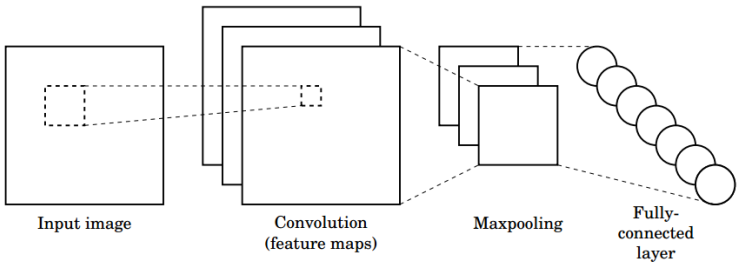


Figure 4.1: Structure of a CNN [1]

CNNs can be used for image classification, object detection or image segmentation. One of the more recent image segmentation network architectures is for example SegNet. It uses first 13 layers from a VGG16[21] network,

together with corresponding decoder layers, that upsample features from max-pooling layer and a fully connected layer at the end for pixel-wise classification [22] [20].

Deep convolutional neural networks are usually accelerated using a GPU, which might not be available for a given task. MobileNet however is an example of a neural network, aimed to run at weaker computing platforms. It uses a 3x3 depthwise separable convolution, which uses between 8 to 9 times less computation than standard convolutions at only a small reduction in accuracy. It also introduces 2 hyper-parameters, width multiplier and resolution multiplier. The former allows to thin a network uniformly at each layer. The number of input channels M becomes αM and the number of output channels N becomes αN. The latter allows to scale down the input image resolution[23].

Terrain classification can also be done without the need for an onboard camera as presented in [24]. Terrain classification was done using already present sensors on the chosen robotic platform (Inertial Measurement Unit (IMU), sensors for measuring motor torque and power consumption, IR range sensor etc.). All classification experiments were performed based on just one single sensor modality at a time. Feed-forward neural network was used for the classification between different types of terrain (gravel, grass, sand, pavement and dirt).

Part II

Ardumower setup and enhancements

Chapter 5

Ardumower setup and ROS integration

5.1 About ardumower

For this thesis, complete ardumower kit was used. This kit includes:

1. ardumower PCB v1.3
2. 3x ultrasonic sensors
3. 2x wheel motors + drivers
4. 1x mower motor
5. GY-801 Inertial measurement unit (IMU)
6. Arduino DUE
7. Bluetooth module

Some other parts, for example perimeter sender and receiver were not used. This kit also includes no battery. The PCB takes a 24V DC input. For this 14 Li-ion 18650 cells were used (7 connected to the series, 2 sets parallel).

The ardumower PCB first needs to be assembled and properly configured based on what sensors will be used. For this, series of jumpers are present on

the board, that set voltage level or enable certain attached modules. After the board is assembled and jumper configuration is made, firmware needs to be uploaded to the used Arduino.

The ardumower firmware allows configuration, testing and calibration by multiple methods. First, but very limited method is via the console from a connected computer. This method allows to test wheel encoders, calibrate IMU gyro and compass and setup bluetooth or WiFi communication. When bluetooth or WiFi is set up, a user can also connect via provided mobile app. This method is the main method for configuring, or debugging the ardumower firmware, as it allows to configure every parameter used by the mower, log errors and values and even plot them.

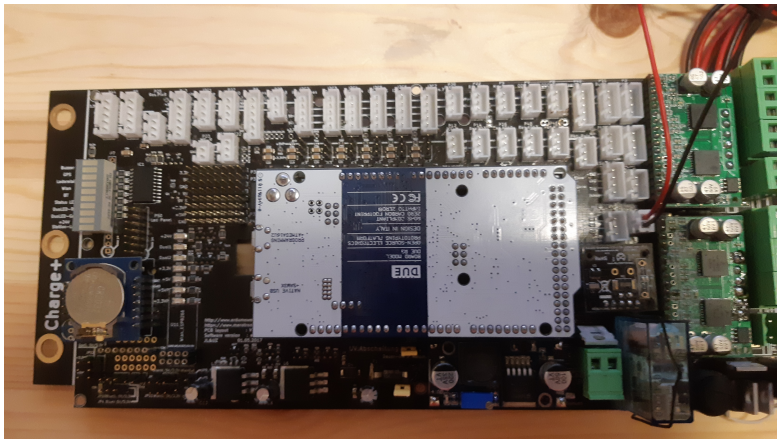


Figure 5.1: Assembled ardumower PCB

5.2 About Robot Operating System

Robot Operating System (ROS) provides libraries and tools to help developers create robot applications. The main component of the ROS framework is a ROS node. ROS nodes are able to communicate with each other using both synchronous and asynchronous communication. For asynchronous communication, ROS Topics are used. Any node can publish a ROS message to a ROS Topic, where multiple other nodes can listen for these messages. ROS Services are used for synchronous communication, where one Node calls the service, advertised by a different node.

5.2.1 ROS TF Library

One of the tools, provided by ROS are the TF and TF2 libraries. These libraries allow to transform ROS messages, that carry information about a position of some object between different coordinate systems (ROS frames).

More about ROS can be found in [25][26].

5.3 ROS with ardumower

The experimental version of the ardumower firmware comes with support for ROS. The integration works by sending values, read by sensors connected to the board, over the USB to a connected computer which runs the main ardumower node. This node then parses the data and saves them to variables. The original ardumower node only uses odometry readings to create odometry message and appropriate transformation from the odom frame to the base_link frame, that is attached to the body of the robot and had to be modified heavily to work together with localization methods, used in this thesis. Secondly, the node listens to messages, published to the cmd_vel node and sends commands for the motors to the PCB.

For better control over the robot, another node was created. This node provides a service for simple path following. The currently used localization method could be changed by simply changing the parameter of the source frame. All transformations and frames were defined according to REP-105[27]. This allows for redundancy if any more complex localization method fails as the node will only lose the information about an error made by another method with respect to the main one.

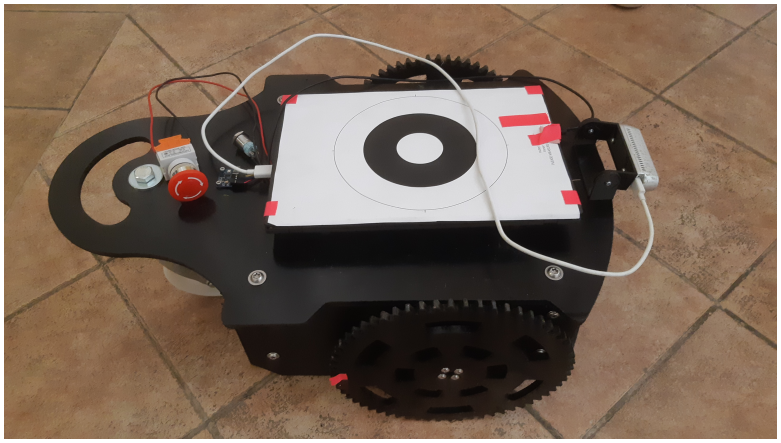


Figure 5.2: Assembled ardumower with a laptop for controll

Chapter 6

Implementation of robot localization methods

In this chapter, the implementation of selected localization methods will be described. The selected methods are

1. Odometry
2. SLAM
3. Whycon

6.1 Odometry + IMU

Since ardumower kit already comes with an IMU and motors with encoders, the first method used for robot localization was fusing odometry and IMU data together. For this, `robot localization` node was used. This node uses an extended Kalman Filter (EKF) to fuse multiple localization inputs together. Possible inputs include Odometry, IMU and visual odometry[28]. To be able to use this node, data from IMU need to be published onto ROS topic as IMU message type. Also, both odometry and IMU messages should have their covariance matrices for each sensor set. As there were no available information about any of the sensors used, these values were set at some small, nonzero positive values on the main diagonals of each matrix.

After the messages are prepared, the robot localization node needs to be configured. The configuration is done by providing a configuration file in YAML format. The parameters in the config file set up what information is available from each message and on which topic the message is being published. There is also an option to assume only 2D space, which was set to true. The list of values provided by odometry is

$$[x, y, \dot{x}, \dot{y}, \dot{\theta}], \tag{6.1}$$

where x is the X coordinate of the robot, y is the Y coordinate, $\dot{x}$,$\dot{y}$ respective velocities and $\dot{\theta}$ the angular velocity of the robot. The list of values provided by IMU is

$$[\theta, \dot{\theta}, \ddot{x}, \ddot{y}]. \tag{6.2}$$

The IMU was also set to a relative mode, which means, that the first measurement is treated as ”zero point” for all future measurements.

The set up robot localization node publishes filtered odometry messages together with a transformation information from the odom coordination frame to the base_link frame, that is attached to the body of the robot.

6.1.1 GNSS module

There is also an option to launch a second node, provided by the robot localization package, that uses information from a GNSS module and publishes odometry messages to a separate topic, but with lower frequency (approx. 1 Hz). This topic subscribes to a GNSS Fix message, that is published for example by `nmea navsat driver` package, that can connect to a GNSS module via USB. The module used for this was U-blox ZED-F9P integrated onboard of a Sparkfun GPS-RTK board.

6.2 RGBd camera + SLAM

Another method implemented was combination of a depth camera and SLAM algorithm. The camera used for this was Intel RealSense D435, that can be used together with Intel RealSense ROS node to publish all necessary information to ROS topics. The depth and image information was then used by ROS implementation of the ORB-SLAM2 algorithm[29], that publishes position and orientation of the camera with respect to the map frame and

transformation to the camera_link frame from the map frame. The ORB-SLAM2 algorithm publishes 3D position and orientation of the camera. For this case, the Z coordinate, together with roll and pitch of the camera was ignored.

6.3 Whycon

Last localization method, that was later used as ground truth in experiments, was external localization using whycon and external camera. To use this method, a camera first needs to be calibrated to compensate for its lens distortion. The calibration is done using a black and white board with checkered pattern. Given known size of one square, it is possible to estimate the projection matrix of the camera.

Figure 6.1: An example image from the camera calibration with the calibration pattern present at the bottom of the frame. The pattern appears smaller at the bottom.

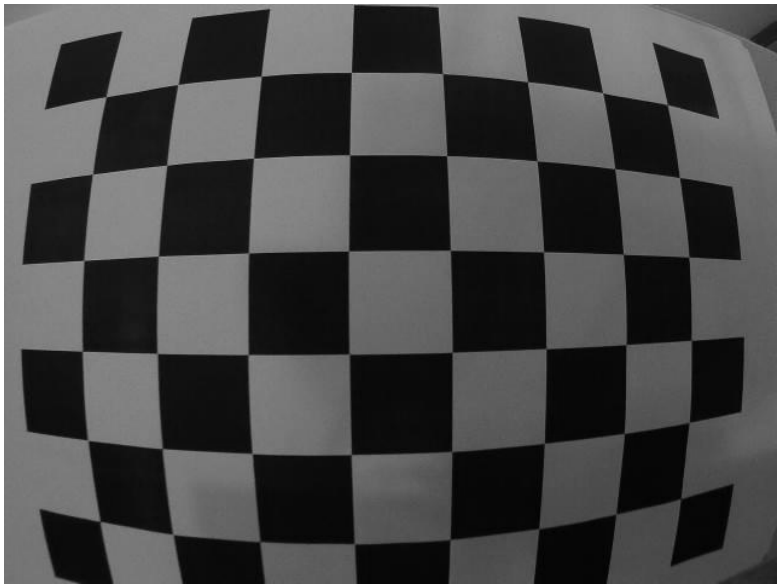


Figure 6.2: An example image from the camera calibration. The distortion of the lens is clearly seen.

After the camera calibration is finished, the whycon node is able to estimate 3D position and orientation of a black and white circle pattern. To obtain point's position in a 2-D user defined plane, further calibration is needed. This calibration is done using 4 circular patterns in a rectangle. These circles represent coordinates $[0, 0]$, $[1, 0]$, $[0, 1]$ and $[1, 1]$ of the 2-D plane. Scale parameter can be input for both x and y axises, if the distance between circles differs from the desired distances. After the calibration, 2-D coordinates of the tracked target are known. It's orientation unfortunately cannot be estimated by this method because of the circular shape of the pattern. For this, another method has to be used.

6.3.1 Odometry + IMU for orientation estimation

In order to estimate robot's orientation, another method has to be used alongside Whycon. For this, orientation estimation from 6.1 was used. The orientation of the robot from 6.1 is given with respect to the odom coordination frame. To be able to use the orientation information, first the orientation difference between the whycon 2-D plane coordinate system and the odom coordination system has to be found. The difference can be found as

$$\Delta\theta = \text{atan}(\frac{y_2 - y_1}{x_2 - x_1}) - \theta_{odom}, \tag{6.3}$$

where y_2 is the y coordinate of the robot with respect to the whycon frame after some movement has occured, y_1 is the origin y coordinate of the robot with respect to the whycon frame, x_2,x_1 are the final and origin x coordinates of the robot with respect to the whycon frame and θ_{odom} is the orientation of the robot with respect to the odom frame. The orientation of the robot with respect to the whycon frame is then

$$\theta_{whycon} = \theta_{odom} + \Delta\theta \tag{6.4}$$

Chapter 7

Implementation of a terrain classifier

The chosen approach was to use an image classifier to classify terrain into one of two classes - grass and not grass. Images were taken from already present Intel RealSense camera. This camera however had to be slightly tilted towards the ground. Furthermore, a region of interest had to be extracted from an incoming image. This region was a 224x224 square in the bottom center of an image. This resolution was chosen, because it is the input resolution of the selected image classifier.

Figure 7.1: Selected region of interest

Image classification was done using transfer learning method. When using this approach, a pre-trained convolutional neural network model is repurposed

for a different task. This results in reduced training time and training dataset needed, because some layers of the model can be frozen. This method was chosen because the training dataset, available at the time of writing this thesis, is too small to be able to train a given model, so that it would achieve better performance, than when using transfer learning method. The chosen model was MobileNetV2[30], that was pretrained on the ImageNet[31] dataset. The final fully connected layers were replaced by the following structure:

1. Fully connected (in: 1280, out: 320)
2. ReLU
3. Dropout
4. Fully connected (in: 320, out: 2)
5. log Softmax

The training images were labeled as belonging to one of the two defined classes, grass and not_grass. All images, that belong to one class were stored in one directory. The name of the directory was then the name of the class.

For training, first 12 layers of the model were frozen. The model was trained for 11 epochs with 0.0001 learning rate and with weighted loss function, with weights 1 for grass and 3 for non grass. The dataset consisted of approx. 270 images of grass and 80 images of other objects for non grass class. This dataset was split to training and validation datasets with 4:1 ratio.

The final trained classifier was then implemented as a set of 2 ROS nodes. The first node takes an image from a Intel RealSense camera as an input and extracts region of interest for the classifier. The second node takes the extracted region of interest and performs a classification. The result of the classification is then published to a ROS topic.

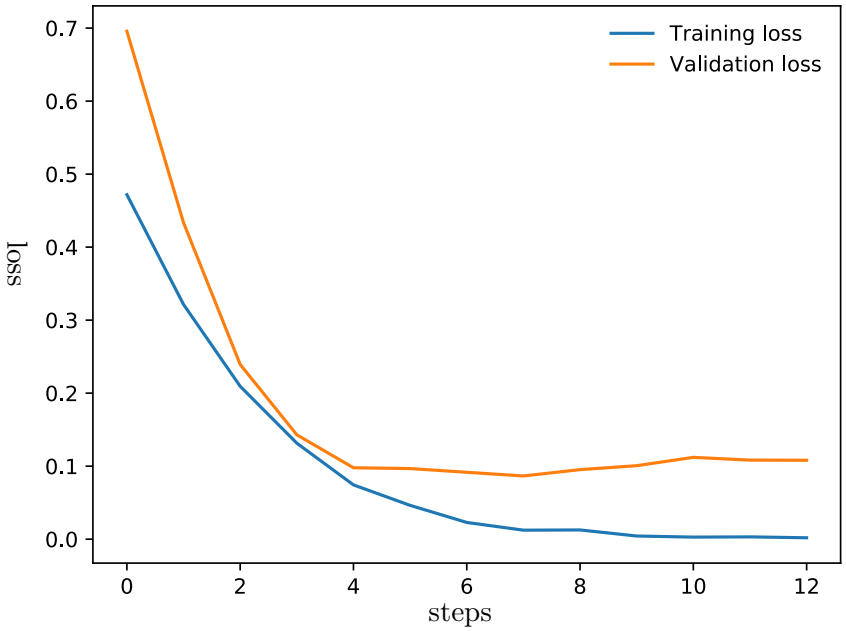


Figure 7.2: Training and validation losses of the final model

Chapter 8

Experiments

In order to perform experiments, all localization methods had to be connected. Each localization method publishes robot's position and orientation with respect to a different frame (odom for odometry, map for SLAM and world for whycon). Since the pose of the robot is known with respect to each frame, their respective transformations can be found. According to REP-105[27], the chain of coordinate frames was defined as

$$\texttt{world} \rightarrow \texttt{map} \rightarrow \texttt{odom} \rightarrow \texttt{base\_link}, \tag{8.1}$$

where base_link is the coordinate frame attached to the body of the robot. The transformation from map frame to base_link frame can be obtained as

$$T_{Mo} = T_{Mb} T_{Ob}^{-1}, \tag{8.2}$$

where T_{Mb} is the transformation from map frame to base_link frame and T_{Ob} is the transformation from odom frame to base_link frame. The transformation from the world frame to the map frame can be obtained as

$$T_{Wm} = T_{Wb} T_{Ob}^{-1} T_{Mo}^{-1}, \tag{8.3}$$

where T_{Wb} is the transformation from the world frame to the base_link frame

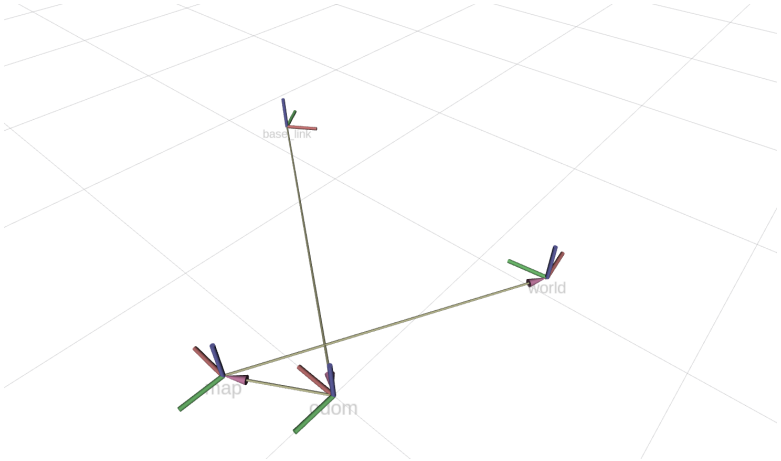


Figure 8.1: Coordinate frames chain

The definition of relations between different localization methods allows for redundancy. If some of the more high level localization method fails, the robot is still able to localize itself as the only information, that is lost is the error of the lower level localization method with respect to the higher level one. All transformations and relations were implemented using ROS TF2 library.

For evaluation of the implemented localization methods' precision, two experiments were proposed. In both of the experiments, the robot moves along pre-defined closed loop. The evaultaion of the performance of all the tested localization methods is done by measuring the final distance of the robot's position estimated from the tested method to the position according to the ground truth. All paths were defined as closed loop to eliminate any coordinate scale errors, that could be made during the calibration of the whycon localization method. In the first experiment, the robot was following a short, pre-defined path. This path was designed so the robot would always be visible to the whycon camera, that was used as ground truth and so the ground truth would be present throughout the whole experiment. The second experiment was run on a longer pre-defined path with ground truth being available only at the beginning and at the end of the path.

In order to compare the methods, all pose information, that are given with respect to each localization method's frame, had to be transformed to a one common frame. This frame was the world frame.

8.1 Localization results

In the first experiment, robot was performing 2 subsequent runs along the short pre-defined path. This was done to allow SLAM to perform loop closure. During this run, compass was not working properly due to the interference from the environment. This was also true for all later experimental runs and calibration of the sensor did not fix this issue. Therefore, the orientation information for the whycon frame was taken from the SLAM. Furthermore, for all later runs, orientation information from the IMU was not used.

	odometry	**SLAM**
distance travelled (**m**)	16.08	
distance from GT (**m**)	0.24	0.36
relative error (%)	1.5	2.2

Table 8.1: Absolute and relative error of both methods with respect to the ground truth at the end of the first experiment.

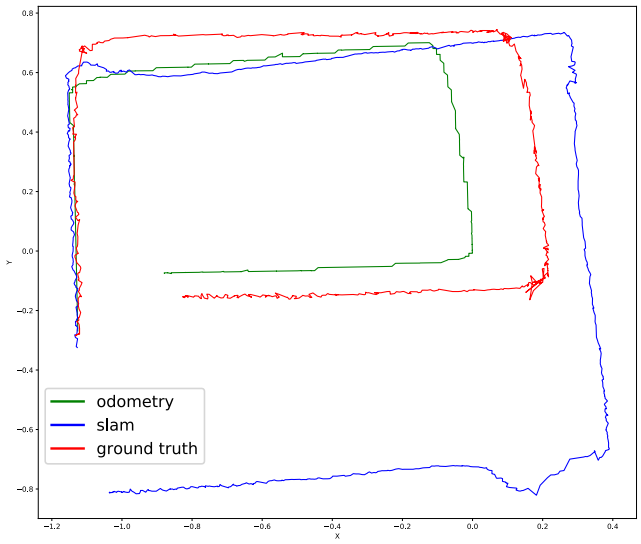


Figure 8.2: Path of the robot according to different available localization methods after the first of two loops

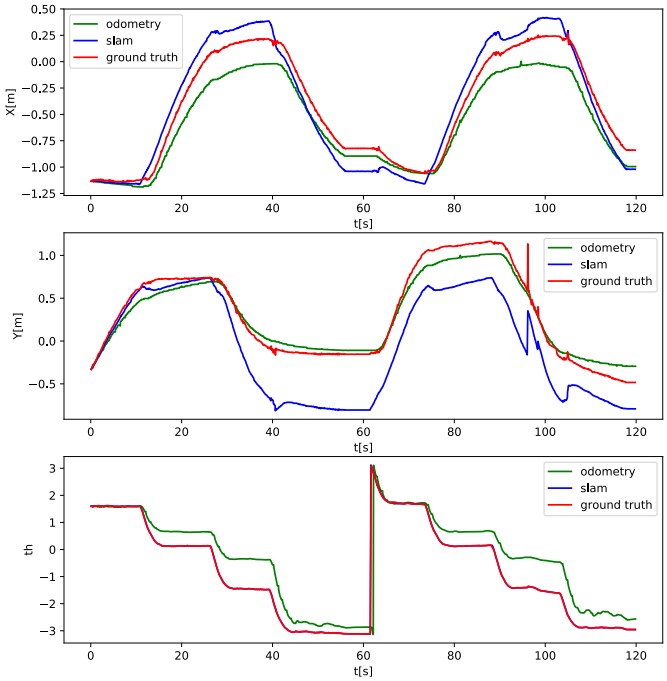


Figure 8.3: 2D pose of the robot according to different available localization methods, loop closure occured for the SLAM method

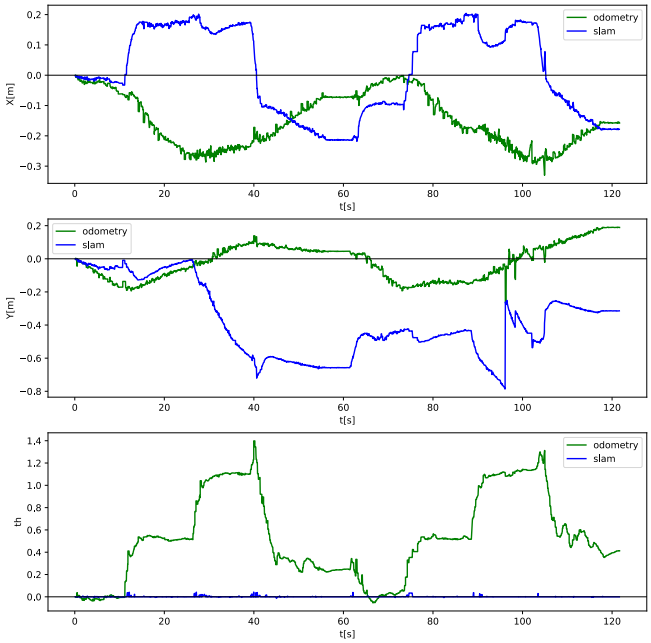


Figure 8.4: Difference of the SLAM and odometry localization methods with respect to the ground truth, ground truth for orientation is the SLAM method

As can be seen in figure 8.2, the position estimation from the odometry did not deviate too much from ground truth in the first loop, but the error of the SLAM method in the Y coordinate started to increase rapidly in the middle of the first loop. This was because the robot was facing towards much closer objects, than at the start of the loop. The depth accuracy of the Intel RealSense camera is dependent on the distance of the objects in front of the camera. At the start of the run, the opposite building and fence were more than 10 meters away, while in the second part, trees and stomps seen in the image are only 2-3 meters away. The error from the inaccurate distant readings was corrected using information from odometry, but when facing closer objects, it resulted in the error, seen on the presented figures.

Figure 8.5: An image from the start of the run, most of the the tracked features (green squares) is more than 10 meters away

Figure 8.6: Closer objects resulted in localization error of the SLAM method

During the second loop, a discrete jump in the Y position of the SLAM method can be seen. This occurs, if the SLAM algorithm recognizes the current image as already previously visited place. The algorithm can then estimate an error in position and propagate it back through the map. This is called loop closure.

The orientation error of the odometry method, visible in figure 8.4 is the result of erroneus compass readings and was corrected for later runs.

At the end of the first experiment (table 8.1), the final odometry error was smaller, than the final SLAM error. This was in part because of the large SLAM error in the Y coordinate, but also in part, because odometry tends to be very accurate on small distances if set up correctly. Both methods however have their final error less than 1 meter, which is the desired precision for this task.

	odometry	**SLAM**
distance travelled (**m**)	14.48	
distance from GT (**m**)	0.05	1.07
relative error (%)	0.3	7.4

Table 8.2: Absolute and relative error of both methods with respect to the ground truth.

In the second run (table 8.2), loop closure did not occur for the SLAM

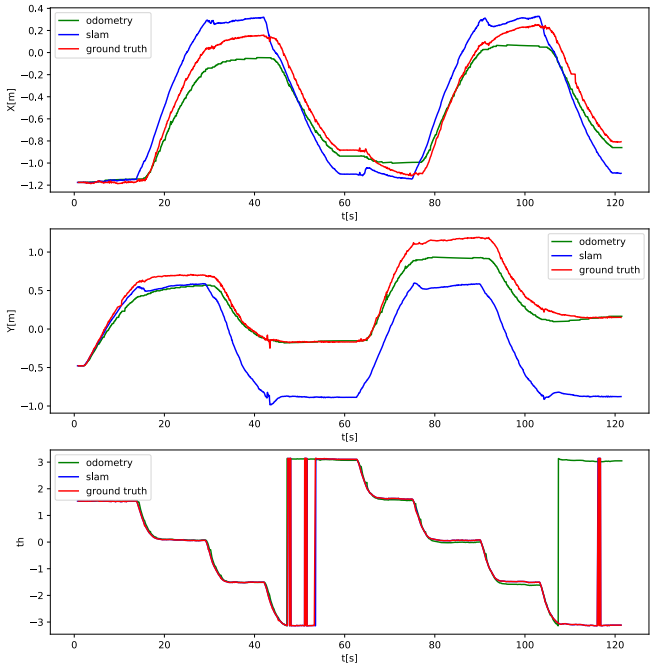


Figure 8.7: 2D pose of the robot according to different available localization methods, loop closure did not occur for the SLAM method

method. This resulted in the final error of this method being more than 1m. The corrected odometry performed extremely well during this run, with the final distance between odometry position estimation and the ground truth of only 5cm. The orientation estimation from the odometry started to deviate from the orientation estimation of the slam method in the middle of the run. There can be also seen spikes in the odometry orientation estimation during the whole run. These might be a result of wrongly set up robot localization node, or erroneous readings from the IMU. The IMU was still used as an input source for the angular speed of the robot, which is then integrated by the robot localization node to estimate orientation.

In the final experiment, the robot was expected to move twice along longer pre-defined path. This experiment however was not finished as the robot crashed at the end of the second run. This was more in part because of wrongly chosen path, that expected the robot to move very close to an obstacle at one part of the run and any slight error in localization could result in the robot crashing into the obstacle. The results presented in table 8.3 are therefore after the first run along the path.

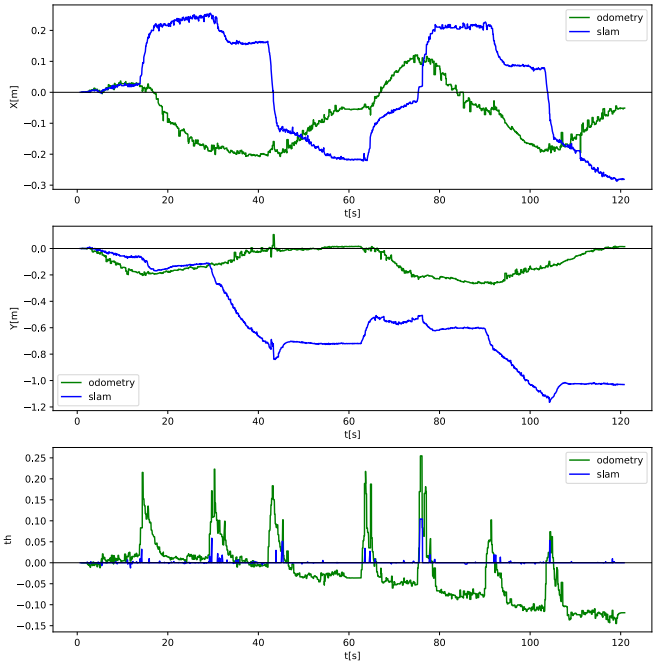


Figure 8.8: Difference of the SLAM and odometry localization methods with respect to the ground truth, orientation error of the odometry method is corrected

	odometry	**SLAM**
distance travelled (**m**)	35.91	
distance from GT (**m**)	2.41	1.68
relative error (%)	6.7	4.7

Table 8.3: Absolute and relative error of both methods with respect to the ground truth.

As can be seen in figure 8.9 and , during the longer run the error of the odometry is larger than the error of the SLAM method at the end of the first loop. This is to be expected as any errors of the odometry accumulate over time. Both methods deviated from the ground truth more than 1 meter at the end.

In all of the previous runs, GNSS module was not working properly and therefore was not used as another input for the odometry filter. Furthermore, the orientation of the SLAM camera proved to be unusable for any reliable

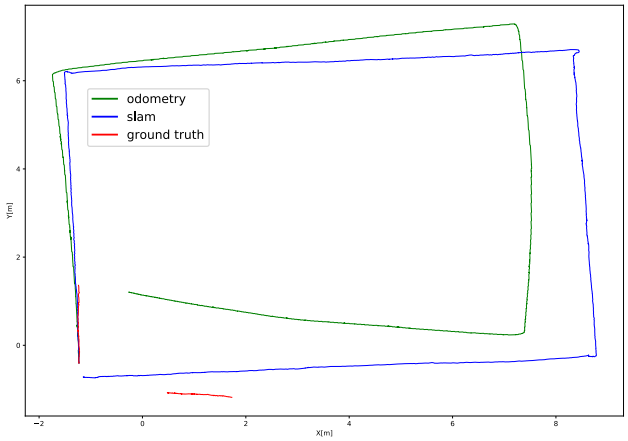


Figure 8.9: Path of the robot according to different available localization methods along the long pre-defined route

terrain classification as the bottom of the camera view is too far in front of the robot. Therefore, the Intel RealSense camera was tilted down. This resulted in the need for another transformation as the position and orientation, obtained from the SLAM method does not correspond exactly to the position and orientation of the robot. The definition of the transformation from the map frame, presented in eq. 8.2, to the odom frame needs to be redefined as

$$T_{Mo} = T_{Mc} T_{Bc}^{-1} T_{Ob}^{-1}, \tag{8.4}$$

where T_{Bc} is the transformation from the base_link frame to the camera frame.

	odometry	**SLAM**	**GPS**
distance travelled (**m**)	13.77		
distance from GT (**m**)	0.34	0.23	2.03
relative error (%)	2.5	1.6	16.4

Table 8.4: Absolute and relative error of all methods with respect to the ground truth. The camera for SLAM is tilted slightly towards the ground.

Thanks to the tilt of the camera, the SLAM method proved to be more accurate, than in previous experiments. This is mainly because more features are tracked on the ground. These features are much closer to the camera and do not move towards camera. This further helps the accuracy as the features also change their position in the image and not only their distance from the camera. GNSS however was by far the most inaccurate localization option tested with error in range of meters. This was expected as no RTK-GPS was used.

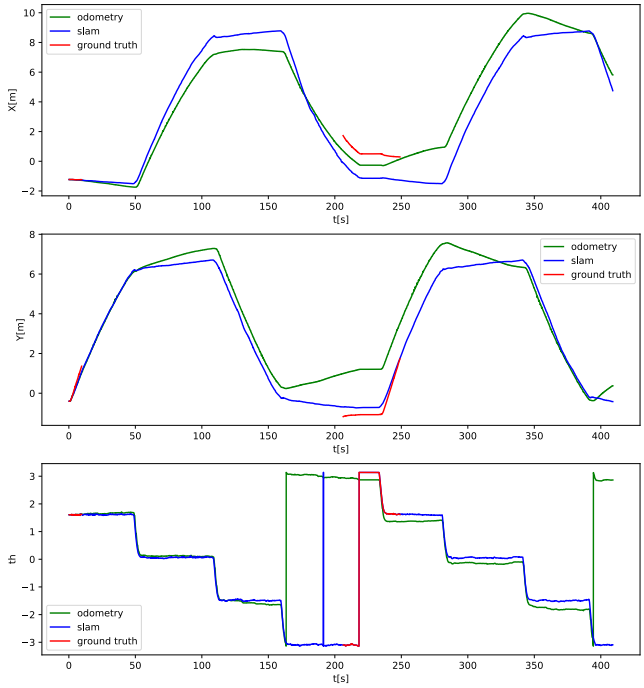


Figure 8.10: 2D pose of the robot according to different available localization methods along the long pre-defined route

During the longer run, the final error of the SLAM method stayed the same, as during the previous longer run with camera oriented parallel to the ground. This is because the distance of the tracked features is mostly consistend throughout the whole run and the error, visible in the first experiment did not show.

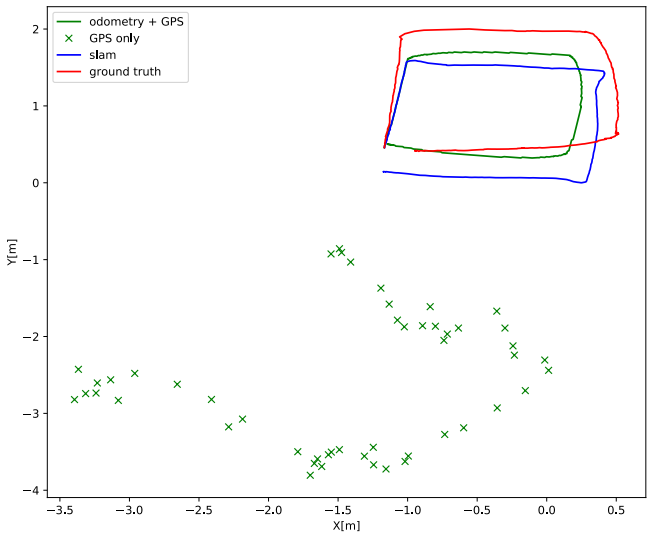


Figure 8.11: Path of the robot according to different available localization methods, GNSS data are also included in the odometry

Figure 8.14: The tilt of the camera also resulted in much more common loss of tracking

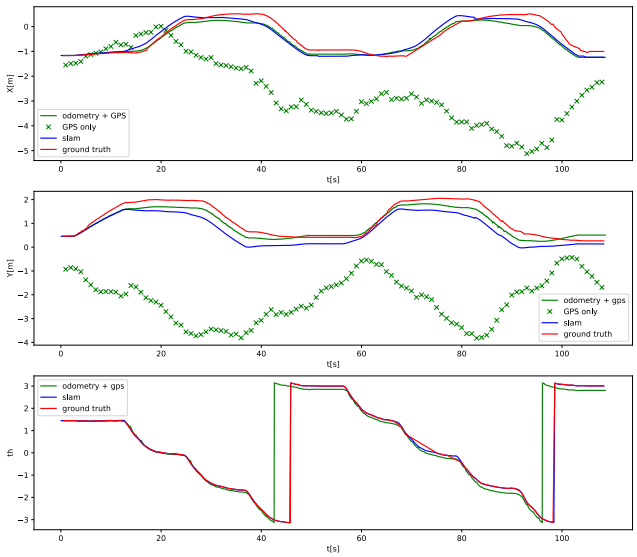


Figure 8.12: 2D pose of the robot according to different available localization methods, the camera tilt resulted in much smaller SLAM error

	odometry	**SLAM**	**GPS**
distance travelled (**m**)	31.70		
distance from GT (**m**)	1.47	1.32	9.5
relative error (%)	4.6	4.2	29.9

Table 8.5: Absolute and relative error of all methods with respect to the ground after moving along longer path. The camera for SLAM is tilted slightly towards the ground.

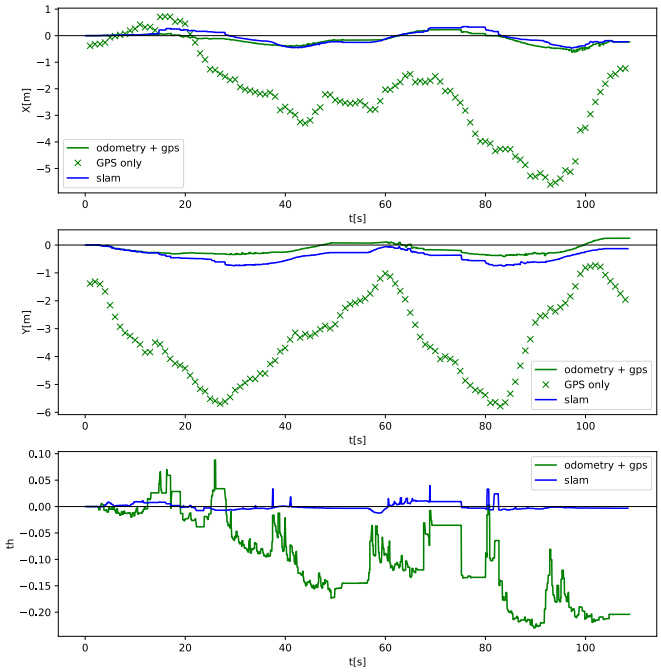


Figure 8.13: Difference of the SLAM and odometry localization methods with respect to the ground truth, the camera tilt resulted in much smaller SLAM error

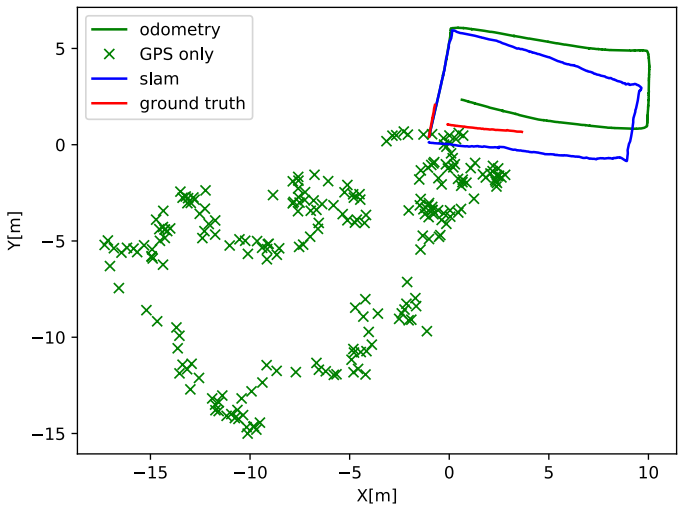


Figure 8.15: Path of the robot according to different available localization methods, GNSS data are also included in the odometry

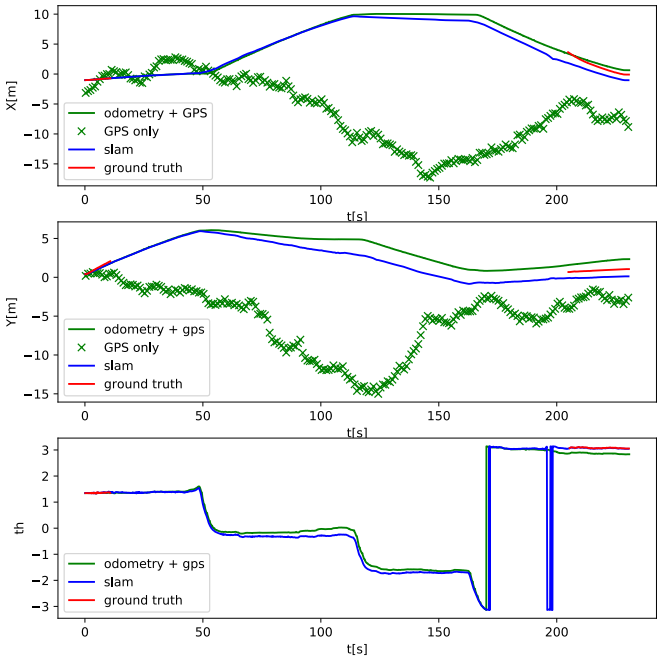


Figure 8.16: 2D pose of the robot according to different available localization methods, the camera tilt resulted in much smaller SLAM error

Overall both tested methods managed to estimate robot's position with relative error with respect to the ground truth being less than 10%. The tilt of the camera, used for SLAM eliminated the variation of the performance based on the distance of the tracked features. The localization using the tilted camera outperformed localization from odometry in both the experiments. Furthermore, loop closure is able to bind the error of the SLAM method, whereas odometry error will continue to increase over time.

Localization using GNSS only proved to be unusable in the current state. The final relative localization error of the GNSS module during the longer run was 30%. In order to improve performance of this method, second module, mounted on a base station is needed to perform RTK-GPS localization. This however further increases the cost of this method.

8.2 Terrain classification results

Evaluation of the terrain classification performance was done on a set of 60 images that were not used during the training of the model. The classification was done using CPU only. The CPU used was Intel i5 5200U @ 2.2 GHz. The model achieved 98% accuracy with an average classification time of 0.1s.

actual\predicted	grass	not_grass
grass	42	0
not_grass	1	17

Table 8.6: Confusion matrix of the model on the evaluation dataset

Figure 8.17: An example of classified images from the evaluation dataset, the model was able to successfuly classify all the images.

Chapter 9

Future work

Implemented nodes and methods, presented in this thesis are only a foundation of a fully functional robotic lawn mower. Therefore, multiple future work options will be presented in this chapter.

9.1 Controller implementation

All nodes, implemnted for this thesis do not communicate with each other. For this another node is needed. This node should take position information from one of the localization methods and terrain information from the image classifier to implement simple reactive behaviour.

Another option is to use point cloud, available from SLAM, together with the information from the image classifier to build a map of the environment. Using this map, planning and frontier exploration algorithms can be used to navigate the environment. The robot can for example first map the boundaries of the area and then calculate optimal mowing trajectories inside this area.

9.2 Charging station

Another topic, that was not researched in this thesis, but is essential for a robotic lawn mower, is search of and return to the charging station. For this, centimeter level localization accuracy is needed. This accuracy was not achieved by any of the tested methods in this thesis and further improvement of the localization precision or a different approach is needed.

9.3 Dedicated computing platform

Last, but not least a dedicated computing platform is needed. All the experiments, made in this thesis were run on a connected laptop. In the future, dedicated computing platform, presented all the time on the robot is needed.

Chapter 10

Conclusions

In this thesis, multiple low cost localization methods were researched. These methods were tested as a replacement for currently used localization methods for robotic lawn mowers, which mostly use buried wire as a boundary of the mowing area and a GPS module. The tested methods were odometry, IMU and GNSS data fused together by an extended Kalman filter[28] and an Intel RealSense D435 depth camera with ORB-SLAM2[29] algorithm. All methods were tested using a DYI robotic lawn mower called Ardumower. All localization methods were implemented as ROS nodes. To controll the robot, another ROS node was implemented that took path information as an input and outputed the desired linear and angular speed of the robot, that was then processed by Ardumower ROS driver node. As a ground truth for localization, Whycon external localization method[14] was used.

Both methods were able to achieve relative error with respect to the ground truth lower than 10%. The SLAM algorithm with the camera tilted slightly towards the ground proved to be more insensitive to the distance of the tracked features and managed to achieve better accuracy, than odometry with relative error with respect to the ground being less than 5% in both experiments. Furthermore, if loop closure occurs, the localization error of this method becomes bounded, whereas the localization error of odometry will continue to increase over time.

If both methods are used together, the robot is able to switch to localization using odometry, if the SLAM method fails to localize. The SLAM method can also be used to correct the odometry error.

For better awareness of the environment, visual classifier of the mowed terrain is presented. This classifier used the images from Intel RealSense camera and extracted a region of interest, present in front of the robot. On this region, classification to one of the two defined classes was made. These classes were grass and not_grass with the idea being to mow everything, classified as grass and avoid everything, classified as not_grass.

For the image classification, MobileNetV2[30] model was used. This model was pretrained on the ImageNet[31] dataset, its fully connected layers were replaced with custom structure, specific for the desired task. For training, first 12 layers were frozen.

The final model achieved 98% accuracy on the evaluation dataset consisting of 60 images, with an average classification time of 0.1s. This classifier was implemented as a set of 2 ROS nodes, where the first node performed extraction of the region of interest from the image and the second node performed the classification.

Finally, multiple future work options were presented, that are needed for a fully functional robotic lawn mower.

Appendices

Appendix A

Bibliography

[1] "Introduction to convolutional neural networks." https://www.vojtech.net/posts/intro-convolutional-neural-networks/. Accessed: 2020-05-17.

[2] H. Sahin and L. Guvenc, "Household robotics: autonomous devices for vacuuming and lawn mowing [applications of control]," *IEEE Control Systems Magazine*, vol. 27, no. 2, pp. 20–96, 2007.

[3] "Is there a robot lawn mower without perimeter wire?." https://myrobotmower.com/is-there-a-robot-lawn-mower-without-perimeter-wire/. Accessed: 2020-05-20.

[4] "Ardumower - do-it-yourself robot lawn mower project." https://www.ardumower.de/index.php/en/. Accessed: 2020-05-05.

[5] M. Ben-Ari and F. Mondada, *Robotic Motion and Odometry*, pp. 63–93. Cham: Springer International Publishing, 2018.

[6] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV '03, (USA), p. 1403, IEEE Computer Society, 2003.

[7] E. Royer, M. Lhuillier, M. Dhome, and T. Chateau, "Localization in urban environments: monocular vision compared to a differential gps sensor," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, pp. 114–121 vol. 2, 2005.

[8] J. Campbell, R. Sukthankar, and I. Nourbakhsh, "Techniques for evaluating optical flow for visual odometry in extreme terrain," in *2004*

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), vol. 4, pp. 3704–3711 vol.4, 2004.

[9] M. Agrawal and K. Konolige, "Real-time localization in outdoor environments using stereo vision and inexpensive gps," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 3, pp. 1063–1068, 2006.

[10] J.-M. Codol, M. Poncelet, A. Monin, and M. Devy, "Safety robotic lawnmower with precise and low-cost l1-only rtk-gps positioning," in *Proceedings of IROS Workshop on Perception and Navigation for Autonomous Vehicles in Human Environment, San Francisco, California, USA*, pp. 69–72, 2011.

[11] S. Thrun, "Robotic mapping: A survey," in *Exploring Artificial Intelligence in the New Millenium*, Morgan Kaufmann, 2002.

[12] H. P. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI Magazine*, vol. 9, p. 61, Jun. 1988.

[13] J. D. Tardós, J. Neira, P. M. Newman, and J. J. Leonard, "Robust mapping and localization in indoor environments using sonar data," *The International Journal of Robotics Research*, vol. 21, no. 4, pp. 311–330, 2002.

[14] T. Krajník, M. Nitsche, J. Faigl, T. Duckett, M. Mejail, and L. Přeučil, "External localization system for mobile robotics," in *16th International Conference on Advanced Robotics (ICAR)*, Nov 2013.

[15] M. Nitsche, T. Krajník, P. Čížek, M. Mejail, and T. Duckett, "Whycon: An efficent, marker-based localization system," in *IROS Workshop on Open Source Aerial Robotics*, 2015.

[16] M. Franzius, M. Dunn, N. Einecke, and R. Dirnberger, "Embedded robust visual obstacle detection on autonomous lawn mowers," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 361–369, 2017.

[17] P. Filitchkin and K. Byl, "Feature-based terrain classification for littledog," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1387–1392, 2012.

[18] M. Y. W. Teow, "Understanding convolutional neural networks using a minimal model for handwritten digit recognition," in *2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, pp. 167–172, Oct 2017.

[19] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 253–256, May 2010.

[20] L. Bauer, "Implementation of a computer vision algorithm for onboard detection of unmanned aircraft," bachelor thesis, Czech Technical University in Prague, 5 2018.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[22] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[23] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[24] L. Ojeda, J. Borenstein, G. Witus, and R. Karlsen, "Terrain characterization and classification with a mobile robot," *Journal of Field Robotics*, vol. 23, no. 2, pp. 103–122, 2006.

[25] "Robot operating system wiki." `http://wiki.ros.org/`. Accessed: 2020-05-19.

[26] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," vol. 3, 01 2009.

[27] "Rep 105, coordinate frames for mobile platforms." `https://www.ros.org/reps/rep-0105.html`. Accessed: 2020-05-09.

[28] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system," in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*, Springer, July 2014.

[29] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2018.

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.