UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

BACHELOR'S THESIS

Traversing algorithm based on proprioceptive measures for sensory deprived environments

Author:
Albert CLÉRIGUES GARCÍA

Supervisor:
Tomáš SVOBODA

Co-supervisor:
Germán RAMOS PEINADO

A thesis submitted to Escuela Técnica Superior de Ingenieros de Telecomunicación *from the* Universitat Politècnica de València *for the* Degree in Telecommunications Technology Engineering

Year 2015-16

Valencia, June 24, 2016

Abstract

The main goal of the diploma project is to develop a combined robot-motion and tilting-flippers algorithm for traversing harsh terrains without exteroceptive measurements, i.e. without Lidar, RGB-D cameras, etc. The robot should advance in the direction given by the operator as autonomously as possible, deciding how much distance to advance, positioning the flippers to aid in traversal and ensuring its safety. The features considered in the planning will be exclusively proprioceptive such as tactile sensor arrays in each flipper, gyroscopes, accelerometers or odometry.

Resumen

El principal objetivo de este proyecto es desarrollar un algoritmo de control del movimiento y de las aletas del robot con el objetivo de avanzar sobre el terreno prescindiendo de sensores extereoceptivos, es decir, sin información de LIDAR, camaras RGB-D, etc. El robot debe avanzar en la dirección indicada por el operador de la forma mas autónoma posible, decidiendo cuanto avanzar, moviendo las aletas para ayudarse y procurando su seguridad. Los sensores considerados para tomar las decisiones serán exclusivamente proprioceptivos como el array de sensores de presión en las aletas frontales, giroscopios, acelerómetros y odometría.

Resum

El principal objectiu d'aquest projecte es desenvolupar un algoritme de control del moviment i de les aletes de un robot amb el objectiu de avançar sobre terrenys prescindint de sensors extereoceptius, es a dir, sense informació de LIDAR, càmeres RGB-D, etc. El robot deu avançar en la direcció indicada per el operador de la forma mes autònoma possible, decidint cuant avançar, moguent les aletes per a ajudarse i mirant per la seva seguretat. Els sensors considerats per a prendre les decisions seràn exclusivament proprioceptius, com per exemple els sensors de presio de les aletes frontals, giroscopis, acceleròmetres i odometria.

Contents

List of Figures

Chapter 1

Introduction

1.1 The TRADR Research Project

This work has been entirely developed for the *NIFTi ground rover* (UGV) robot as part of the TRADR Research Project, which stands for Long-Term Human-Robot Teaming for Robot-Assisted Disaster Response. This is an EU-funded 4-year Integrated Project in the FP7 which develops technology for human-robot teams to assist in disaster response efforts, with the aim of developing persistent models for environment perception, multi-robot acting and human-robot teaming. The TRADR programme will be now briefly introduced as a way to put in context the development of this thesis.

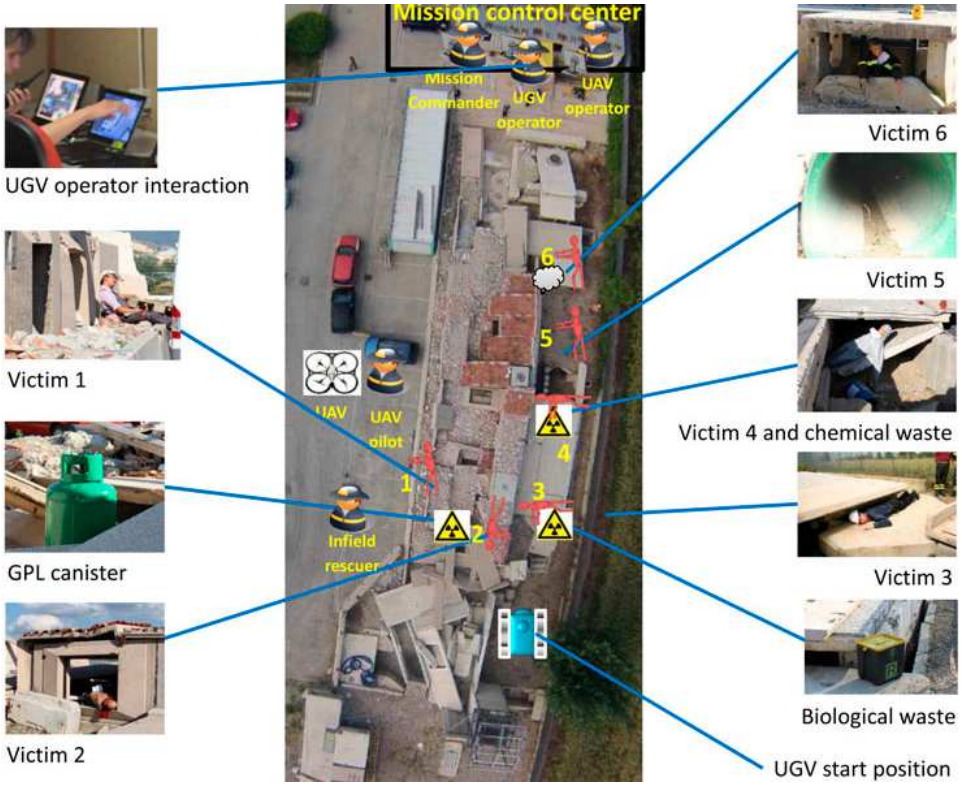


FIGURE 1.1: Scenario used to evaluate the TRADR project. [1]

1.1.1 TRADR Project Motivation

The TRADR Project spins-off from the NIFTI project (Natural human-robot cooperation in dynamic environments) specializing in disaster response, such as train accidents, tunnel disasters or earthquakes. In these scenarios, various kinds of robots *collaborate* with human team members to explore and gain consciousness of the environment. The team can consist of humans operating on the area and/or from a remote command post, NIFTi ground rovers (UGV), and quadcopters/microcopters (UAV), with an initial Human-to-Robot ratio of about 1:1, which could be further decreased in the future.

The goal is that throughout this collaborative effort team members gain a faster and deeper understanding of the disaster area, improving the coordination, planning and response time. Figure 1.1 shows an example of the kind of scenarios the project is aimed for.

1.2 The NIFTi ground rover

The robot used for this work is a small search-and-rescue robot designed to move through rough terrain and hostile scenarios. The robot is remotely controlled by an operator using the data from the robots sensors that, in its most basic configuration, include an omnidirectional camera, a LIDAR (a rotating 2D laser scanner), IMU and GPS. New sensors for the robot are being developed and improved, as this is much an R&D ongoing project.

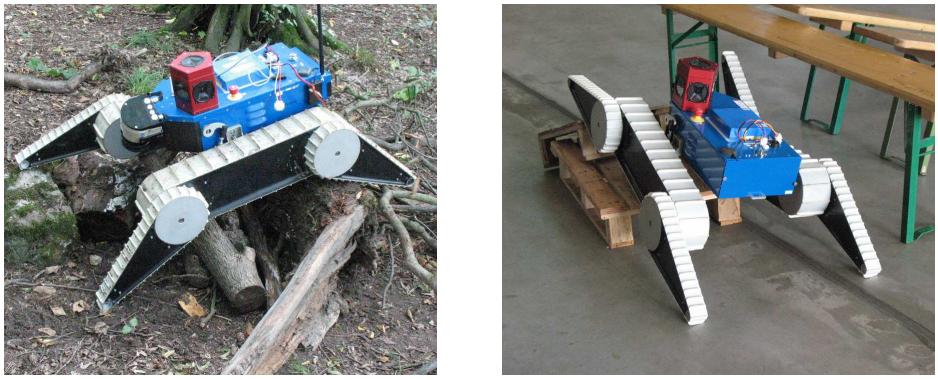


FIGURE 1.2: The NIFTi ground rover

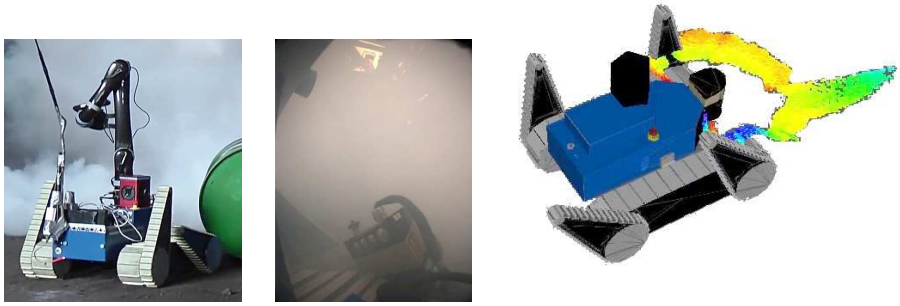
The robots traversing capabilities rely on two main caterpillar tracks to deliver most of the push, and four independent tilting flippers that aid in traversing of terrain, such as climbing obstacles or helping to stabilize. The flippers are actuated by powerful servos with a current of up to 4 A, which is enough for the robot to lift itself through obstacles of up to 40 cm.

The flippers can also be used to increase the effective contact surface with the ground and gain traction, this is critical to traverse obstacles such as stairs where the step corners are the only contact points.

1.3 Goals of this work

1.3.1 Motivation

One important role for the TRADR UGV in fire scenarios is to help firefighters by scouting indoor disaster sites and providing valuable information prior to human deployment. A significant problem has been spotted in the project reviews, where UGV going through areas filled with dense smoke has seen its camera and LIDAR rendered useless.



(A) UGV robot (B) Camera view (C) LIDAR pointcloud

FIGURE 1.3: Examples of dense-smoke related problems

The LIDAR sensor fails completely as laser beams are randomly reflected by smoke particles, creating a cloud of noise around the robot as seen in Figure 1.3 (c). This fact, combined with nearly zero visibility from the camera, leads to poor robot placement perception from the operator, whose only information for remote control comes essentially from this two sensors.

1.3.2 Flipper's force sensor array

A new branch of development was started to solve this problem with the addition of new proprioceptive sensors whose performance is not affected by smoke [2]. The most relevant is the addition of force sensors to the front flippers, consisting on an array of six force sensing elements.

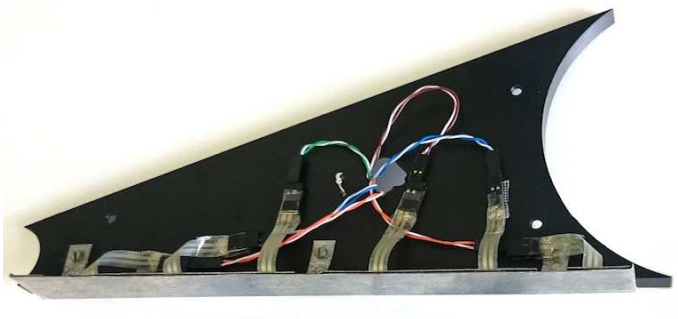


FIGURE 1.4: The flipper force sensor array

The main idea being that by pressing the flippers against the ground, the different pressures on each sensor of the array will convey information about the shape of underlying terrain as seen on Figure 1.5

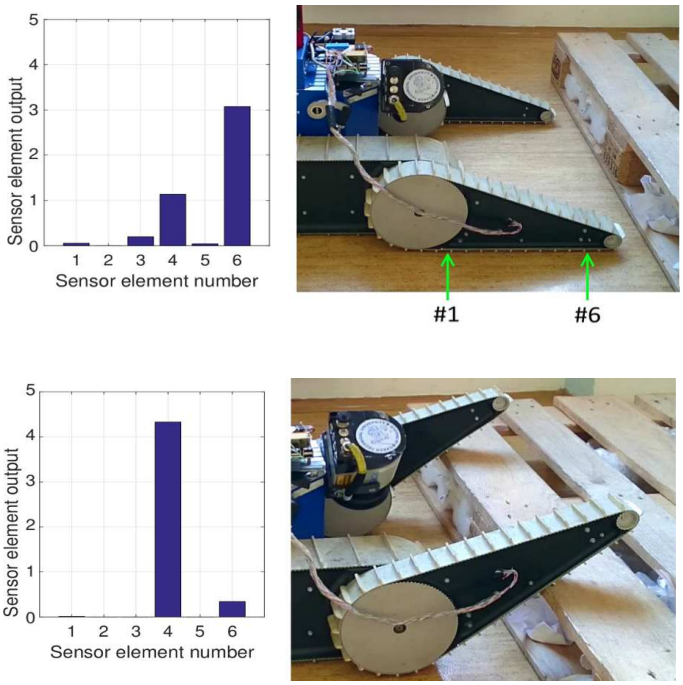


FIGURE 1.5: Force sensor array placed in the flipper along the readings

A terrain estimator was then trained using features such as the pressure sensor readings along with other, as the flipper angles at touch or internal odometry, to estimate the height profile of the terrain as discussed in [2]. The estimation gives an average height profile of the terrain with a resolution of 10 cm. under and in front of the robot as seen in Figure 1.6. Additionally, it gives the upper and lower quartiles of the height probability distribution for each estimated height, which can be used as a measure for the certainty and accuracy of estimation.

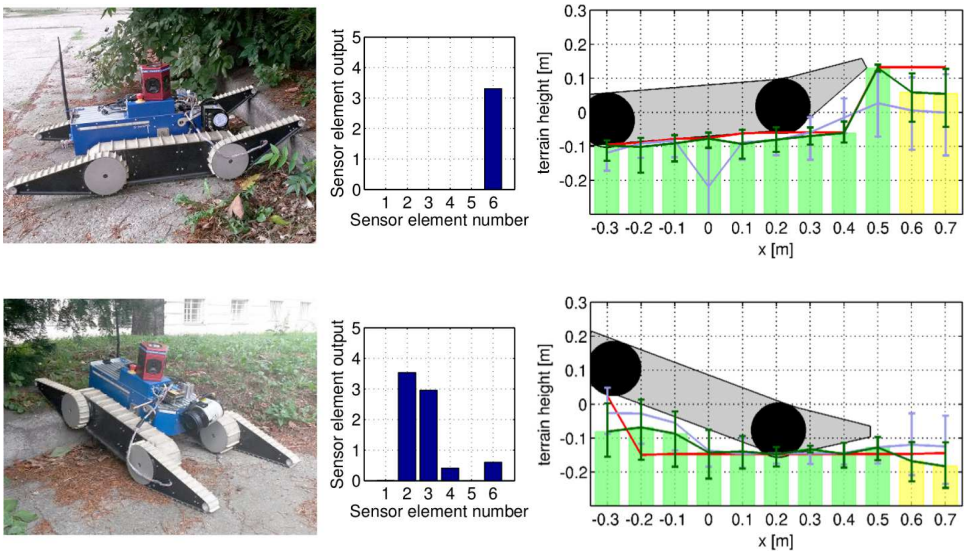


FIGURE 1.6: From left: robot in front of obstacle, measured forces in sensing elements, estimated terrain height in bins

1.3.3 Development of a Blind Traversing Algorithm

The main *goal* of this work will be to develop a traversing algorithm for the so called "blind mode" using proprioceptive sensors data, such as the estimated terrain from the tactile measurements, and setting the robot morphology as needed to traverse the terrain in forward direction while ensuring the safety and stability of the robot. The traversal has to be as autonomous as possible since teleoperation of the complex robot morphology causes high cognitive load of the operator, whose attention should be rather focused on reaching the higher-level search and rescue goals. The operator should only have to set the desired advance direction for the robot and the algorithm should do the rest.

Although other work has been made for this robot in the area of adaptive traversability [4], where the morphology of the robot is dynamically changed for successful traversal, this is not transferable to our problem as its traversing strategies rely heavily on the flippers for support as seen on Figure 1.7. These are not suited for the blind traversal, as front flippers should be free for tactile sensing to the extent possible. Instead, a new set of flipper positioning traversing strategies will be proposed to solve this problem.

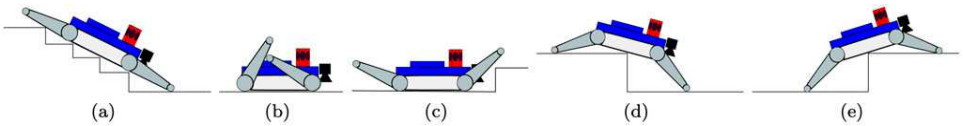


FIGURE 1.7: Flipper configurations for the adaptive traversing algorithm, inadequate for blind traversing

The scope of this work aims at traversing symmetrical relatively simple obstacles such as steps, stairs or ramps. Asymmetrical obstacle traversing or angled approach to obstacles would require the addition of tactile sensors to the back flippers and independent estimation of terrain under left and right track, which at the time was still being implemented. Thus, this work aims to serve as baseline and to uncover the challenges and problems to account for future development.

1.4 Technical Specifications and Planning

This work will be developed on ROS.org (Robot Operating System) which will be introduced later, and the algorithm will be implemented in Python. It will be deployed on the UGV introduced in Section 1.2, which will be equipped with two force sensor arrays in each of the front flippers. These will produce a terrain estimation consisting on a single one-dimensional array representing the average height of the terrain between the two flippers. In parallel to the development of this work two more force sensor arrays were being developed and added to the rear flippers to extend the profile to also include the rear section of the robot. Furthermore, independent terrain profiles for right and left tracks were also being added to the

setup. This work does not benefit of those advantages but has been made and implemented accounting for this future imminent additions.

The work has been developed in the span of 4 months, from the 18th of February up to the end of June of 2016, being developed in two main stages:

First stage (18 February - 18 March) – consisted on the introduction to ROS.org and the robot environment by developing a simple prototype of the algorithm that can climb and descend a standard ISO European pallet for the 2nd Year review of the TRADR Research Project in Dortmund, Germany. This was developed in tandem with a member of the CMP in the beginning but progressively being more individual. The review and evaluation of this first version of the algorithm was satisfactory and encouraged further development.

Second stage (22 March - 20 June) – production of the final version of the Blind Traversing algorithm. The software architecture, blind strategies and tests were done in this phase. The work from the first stage is not included in this document as it was essentially a learning exercise on which the next version would be built.

Chapter 2

The Blind Traversing Algorithm

2.1 Algorithm Overview

We have approached the blind traversing algorithm real-time operation dividing its operations in an arbitrary number of sequential stages, each stage is comprised of three consecutive actions that the robot will execute. The operation will consist on executing stage after stage, where each action of the stage is executed according to the settings associated with it for that particular stage:

1. Set flippers to desired shape.
2. Advance forward a certain distance (20 cm).
3. Lower the flippers to perform tactile sensing (or touch) with the front flippers in order to estimate the height profile of the terrain.

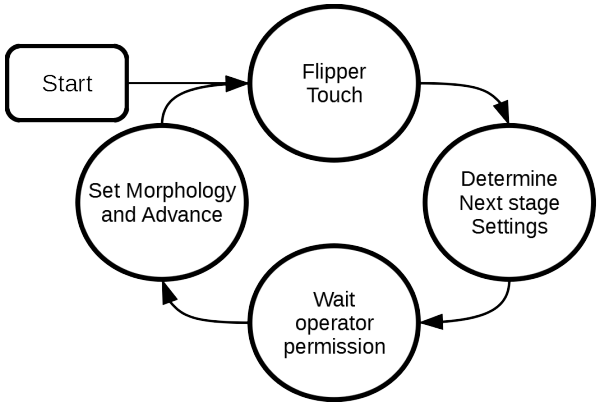


FIGURE 2.1: Diagram describing actions executed in one stage

The first action will apply the optimal robot morphology to traverse the next stage, this is achieved by adjusting the angle and torque of each flipper according to the values of the stage options. Then the robot moves forward using its caterpillar tracks until it reaches the next measuring point and stops. The robot then lowers the front flippers and pushes them to the ground to get information about the terrain shape as explained in 1.3.2. After this touch action, an estimation of the terrain is made using the force sensor array data in each of the front flippers, which is then used along with other proprioceptive sensor measures to determine the optimal settings for

the actions in the next stage, i.e. set morphology, advance and touch. The information and decisions made for the next stage are then presented to the operator, that should acknowledge them and give the algorithm permission to execute the next stage as planned.

2.1.1 Traversing with only proprioceptive sensors

The approach described above responds to many limitations involving the use of proprioceptive sensors. The algorithm has an stop-and-go motion, where the robot has to stop to perform the touch action, then wait for the height profile estimation, compute the optimal settings to traverse the estimated terrain and finally wait for the operator to validate the plan to continue its movement. Among other things, this is motivated by the static tactile measuring procedure used to train the terrain estimator and the range of the flippers tactile sensors of about 35 cm. that hugely limits the anticipation of obstacles and operator awareness of the environment.

This operation, although perfectly functional, presents many drawbacks such as low operator interactivity, that receives information of the terrain shape every 10s., and slow effective advance speed. However, this operation has been chosen as it is reasonably effective and copes better with the restrictions of working with this kind of sensors:

- Having null terrain anticipation, the robot has to advance with their flippers up in case it encounters an obstacle, so it will begin to climb it instead of crashing into it. If the flippers have to be upwards while advancing, this means it is unsafe to perform the tactile sensing while in movement and it is necessary to stop and remain still while this action is happening.
- Each time the touch action is performed, an estimate of the front 30 cm. is made, hence it is safe to advance only around 20 cm. before the terrain in front of the robot is sufficiently uncertain to have to measure it again.
- A constant and specific torque has to be applied downwards when performing the touch in order for the force sensors to be in their linear zone. The robot has to remain still as moving tracks would interfere with the tactile sensing, and moreover, the terrain estimator had already been trained with static measuring.

2.2 Software Architecture

2.2.1 The Robot Operating System (ROS)

The robot is controlled and operated using the software ROS.org, a highly modular and flexible framework for writing robot software. ROS organizes the control and operation of a robot using three main logical components: nodes, topics and messages. A node really isn't much more than an executable file that uses the ROS client library to communicate with other

nodes. For that purpose, a node will register as a publisher or subscriber to a *topic*, where *messages* flow from publisher to subscribers.

The components described above are the building blocks of a very powerful architecture where each node can receive and broadcast information system-wide. ROS.org uses a hybrid P2P system for communications, which allows for nodes controlling the same robot or robots to be distributed on multiple computers, sharing data and commands. A basic example diagram illustrating this concepts is found on Figure 2.2.

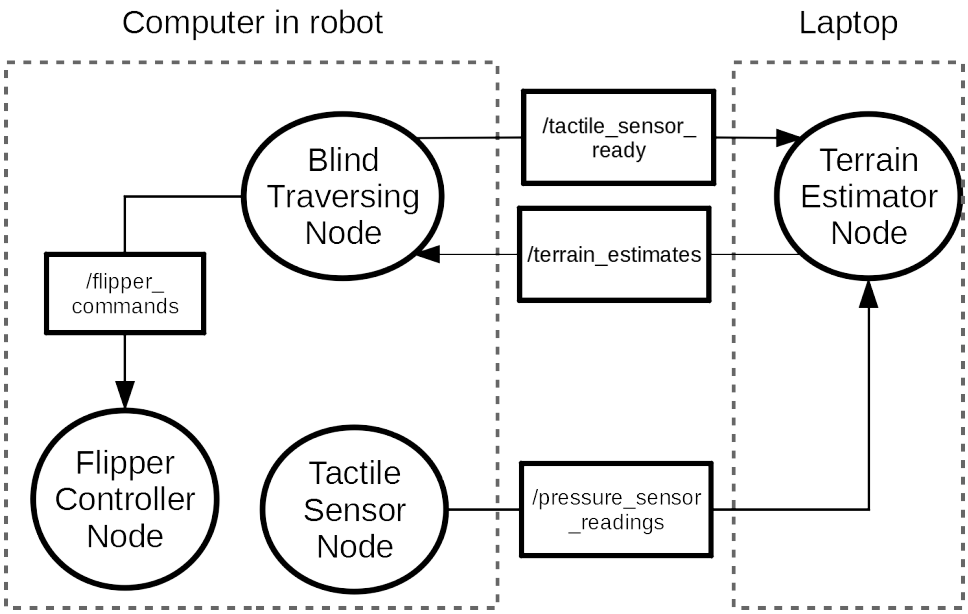


FIGURE 2.2: Part of the ROS node diagram of the Blind Traversing node. Nodes are circled and topic names are squared.

2.2.2 Blind Traversing Node Graph

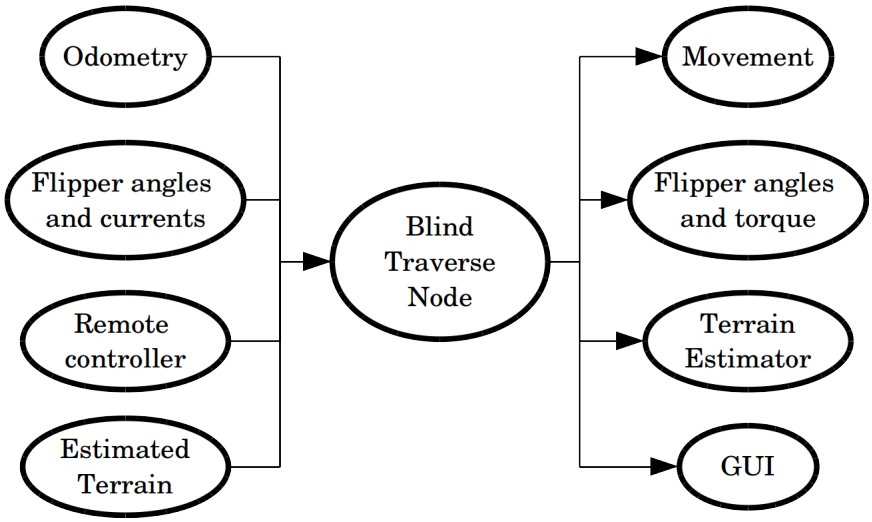


FIGURE 2.3: Interaction between the Blind Traverse node and the nodes present in the system

The *Blind Traverse* node is also part of a bigger system of nodes present in the robot. The most important subscriptions include the remote controller commands, the flippers actual current and angle, the odometry, which includes position, pitch, roll and yaw, and the estimated terrain. Similarly, it publishes to topics controlling the movement of the robot, the angle of the flippers and its maximum torque, the Graphical User Interface (GUI) and sending the tactile sensor data to the terrain estimator.

2.2.3 The Blind Traversing Node

The blind traversing node internal class architecture has been divided into four main classes, to provide flexibility and modularity in future development:

- **Blind Controller** class which contains the main loop, serves as a central hub for control, coordination and staging of the blind mode.
- **Kinematic Controller** class that serves as a high level interface for the node to easily control the robot actuators.
- **Stage Determination** class which receives sensors data and terrain, analyzes them and determines the settings for the next stage.
- **IOController** class receives and packs the robot sensor information for the other controllers as well as communicating other system related functions.

All the algorithm has been implemented as a standard real-time control loop where updated sensor data is gathered at least once per iteration. The functions have been implemented to work on a step-by-step basis with persistent state, processing the updated real-time data in each iteration and then updating its state variables or executing the appropriate instructions.

IOController

The IOController class manages, among other things, the Blind Mode control service, which serves as an interface for other ROS.org nodes to toggle the operation of this node. It can also display text in RViz, a visualization tool that reconstructs and shows the 3D maps and position of the robot, an example of RViz and the displayed text can be seen in Figure 3.7(A).

Most importantly, it manages the sensor reading updates: everytime a new reading is made from any sensor, the corresponding IOController callback function is called that updates the value in the Loop Variables. Every time an iteration of the main loop is performed, a copy of the updated Loop Variables is requested from the IOController and distributed to all controllers in the node. The Loop Variables include information such as:

- **Pressed Buttons** pressed state of the buttons in the remote controller. Currently using the '1' button to toggle the Blind Mode, the '4' to give permission to execute next stage and '3' to repeat the touch operation.
- **Flipper angles** tilted angle of each flipper.

- **Flipper currents** electrical current the servo in each flipper is using to move it or to keep it in place.
- **Internal odometry** Position vector and euler angles.
- **Estimated Terrain** represented by the median, upper and lower quartile height profiles.

Stage Determination

The Stage Determination class analyzes and processes the sensor readings and estimations to produce the optimal settings to traverse the next stage. The decision algorithm will be explained later in Section 3.2.2. The stage settings that the algorithm determines are composed of the next options:

- **Movement Options** This include a boolean to enable/disable movement for that stage, the distance to advance in meters and the speed to do so.
- **Front and Rear Flipper Options** In each stage, two independent settings are computed for front and rear flippers, these include:
  - **Marker Text** Text to be displayed in RViz for operator.
  - **Angles** Angles of flippers to get the right morphology. If this value is null, no change is made to the angle of the flippers in next stage.
  - **Torque** referred to the maximum electrical current that the servos can apply to move the flipper or keep it in place.
  - **Touch Options** settings for the touch operation, that include a boolean to toggle touch for next stage and an option to return the flippers to the original position after the touch operation.

Kinematic Controller

The Kinematic Controller provides an easy interface to control the movement speed of the robot and the position of the flippers. These are used to form high-level function to perform the three actions in each stage: set morphology, move and touch. All actions are executed in a step-by-step basis, each step corresponding to one iteration of the control-loop, resulting in the functions `flipper_shape_step(stage_options)`, `touch_step (stage_options)` and `movement_step(stage_options)`, where each one uses only the required options from stage_options.

The `touch_step` function returns a Touch Report object in each iteration, which contains:

- **Touch Completed** boolean that indicates Blind Controller if the touch action has finished.
- **Dangerous Hole**, boolean indicating that the terrain in front of the robot is a dangerously deep hole, this will be seen in Section 3.3.
- **Angle at touch** angle of each of the four flippers when touching the ground used by the Stage Determination class to detect dangerous terrain and output the best settings for the next stage.

Stage Operation Overview

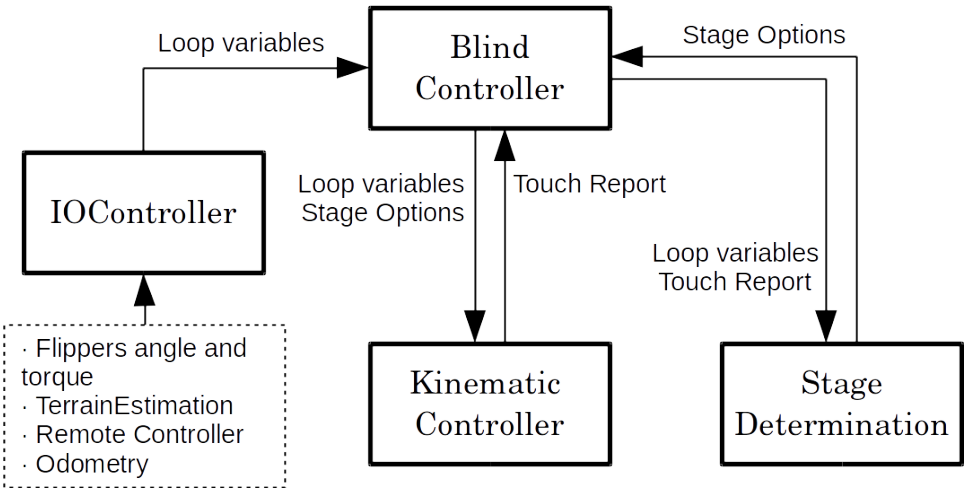


FIGURE 2.4: Diagram illustrating the flow of data through the classes

For a given stage, the data flow and class interaction is exemplified in the Figure 2.4, and proceeds as follows:

- The Kinematic Controller sets the flippers to the desired shape, moves forward and performs the touch operation. In initialization, the shape and movement options are given blank, so they are not executed.
- While executing the touch action, when all flippers are in contact with the ground, the terrain estimator node is informed that the tactile sensor array data is ready to be captured and processed.
- Once the touch action is finished, it wraps important information from the action into a packet called *touch report* where information such as the flippers angle at touch are included. The robot then waits for the terrain estimation to be done by the Terrain Estimation node.
- When the terrain estimation has been computed, it is then inserted into the *loop variables*, and then passed to the Stage Determination class along with the touch report. Then, the optimal settings for the next stage actions are computed using this information and are then sent to the Blind Controller.
- The Blind Controller will display the planned settings for the next stage to the operator and await for permission to start the new stage.

Chapter 3

Blind Traversing Strategies and Stage Determination

3.1 Blind Traversing Strategies

As stated in Section 1.3.3 a new set of traversing strategies is needed in order to traverse the terrain while leaving the front flippers free as much as possible. A *strategy* is defined as a combination of certain settings for the shape, move and touch actions to traverse a particular terrain.

To implement these strategies the decision has been made to independently compute and apply *flipper settings* for front and rear sections, configuring parameters such as flippers angle and torque, performing or not tactile sensing and the distance to advance. This provides great flexibility when facing situations with multiple obstacles. With this in mind, different predefined flipper settings have been proposed to implement the strategies:

Predefined Front Flipper Settings

Detection Default position at 45º up for traversing.

Flat Aligned with ground plane, used to traverse slopes.

Hole Keeps the flipper on angle at touch.

Soft Softens flippers to pivot smoothly.

Predefined Rear Flipper Settings

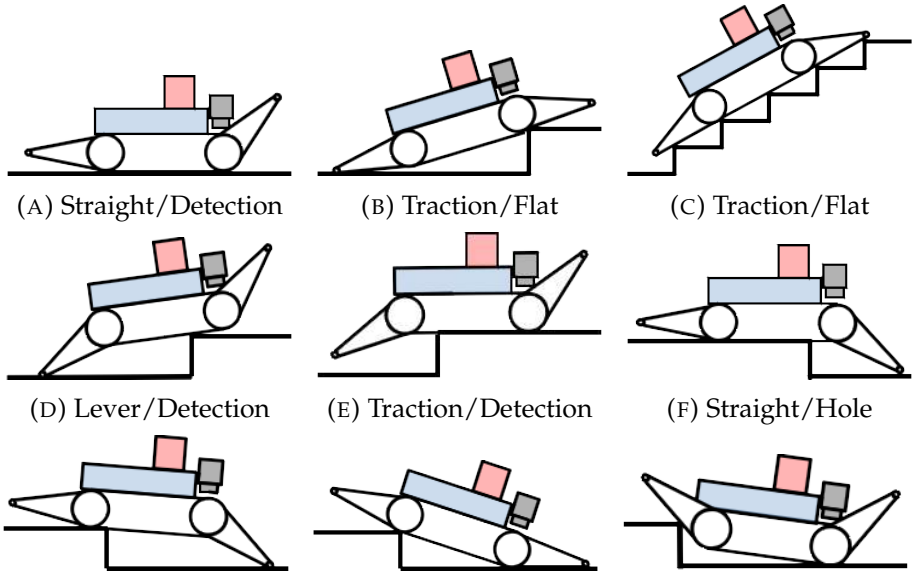
Straight Default position at 0º for traversing.

Lever Lifts the body to pivot over obstacle.

Traction Soft torque when flipper is lower than robot ground plane. Strong torque when flipper tries to go up getting locked at 0º.

Soft Softens flippers to pivot smoothly.

The situations depicted in Figure 3.1 exemplify the expected computed settings given the terrain. These strategies try to maximize the control and predictability of the blind traversing, where one strategy shape naturally leads to the next stage.



(A) Straight/Detection (B) Traction/Flat (C) Traction/Flat

(D) Lever/Detection (E) Traction/Detection (F) Straight/Hole

(G) Straight/Hole (H) Straight/Soft (I) Soft/Detection

FIGURE 3.1: Rear/Front settings for the Blind Traversing Strategies.

These strategies are more passive than the Adaptive Traversing (AT) ones in Figure 1.7 and try to address some issues such as:

Uncertainty of obstacle top – it is not known if the obstacle being climbed continues up, flattens or goes down, hence, we try to climb with the front flipper upwards to prevent getting stuck or not sensing the upcoming terrain correctly. Figure 1.7(e) shows how in AT the front flipper is pushed down to get grip on top while climbing an obstacle, to compensate for the loss of traction the rear flipper is used to provide the necessary grip.

Holes – As the terrain after and at the bottom of the hole is unknown, the pivoting and descent is delayed as much as possible to avoid descending on a dead end. We say a hole is detected if the front flippers descend beyond a certain threshold when executing the touch operation. When a hole is detected, the front flippers are kept a little bit up from the touch angle as seen in Figure 3.1(F), this way we increase stability and speed of touch operation and delay the descent. We descend only when the robot naturally leans towards the front and the terrain has been sufficiently probed.

Stairs – Given the uncertainty of obstacle top commented earlier and the inability to properly recognise stairs and complex slopes that need critical traction, a simple solution has been adopted: when the robot is pitched up the front flippers go flat aligned with the ground plane. This simple change increases the grip when the obstacle continues up, like in stair climbing Figure 3.1(C), without reducing the ability to sense and prevent from unknown terrain features, as the flippers are still raised up.

Predictability – The strategies have been designed to provide clear and meaningful magnitude changes to clearly mark the transition from one strategy to another. For example, the decision to soften front flippers to end descent over a hole as in the Figure 3.1(H) is given by the increase in force the front flippers have to do to keep shape when the robot starts supporting in them, which is a natural consequence of advancing over a hole. This will be seen and explained in the next section.

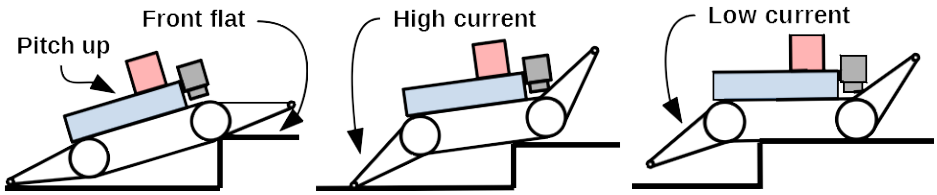
3.2 Stage Determination

Once the strategies have been laid out, a decision algorithm is needed to correctly identify the terrain based on sensor readings and output the optimal strategy to traverse it. The decision algorithm selects the optimal predefined flipper settings to traverse the next stage and returns them.

3.2.1 Strategy Design Overview

As stated before, the Blind Traversing strategies have been designed to naturally flow from one another at the time of traversing an obstacle. Despite this fact, the decision algorithm has been implemented with flexibility in mind without using state-machines or anything like that, this will be detailed later. Before describing the algorithm it seems logical to explain and show the main markers and indicators for choosing the correct strategy.

Climbing Strategies Overview



(A) Traction/Flat (B) Lever/Detection (C) Traction/Detection

FIGURE 3.2: Rear/Front settings for obstacle climbing with main variable indicators for transition decision.

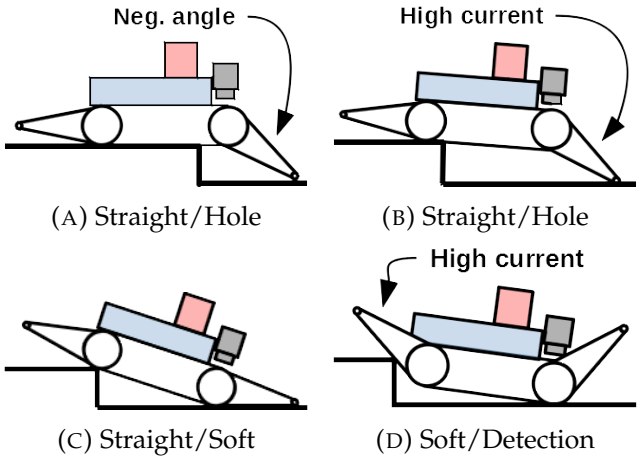
1. Starting from the default position Figure 3.1(A), we encounter an obstacle and climb over it, as the pitch raises above the *slope threshold* the front flippers automatically alineate with the robot ground plane as seen on Figure 3.2(A).

2. If pitch is above the *slope threshold* and the ground in front of the robot is flat, as Figure 3.2(A) points out, this indicates that we have topped the climb and should now pivot over the obstacle by levering with the rear flippers as seen on Figure 3.2(B).

   If the ground in front of the robot was not flat it would mean that the obstacle continues to change elevation and the flippers would keep the optimal configuration for slope climb which is the same as in Figure 3.2(A).

3. While the pivoting hasn't been completed the robot rests over the back flippers causing a high current in the servo to keep the flipper in position. The current in rear flipper servos is used as an indicator to detect when the robot pivots and tops the obstacle, as seen on Figure 3.2(C), where the rear flipper indicates now a low current. At this point we engage back traction in the back flippers to finish the climb safely and stable.

Descent Strategies Overview



(A) Straight/Hole (B) Straight/Hole

(C) Straight/Soft (D) Soft/Detection

FIGURE 3.3: Rear/Front settings for obstacle descent with main variable indicators for transition decision.

1. While traversing terrain and measuring, the *touch report* indicates that the angle of touch is below the *hole threshold*, the Hole preset is selected and flippers are set a little bit higher than the angle of touch as seen in Figure 3.3(A).
2. The robot keeps advancing until it naturally pivots and starts supporting on its front flippers, consequently the current in front flippers raises to keep them in place as seen on Figure 3.3(B). This raise in current is used to determine when the robot is ready to end its descent by softening the torque in front flippers as Figure 3.3(C) shows.
3. Once the robot has descended, it keeps advancing until the rear part starts to support onto the rear flippers, this raises the current in the rear servos to keep flippers in place. The increase in current in back flippers is then interpreted to soften the back flippers and softly descend the rear part of the robot to the lower level like in Figure 3.3(D).

3.2.2 Stage Determination Decision Algorithm

The decision algorithm has been implemented as a handcrafted `if-else` decision tree where the front and rear flipper configurations are assigned independently. A machine learning algorithm would certainly perform better than a handcrafted if-else algorithm, however, the decision was made to select the last one given that there was no prior work to rely on and the terrain estimation awaiting for better and richer estimations.

Selecting and training a machine learning algorithm could be on its own a complete bachelor's thesis, but in this case, also the software architecture had to be designed and implemented. The final decision, given the time and amount of work restrictions, was to do a handcrafted if-else decision tree that suffices for the scope of this work.

The front flipper settings decision is made according to Algorithm 1:

Algorithm 1 Decision algorithm for front flipper settings

```
1: if front flipper angle at touch < hole angle threshold and pitch < pitch threshold for slope then
2:     if front flippers current > min current threshold support then
3:         front flipper settings = soft
4:     else
5:         front flipper settings = hole
6:     end if
7: else
8:     if abs(pitch) > pitch threshold for slope then
9:         front flipper settings = flat
10:    else
11:        front flipper settings = detection
12:    end if
13: end if
```

The front flipper decision algorithm is rather simple because the touch operation is really the main usage of the front flipper, it is divided in two conditional decision blocks: hole and obstacle case.

- The hole case (line 1) is triggered when the front flippers have touched the ground at a sufficient negative angle and the robot is not in a slope, it then checks if the robot is supporting on the front flippers or not to produce the final descent as explained previously in Descent Strategies Overview (Section 3.2.1).
- The case for obstacles (line 7) then checks if the robot is on a slope to assign either front detection or flat settings to deal with stairs as explained in Section 3.1.

Algorithm 2 Decision algorithm for rear flipper settings

```
1: if previous rear settings == lever then
2:     if rear current > max current threshold not support then
3:         rear flipper settings = previous rear flipper settings
4:     else
5:         rear flipper settings = traction
6:     end if
7: else if pitch horizontal and rear current > min current threshold support then
8:     rear flipper settings = soft
9: else if pitch > pitch threshold for slope and terrain front flat then rear flipper settings = lever (big or small)
10: else
11:     if abs(pitch) > pitch threshold for slope then
12:         rear flipper settings = traction
13:     else
14:         rear flipper settings = straight
15:     end if
16: end if
```

The rear flipper settings are assigned using rather more complex conditionals, given that they are the main traversing flippers as seen in Algorithm 2:

- The first conditional aims at delivering levering consistency once it has been triggered, it was seen in experiments that interrupting levering caused serious safety issues. The current applied to the rear flippers helps determine if we have stopped supporting on them and it is safe to stop the levering.
- The second conditional in line 7 checks the conditions for the end descent strategy to lower the rear part of the body smoothly as seen in Figure 3.1(I).
- In line 9, the conditional checks if the robot is on top of an obstacle and should start the levering maneuver, it does so by looking if the robot is on a sufficiently ascending pitch and if the terrain in front is flat. There are two levering presets, one for small obstacles and another one for big ones, that are chosen based on the pitch of the robot, the taller the obstacle the bigger the pitch.
- Lastly there is the default case if none of the above applies, which also checks the pitch of the robot to apply either the stairs/slope setting or the flat settings.

Decision thresholds and parameters definition

The Stage Determination decision algorithm features many thresholds and parameters in the conditionals, these have been defined attending to specific criteria:

Pitch thresholds – The thresholds related with the pitch have been assigned attending to some basic parameter definitions. First, an obstacle has been defined as a sufficiently high change in height to cause abrupt transition, which has been experimentally set to a change in height bigger than 6cm. between bins (10cm. long).

- The *pitch horizontal* threshold has been defined as the pitch of the robot when standing on top of the smallest obstacle. Which is computed as the inverse sin of the length of the track (0.5 m.) divided by the height of the obstacle (0.06m.):

$$\arcsin\left(\frac{0.06}{0.5}\right) = 6.89^\circ \approx 7^\circ \tag{3.1}$$

- The *slope threshold* has been defined as the pitch that the robot would be in at half climb on the 6cm. obstacle, this roughly being two times the *pitch horizontal* threshold at 14°.
- Finally, the *pitch threshold for big or small lever* has been determined experimentally at 30°, this is used when starting the lever maneuver, where the pitch is checked to either do a big or small lever to lift the robot more.

Current thresholds – The current thresholds have been entirely determined through experimental results, resulting in:

- **Min current threshold support** is the minimum current from which we determine the robot is supporting on those flippers, set at 1 A.
- **Max current threshold not support** is the maximum current up to which we determine the robot is not supporting on its flippers, with a value of 0.5 A.

Hole angle threshold – is the angle of the flippers at touch from which it is determined that there is a hole in front, it has been experimentally set to avoid false positives at -22.5°.

Terrain front flat – is a measure that the terrain in front of the robot is sufficiently horizontal to consider it flat. Its calculation will be detailed in Section 3.2.3.

3.2.3 Terrain Analysis

In the beginning of the project, a huge portion of the Stage Determination was done based on a terrain analysis of the height differences of the estimate. However, as development advanced, many issues appeared related with poor under-track estimation, outliers and a prototyping training data set which, for example, did not include stairs.

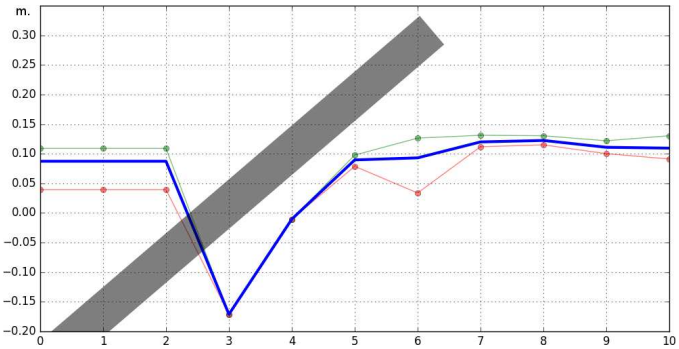
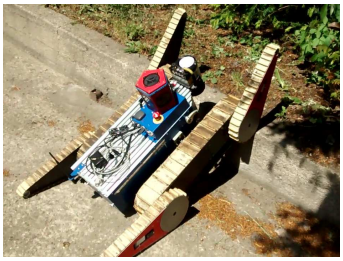
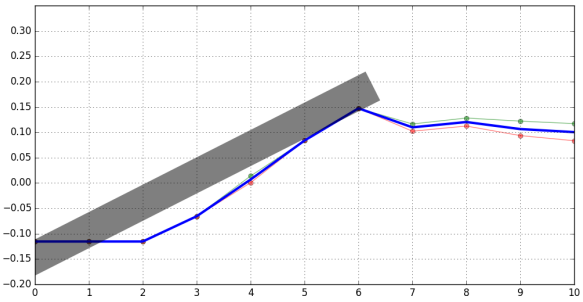


FIGURE 3.4: Terrain height profile with a rear estimation glitch, the black line symbolizes the robot, it is clear that the height of bins 0, 1 and 2 is not right as the robot was not underground.



(A) Ground Truth



(B) Estimated Terrain

FIGURE 3.5: Step being aproximated as a ramp due to poor under-track estimation.

As these problems appeared, fewer and fewer decisions where based on information extracted from the terrain. In the end only one parameter is computed from the terrain estimate, this being the flatness of the terrain in front of the robot, used for deciding if the obstacle top has been reached and we should lever with the back flippers. The way this parameter is computed is by comparing the maximum upper quartile in bins 6 to 9 with the minimum lower quartile in the same bins, if they don't differ more than 6 cm. we can classify the terrain in front of the robot as flat.

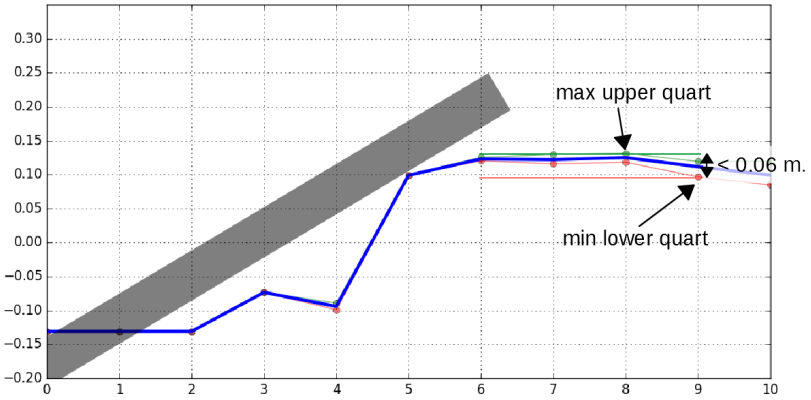
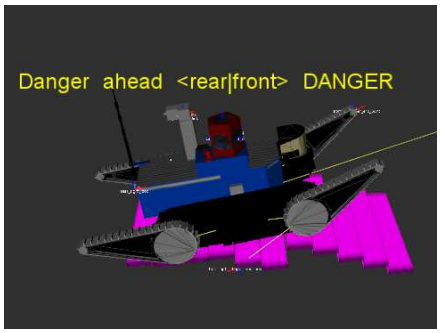


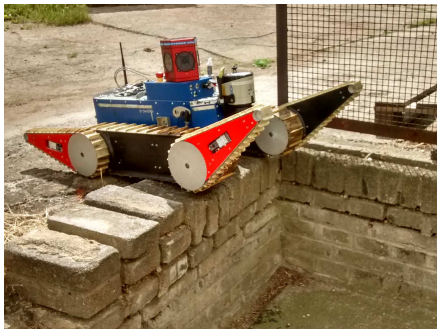
FIGURE 3.6: Example of front flat terrain and its calculation.

3.3 Dangerous Terrain

Currently, only one safety measure has been implemented in the Blind Traversing algorithm: if the angle at touch reaches the *touch bottom angle limit*, then the hole is considered too deep, the movement is disabled and the operator is prompted to retake manual control to manage the situation.



(A) Operator view



(B) Robot before hole

FIGURE 3.7: Example of a dangerous hole and the operator view.

Safety in Blind Traversing is absolutely vital, however, it has not been a priority for this work, given that this branch of development is still in very early stages and there is still a lot of work to do before being able to apply safety policies.

Chapter 4

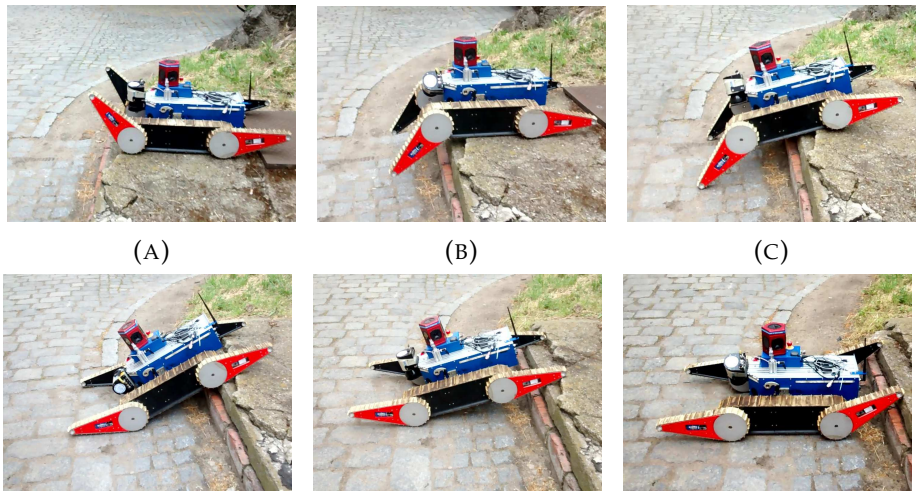
Experiments and Conclusions

4.1 Experiments

The evaluation of the algorithm has been done with several experiments traversing obstacles such as stairs, steps and some complex obstacles simulating debris or complex terrain. The traversal was succesful in any case despite some glitches that didn't affect beyond excess push with flippers or abrupt movement.

The experiments are only qualitative, meaning that no metrics have been recorded such as time to traverse or energy consumed. Given the lack of time to develop the proper Blind Traversing algorithm, after programming the software architecture, only one iteration of the Blind Strategies has been done, limiting the ability to measure and compare improvements over several versions.

4.1.1 Descending step



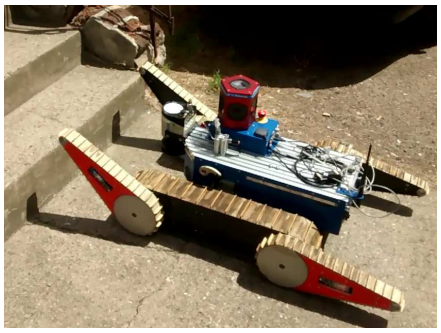
(A) (B) (C)

(D) (E) (F)

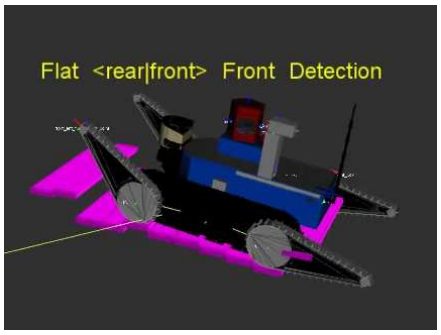
FIGURE 4.1: Different stages of a step descent experiment

4.1.2 Stairs traversal

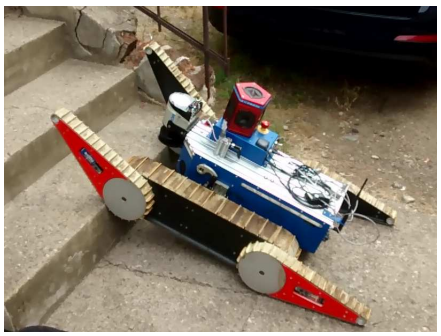
Stair Climb



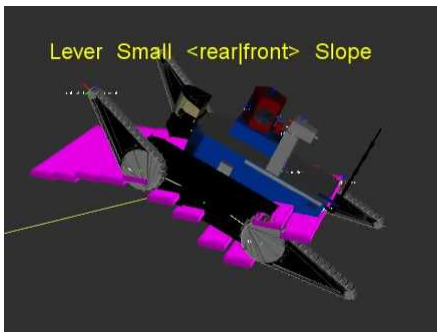
(A)



(B)

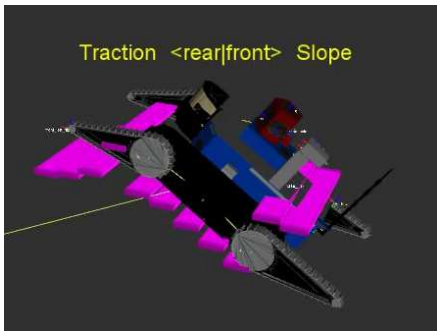


(C)

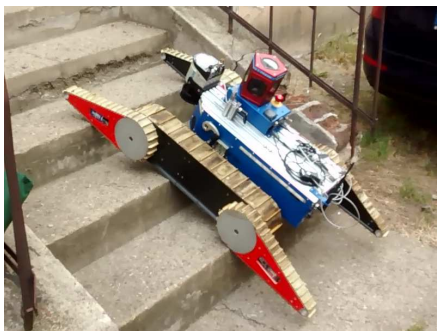


(D)

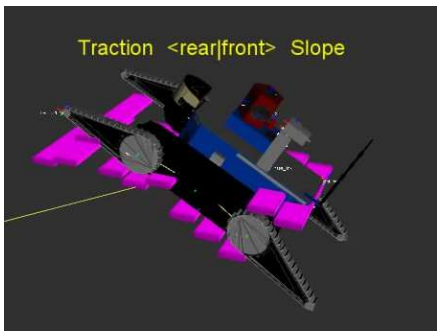
(E)



(F)

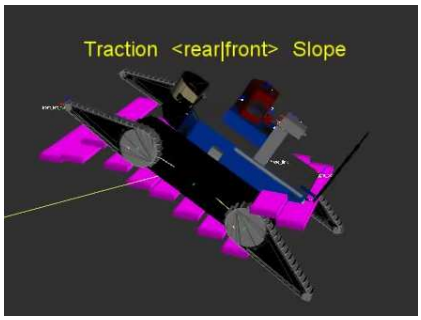


(G)



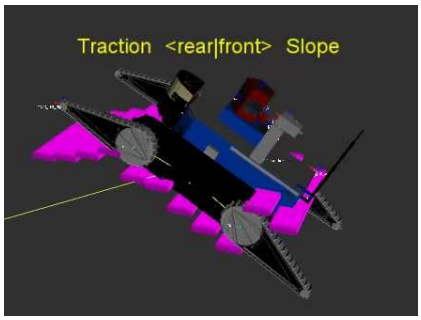
(H)

(I)



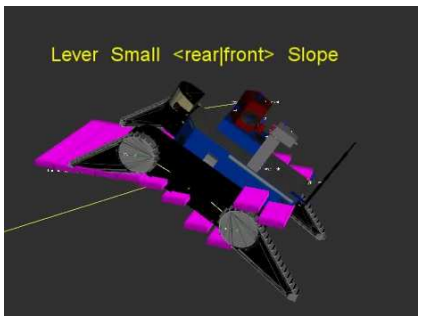
(J)

(K)



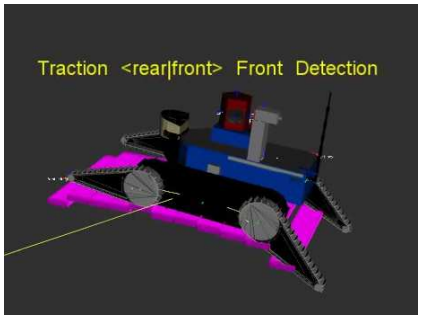
(L)

(M)



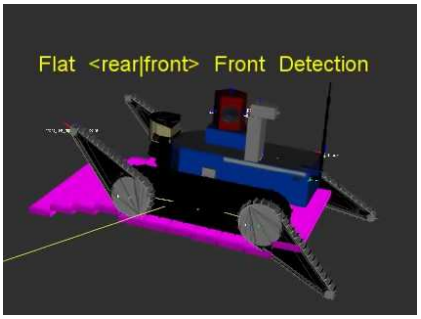
(N)

(O)



(P)

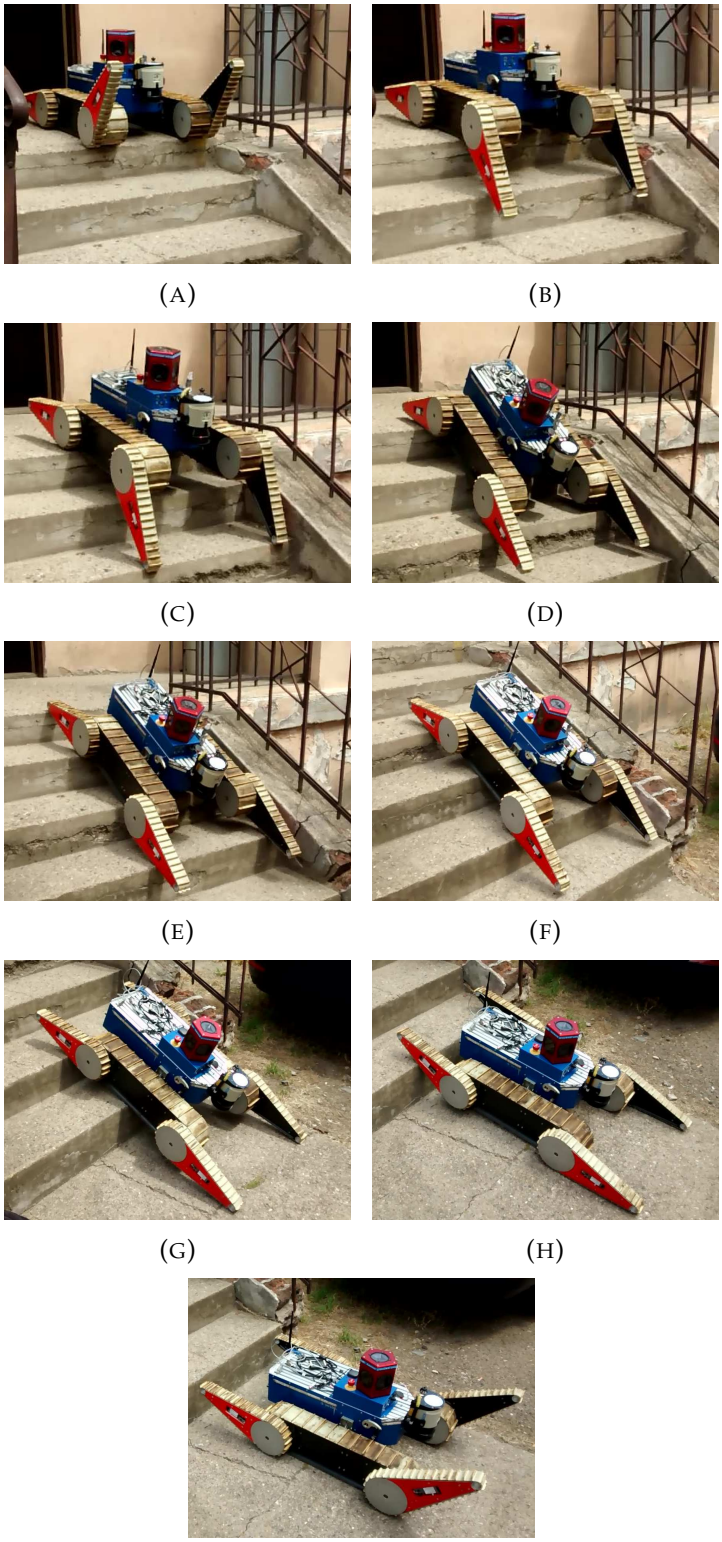
(Q)



(R)

FIGURE 4.2: Different stages of a stair climb experiment along with the operator simulation view

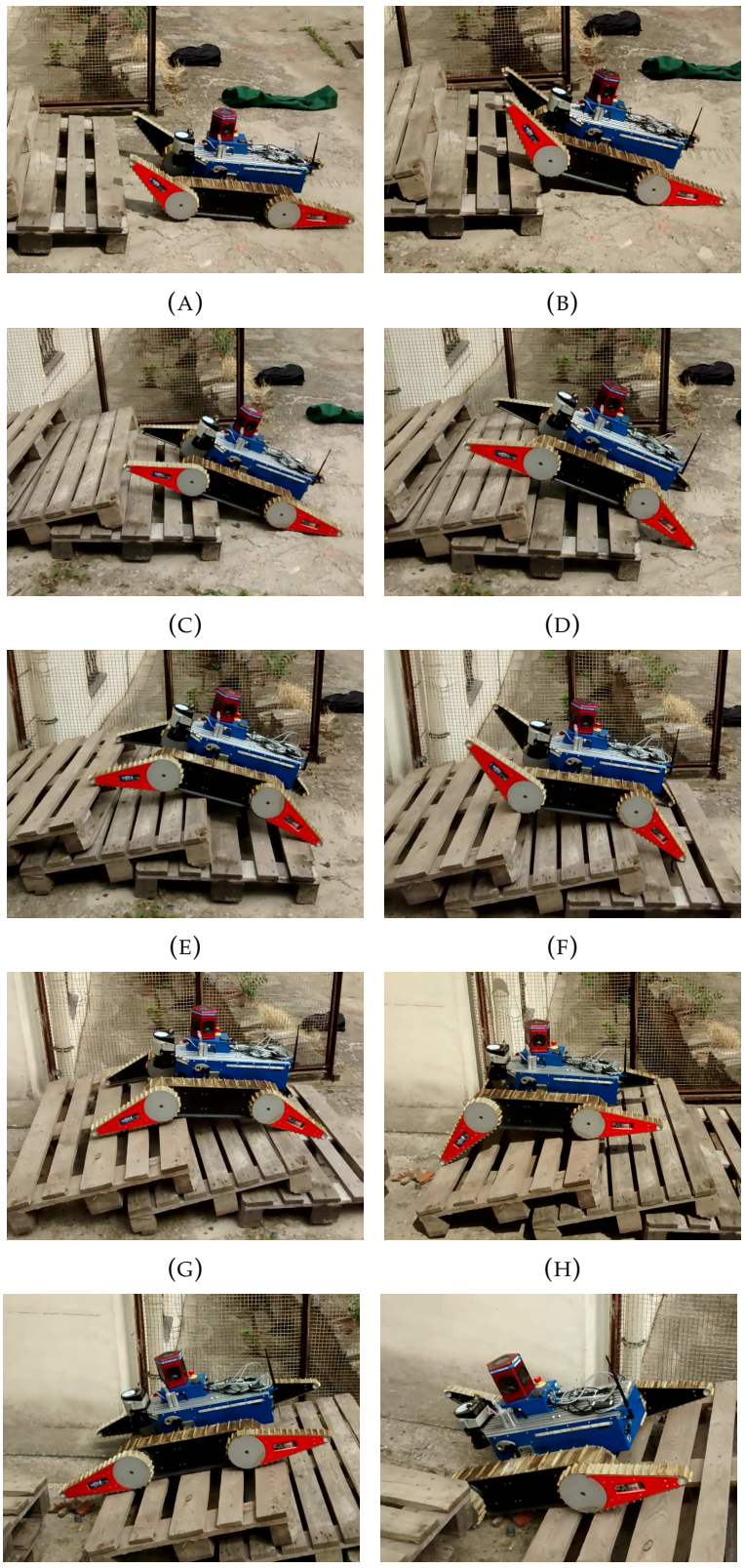
Stair Descent



(A) (B) (C) (D) (E) (F) (G) (H) (I)

FIGURE 4.3: Different stages of a stair descent experiment

4.1.3 Complex Obstacle



(A) (B) (C) (D) (E) (F) (G) (H) (I) (J)

FIGURE 4.4: Different stages of composed pallet obstacle traversal

4.1.4 Known Issues

Although the algorithm has been refined and tuned, there are some issues hard to solve at this point of the development but should easily addressable with the forthcoming addition of independent left/right track estimations, rear flipper force sensors and more richer training data set for the terrain estimator which will include stairs and walls.

- The lever shape can get stuck in loop if the low force spot is missed or shape is activated when not necessary. The lever shape is the only special case of temporal coherence given that if it was disengaged too soon the robot could destabilize heavily and even turn over. To avoid this, when levering, the algorithm waits for the robot to stop supporting on back flippers, which signals that the levering can be finished. However, if the lever is triggered where it should not (e.g. in flat terrain) it will enter a loop where the force in the back flippers is never released and they will keep pushing the ground hopelessly. This will be easily solvable once the rear flipper terrain estimation is implemented.
- The front flippers can get stuck inbetween the steps of stairs when climbing a structural stair. This is a difficult problem to solve for the general case which is very specific for this kind of stairs. Some ideas and possible solutions will be proposed in Section 4.3.

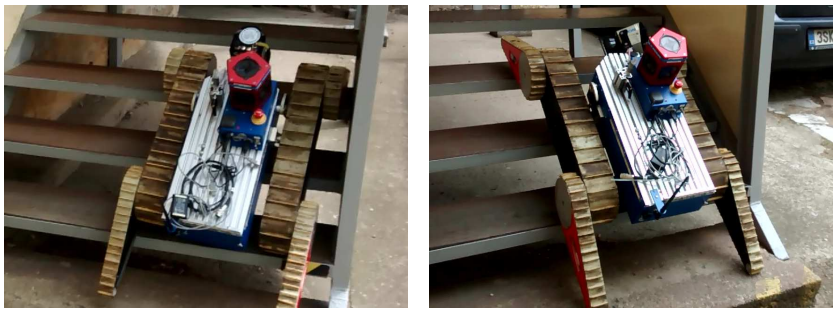


FIGURE 4.5: Examples of flippers stuck inbetween the steps of stairs.

- The robot cannot climb steps bigger than 20 cm. because measuring the top pushes the robot down, the rear flippers move from the ideal 0^{o} angle, rear levering fails to push the back of the robot up to climb the obstacle and it starts going vertical as the robot tries to advance.

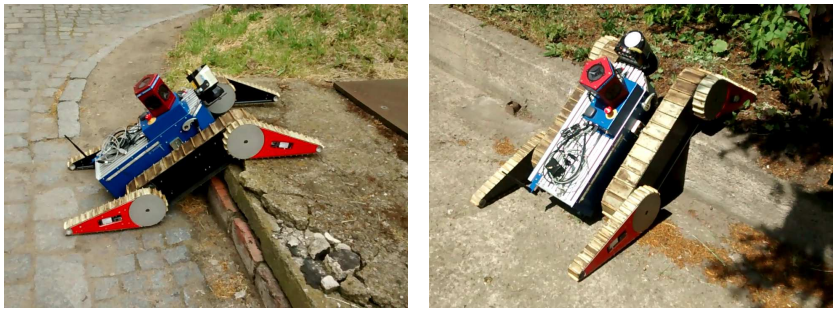


FIGURE 4.6: Examples where measuring the top of the obstacle pushes the robot down.

- The touch operation with all flippers can lead in many cases to unsafe situations. Especially when hovering over a downwards step, where the rear flippers touch first the ground, lift the rear of robot and could potentially make it fall downwards. This should be definitely considered when implementing and deploying safety policies.
- The algorithm fails to recognize unclimbable obstacles, such as a wall or a high step. This is mainly due to the lack of training examples for the terrain estimator, which will be further trained for this matter. Other policies can be implemented for this matter, such as stopping the robot if its pitch goes beyond a dangerous threshold.

4.2 Conclusions

The Blind Traversing algorithm presented in this work has been the first attempt to address this particular problem:

- A software architecture has been built nearly from scratch that provides a flexible and modular framework for future developments.
- The algorithm is functional and reliable when used on symmetrical simple obstacles such as steps, stairs or slopes.

The *aim* of this project has been to serve as a baseline to benchmark future improvements, but more importantly, to uncover the challenges and problems to account for future development. It has also been a first approach to this kind of traversing, that had never been implemented nor tested for this robot, and has demonstrated that the force sensors fulfill their role and provide the functionality and information that was expected.

The terrain estimation has proved to work better as a way for the operator to be aware of the context and environment of the robot than to use it for robot morphology determination. At least this holds true for the handcrafted algorithm presented in this work, it is nearly guaranteed that a machine learning algorithm could extract and use the information in the estimates far more effectively.

Throughout the development of this work it has come clear that the challenges and considerations to make for the Blind Traversing case are far more complex and restrictive than usual:

- The environment awareness of the operator is key for this kind of traversing, as it is critical for the operator to understand the proposed plans and evaluate if they are adequate and safe.
- The touch operation and the stop-go motion of the algorithm is really slow and makes for long stages that make the traversing really sluggish. This long operation makes it hard for the operator to stay focused in the task.
- It is not clear how to return the manual control to the operator in danger situations or when requested as the LIDAR and camera could still not work as expected and he/she would have to maneuver the robot over unknown terrain, leading to potentially unsafe situations.

4.3 Future Development

The Blind Traversing poses many issues and challenges that should be addressed before it is really safe and functional to deploy. The most relevant and important considerations to account for future development are:

- The handcrafted Stage Determination algorithm, which should eventually be replaced by a machine learning algorithm that can produce better and safer decisions. However, the training of this particular feature can be really tricky given the reactive nature of the traversing strategies and measured terrain. To illustrate, two different strategies to descend a step will make the tactile measuring in different places, angles and forces, resulting in different measurements. Hence, the training procedure and traversing strategies should be engineered for high repeatability and consistent measures and results.
- Speeding up the overall operation of the algorithm is key. Some of the bigger time sinks lie on the tactile measuring procedure, the terrain estimation and the slow (but safe) advance speed and distance. Efforts should be made to make the process more interactive and quick, such as dynamically deciding where to measure and the distance and speed for each stage. Overlapping some part of the actions on a stage could also be an option, for example starting to lower the front flippers some centimeters before reaching the measuring point. Different advance strategies could also speed up the operation such as raising less the front flippers if the terrain has been flat for a while, which would reduce the time to lower and raise them. It could even be considered to automatically execute the actions if the terrain estimation is sufficiently certain and flat (which under normal conditions should happen quite often).
- The creation of a Stairs mode independent of the normal Blind Traversing should be created given the critical requirements for traction and special treatment of stairs. A simple clasifier could be trained from raw tactile sensing data to detect the patterns of stair steps, once detected the Stairs mode could be engaged that maintained the flat shape of the robot while ensuring the flippers didn't get stuck between structural stairs steps as seen in Figure 4.6.
- The independence of terrain estimations is also a big drawback, merging the different estimations in a persistent map could help to statistically improve the accuracy of estimates and greatly increase the environment awareness of the operator. Merged with the LIDAR 3D maps could potentially avoid measuring the terrain of previously LIDAR scanned areas that could have been filled with smoke afterwards.
- The situational awareness by the operator is one of the most critical parts of the project, given that a poorly informed decision could end in a tripped over robot or even a broken one. The persistent mapping and merging of estimations should be enough to address this issue, however it is highly recommended to consider this problem more thoroughly. Partial fuzzy classification on the type of terrain could help the operator mentally classify the terrain it is traversing (stairs,

carpet, pile of rocks, gravel, wood, road...) which could be inferred from IMU data [3].

- The proposed interactivity between algorithm and operator consists on proposing a plan for the next stage, the operator then has to consider if this is safe and adequate and then confirm the decision. Other schemes could be considered that could significantly reduce the cognitive load and lead to faster results, for example, instead of waiting for a confirmation, execute the proposed plan some seconds after the proposition if no button is pressed. Essentially, reduce the requirement for the operator to quickly understand and process the information before the algorithm can continue.

Bibliography

[1] G.J.M. Kruijff et al. "Designing, developing, and deploying systems to support human–robot teams in disaster response". In: *Advanced Robotics* 28.23 (2014), pp. 1547–1570. URL: http://dx.doi.org/10.1080/01691864.2014.985335.

[2] Vojtech Salansky et al. "Touching without vision: terrain perception in sensory deprived environments". In: *21st Computer Vision Winter Workshop* 21 (Feb. 2016). URL: http://vision.fe.uni-lj.si/cvww2016/proceedings/1.

[3] Christian Weiss, Holger Fröhlich, and Andreas Zell. "Vibration-based Terrain Classification Using Support Vector Machines". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Oct. 2006), pp. 4429 –4434.

[4] Karel Zimmermann et al. "Adaptive Traversability of unknown complex terrain with obstacles for mobile robots". In: (2014), pp. 5177–5182. ISSN: 1050-4729. DOI: 10.1109/ICRA.2014.6907619.