José Rodrigo Ferreira Baleia

Licenciado em Ciências da Engenharia Eletrotécnica e de Computadores

Haptic Robot-Environment Interaction for Self-Supervised Learning in Ground Mobility

Dissertação para obtenção do Grau de Mestre em Engenharia Eletrotécnica e de Computadores

Orientador: Prof. Doutor José António Barata de Oliveira, FCT-UNL
Co-orientador: Prof. Doutor Pedro Figueiredo Santana, ISCTE-IUL

Júri:

Presidente: Doutor Ricardo Luís Rosa Jardim Gonçalves – FCT-UNL
Arguente: Doutor Pedro Alexandre da Costa Sousa – FCT-UNL
Vogal: Doutor Pedro Figueiredo Santana – ISCTE-IUL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março 2014

Copyright

Haptic Robot-Environment Interaction for Self-Supervised Learning in Ground Mobility

Acknowledgements

I would like to thank my thesis supervisor Prof. Jose Barata for his support and for giving me the opportunity and enabling the resources for the hardware necessary for the robot implementation. I would also like to thank other professors from the institution, whose teachings had a big role in my growth not only as an engineering student but also a human being.

A special note of gratitude to my co-supervisor Prof. Pedro Santana, whose vision and knowledge on the field of robotics motivated and inspired me to accomplish more as an engineer. I'm grateful for the constant inputs on the overall work and readiness to help and improve. I would also like to thank Eduardo Pinto, whose hardware expertise were essential for the development of the robot.

My next acknowledgments go to all the ones whose paths crossed mine in this long journey of becoming an electrical engineer. A special word of thanks to Gustavo Fradique, Nuno Barata, Fernando Costa, Pedro Carvalho, Rui Borrego, Pedro Barreira, André Pereira, João Patrício, João Silva, and many others, that started out as colleagues and ended up as friends. I would also like to thank my work colleague Pedro Alves for his support in the roughness that is field testing.

Finally, I would like to thank all my family and friends for the support shown during the years.

Abstract

This dissertation presents a system for haptic interaction and self-supervised learning mechanisms to ascertain navigation affordances from depth cues. A simple pan-tilt telescopic arm and a structured light sensor, both fitted to the robot's body frame, provide the required haptic and depth sensory feedback. The system aims at incrementally develop the ability to assess the cost of navigating in natural environments. For this purpose the robot learns a mapping between the appearance of objects, given sensory data provided by the sensor, and their bendability, perceived by the pan-tilt telescopic arm. The object descriptor, representing the object in memory and used for comparisons with other objects, is rich for a robust comparison and simple enough to allow for fast computations. The output of the memory learning mechanism allied with the haptic interaction point evaluation prioritize interaction points to increase the confidence on the interaction and correctly identifying obstacles, reducing the risk of the robot getting stuck or damaged. If the system concludes that the object is traversable, the environment change detection system allows the robot to overcome it. A set of field trials show the ability of the robot to progressively learn which elements of environment are traversable.

keywords: autonomous robots, self-supervised learning, affordances, terrain assessment, depth sensing, robotic arm.

Resumo

Esta dissertação apresenta um sistema para interação háptica e mecanismos de aprendizagem auto-supervisionada para averiguar as possíveis ações sobre objetos a partir de informação sensorial. Um braço telescópico para obter informação háptica e um sensor de profundidade baseado na projeção de luz estruturada, ambos incluídos no chassis do robô, fornecem os requisitos para a geração de metodologias de interação háptica. O objetivo do sistema é o de continuamente avaliar o custo de navegação em ambientes naturais. Para alcançar este objetivo, o robô aprende o mapeamento entre a aparência dos objetos, dada a informação sensorial, e a sua rigidez, percecionada através do braço telescópico. O descritor do objeto, representação do objeto em memória e utilizado para a comparação com outros objetos, é rico para permitir uma comparação robusta e simples para ser de rápida computação. O resultado do mecanismo de aprendizagem aliado com a análise geométrica do objeto prioriza pontos de análise para aumentar a confiança na interação e corretamente detetar obstáculos, reduzindo o risco do robô ficar preso ou danificado. Se o sistema conclui que o objeto é trespassável, o sistema de deteção de mudança de ambiente permite ao robô atravessá-lo. Um conjunto de testes no terreno demonstra a capacidade do robô de aprender progressivamente que elementos do ambiente são trespassáveis.

palavras-chave: Robôs autónomos, affordances, aprendizagem auto-supervisionada, percepção de terreno, sensor de profundidade, braço robótico.

Contents

List of Figures

List of Tables

List of Notations

α	Memory confidence factor
α_C	Weight of parameter $C_h(P)$
α_S	Weight of parameter $N^j_{\mathbf{H}}(\mathbf{H'})$
α_ρ	Environment change difference threshold
β_C	Weight of parameter $C_d(P)$
β_S	Weight of parameter $W^j_{\mathbf{H}}(\mathbf{H'})$
Δ_ρ	Relation between $\rho^j_{\mathbf{H}}$ and $\rho^j_{\mathbf{H'}}$
Δ_P	Relation between $P^j_{\mathbf{H}}$ and $P^j_{\mathbf{H'}}$
δ_S	Weight of parameter $P^j_{\mathbf{H}}(\mathbf{H'})$
δ_ρ	Environment change difference
γ_C	Weight of parameter $C_\rho(P)$
γ_S	Weight of parameter $\rho^j_{\mathbf{H}}(\mathbf{H'})$
$\mathcal{A}$	Robot Arm frame of reference
$\mathcal{C}$	Sensor frame of reference
$\mathcal{H}$	Histogram formalization
$\mathcal{O}$	Base of the robot frame of reference
μ	Distance from sensor to wheel height
ω	Comparison factor for $W^j_{\mathbf{H}}(\mathbf{H'})$ score
$\rho^j_{\mathbf{H}}(\mathbf{H'})$	Comparison of $\rho^j_{\mathbf{H}}$ score between histogram **H'** and **H**
$\rho^j_{\mathbf{H}}$	Point density per line j in histogram **H**
σ	Robot architecture and sensor adaptive factor
τ	Distance to the sensor dependence factor
c	Coordinates to which the point is found in the $\mathcal{C}$ and $\mathcal{A}$ frame of reference
H	Existing histogram

S	List of the n closest neighbors by similarity
υ	Comparison factor for $N^j_{\mathbf{H}}$(**H'**) score
ζ	Centroid of a point cloud
$C_d(P)$	Distance to centroid score for point interaction
$C_h(P)$	Height score for point interaction
C_k	Confidence of knowing a new object from memory evaluation
$C_T(P)$	Final score for point interaction
$C_{\mathcal{A}}$	Point in the $\mathcal{A}$ frame of reference
$C_{\mathcal{C}}$	Point in the $\mathcal{C}$ frame of reference
$C_\rho(P)$	Number of neighbor points score for point interaction
C_{int}	Cloud used to generate the interaction points
Cl_f	Cloud with the background removed
Cl_n	New captured cloud
Cl_{bkg}	Cloud without robotic arm in range
E_o	Object labeled as obstacle
E_t	Object labeled as traversable
F	Furthest point in a cloud from the robots arm referential
H_f	Filtered cloud used for histogram creation
H_v	Voxelization of the filtered cloud
i	Histogram column iterator
j	Histogram line iterator
j_o	Number of bins occupied in a line
M	$\mathcal{C}$ to $\mathcal{A}$ transformation matrix
M^{-1}	Inverse M matrix, $\mathcal{A}$ to $\mathcal{C}$ transformation matrix
M_{n_n}	Maximum number of points in a sphere
$N^j_{\mathbf{H}}$(**H'**)	Comparison of $N^j_{\mathbf{H}}$ score between histogram **H'** and **H**
N_l	Number of points in a line
n_p	Number of points used for calibration
$N^j_{\mathbf{H}}$	Number of clusters per line j in histogram **H**
n_c	Number of **c** found
$n_n(P)$	Number of points inside a sphere

P_ρ	Relation between $P^j_{\mathbf{H}}$ and $P^j_{\mathbf{H}'}$
$P^j_{\mathbf{H}}(\mathbf{H}')$	Comparison of $P^j_{\mathbf{H}}$ score between histogram **H'** and **H**
P_o	Probability of the object being an obstacle
P_t	Probability of the object being traversable
P_ρ	Density change input cloud value
$P^j_{\mathbf{H}}$	Maximum points in a cluster per line j in histogram **H**
Ps_ρ	Density change first input cloud value
R	Rightmost point from the arm
r_{int}	Robot end effector interaction radius
r_n	Neighbor points search radius
$S^j_{\mathbf{H}}(\mathbf{H}')$	Score from comparing a single line between two a histogram **H** and **H'**
S_{cl}	Size of the input cloud
S_{vox}	Size of the cloud after voxelization
T_o	Traversability of a object in memory
$W^j_{\mathbf{H}}(\mathbf{H}')$	Comparison of $W^j_{\mathbf{H}}$ score between histogram **H'** and **H**
$W^j_{\mathbf{H}}$	Largest cluster per line j in histogram **H**
$x_{\mathbf{H}}$	Number of columns in a histogram
$y_{\mathbf{H}}$	Number of lines in a histogram
$S^{\mathbf{H}'}_{\mathbf{H}}$	Similarity score between two histograms **H** and **H'**
P	Point in a x, y, z coordinate system
p	Point in a x, y coordinate system

Chapter 1

Introduction

Since the first invertebrate ventured out of the Panthalassa, the ability to navigate through the environment became crucial for the species survival. Nature evolved in order to allow for different means of interaction and learning mechanisms, some insects developed antennas to sense the surroundings, while mammals brains grew to support the big influx of information provided by non-haptic feedback, like hearing, vision and olfactory perception.

Inspired by Nature, in which visual and haptic sensory feedback are known to be jointly exploited in the Human brain (Lacey et al., 2010; Schwenkler, 2013) this thesis presents an haptic robot-environment interaction system for self-supervised learning of vision skills for safe navigation. For this purpose, the robot is provided with a mechanism to learn a mapping between the volumetric appearance of obstacles, given sensory data provided by a depth sensor, and their bendability, perceived by physically interacting with them with a small arm. As interactions unfold, the robot grows its ability to properly assess the cost of navigating the environment from its depth sensor and, consequently, reducing the need for physical interactions. As a consequence, the robot's spatial reasoning look-ahead grows significantly, which is key to ensure a safe navigation.

Models like Kim and Möller (2007) and Wijaya and Russell (2002) aim to mimic the simplicity and low processing power necessary to navigate the environment as used by some invertebrate species, like ants, or small mammals, like rats. On the other hand, some models went for a different route and created more complex interaction methods, aiming to give a deeper insight provided by interaction. An example of that is the model developed by Edsinger-Gonzales (2005), which recreated a complex hand which can sense force, or the model of Shin et al. (2010), recreating a force sensing arm. To learn how the species interact with the world is crucial to learn about the perception they have of the environment.

The concept of affordances link the ability of a subject though its actions to the features of the environment, so in order to learn an affordance it is necessary to interact with the world. Object aspect by itself can give information about the interaction possibilities (Gibson, 1977). For example, to move an object it is easier to roll it if the shape allows it rather than lifting it, or holding certain objects are only possible to be lifted if the user can grasp them. By combining the interaction method and the information gathered by the object aspect, it is possible to make a better guess about the objects

reaction to the interaction and use that experience to learn about its traversability. This non-invasive method offers safer results for the environment and to the subject under test, a desirable quality for field robots. Uğur and Şahin (2010) considered learning affordances from full-body interactions by moving the robot against the objects. Conversely, the system developed in this thesis proposes assessing navigation cost with a robotic antenna, which reduces the robot's risk of getting stuck or damaged as well as it allows for a finer analysis of the object.

While studying the aspect is important to decide the best interaction, remembering past efforts allows for a long term evolution in the quality of those interactions. Lower animals use a match between the input image and the stored images (Tanaka, 1993) so inspired by this, a comparison method for self-supervised learning applied to robotics improves the interaction dynamics in the long run. The aspect of the object and the result of that interaction allows to enhance the knowledge of the surrounding world, and by applying an object descriptor and a memory recall system, both an empirical and theoretical weight is given to the decision. With this, the speed of interaction, meaning the time it spends with each object, and its quality is expected to improve as more object data is collected.

To correctly identify the objects found in nature, a robust descriptor is needed. Vegetation is not the same all around the world so it is difficult to find methods of classification and biomass estimation, and as shown by Lu (2006), this an area with room for more research. Flora can vary by size, mass, color, shape and density, making the correct classification of similar types of environments a challenging task. A correct flora descriptor is important to be able to learn from previous experiences, so a good descriptor is necessary. The descriptor has to be broad enough to be able to work in every kind of environment but specific enough to be able to detect flora changes in the surroundings. While flora is present in field environment, other natural objects such as rocks, trunks or man made objects can still be found and have to be correctly described.

In a natural environment, the system may encounter vegetation or other types of objects, such as rocks. To assess the traversability, the system must create a 3D descriptor for comparison with its memory. The knowledge of the common vegetation is important to create the descriptor. When comparing to previous encounters, the system applies what it has learned in order to generate a haptic interaction methodology by taking into account not only what was learned but also the probable best interaction points to assess the bendability affordance based on the object's structure. This interaction is more detailed if the confidence in the new object is low, or coarser if the confidence is higher. If after the interaction the system concludes that the bendability affordance is present in the object, it proceeds to overcome it.

To validate the proposed model, a prototype was built with a custom telescopic pan-tilt antenna, a structured light sensor and a mobile platform. To test the haptic interaction, controlled test subjects were used and controlled interactions were performed in order to assess the bendability, and a database was constructed to test the memory recall system and the classification system. Field trials were performed to test the environment change detection implementation.

1.1 Dissertation Outline

This dissertation is organized as follows:

Chapter 2 reviews the state of the art in affordances, traversability, self-supervised learning, interaction methods and vegetation characterization;

Chapter 3 presents the supporting concepts of this work, like software libraries used, voxel grids and histograms;

Chapter 4 describes a possible model for the existing problems and the methods used for implementation;

Chapter 5 shows and explains the prototype developed for testing the proposed model. Both hardware and software perspective are approached;

Chapter 6 presents the experimental results based on the model and prototype developed, as well as the testing parameters used;

Chapter 7 aggregates a set of conclusions, main contributions of this dissertation and further research opportunities on the subject.

Chapter 2

Related Work

This chapter includes a description of the state of the art in robotic interaction methods, including projects offering different perspectives on the subject, the state of the art of recognition methods mainly based on natural environments, projects and studies about affordances, studies and systems about traversability and finally an explanation of self-supervised learning, including the merits of this learning mechanism.

2.1 Interaction Methods

Robotics often take biological inspiration in order to create robots that mimic what happens in nature to solve complex problems (Pfeifer et al., 2007). In nature one can find various methods of interaction, from seemingly simple mechanisms, like whiskers, to more mechanically complex, such as an human arm. These topics are deeply studied and a big array of solutions is available.

In a complex perspective, in order to simulate a human arm, a system needs to emulate the fine balance between stiffness and force control. To achieve this it is necessary to understand the actuators available and assemblies used. A commonly used actuator is based in series elastic actuators (Pratt and Williamson, 1995). These actuators place an elastic element between the output of the actuator and the robotic link to limit the high-frequency impedance of the actuator to the stiffness of the elastic coupling. To limit the low-frequency impedance a linear feedback system is implemented to regulate the output torque of the actuator-spring system. Therefore, the series elastic actuators provide low impedance across the frequency spectrum (Zinn et al., 2004). These actuators are mainly developed to provide safe human-robot interaction, by finding a safe balance between torque and speed. Recently, a different approach has been used in order to reach the same goal. The new trend is the use of variable impedance actuators, to achieve safe, energy-efficient, and highly dynamic motion (Vanderborght et al., 2013).

Alternatively, some researchers go to the route of mimicking low animals by using whiskers to receive haptic feedback from the environment. Whiskers (mostly rats) have been a subject of study for a long time. Vincent (1913) started a detailed study of whiskers physical abilities followed by a

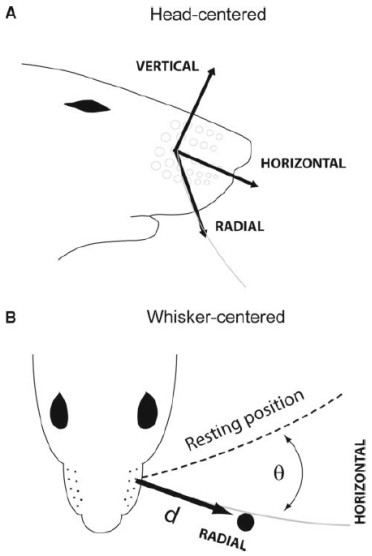

Figure 2.1: Coordinate systems of the vibrissal system (Ahissar and Knutsen, 2008).

deeper study of its capabilities (Vincent, 1915). Rodents use their whiskers to detect and identify objects in their proximal three-dimensional space. Each whisker shaft is embedded in a follicle structure and mechanoreceptors surrounding the shaft measure the deflection in all directions. Whisker behavior involves repetitive (periodic or non-periodic) forward (protraction) and backward (retraction) movements of the whiskers. Whisker movements are largely synchronous on one side, but often occur with a phase-shift across the two sides of the snout. Whiskers are usually grouped in vertical and horizontal rows, and each row is used to sense different information about the surroundings (Ahissar and Knutsen, 2008). Figure 2.1 shows the coordinate systems of the vibrassal system found in most rodents.

In robotics several sensors were developed in order to simulate whiskers found in small rodents. Whisker probes have the potential to be fast, accurate and cheap, while still providing enough information to be usable in small robots and with low bandwidth requirements. Russell (1992) developed a tactile sensor array, with each sensor consisting of a potentiometer and a long inflexible beam, and the potentiometer sensor at the whisker root measured the rotational angle, proportional to the contact force applied to the antenna tip, providing the ability to obtain the surface profile of an object. This kind of sensors were successfully applied in robots in order to improve the navigation capabilities (Jung and Zelinsky, 1996). More recently, whisker sensors have been refined to be able to detect minute differences in texture and shape of the encountered objects. Scholz and Rahn (2004) and Fend (2005) show how it is possible to use whiskers to detect textures and fine details on the objects. All these improvements require more processing power due to signal processing and elimination of false positives and self-generated signals from the whiskers (Anderson et al., 2010). More recent works rely on using several arrays of whiskers associated with whisker movements and signal processing to quickly determine characteristics of the objects (Kim and Möller, 2007), by using flexibility, friction, sweeping movements and different whisker width.

In this thesis, a robotic antenna simulating a whisker is used to assess information about objects in the environment, namely the bendability, and by a mapping with the appearance of the obstacles assess the cost of navigation. The interaction with the object should be as efficient as possible in order to preserve energy and save time, while still maintaining confidence on the interactions. This

is made by generating a motion plan of interaction with the antenna depending on the characteristics of the object and confidence of the knowledge.

2.2 Haptic-visual relation

One of the parameters to generate the interaction methodolgy is the object's geometry in the environment. Haptic-visual perception is a complex and old subject in studies. As early as 1690, the Molyneux problem raised the issue by questioning "if a man born blind can feel the differences between shapes such as spheres and cubes, could he similarly distinguish those objects by sight if given the ability to see?" (Locke, 1700). The haptic system uses sensory information derived from mechanoreceptors and thermoreceptors embedded in the skin ("cutaneous" inputs) together with mechanoreceptors embedded in muscles, tendons, and joints ("kinesthetic" inputs) (Lederman and Klatzky, 2009). Studies concluded that the haptic feedback obtained from an object surface and the real object properties are tighly bound to the nature of the interaction, meaning that different interaction methods can change the perception of the objects real properties (Lederman and Klatzky, 1987). The usual pattern to learn information from the object by active perception is to explore the objects texture, weight, hardness, volume, temperature and global shape. Lacey et al. (2010) shows that the multisensory view-independent object representation underlying visuohaptic object recognition integrates both structural and surface properties, while Phillips et al. (2009) concludes that there is a high degree of perceptual equivalence between vision and haptics. However, Held et al. (2011) concludes that the people that were previously blind failed to correspond the object's visual properties to the haptic feedback provided from interaction, but Schwenkler (2013) concludes that there is a relation between visual properties and haptic feedback. As described here, there is not a clear consensus on the subject.

2.3 Vegetation Characterization

As seen in the previous section, the haptic feedback obtained depends on the interaction approach so to achieve better results is it important to know the type of subject under evaluation. Vegetation charaterization is used in order to estimate the biomass in a determined area. Remotely sensed data have become the primary source for biomass estimation. Biomass estimation (derived from living organisms) remains a challenging task, especially in those study areas with complex forest stand structures and environmental conditions. Biomass, in general, includes the above-ground and below-ground living mass, such as trees, shrubs, vines, roots, and the dead mass of fine and coarse litter associated with the soil. Due to the difficulty in collecting field data of below-ground biomass, most previous research on biomass estimation focused on above-ground biomass (AGB). Either optical sensor data or radar data are more suitable for forest sites with relatively simple forest stand structure than the sites with complex biophysical environments. A combination of spectral responses and image textures improves biomass estimation performance. More information on the subject can be found in Lu (2006) survey where the state of the art in biomass estimation from high altitude perspective is summarized.

Above-ground biomass estimation acquired with optical sensor data can be directly estimated with different approaches, such as multiple regression analysis (Franklin and Hiernaux, 1991), K nearest-neighbour (Halme and Tomppo, 2001), and neural network (Zheng et al., 2004), and indirectly estimated from canopy parameters, such as crown diameter (Popescu et al., 2003), which are first derived from remotely sensed data using multiple regression analysis or different canopy reflectance models. The characteristics of the biomass estimation can be used to identify the vegetation type presented. Fine spatial-resolution data can be airborne, such as aerial photographs, or spaceborne, such as IKONOS (Grodecki, 2001) and QuickBird (Toutin and Cheng, 2002) images, with spatial resolutions of less than 5 m (e.g. the spatial resolutions of panchromatic images of IKONOS and QuickBird are 0.83 and 0.61 m). They are frequently used for modelling tree parameters or forest canopy structures. The medium spatial-resolution ranges from 10 to 100 m. The most frequently used medium spatial-resolution data may be the time-series Landsat data, which have become the primary source in many applications, including AGB estimation at local and regional scales (Sader et al., 1989). Lefsky et al. (2001) evaluated the utility of several remotely sensed data for estimating stand structure attributes-age, basal area, biomass, and diameter at breast height (DBH). The coarse spatial resolution is often greater than 100 m. Common coarse spatial resolution data include NOAA Advanced Very High Resolution Radiometer (AVHRR), SPOT VEGETATION, and Moderate Resolution Imaging Spectroradiometer (MODIS). They are often used at national, continental, and global scales. The AVHRR data have long been the primary source in large-area surveys because they offer a good trade-off between spatial resolution, image coverage, and frequency in data acquisition. The AGB estimation using coarse spatial-resolution data is still very limited because of the common occurrence of mixed pixels and the huge difference between the size of field-measurement data and pixel size in the image, resulting in difficulty in the integration of sample data and remote sensing-derived variables.

In many areas of the world, the frequent cloud conditions often restrain the acquisition of high-quality remotely sensed data by optical sensors. Thus, radar data become the only feasible way of acquiring remotely sensed data within a given time framework because the radar systems can collect Earth feature data irrespective of weather or light conditions. Due to this unique feature of radar data compared with optical sensor data, the radar data have been used extensively in many fields, including forest-cover identification and mapping, discrimination of forest compartments and forest types, and estimation of forest stand parameters (Treuhaft et al., 2004).

Different vegetation structures present different image, radar and lidar information, so this information can also be used to detect and identify vegetation at ground level. Along the years projects were developed using this knowledge. Lalonde et al. (2006) created a system to identify porous volumes like grass and tree canopy, thin objects like wires or tree branches and solid objects like ground surface, rocks or large trunks using 3D point cloud data and off-line labeled learning data. Using a Kinect sensor Azzari et al. (2013) developed a system of rapid characterization of vegetation structure by analyzing the canopy structure from point clouds. Moorthy et al. (2011) used laser information to learn the characteristics of olive trees and showed that is is possible to use this technology as a new observational tool and benchmark for precise characterization of vegetation architecture for improved agricultural monitoring and management, by calculating crown width, crown height, crown volume, and plant area index. More recently, Wurm et al. (2012) tested laser scanners capturing 3D point clouds and laser scanners mounted at a fixed angle to detect low vegetation by using the remission values of the laser scanners. The tests showed that laser scanners can successfully detect vegetation. More research and projects are needed to improve the vegetation characterization

in natural environments due to its complexity and wide variety of types of vegetation, in both vegetation analysis and types of sensors used for this task. There is still room to improve for developing a system that is able to correctly identify and learn the characteristics of the vegetation.

The knowledge of the characteristics of natural environments is important for the system developed in this thesis for the generation of the object descriptors and the memory recall. Low vegetation is structured in a way that is usually denser at the bottom, and small plants are often harder to traverse if they do not bend in the middle. These factors were taken into account in the generation of the haptic interaction points classification. The type of sensor is also important, and it is known that structured light sensors are capable of detecting plants and vegetation correctly.

2.4 Affordances

The result of the interaction with vegetation can give an indication about its traversability. The term affordance was originally used by psychologist James J. Gibson (Gibson, 1979), where he explained how inherent "values" and "meanings" in the environment can directly be perceived, and how that information can be linked to the action possibilities offered to the organism by the environment (Gibson, 1977). Gibson defined affordances as all "action possibilities" latent to the environment, objectively measurable and independent of the individuals ability to recognize them, always in relation to agents and therefore dependent on their capabilities. For instance, a set of steps which rises four feet high does not afford the act of climbing if the actor is a crawling infant. Gibson's is the prevalent definition in cognitive psychology (Jones, 2003). Affordances are closely related to perception. Gibson stated that when the constant properties of constants are perceived the observer can go to detect their affordances. However, this concept has been a target of changes of perspective and meaning. Turvey (1992) claims that affordances are dispositional properties of the environment, their effectivities are dispositional properties of the subject and when they meet in space they get updated. However Stoffregen (2003) defends that the environment does not offer proprieties and they are only the result of the subject-environment interaction. Chemero (2003) introduces a new concept, the concept of abilities. For Chemero, an affordance is the result of the interaction of an ability of a subject with the features of the environment. Şahin et al. (2007) created a model where he states that an affordance is an acquired relation between a behavior of an agent and an entity in the environment such that the application of the behavior on the entity generates a certain effect, but later revised to model to add the view of a third perspective, so an affordance became a relation between a entity-behavior perception of an agent such that the application of the behavior on the entity generates a certain effect.

There is still the question of how these affordances are acquired. Some authors defend that they are acquired by learning (Eleanor J. Gibson (2000)) while others defend that they are naturally acquired by evolution (Norman, 2002). Eleanor J. Gibson (2000) concluded that learning an affordance is a matter of perceptual learning. In her studies she concluded that an affordance is not association of elementary processes, construction from elements of any kind, or formation of a representation, but a process of differentiation that results in specification of information for an affordance, a functional relation between a agent and its environment. The process is one of selection, not addition. Learning is a result of variation, accomplished through exploratory activity that leads to perception

of consequences (new information) and of selection. Selection is based on two principles: the affordance fit, link between the actions performed and the ensuing consequence of making contact with the resource offered; and reduction of uncertainty. Reduction of uncertainty is achieved by discovery of unity, order, and economy of actions. Unity is what is called to the detection of invariance provided by perceiving order, per example, one's own moving hand. The minimal information that is invariant over transformations and contextual change will be preserved, meaning that the different detected features are the ones that abide. The principles of affordance fit and reduction of uncertainty operate together to determine what is learned in perceptual learning.

The concept of affordances have been applied to robotics in order to improve a robot's ability to learn about the environment surrounding it, and how to interact with it. Montesano et al. (2008) using the concept of affordances and a Bayesian network, managed to teach a robot on how to interact with different objects by imitation and repetition. The system correctly learned the relations between actions, objects and effects, the affordance model according to Chemero. As stated above, Şahin et al. created a new model and applied it to robotics. He created an affordance model based on three perspectives and applied it to robot control. Sinapov and Stoytchev (2008) took that concept and created a model that managed to learn the similarity between several tools and associate that similarly with actions on different objects. Santana et al. (2010) created a model where affordances of a scene are predicted by using visual context by utilizing gist descriptors (perceptual context), in order to prioritize perceptual resources and visual attention. The model was based on lazy learning and by associating gist with behaviors and successfully showed that self-supervised learning can be improved by using behaviors on context rather than using object descriptors. Detry et al. (2009) implemented a system that learned grasp hypothesis from previously learned sources, like imitation or visual cues, and correctly managed to perceive the grasp affordance.

2.5 Traversability

In this thesis, the affordance bendable is related to the affordance traversable. In the Oxford dictionary, the verb "traverse" means to travel across or through, so it is a relation between an ability from the agent and a feature of the environment, and as seen in Section 2.2, it is an affordance. Since most actions depend on mobility, traversability is a fundamental affordance for autonomous robots. Robotic applications such as planetary exploration, search and rescue, forestry and mining are made feasible by designing robots with reconfigurable components and learning mechanisms that passively, or actively adapt to the environment. Historically, terrain analysis through traversability estimation was addressed as binary classification, but the trend is for finer classification to englobe not only traversability in its most common definition, but also to include the concept of time and energy efficiency, an important aspect to model artificial intelligence (Horton et al., 2012).

According to Papadakis (2013), the traversability estimation can be separated by proprioceptive sensory data processing and exteroceptive sensory data processing. Proprioceptive traversability analysis capture the difficulties of a vehicle while traversing the environment by analyzing sensor information given by vibrations, wheel slips, bumper hits, etc. This is usually used in conjunction with other sensory information in order to correlate with the information learned while traversing the environment. This approach has high risks because it can easily damage or destroy the vehicle so

exteroceptive methods are usually preferred. Exteroceptive sensory data can be divided into two groups; geometry based and appearance based.

The majority of terrain traversability analysis methodologies rely on geometric processing. Geometric analysis is based on creating models or representations of the robot and the environment and through diverse methodologies compute a likely traversability classification of the environment. A set of common features that characterize traversability analysis methods are the analysis of the terrain properties, robotic attributes, robot stability and robot kinematic constraints. The most common methodologies used in robotics to predict the traversability affordance based on geometry are signal processing methods, convolution with kernel and statistic processing. Signal processing, either single-scale or multi-scale space analysis, is made by obtaining a set of roughness parameters of the environment by employing Fourier analysis or by utilizing wavelet decomposition. On the other hand, a more popular method, convolution with kernel, is obtained by simulation the vehicle as a fixed size 2D kernel and convoluting this kernel with the 2D terrain map. The idea is to iteratively process a window and try different orientations in order to obtain a traversability estimation. Finally, by using statistic processing, traversability grid maps are constructed by computing elevation statistics from the set of 3D points residing within each grid cell, namely, the maximum, minimum, variance of height and slope. These methods were first introduced by Langer et al. (1994) and in the first projects it relied on hard thresholds according to the vehicle capabilities. By definition, affordances cannot rely solely on the aspect of the object but also on the abilities of the subject, therefore it is also important to include robot dependent variables in order to correctly estimate traversability. Ugur et al. (2007) created a robot (KURT3D) that learned to perceive traversability by geometric analysis of objects. The robot, equipped with a 3D laser scanner, could navigate through a room filled with spheres, boxes and cylinders and it managed to distinguish between non-traversable objects (boxes, upright cylinders or lying cylinders in certain orientation) and traversable objects (spheres and lying cylinders in a rollable orientation). The system proved that geometric study of the environment can be successfully used to perceive the traversability affordance.

Appearance based analysis methods rely on image-processing classification in other to estimate the traversability. This kind of analysis usually have different sets of terrain classes labeled as traversable or not and by image comparison it determines the possible traversability. Some features used in geometric analysis, such as roughness, slope, discontinuity and hardness can also be obtained by means of image processing. This can give the system another mean of data collection to further refine the decision. With the improving quality of digital cameras and the growing processing power of computers, appearance based traversability can be further refined from the point of view of the structure of the underlying raw feature space.

These methods can be combined in order to create a system that offers the best qualities of each method. An evident complementarity exists between LIDAR and vision sensors which has been exploited within several works in order to extend the range of operation conditions and increase the overall robustness. These approaches may be referred as hybrid to denote cases where traversability analysis is being performed by fusion of the two main categories of sensory data and occasionally from other heterogeneous sources of data. Kim et al. (2006) created an on-line learning mechanism to accurately predict the traversability affordance. The system was based on a few assumptions. Firstly that visual features derived from stereo vision and color imagery are sufficient to discriminate between terrain regions from the standpoint of the traversability affordance. It was assumed that the system could determine the navigation experience of the robot reliably enough to label the

traversability of terrain regions as the robot attempts to drive over them. Also, the system could establish the correspondence between terrain regions in the local neighborhood of the robot and visual features that result from imaging the terrain regions using a standard stereo rig. Finally, that the system could afford to explore the terrain features in its environment without endangering the overall success of its mission was also an assumption. Starting with the observation that traversability is in the most general sense an affordance, the system implemented an on-line learning method that could accurately predict the traversability properties of a complex terrain using a stereo camera, and both geometric features of the terrain and appearance data. By separating the traversability classifier into two different steps, close range and long range, Manduchi et al. (2005) created a novel system that implemented two different algorithms to two different perspectives on the same problem. Using a long-range 3D obstacle detection and terrain color classification, implemented through a color stereo camera based on stereo range measurement and a color-based classification system to label the detected obstacles according to a set of terrain classes appearance, and a single-axis LIDAR for close-range analysis, to allow the system to discriminate between grass and obstacles such as tree trunks or rocks, the system proved viable and robust for unsupervised autonomous navigation in off-road environments. Dang and Hoffmann (2005) created a model that does not need any a priori information about the shape of the observed objects, but relies on the basic assumption that 3D points standing out of the estimated ground-planes are rigid and therefore obstacles. Santana et al. (2011) model introduced a hybrid approach. Large non planar objects were classified as obstacles while on smaller objects the geometrical relationships between neighbor 3D points were considered.

On the system developed in this thesis, the goal was to join the advantages of a proprioceptive method allied with a exteroceptive method in order to perceive the traversability information without the danger of damaging the system or the environment, resorting to a proprioceptive sensor conceived to determine traversability. As a proprioceptive sensor, a novel pan-tilt telescopic antenna was created and used. The robot's characteristics and the interaction methods are closely related to be able to predict the traversability of the environment based on a controlled interaction. The dimensions of the interaction sensor and the torque it provides must be adequate to the size of the platform. The system was further refined by implementing a learning mechanism based on close range geometry and memory recall.

2.6 Self-Supervised Learning

Autonomous navigation in unstructured natural environments is quite challenging because as opposed to more traditional urban environments, the lack of structured components in the scenes complicates the design of even basic functionalities such as obstacle detection. As seen in Section 2.5, obstacles can be labeled resorting to geometric descriptors, appearance descriptors or by exteroceptive sensor data, but the affordances that are available to the robot environment system are difficult to hard code due to the unpredictability of unstructured environments. Self-supervised learning refers to the ability of systems to generate their own general rules based on the sensory input, ability of the subject towards the environment and the result of that interaction.

For the 2005 DARPA Grand Challenge robot race, a driverless car competition consisting of 212 km of off-road course near the California/Nevada state line, Dahlkamp et al. (2006) won the race with

a vehicle that implemented a self-supervised learning module that added increased robustness to detect drivable paths. By combining data from a laser range finder and a pose estimation system, the system could identify drivable surfaces, and using a color camera the system would generalize that patch of drivable surface outward into the far range. This was important to allow the vehicle to achieve greater speeds when it had higher confidence that a drivable path was ahead.

Kim et al. (2006) implemented a self-supervised system that could predict traversability. Based on close range exteroceptive sensors, the system could sense the traversability of the environment and learned from its appearance. Then it applied the learning vectors to a fixed radius around the robot and predicted the traversability of the surroundings, even on previously unseen terrain. Bajracharya et al. (2009) applied the concepts of self-supervised learning into a real-time system for autonomous off-road navigation that based on proprioceptive sensors, operator input and stereo cameras, adapted to local terrain and generalized those rules to the extended terrain. The short-range geometry-based classifier learned from proprioceptive examples and the image-based long-range classifier learned from the geometry-based classification and generalized those rules to appearance and to further distances. Contrary to more traditional autonomous off-road approaches that rely on traversability based on fixed parameters, this system obtained good results in correlating its observations of the terrain with signals from its proprioceptive sensors while exploring its environment, enabling it to learn the traversability of the terrain on-the-fly. This created a system that could perceive the traversability on-line and autonomously. Wellington et al. (2006) model shows a solution for navigation in unstructured outdoor environments. A terrain model is used in combination with the of the vehicle to find a dynamic trajectory that avoids obstacles while protecting against roll-over, body collisions, high-centering, and other safety conditions. Using a generative, probabilistic approach to modeling terrain, the model exploits the 3D spatial structure inherent in outdoor domains and an array of noisy but abundant sensor data to simultaneously estimate ground vegetation height and classify obstacles. The system applied two Markov random fields and a latent variable that encodes the assumption that vegetation of a single type has a similar height.

In the presented work, the system implements a self-supervised learning mechanism where all the previous experiences are considered when a new scenario is presented. The more objects the system interacts with, the faster and more productive the next encounters will be, while if something different is presented, the system recognizes it and takes more time to correctly assess the new objects properties.

Chapter 3

Supporting Concepts

This chapter introduces the reader to some concepts used in order to develop and implement the system described in this document. Sensor information storage methods, like Point Clouds, is approached (see Section 3.1). Point Clouds was the primary data format used for developing the system. To simplify the Point Cloud data and allow for faster computation, additional data representation and storage methods were used (see Sections 3.3, 3.4 and 3.5). A method for parameters estimation, RANSAC, is explained in Section 3.2. RANSAC was applied in the system in order to recognize and identify the antenna appearance in the Point Cloud data. Finally, an introduction to the software frameworks used for real world implementation are presented in Section 3.6.1 and Section 3.6.2.

3.1 Point Clouds

A point cloud is a data structure used to represent a collection of multi-dimensional points and is commonly used to represent 3D data. The points are usually represent in a X, Y and Z geometric coordinates of an underlying sampled surface. When color information is present, the point cloud becomes 6D. Contrary to images, where only 2D information is stored, this method of acquisition and structure of data provides more information about the sampled surface. Relative distances between points and absolute distance to the sensor can be easily calculated and with great accuracy. This enables greater interaction detail due to the fact that the geometry of the environment is more deeply understood by the system. Figure 3.1 shows and example of a point cloud representation of a torus.

Point clouds are usually converted to polygon mesh or triangle mesh models to represent models with a more familiar look. This process is commonly referred to as surface reconstruction. Some techniques commonly used involve building a network of triangles over the existing vertices of the point cloud, while other approaches convert the point cloud into a volumetric distance field and reconstruct the implicit surface so defined though a marching cubes algorithm (Linsen, 2001).

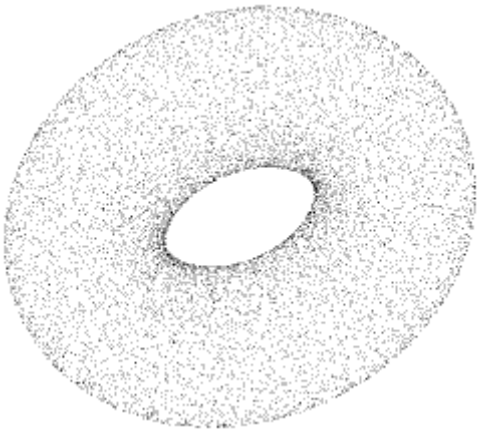

Figure 3.1: A point cloud image of a torus.

There are several methods and sensor types to obtain a point cloud. Some sensors use lasers and measure the reflected light from the object, in order to calculate distances (LIDAR), other sensors use a range imaging camera that resolves distance based on the known speed of light, measuring the time-of-flight of a light signal between the camera and the object for each point of the image (time-of-flight camera), while stereo cameras use two or more lenses with a separate image sensor to simulate human binocular vision, giving the ability to capture 3D images. Other method to obtain point cloud data is by using structured light. This process consists in projecting a known pattern (pixels, grids or horizontal bars) on an object and by verifying the deformation when striking the surface it allows the vision systems to calculate the depth and surface information.

3.2 Random Sample Consensus (RANSAC)

RANSAC is an iterative algorithm, proposed by Fischler and Bolles (1981), used to estimate parameters of a mathematical model from a set of observed data which contains outliers. The results improve the more iterations are allowed.

A basic assumption is that the data consists of inliers (data whose distribution can be explained by some set of model parameters, though may be subject to noise), and outliers (data that do not fit the model). The outliers can come from extreme values of the noise or from erroneous measurements or incorrect hypotheses about the interpretation of data. RANSAC also assumes that, given a (usually small) set of inliers, there exists a procedure which can estimate the parameters of a model that optimally explains or fits this data. The percentage of outliers which can be handled by RANSAC can be larger than 50 % of the entire data set. This percentage is known as the breakdown point and is commonly assumed to be the practical limit for many other commonly used techniques for parameter estimation.

Despise many modifications, the RANSAC algorithm is essentially composed of two steps that are repeated in an iterative fashion (Linsen, 2001):

Hypothesize - First minimal sample sets (MSSs) are randomly selected from the input dataset

and the model parameters are computed using only the elements of the MSS. The cardinality of the MSS is the smallest sufficient to determine the model parameters (as opposed to other approaches, such as least squares, where the parameters are estimated using all the data available, possibly with appropriate weights).

Test - In the second step RANSAC checks which elements of the entire dataset are consistent with the model instantiated with the parameters estimated in the first step. The set of such elements is called consensus set.

RANSAC terminates when the probability of finding a better ranked consensus set drops below a certain threshold. In the original formulation the ranking of the consensus set was its cardinality (consensus sets that contain more elements are ranked better than consensus sets that contain fewer elements).

An advantage of RANSAC is its ability to do robust estimation of the model parameters with a high degree of accuracy even when a significant number of outliers are present in the data set. A disadvantage is that there is no upper bound on the time it takes to compute these parameters. If the number of iterators is insuficient the solution obtained might not be optimal and it may not even be one that fits the data in a good way. RANSAC can only estimate one model for a particular data set. If two or more model instances exist, RANSAC may fail to find either one.

3.3 Voxel Grid

A voxel can be described as a set of small 3D boxes in space. It represents a value on a regular grid in three dimensional space. A voxel is a combination of "volume" and "pixel", where pixel is a combination of "picture" and "element". Voxels do not typically know their absolute position but are situated by their relative position to other voxels, in contrast to points and polygons where their position is often explicitly represented by the coordinates of their vertices. A voxel represents a single sample, or data point, on a regularly spaced, three-dimensional grid. This data point can consist of a single piece of data, such as an opacity, or multiple pieces of data, such as a color in addition to opacity. A voxel represents only a single point on this grid, not a volume; the space between each voxel is not represented in a voxel-based dataset. Depending on the type of data and the intended use for the dataset, this missing information may be reconstructed and/or approximated, e.g. via interpolation. A direct consequence of this difference is that polygons are able to efficiently represent a simple 3D structure with lots of empty or homogeneously filled space, while voxels are good at representing regularly samples spaces that are non-homogeneously filled. Voxel images are primarily used in the field of medicine and are applied to X-Rays, CAT (Computed Axial Tomography) Scans, and MRIs (Magnetic Resonance Imaging).

A voxel grid finds the points inside each voxel (3D box) and all the points present will be approximated (downsampled) with their centroid. This approach is slower than approximating them with the center of the voxel, but it represents the underlying surface more accurately. Figure 3.2 shows the difference between voxel representations.

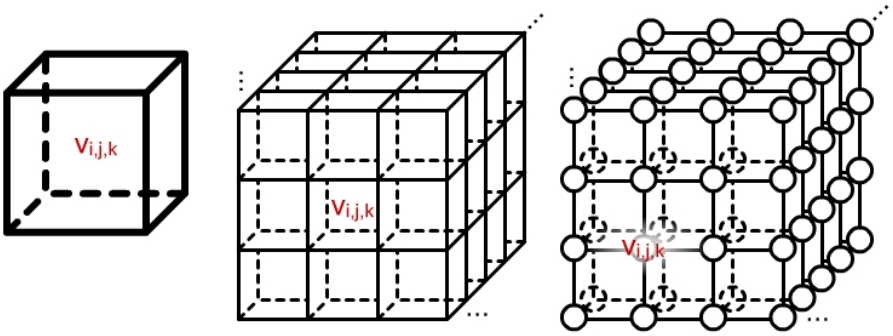

Figure 3.2: Voxel representations. The image on the left represents a single voxel, while the middle represents a voxel set. The image on the right shows a voxel grid. (Zirbes, 2014)

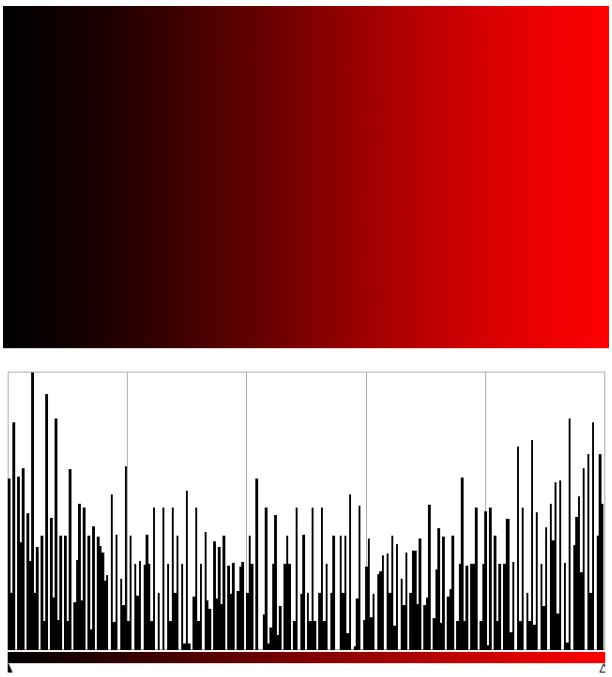

Figure 3.3: Example of a bar histogram representing the distribution of intensity levels for the red channel.

3.4 Histograms

A histogram is a graphical representation of the distribution of data. Tabulated frequencies, shown as adjacent rectangles, are erected over discrete intervals (bins), with an area equal to the interval (frequency divided by the width of the interval). A histogram may also be normalized displaying relative frequencies. Histogram are used to plot the density of data and often density estimation. They are often applied as image descriptors. Acting as a graphical representation of the tonal distribution in a digital image, it plots the number of pixels for each tonal value. Figure 3.3 shows a red gradient image and the corresponding intensity level histogram. As can be seen in the example, this image has a pretty consistent intensity distribution, as it is expected in gradients. In the system developed in this thesis, histograms are used generate comparison methods to different object's point clouds. Each bin contains the frequency of points found in a space interval.

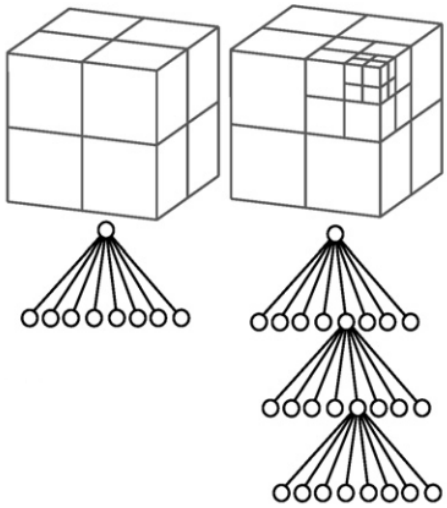

Figure 3.4: Space discretization of a cube and the corresponding octree depth. (Ferrando et al., 2011)

3.5 Octree

An octree is a tree data structure in which each internal node has exactly eight children. First presented by Meagher (1982), this data structure is used to represent arbitrary 3D objects to any specified resolution in a hierarchical 8-ary tree structure. Due to the unpredictable nature of object's shapes (concave, convex, inclusion of holes, either exterior or interior, disjoint parts or sculptured surfaces), 3D representations of objects required high processing power and a large quantities of memory. Also, prior representation techniques were not sufficiently robust to easily handle the object's complexities required in a realistic environment. Manipulation and display algorithms performing functions such as interference detection (two or more objects occupying the same region of space) and hidden surface removal (necessary for realistic display) required extremely large numbers of calculations in practical situations. Their complexity was usually exponential growth and processing power was not available. Octree geometric modeling was created to develop a capability to represent any 3-dimensional or N-dimensionl object to any specific resolution in a common encoding format; to operate on any object or set of objects with the Boolean operations and geometric operations; to implement a computationally efficient (linear) solution to the N-dimensional interference problem; to develop the capability to display in linear time any number of objects from any viewpoint with color, shading, shadowing, multiple illumination sources, transparent objects, orthographic or perspective view and smooth edges (anti-aliasing); and finally to develop a scheme that can be implemented across a large number of inexpensive high-bandwidth processors that do not require floating-point operations, integer multiplication or integer division. Figure 3.4 represents the steps of the subdivision methodology into octants and the octree representation of the object.

3.6 Supporting tools

In this section a brief overview of the frameworks used that implemented the discussed algorithms shown in this chapter is presented.

3.6.1 Robot Operating System (ROS)

The Robot Operating System (Quigley et al., 2009) is a flexible open-source framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. It allows for fast deployment and provides plenty of tools for a quick and inexpensive project implementation.

The primary goal of ROS is to support code reuse in robotics research and development. This aims to reduce the implementation time in new systems. ROS is distributed framework of processes (based on nodes) that enables executables to be individually designed and loosely coupled at run-time. These processes can be grouped into packages and stacks, which can be easily shared and distribuited. ROS also supports repositories to enable collaboration between different entities. ROS is also designed to be thin and usable with other robot software, uses any available library, is language independent, provides tools for testing and debugging (rostest) and allow easy scaling.

ROS has three levels of concepts, Filesystem level, Computation Graph level, and the Community level. In addition to the three levels of concepts, ROS also defines two types of names, Package Resource Names and Graph Resource Names.

The Filesystem level most important concepts are packages, message types and service types. Other concepts include metapackages, specialized Packages to present other type of packages; package Manifests, to describe a package; and repositories, which consist in a collection of packages that share a common VCS system. Packages are the building block in ROS software. A package might contain nodes, a dataset, configuration files or any type of software. The goal of packages is to provide useful functionality in an easy-to-consume manner so that software can be easily reused. Message types and service types describe how the messages are constructed. This structure defines how the ROS nodes transmit messages to each other and how they publish their available services.

The Computation Graph level is the peer-to-peer network of ROS processes that are processing data together. The main concepts of this level are Nodes, Master, messages, services, topics and bags. Nodes are in where the computation is performed. A package can include many nodes, i.e., one for each sensor or actuator in a robot. The ROS Master provides name registration and look up to the rest of Computation Graph. The Master allow the nodes to see each other and to exchange messages and invoke services. To communicate with each other, nodes rely on messages. A message is a data structure comprising typed fields, defined in the Filesystem level. ROS has predefined many message types but custom ones can be created. Similar to messages, nodes can also broadcast the services they provide. Services are published by the ROS node and via a request/reply other nodes can use the published services. Topics are named buses over which nodes exchange messages. Topics have anonymous publish/subscribe semantics, which decouples the production of information from its consumption. This means that nodes are not aware of who they are communicating with, and the connection between nodes is made through topics. There can be multiple publishers and subscribers to a topic. Finally, bags are a format to which ROS relies to save and playing back ROS message data. This facilitates the development and testing of algorithms.

Figure 3.5 exemplifies the ROS communication. When a node needs a service, it consults the available topics, and if the service is available, the topic notifies the offering node. After that, the

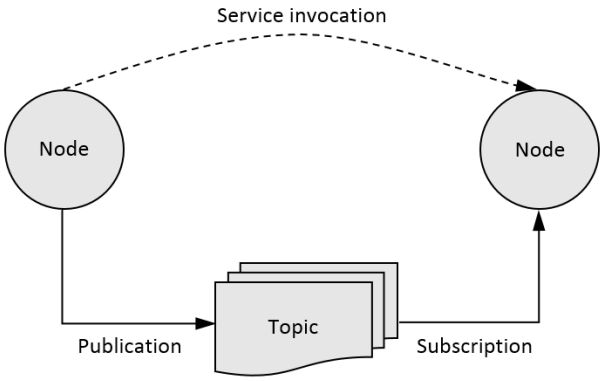

Figure 3.5: ROS service communication between nodes diagram (adapted from ROSWiki (2014)).

requesting node can request services to the node, by a request/reply model.

The last level of concept of ROS, the Community Level, provides the support for the community to exchange software and knowledge. These resources include distribuitions, to facilitate the installation of software and version management; repositories; the ROS Wiki, where all the documentation and tutorials are maintained; bug ticket system and mailing lists.

3.6.2 Point Cloud Library (PCL)

Point Cloud Library (PCL, Rusu and Cousins (2011)) is a standalone open-source framework for 2D/3D image and cloud processing. Written in C++, this cross-platform framework has been successfully compiled and deployed in Linux, MacOS, Windows and Android/iOS. PCL is developed by a large consortium of researchers and engineers around the world. This framework contains numerous state of the art algorithms for filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. These algorithms can be used, for example, to filter outliers, from noisy data, stitch 3D point clouds together, segment relevant parts of a scene, extract keypoints and compute descriptors to recognize objects in the world based on their geometric appearance, and create surfaces from point clouds and visualize them. These algorithms can be used in a wide range of applications, from perception in robotics to reconstruction of the world in 3D.

In its architecture, PCL is split into modular libraries. The used on this thesis are:

Filters - Used to remove noise and reduce measurement errors present in shadow points. This filtering is made by using statistical analysis and compute the distribution of points to decide which are removable.

Features - This library contains data structures and mechanisms for 3D feature estimation from point cloud data. 3D features are representation at certain 3D points, or positions, in space, which describe geometrical patters based on the information available around the point. The data space selected around the query point is usually referred to as the k-neighborhood. The most widely used geometric point features are the underlying surface's estimation curvature and a normal at a point cloud query point.

Registration - Combining several datasets into a global consistent model is usually performed using a technique called point set registration. The key idea is to identify corresponding points between the data sets and find a transformation that minimizes the distance (alignment error) between corresponding points. This process is repeated, since correspondence search is affected by the relative position and orientation of the data sets. Once the alignment errors fall below a given threshold, the registration is said to be complete. The registration library implements a plethora of point cloud registration algorithms for both organized and unorganized (general purpose) datasets. For instance, PCL contains a set of powerful algorithms that allow the estimation of multiple sets of correspondences, as well as methods for rejecting bad correspondences, and estimating transformations in a robust manner.

Octree - The octree library provides efficient methods for creating a hierarchical tree data structure from point cloud data (see Section 3.5). This enables spatial partitioning, downsampling and search operations on the point data set. Each octree node has either eight children or no children. The root node describes a cubic bounding box which encapsulates all points. At every tree level, this space becomes subdivided by a factor of 2 which results in an increased voxel resolution. The octree implementation provides efficient nearest neighbor search routines, such as Neighbors within Voxel Search, K Nearest Neighbor Search and Neighbors within Radius Search. It automatically adjusts its dimension to the point data set. A set of leaf node classes provide additional functionality, such as spacial occupancy and point density per voxel checks. Functions for serialization and deserialization enable to efficiently encode the octree structure into a binary format.

Sample Consensus - The sample consensus library holds SAmple Consensus (SAC) methods like RANSAC (see Section 3.2) and models like planes and cylinders. These can be combined freely in order to detect specific models and their parameters in point clouds. Some of the models implemented in this library include: lines, planes, cylinders, and spheres. Plane fitting is often applied to the task of detecting common indoor surfaces, such as walls, floors, and table tops. Other models can be used to detect and segment objects with common geometric structures.

IO - The IO library contains classes and functions for reading and writing point cloud data (PCD) files, as well as capturing point clouds from a variety of sensing devices.

Visualization - The visualization library was built to allow rapid prototyping and visualization of algorithms operating on 3D point cloud data. The library offers methods for rendering and setting visual properties (colors, point sizes, opacity, etc) for any n-D point cloud datasets, methods for drawing basic 3D shapes on screen (e.g., cylinders, spheres, lines, polygons, etc) either from sets of points or from parametric equations, a histogram visualization module for 2D plots, a multitude of Geometry and Color handlers, and a Range Image visualization module.

Common - Contains the common data structures and methods used by the majority of PCL libraries. The core data structures include the PointCloud class and a multitude of point types that are used to represent points, surface normals, RGB color values, feature descriptors, etc. It also contains numerous functions for computing distances/norms, means and covariances, angular conversions, geometric transformations, and more.

Search - The search library provides methods for searching for nearest neighbors using different data structures, including KdTree, Octree, brute force, and specialized search for organized datasets.

Chapter 4

Proposed Model

This chapter describes the proposed model, in both a software and physical perspective. The main focus of the system is to provide a simple and fast solution in sensor and process power for mobile a autonomous vehicle mainly focused on natural environments. The algorithms used are appliable to any kind of navigation.

4.1 Robot Model

For the purposes of the application of this system, the robot model must comply with some hardware requirement. The system was projected only to use a depth sensor capable of producing 3D point cloud data and a robotic arm with an end effector capable of sensing if it got stuck in the environment. A point cloud is a set of data points in the X, Y and Z coordinate system. The position of the sensor is relevant, because the algorithms described on this thesis are based on the capacity of the robot to overcome an obstacle that is aligned with the sensor, so this should be placed at about wheel height, around the bumper level. Figure 4.1 shows a schematic of the model with the projected sensor and actuator requirements.

Due to the three main hardware components, in this model there are considered three different frames of reference, one for each component. As shown in Figure 4.1, $\mathcal{O}$ is the frame of reference at the base of the robot, $\mathcal{C}$ is the frame of reference from the sensor and $\mathcal{A}$ is the frame of reference from robotic arm.

4.2 Model Overview

In this section the global control system architecture is explained for this model. At the start, the system performs a one time calibration in order to correctly interact with the environments (see Section 4.3). This process occurs to determine the position relation between the depth sensor and

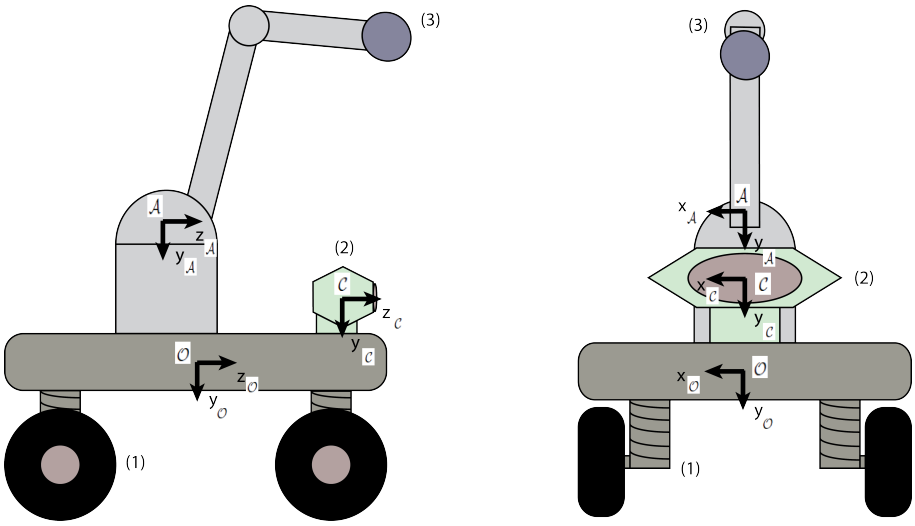

Figure 4.1: Front and side view of the robot model. **(1)** - Locomotion, **(2)** - Depth sensor, **(3)** - Robotic arm and end effector. in the figure are also shown the associated frame of reference for each component.

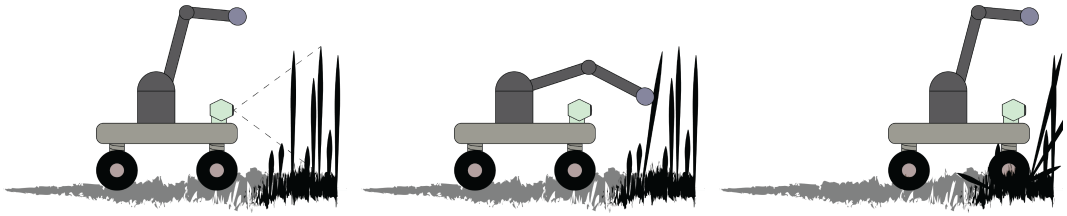

Figure 4.2: Proposed system's major steps. (Left) The robot finding an object with its depth sensor. (Middle) As the object's class is still new to the robot, the latter physically interacts with so as to learn its traversability. (Right) The robot overcoming the traversable object.

the robotic arm. While executing a given mission, e.g., moving towards a specified waypoint, the robot may face an object. This object can be traversable (e.g., vegetation) or not (e.g., a rock). To assess it, the robot creates a 3D descriptor of the found object and uses it to search its memory for the outcome of previous encounters with similar objects. If these previous encounters taught the robot that the object is traversable then the robot does not expend the effort of avoiding it. However, while traversing the object the robot may find itself stuck and, consequently, needs to update the memory to report that the object is not traversable. The outcome of consulting the memory may produce a low confidence result when the object is being seen for the first time or there have been ambiguous previous interactions with it. In this case the robot opts to perform an haptic interaction with the object. The higher the confidence the robot is on the contents of the memory, the coarser the haptic interaction must be. This allows the robot to reduce the time of interaction as the object gets known and, in the limit, when confidences rises to a certain level the interaction is skipped altogether, The result of the interaction is then used to update the memory in terms of how traversable is the object. In Figure 4.3 the global flowchart of the control system is shown. Figure 4.2 shows a schematization of the steps of evaluation, interaction and overcoming the traversable object.

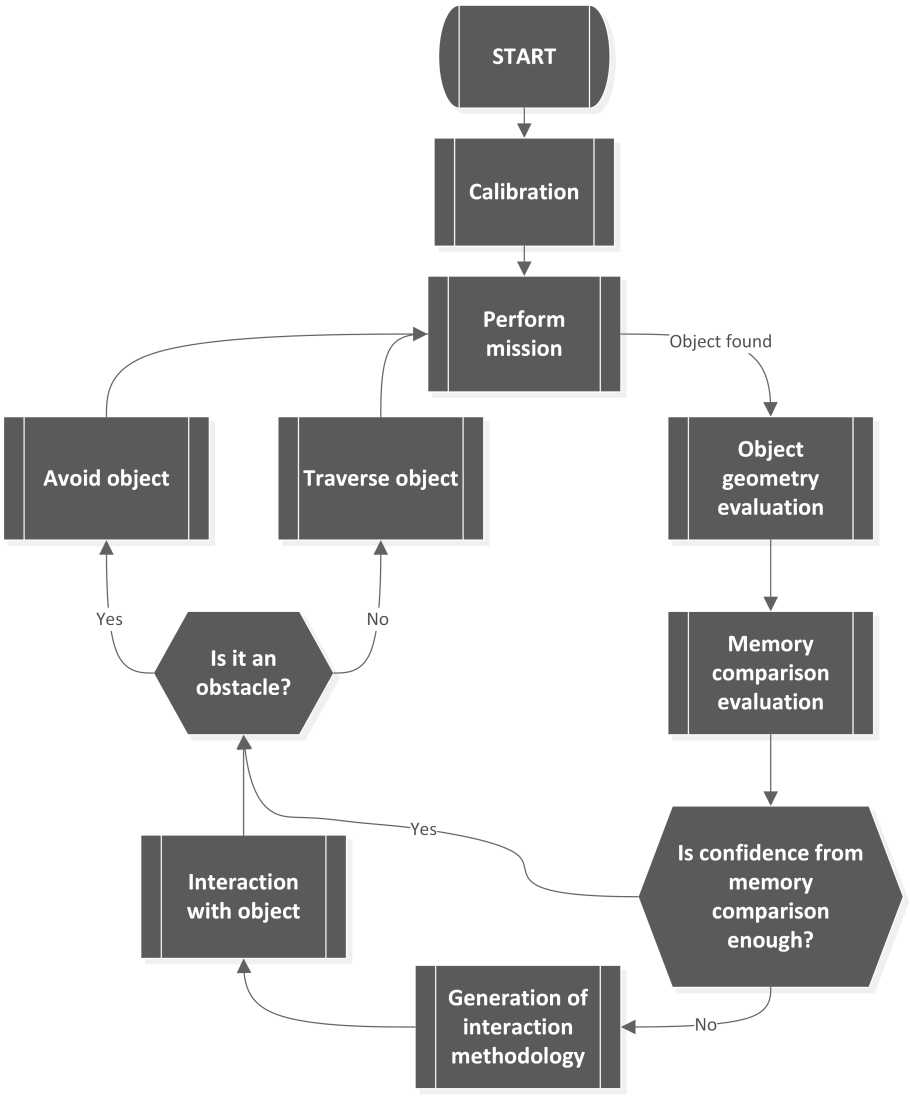

Figure 4.3: Proposed system's workflow.

4.3 Calibration

One of the main purposes of this system is the controlled interaction with the environment surrounding the robot, so it is fundamental to correctly interact with the objects detected by the sensor. Considering both axis from the arm and the sensor, $\mathcal{A}$ and $\mathcal{C}$, through interaction between them is possible to compute the transformation matrix M, of 4x4 dimension. A point represented in the $\mathcal{C}$ referential is shown as $C_{\mathcal{C}}$=($X_{\mathcal{C}}, Y_{\mathcal{C}}, Z_{\mathcal{C}}, 1$), which can also be represented in the $\mathcal{A}$ referential as $C_{\mathcal{A}}$=($X_{\mathcal{A}}, Y_{\mathcal{A}}, Z_{\mathcal{A}}, 1$). Is possible to find the relation between a point in $C_{\mathcal{C}}$ and $C_{\mathcal{A}}$ as

$$C_{\mathcal{C}} = M \cdot C_{\mathcal{A}} \tag{4.1}$$

in a similar way, the relation between $\mathcal{A}$ and $\mathcal{C}$ can be found by using the inverse transformation matrix M^{-1}.

$$C_{\mathcal{A}} = M^{-1} \cdot C_{\mathcal{C}} \tag{4.2}$$

To learn matrix $\mathbf{M}$, the robot arm performs a babbling behaviour in order to cover its configuration space. Simultaneously, the robot tracks the arm's end effector with the depth sensor. This allows the robot to accumulate a set of n correspondences between points in the arm's and in the sensor's frames of reference, $C^j_{\mathcal{A}} \leftrightarrow C^j_{\mathcal{C}}, \forall j = 1, 2, \ldots, n$. Matrix $\mathbf{M}$ is then estimated with a least-square SVD-based closed-form solution to the problem $\sum_{j=1}^{n} ||C^j_{\mathcal{A}} - \mathbf{M}C^j_{\mathcal{C}}||^2$ (Haralick et al., 1989).

To learn the arm localization, the sensor first must learn about its surroundings. The first step is to capture a cloud from the sensor frame of reference without the arm visible. Cl_{bkg} represents the captured cloud without the robotic arm in its range. After that, the arm moves to a set of positions, for a number of times, n_p, set by the user, and every time it captures a new cloud, Cl_n. For performance reasons, a depth filter is applied to both clouds, Cl_{bkg} and Cl_n, with the maximum estimated range from the robots arm in the sensor frame of reference. The next step in the system is to utilize the sensor to detect the arms position in its frame of reference.

Using a 3D change detector background subtraction technique is possible to remove the points that are considered background (Cl_{bkg}) from the new Cl_n cloud. To remove the background, an octree spatial change detection technique was used. An octree (Meagher, 1982) is a tree data structure where the cloud data is subdivided into octants. This allows for faster access, indexations and other operations within the cloud. The octree spatial change detection allows to detect new leaf nodes and serialize their point indices, meaning that the undesired points can be removed from the cloud, creating the filtered cloud, Cl_f. Changes in the background caused by the wind, noise from the acquisition sensor or even changes in lighthing could be enough for the octree change detector algorithm fail at removing points, so a second procedure was implemented to reduce the false positives.

By utilizing a RANSAC (Fischler and Bolles, 1981) implementation the false positives are removed, ensuring that the system detects correctly the arms end effector. RANSAC is an iterative method that allows for the system to detect an appearance from a set inliners. Since it is an iterative method, it is quite slow, so that is why the background subtraction is applied to the cloud first. After the end effector is found, the system calculates the point of its position (either by finding its centroid

or some specific arm part), and stores the point **c** on the $\mathcal{C}$ frame of reference, as well as the point in the $\mathcal{A}$ coordinates.

After every time the system successfully determines the arm position on the sensor, it increments a counter, n_c, until **c** is found n_p number of times. These points are stored in a vector **C** and **A**, corresponding to the $\mathcal{C}$ and $\mathcal{A}$ frame of reference, respectively.

With the information of the coordinates in the $\mathcal{A}$ referential and the corresponding $\mathcal{C}$ coordinates and using a 3D rigid body transformation algorithm, the transformation matrix M is estimated. There are several methods to calculate the 3D rigid transformation matrix, some faster with worse results and some slower but more accurate (Eggert et al., 1997). For best results and fast processing time, Singular Value Decomposition (SVD) method is preferred. Algorithm 1 outlines the entire calibration phase.

Algorithm 1 Calibration pseudo-code.

Input: Vector **P** of arm positions; n_p number of points to capture; d depth filter limit
Output: Transformation Matrix, M
Data: Point C in $\mathcal{C}$ and point A in $\mathcal{A}$, **C** vector for C positions and **A** vector for A positions,

Capture background cloud, Cl_{bkg}, and apply depth filter with d value

Initialize $n_c \leftarrow 0$

Initialize **C** vector position and **A** vector position, (**C** $\in \emptyset$, **A** $\in \emptyset$)

while $n_p > n_c$ **do**

  Move arm position to P(n_c)
  Capture new cloud, Cl_n, and apply depth filter with d value
  Detect changes between Cl_{bkg} and Cl_n and store them in Cl_f [see Section 4.3]

  if $|Cl_f| > 0$ **then**

    Detect end effector appearance using RANSAC in Cl_f

    if End effector detected **then**

      Add C on the $\mathcal{C}$ frame of reference on **C** vector
      Add A on the $\mathcal{A}$ frame of reference on **A** vector
      $n_c \leftarrow n_c + 1$

    end if

  end if

end while

Estimate 3D rigid transformation, M, using correspondences between **C** and **A**

return M

4.4 Object Evaluation

In this chapter the algorithms and methods suggested for a fast object classifier and interaction methodology are presented. There are three main components for this evaluation: (1) Past experiences through memory evaluation; (2) object appearance by evaluating the best interaction points; and combining these two results, (3) an interaction methodology is created. When a new object is encountered a new memory entry is created in order to learn from it. The system stores in memory the object cloud, object point frequency histogram and information about its traversability. Figure 4.4 shows the flowchart of the processes involved in the object evaluation and interaction. For the interaction with the object both evaluation results are taken into account when generation the interaction methodology, as explained in Chapter 4.4.3.

4.4.1 Learning from Memory Evaluation

When the robot encounters a new object, it needs to perform a memory comparison in order to try to estimate the new action possibilities on it. This process can be divided in two key parameters, object description (Section 4.4.1.1), which creates a representation of an object in memory; and memory recall (Section 4.4.1.2), the methodology to compare several object descriptors in memory.

4.4.1.1 Object Descriptor

To create a fast memory comparison algorithm, it is first necessary to simplify the input point cloud. 3D clouds take a long time to compute because of their large size. Some methods allow for faster cloud operations and search, like octrees, or even cloud simplification, like voxelization or filtering, without losing important information. A histogram, $\mathcal{H}$, is a method of simplification where a bi-dimensional matrix stores the frequency of points in a determined area of the cloud, also known as bin.

Since this model is based on close interactions, the depth is filtered to points where the interaction is possible, i.e., inside the robotic arms reach, creating the cloud H_f. Using this approach, the information is expected to be computed faster while delivering consistent results. Similarly to image histograms, a point histogram can be used in order to simplify and later compare the characteristics of each object. The number of points inside each bin is counted, ignoring the depth, meaning that the bin is not 3D and the histogram is a projection of a cloud in a $\{x, y, z\}$ coordinate system in a xy plane, meaning that a point P = $(X_C, Y_C, Z_C, 1)$ is represented as p = $(x_C, y_C, 1)$. A grid of $x_{\mathbf{H}}$ by $y_{\mathbf{H}}$ dimensions is applied to the the point cloud in the $\{x, y, z\}$ coordinates after a simple voxelization, H_v, for faster processing and fewer data manipulation, in order to learn some characteristics about the objects that will be discussed later in this document. To create the grid, the resolution of the sensor and point density must be taken into account to know the bin size. Ideally, this number should come as a result of experimentation.

Algorithm 2 shows the processes involved in the histogram creation. Figure 4.5 shows an example of the grid being applied, to the cloud shown in Figure 4.6.

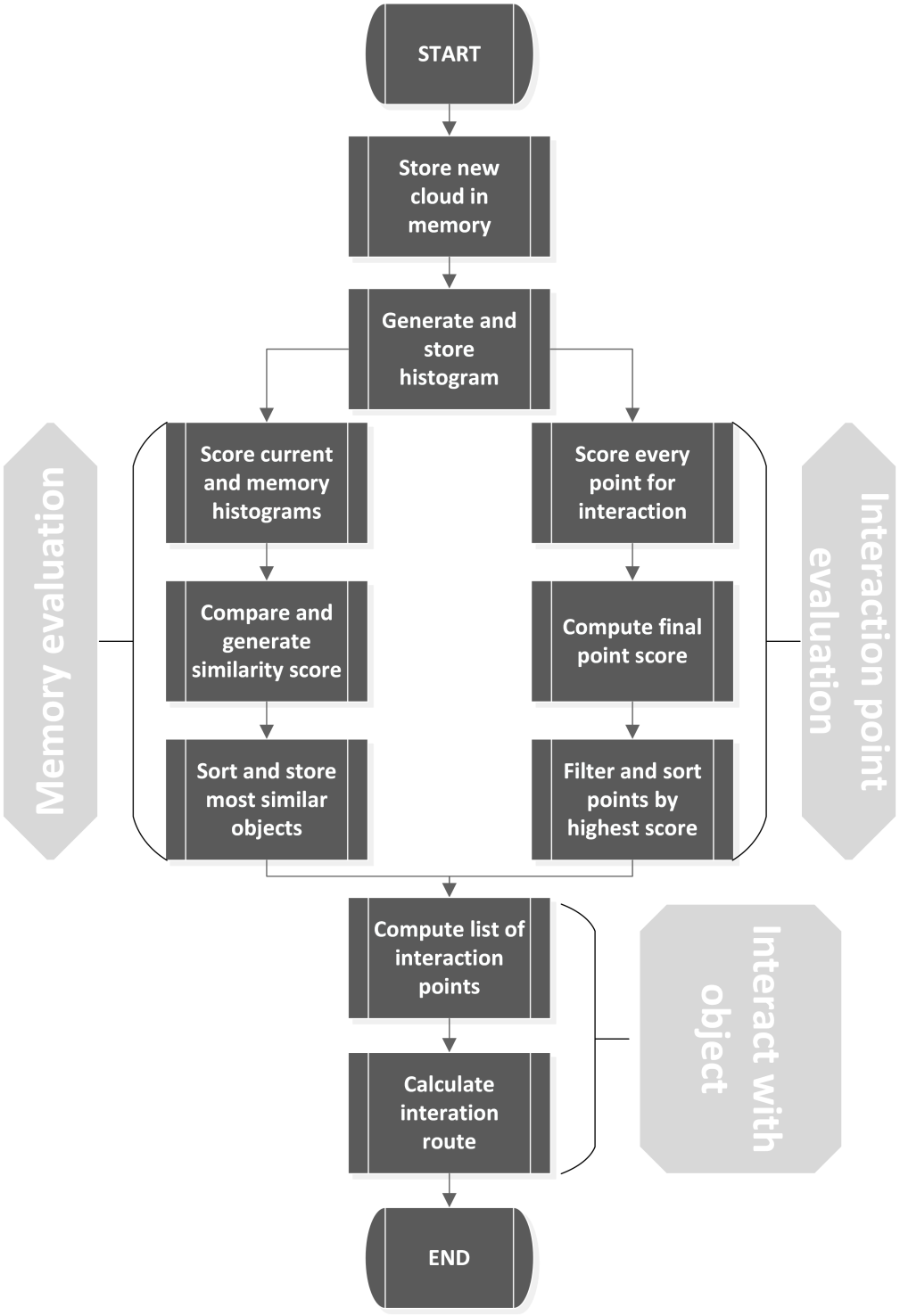

Figure 4.4: Flowchart showing both evaluations and interaction process. The memory evaluation process can be subdivided into three major steps as well as the interaction point evaluation, as shown in the figure. The interaction with the object is only described from the control perspective.

Algorithm 2 Histogram creation pseudo-code.

Input: Object cloud O; Grid size ($x_{\mathbf{H}}$ and $y_{\mathbf{H}}$); voxel size (v_x, v_y, v_z); maximum interaction depth d
Output: Histogram **H**
Data: Filtered cloud H_f; voxelized cloud H_v; Column iterator i; Line iterator j

Apply depth filter on cloud O with d limit, $H_f \leftarrow$ Depth filter (O)

Apply voxel grid with size (v_x, v_y, v_z) on cloud H_f, $H_v \leftarrow$ Voxelization (H_f)

Initialize histogram $\mathcal{H}$ as a bi-dimensional matrix of size ($x_{\mathbf{H}}$, $y_{\mathbf{H}}$)

for each P $\in H_v$ **do**

    Find the matrix position (i, j) bin where P belongs, on p coordinates
    H(i, j) $\leftarrow$ **H**(i, j) + 1

end for

return H

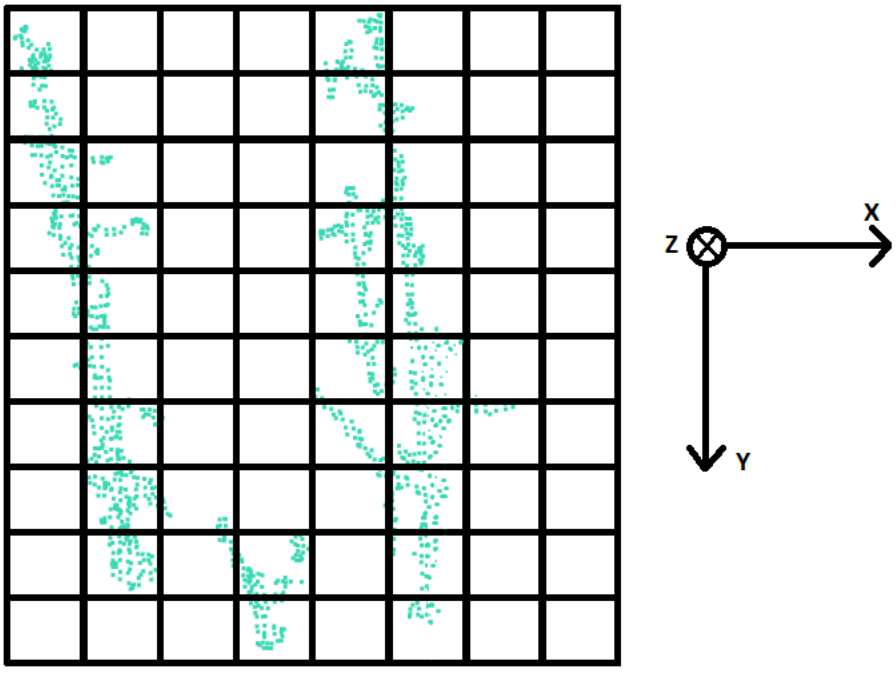

Figure 4.5: Grid of dimensions $x_{\mathbf{H}}$ = 8 and $y_{\mathbf{H}}$ = 10 being applied to a cloud.

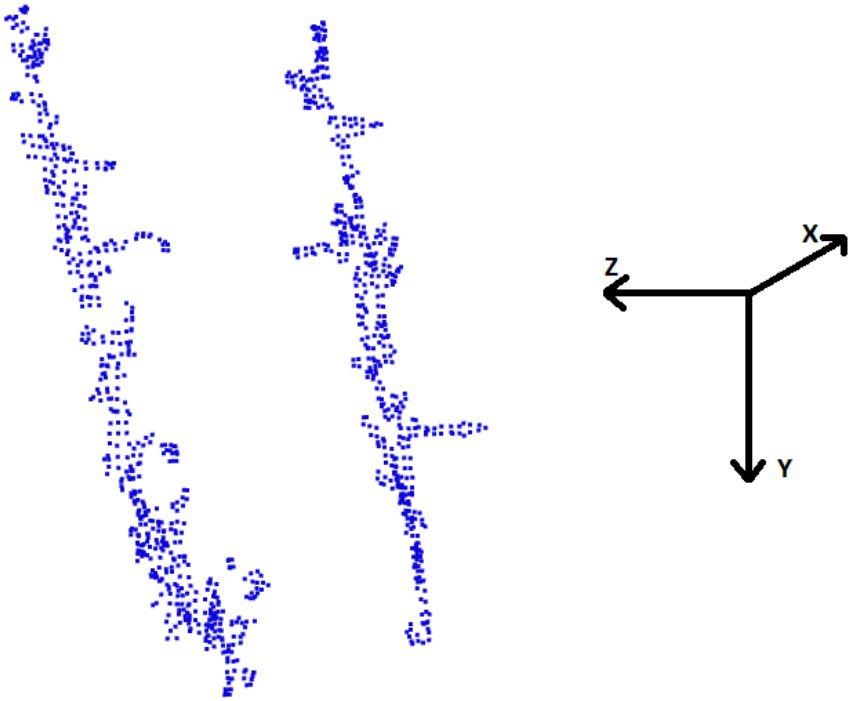

Figure 4.6: Perspective view of the cloud where the histogram grid was applied.

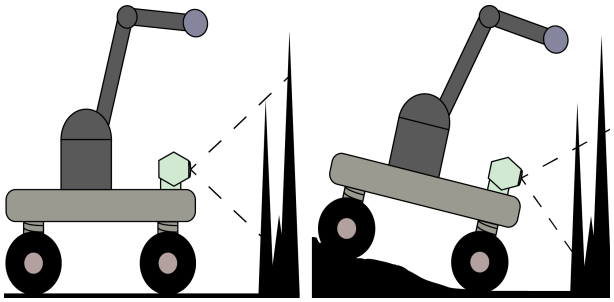

Figure 4.7: Change of inclination impact on the y axis. As seen in the figure, at close range the inclination does not have a big impact on the aspect of the object.

Based on heuristic knowledge, four description parameters were used. Each parameter produces a score per line j of the histogram. Only lines are used for description because since the object evaluation is performed at close range, the height variation is not very significant, presenting more variation on the x axis rather that on the y axis, as schematized on Figure 4.7.

The four parameters used for an object description. These parameters are based on the notion of density, size and continuity. The four metrics are formulated as:

Number of clusters per line - $N_{\mathbf{H}}^{j}$ This counts the number of adjacent bins occupied in line j, separated from each other by an empty bin in histogram **H**. Using as reference Figure 4.8, it can be observed that three clusters exist, because there are three set of bins separated with atleast one empty bin.

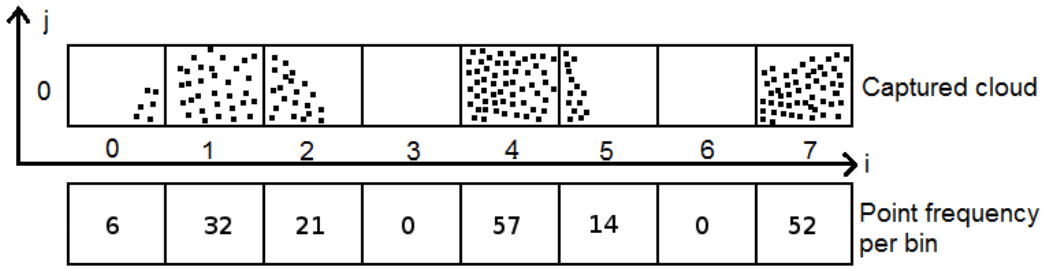

Figure 4.8: Example of a line evaluated by the described method. Since the histogram line j is 0, this line would have a value of $N_{\mathbf{H}}^0$ = 3, $W_{\mathbf{H}}^0$ = 3, $\rho_{\mathbf{H}}^0$ = 30.3 and $P_{\mathbf{H}}^0$ = 71.

Largest cluster per line - $W_{\mathbf{H}}^j$ Counts the number of bins of the largest cluster found in line j. As observed in Figure 4.8, by adding the number of bins in each cluster, it is observed that the largest number is $W_{\mathbf{H}}^0 = 3$.

Point density per line - $\rho_{\mathbf{H}}^j$ Density of points per bin. As seen in Figure 4.8, there are 8 bins in this line, and by adding the frequency of points, N_l, the result is N_l = 182. Being j_o the number of bins occupied in a line, this number is given by

$$\rho_{\mathbf{H}}^j = \frac{\sum N_l}{j_o}, \tag{4.3}$$

meaning that in the example $\rho_{\mathbf{H}}^0 = \frac{182}{6} = 30.3$.

Maximum points in a cluster per line - $P_{\mathbf{H}}^j$ Gives the number of maximum points in a cluster. Note that it does not necessarily mean that is the largest cluster. In the example in Figure 4.8, there are three clusters, and by adding the point frequency in all of then, it comes that the one that has the most points is the second one, so $P_{\mathbf{H}}^0 = 71$.

4.4.1.2 Memory Recall

The memory is composed of descriptor-traversability tuples. A tuple is built by associating the descriptor of the observed object and the physical interaction binary-valued outcome. In the current implementation, forgetting has not been implemented. Therefore, all interactions are stored and maintained throughout the robot's lifecycle.

When facing an object, the robot will search for similar objects stored in memory in order to determine the most likely navigation cost of the object. The object descriptors are compared line by line to the current object being scored, and in the end of the procedure they are merged to provide a similarity score. The calculated scores can not be negative, so if the result of a scoring parameter is negative it is clamped to 0. Being **H'** the histogram being evaluated and **H** a histogram in memory, the scoring for each parameter is computed as:

$N^j_{\mathbf{H}}(\mathbf{H'})$ is the score generated by comparing the $N^j_{\mathbf{H}}$ value from a histogram **H** and a histogram **H'**. If the value of $N^j_{\mathbf{H}}$ is the same in both histograms, the $N^j_{\mathbf{H}}(\mathbf{H'})$ score is 1, while otherwise it is 0. Being $N^j_{\mathbf{H}}$ the score of the histogram line in memory, and $N^j_{\mathbf{H'}}$ the score of the histogram line being evaluated:

$$N^j_{\mathbf{H}}(\mathbf{H'}) = 1 - \frac{1}{\upsilon} \cdot (|N^j_{\mathbf{H}} - N^j_{\mathbf{H'}}|), \tag{4.4}$$

where υ is a value dependent on the robot model. This allows for a maximum number of clusters of difference.

$W^j_{\mathbf{H}}(\mathbf{H'})$ is the score generated by comparing the $W^j_{\mathbf{H}}$ values between a line j of histogram **H** and a histogram **H'**. Being $W^j_{\mathbf{H}}$ the score of the histogram line in memory and $W^j_{\mathbf{H'}}$ the score of the histogram line being evaluated:

$$W^j_{\mathbf{H}}(\mathbf{H'}) = 1 - \frac{1}{\omega} \cdot (|W^j_{\mathbf{H}} - W^j_{\mathbf{H'}}|). \tag{4.5}$$

where ω is a value dependent on the robot model. This allows for a maximum number of bins of difference.

$\rho^j_{\mathbf{H}}(\mathbf{H'})$ is the score generated by comparing the $\rho^j_{\mathbf{H}}$ value in a line j between two histograms, **H'** and **H**. The score is given in relation with the difference factor between the two lines, of the histogram, Δ_ρ, which is given by

$$\Delta_\rho = \frac{|\rho^j_{\mathbf{H}} - \rho^j_{\mathbf{H'}}|}{\rho^j_{\mathbf{H'}}} \tag{4.6}$$

The relation between $\rho^j_{\mathbf{H}}(\mathbf{H'})$ and Δ_ρ is obtained experimentally because it depends on the sensor and robot used. Similarly with $\rho^j_{\mathbf{H}}(\mathbf{H'})$, the $P^j_{\mathbf{H}}(\mathbf{H'})$ score is generated by the comparing the $P^j_{\mathbf{H}}$ value in a line j between two histograms, **H'** and **H**. The score is given in relation with the difference factor between the two lines, of the histogram, Δ_P, which is given by

$$\Delta_P = \frac{|P^j_{\mathbf{H}} - P^j_{\mathbf{H'}}|}{P^j_{\mathbf{H'}}}. \tag{4.7}$$

The relation between Δ_P and $P^j_{\mathbf{H}}(\mathbf{H'})$ is also obtained experimentally. The similarity score per line j between two histograms **H'** and **H**, $S^j_{\mathbf{H}}(\mathbf{H'})$, is given by:

$$S^j_{\mathbf{H}}(\mathbf{H'}) = \alpha_S \cdot N^j_{\mathbf{H}}(\mathbf{H'}) + \beta_S \cdot W^j_{\mathbf{H}}(\mathbf{H'}) + \gamma_S \cdot \rho^j_{\mathbf{H}}(\mathbf{H'}) + \delta_S \cdot P^j_{\mathbf{H}}(\mathbf{H'}) \tag{4.8}$$

where

$$\alpha_S + \beta_S + \gamma_S + \delta_S = 1. \tag{4.9}$$

The next step is to evaluate the final similarity score between two complete histograms **H** and **H'** based on these scoring values, $S^{\mathbf{H'}}_{\mathbf{H}}$. The final similarity is computed by comparing lines at the same height, where 1 means it is quite similar and 0 that it is very different. Each parameter has its own weight in the final similarity score.

The final similarity score between two histograms is given by

$$S_{\mathbf{H}}^{\mathbf{H'}} = \frac{\sum_{k=0}^{j} S_{\mathbf{H}}^{k}(\mathbf{H'})}{y_{\mathbf{H}}}. \tag{4.10}$$

After computing the similarity between the new object histogram $\mathbf{H}'$, and all the other objects in memory **H**, the system stores the n closest neighbors for subsequent use in the learning procedure. All these processes are presented in Algorithm 3.

4.4.2 Interaction Points Evaluation

Even though learning from past experiences is important, the object alone can give information about its traversability. Since the robot model is equipped with a robotic arm for interaction, the system must analyze which would be the best areas to interact in order to learn about the object's traversability. Based on heuristic knowledge three classification categories were created for each point P in the $\mathcal{C}$ frame of reference; Height $C_h(P)$, distance to object centroid $C_d(P)$, and number of neighbors points $C_\rho(P)$. The final classification score, $C_T(P)$, is computed using all the other scores. Algorithm 4 shows the process of evaluation and sorting.

Height Evaluation The height evaluation is based on a gaussian function, where the height of the points of the sensor in relation with the robot are scored.

$$C_h(P) = e^{(-\frac{(y_{\mathcal{C}}-\mu)^2}{2\cdot\sigma^2})} \tag{4.11}$$

Where μ is the distance of the sensor to the wheel height, and σ is a value dependent of the robot and sensor architecture.

Distance to centroid Evaluation Similarly to the Height Evaluation, the Distance to centroid Evaluation is based on a normal distribution model to the distance of each point to the object centroid. Being ζ the centroid point of the object cloud, defined as ζ = ($x_{\mathcal{C}}$,$y_{\mathcal{C}}$,$z_{\mathcal{C}}$, 1), the score $C_d(P)$ is given by:

$$C_d(P) = e^{(-\frac{||P-\zeta||^2}{2\cdot\sigma^2})} \tag{4.12}$$

where σ is a value dependent of the robot and sensor architecture.

Number of neighbor points The final classification category the system uses is the notion of point density. In order to get consistent results, a voxelization of the cloud must be performed first.

Assuming that x_v, y_v and z_v is the voxel leave size in the x, y and z axis, V = (v_x,v_y,v_z), the neighbor radius (r_n) used is

Algorithm 3 Learning from Memory Evaluation pseudo-code.

```
1:
2: Input: Current object histogram, H'
3: Output: n clouds with highest similarity
4: Data: Similarity vector S, total score vector T
5:
6: Initialize vector S ∈ ∅
7:
8: // Generation of scores for object histogram
9: for each j from current object histogram H' do
10:
11:    Count number of clusters to generate N^j_H', and store value
12:    Find largest cluster to generate W^j_H', and store value
13:    Calculate point density to generate ρ^j_H', and store value
14:    Find maximum points in a cluster to generate P^j_H and store value
15:
16: end for
17:
18: // Generation of scores from memory stored objects
19: for each histogram in memory H do
20:    for each j of the histogram do
21:
22:       Count number of clusters N^j_H
23:       Find largest cluster W^j_H
24:       Calculate point density ρ^j_H
25:       Find maximum points in a cluster P^j_H
26:
27:       // Score comparison from memory objects and current object
28:       Generate N^j_H(H') [Equation 4.4]
29:       Generate W^j_H(H') [Equation 4.5]
30:       Generate ρ^j_H(H')
31:       Generate P^j_H(H')
32:       Compute similarity score S^j_H(H') for the line [Equation 4.8]
33:       Store score S^j_H(H') in vector S
34:
35:    end for
36:
37:    Compute total similarity score S^H'_H from vector S [Equation 4.10]
38:    Store S^H'_H score in vector T
39:
40: end for
41:
42: Sort T by highest S and store the n closest neighbours
43:
44: return n closest neighbours
```

$$r_n = \sqrt{(v_x)^2 + (v_y)^2 + (v_z)^2} + \tau \tag{4.13}$$

where τ is a distance dependent on the sensor resolution and accuracy desired.

It is assumed that the best possible score is an approximation of the maximum number of neighbors in the a sphere. Because the nature of the point clouds, the depth is ignored and it is the same for all the points in the cloud, because the voxel size and r_n is constant. The best score is estimated as:

$$M_{n_n} = \frac{2 \cdot \pi \cdot r_n^2}{v_x \cdot v_y} \tag{4.14}$$

Being $n_n(P)$ the number of neighbors inside a radius of a point P, the score for this parameter is given as

$$C_\rho(P) = \frac{n_n(P)}{M_{n_n}} \tag{4.15}$$

Total interaction score Having all three classification categories, the final point score for interaction, $C_T(P)$, is given by

$$C_T(P) = \alpha_C \cdot C_h(P) + \beta_C \cdot C_d(P) + \gamma_C \cdot C_\rho(P) \tag{4.16}$$

where

$$\alpha_C + \beta_C + \gamma_C = 1 \tag{4.17}$$

this allows for the values of α_C, β_C, and γ_C to be tuned, depending on the robot used and to the type of environment it is planned to interact with.

After the final interaction score is computed for every point in the cloud, the model ignores the points that are close to each other. Since the cloud points are usually very dense, a physical interaction will in practice interact with several points at once. The final sorting for the cloud interaction must take this factor into account. When the system is generating the sorted cloud point index by classification, it flags the surrounding points to the current highest scoring point so they cannot be listed. The interaction radius, r_{int}, must be calculated by taking into account the physical characteristics of the robot model arm end effector.

The classification is made before the sorting so all points are already classified beforehand, so when the system flags the cloud points that are close to the current highest scoring point, it will not affect the surrounding points classification. In Figure 4.9 is exemplified this sorting method.

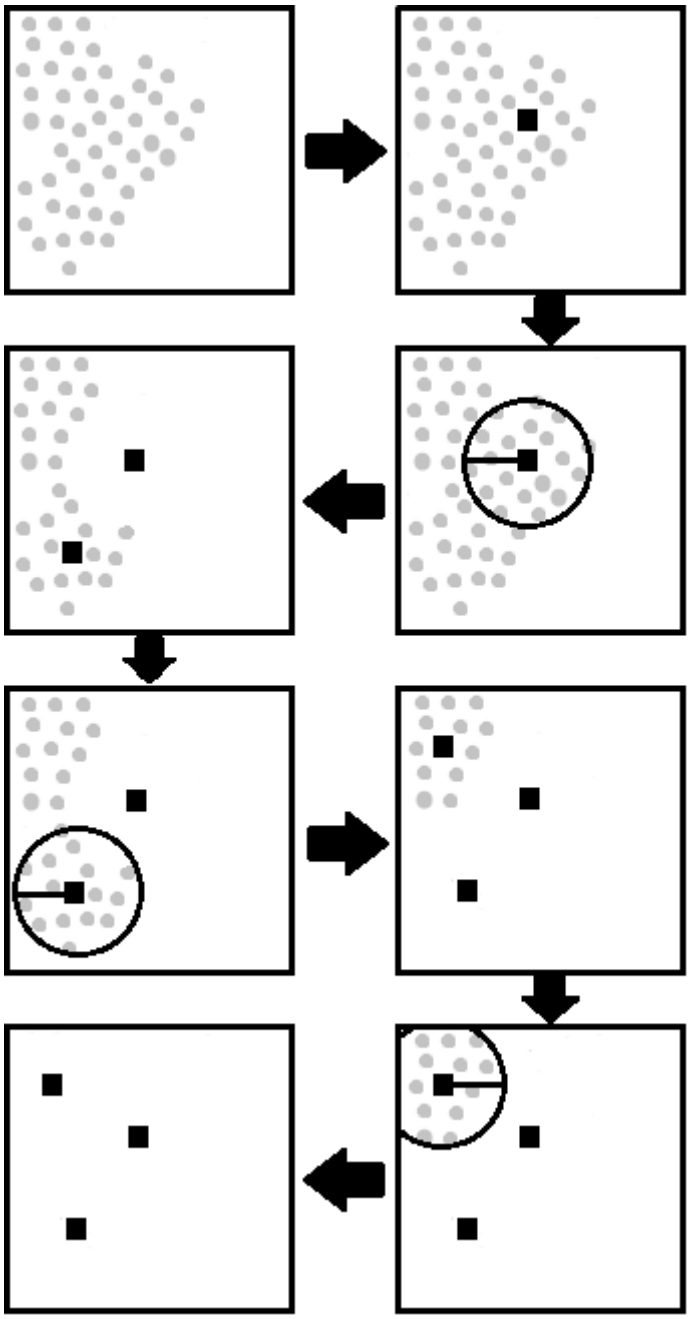

Figure 4.9: Example of the sorting algorithm filter being applied. The darker points are the current highest scoring points, and after removing points closer that a radius of r_{int}, the next higher scoring point is found and the process is repeated, until there are no more points left.

Algorithm 4 Interaction point evaluation and sorting pseudo-code.

```
1:
2: Input: Object cloud O; Voxel size
3: Output: Sorted point list P
4: Data: Neighbor points V_o
5:
6: P ← ∅
7: O_v ← Apply voxel grid on cloud O
8:
9: // Classification cycle for every point
10: while O_v ≠ ∅ do
11:
12:    o = argmax C_T(o) [Equation 4.16]
13:    o in O_v
14:    Find neighbor points, V_o [Figure 4.9]
15:    O_v ← O_v \ V_o
16:    P ← P ∪ o
17:
18: end while
19:
20: return P
```

4.4.3 Interaction with the Object

The final step in the object evaluation is the creation of the methodology to proceed to the interaction. To do so both scores, memory evaluation and interaction point evaluation, are taken into account. Since the first returns a list of n closest neighbours and the latter a list of the best points for interaction, it is necessary to merge the data together to make a decision. Every object in memory contains information about its traversability, T_o, which is a binary value of information about traversability, with 0 meaning that is not traversable and 1 that it is traversable. Being E_o the scores with the T_o label equal to obstacle, and E_t the scores with the T_o label equal to traversable, the score for each label similarity to a given object is

$$P_o = \frac{\sum_{u=0}^{k-1} E_o(u)}{k} \tag{4.18}$$

and

$$P_t = \frac{\sum_{u=0}^{k-1} E_t(u)}{k}. \tag{4.19}$$

The confidence on the score, C_k, is given as

$$C_k = max(P_t, P_o) \tag{4.20}$$

With the probability of knowing the object and the final score for point interaction, $C_T(P)$, a line is used to compute the points with which the system interacts with: (Equation 4.21)

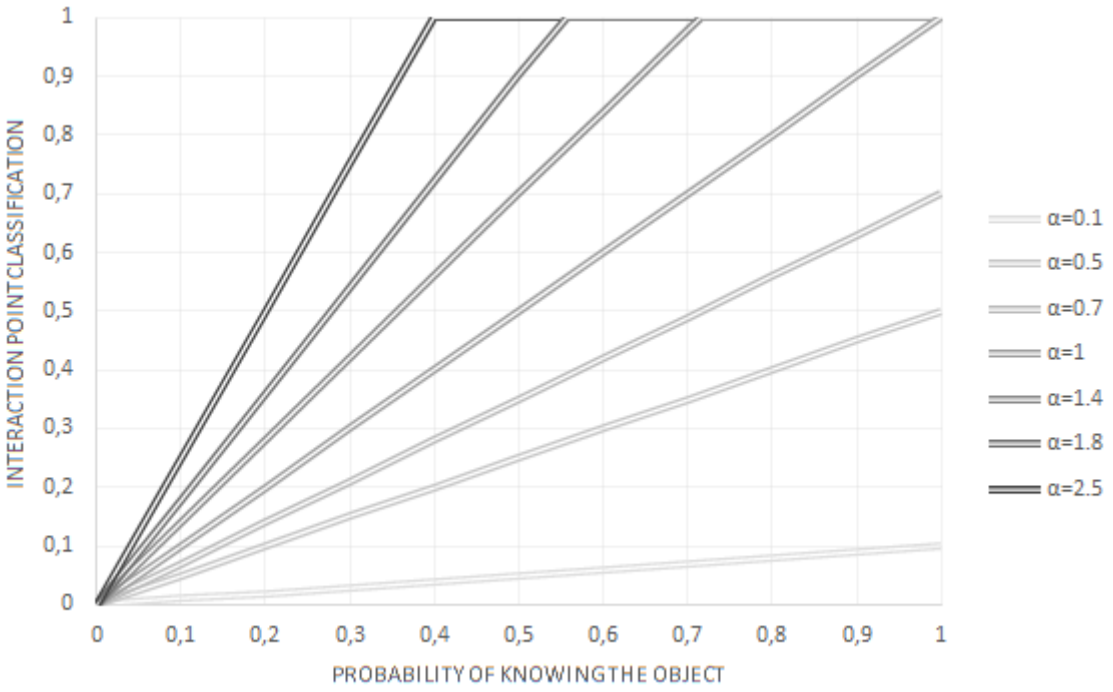

Figure 4.10: Effect of α on the confidence of the system. The system interaction occurs when the comparison between $C_k \cdot \alpha$ and $C_T(P)$ falls above the shown lines. For α = 1 both evaluations are equally weighted.

$$C_T(P) > \alpha \cdot C_k. \tag{4.21}$$

As shown in 4.10, by adjusting the α factor is possible to change the dynamics of the system. A higher α value allows for a bigger confidence on the memory and learning procedure, and a faster system as well because it will result in fewer interactions with the environment. On the other hand, a lower α value means that the system has a higher confidence on the interaction procedure rather than on the memory. It also means a slower system because it will result in a higher number of interactions with the environment. If the control system decides on not interacting with the new object, the system will assume that it is an obstacle or not, depending on the P_o and P_t value, choosing the highest.

The final step in the object evaluation process is the interaction with the object. After all the calculations are done the system stores a list of optimal cloud points for interaction considering the object appearance and what it was learned from previous experiences. If there are no optimal points for interaction, the control system assumes the traversability from the memory evaluation. When the points are computed, the first step is to find the furthest point from the interaction point cloud, C_{int}, in the z axis, F, and sweep all the points at that depth. After, the arm interacts with the rightmost point suggested, and goes to the next nearest point until every point is swept, or the arm gets stuck on the way. If the arm gets stuck on the way from each point, the system classifies the object as not traversable and avoids the object, otherwise it will try to traverse it. An example of the physical interaction of arm with the object is shown in Figure 4.11, and the psudo-code is explained in Algorithm 5.

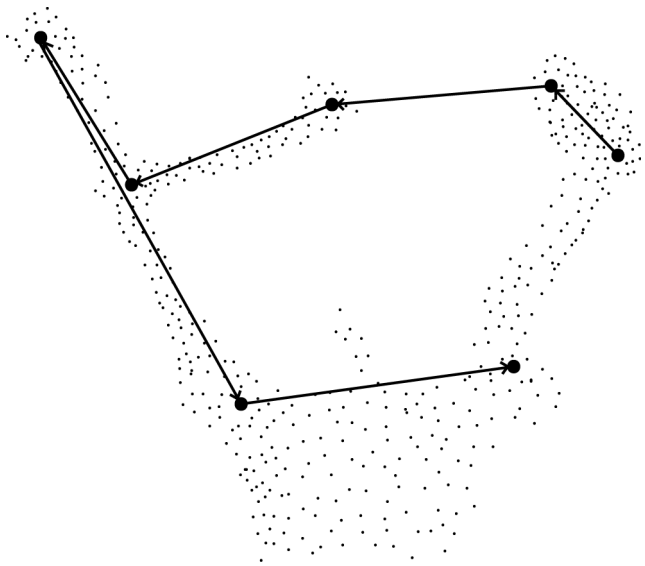

Figure 4.11: Example of a cloud being swept after evaluation. The arm goes to the rightmost point and continues to the consecutive closest points.

Algorithm 5 Interaction procedure pseudo-code.

Input: Cloud of interaction points C_{int}
Output: Information on traversability (0 not traversable, 1 traversable)
Data: Furthest point from the arm F, Rightmost point from the arm R

Find furthest point from the arm, F, in cloud C_{int}

Find rightmost point from the arm, R, in cloud C_{int}

Move arm to R at depth F

for each P in C_{int} **do**

  $C_{int} \leftarrow C_{int} \setminus P$
  Find closest point to actual position, P' (See Figure 4.11)
  Move arm to P' at depth F

  if stuck **then**

    return 0

  end if

end for

return 1

4.5 Environment Change Detection

When the system decides that the object is traversable, it proceeds to try to go through it. A method of understanding when the environment it is crossing changed to something different, or when the object was successfully traversed, is important to change the behavior or the robot. This must be computed quickly because it has to be evaluated while the robot is in motion, and a long processing time could put the robot in danger.

Different environments have different appearances, and the metric that was used in this model to identify an environment change was the shift of density. By applying a cloud voxelization and comparing to the original number of points in the cloud, it is possible to calculate the factor reduction in the process and, therefore, a metric of cloud density is created. Being S_{cl} the size of the cloud and S_{vox} the size of the cloud after voxelization, P_ρ, the factor of the cloud reduction can be found by using Equation 4.22.

$$P_\rho = \frac{S_{vox}}{S_{cl}} \tag{4.22}$$

Before entering the new environment, a cloud is captured at close range by the control system, O, and its density reduction, Ps_ρ, is calculated. This number is stored in memory for future reference. While the robot traverses the terrain, it continuously captures new clouds, O' while P_ρ is calculated and compared to the reference value. If the system detects a big change in P_ρ, by calculating δ_ρ, it assumes that the environment has changed and it needs to evaluate the new situation. The change threshold, α_ρ, is used as a stopping condition for the environment change algorithm, and shown in Algorithm 6.

$$\delta_\rho = |Ps_\rho - P_\rho| \tag{4.23}$$

Algorithm 6 Environment change detection pseudo-code.

Input: α_ρ, Voxel size (v_x, v_y, v_z)
Output: None (stopping condition)
Data: P_ρ, δ_ρ

Capture cloud at the entrance of the object, O

$O_v \leftarrow$ Compute voxelization of cloud O

Calculate cloud reduction reference Ps_ρ, with cloud O_v and O [Equation 4.22]

Move robot forward

while $\delta_\rho < \alpha_\rho$ **do**

    Capture cloud, O'
    $O'_v \leftarrow$ Compute voxelization of cloud O'
    Compute P_ρ with cloud O'_v and O' [Equation 4.22]
    Calculate δ_ρ [Equation 4.23]

end while

return Stop

Chapter 5

Prototype

This chapter presents the prototype development to test the proposed model, in both software and hardware perspectives. The control system model was implemented in the C++ programming language and it was integrated with the Robotics Operating System (ROS) (Quigley et al., 2009). The controller was tested in an Intel Core i5-2140M CPU @ 2.30GHz, 4GB of RAM running a 64-bit Linux distribution Ubuntu 12.04 and using Point Cloud Library (PCL) (Rusu and Cousins, 2011) for cloud operations. Hardware wise, a telescopic antenna was created for object interaction, a Microsoft Kinect sensor was used as a sensor and a heavily modified Fast Lane Wild Fire RC Monster Truck was used as a mobile platform. For direct actuator control, two Microchip microcontrollers (PIC18F4550) were used, both connected by USB to the control system. The firmware for the microcontrollers was written in C programming language using Mikro C IDE.

5.1 Prototype Overview

As explained in the previous chapter, the robot model has three major components; a robotic arm, a sensor capable of producing point clouds and a mean of locomotion. Taking into account these factor, a robot was built to test the system portraited in this thesis. The prototype is based on a $45\,\text{cm} \times 35\,\text{cm} \times 65\,\text{cm}$ 4-wheeled robot with differential locomotion, fitted with a custom telescopic antenna with pan tilt control. The antenna is capable of stretching up to 1 m and its pan and tilt cover 180 in both axis, respectively. As depth sensor, the robot uses a Microsoft Kinect, which employs modulated light to capture tridimensional point clouds of the environment. It is applicable robustly outdoors during the night and at most in the presence of dim daylight. For daylight operation the robot would have to be equipped, for instance, with a binocular vision sensor. As the noise model of these two sensory modalities is rather similar, the proposed model should be easily applicable to binocular vision and, as a result, enable daytime outdoors operation. The system is implemented on the top of the Robotics Operating System (ROS) Quigley et al. (2009) and relies on the Point Cloud Library (PCL) Rusu and Cousins (2011) for low-level point clouds processing. Figures 5.1 and 5.2 show the front and side view of the prototype robot.

For communications and control, two microcontrollers (PIC18F4550) were used, along with a

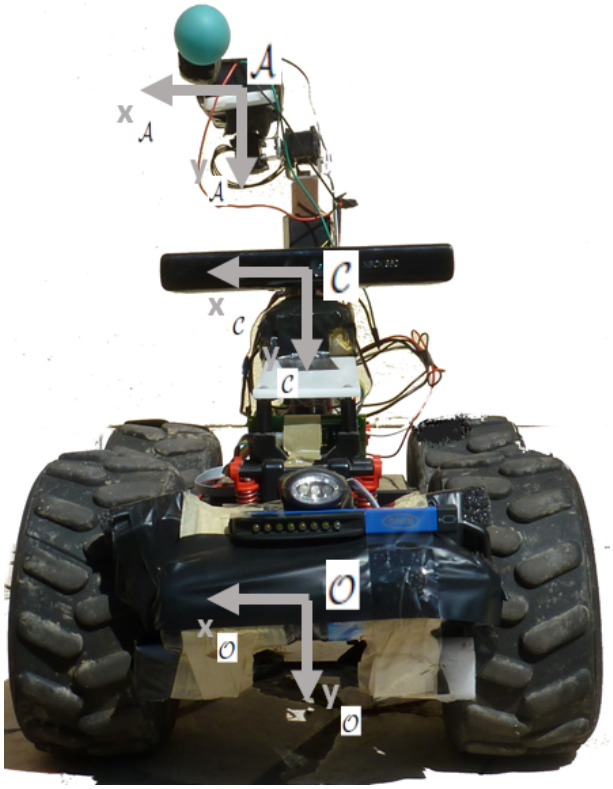

Figure 5.1: Front prototype view with the associated frames of reference.

Figure 5.2: Side prototype view with the associated frames of reference.

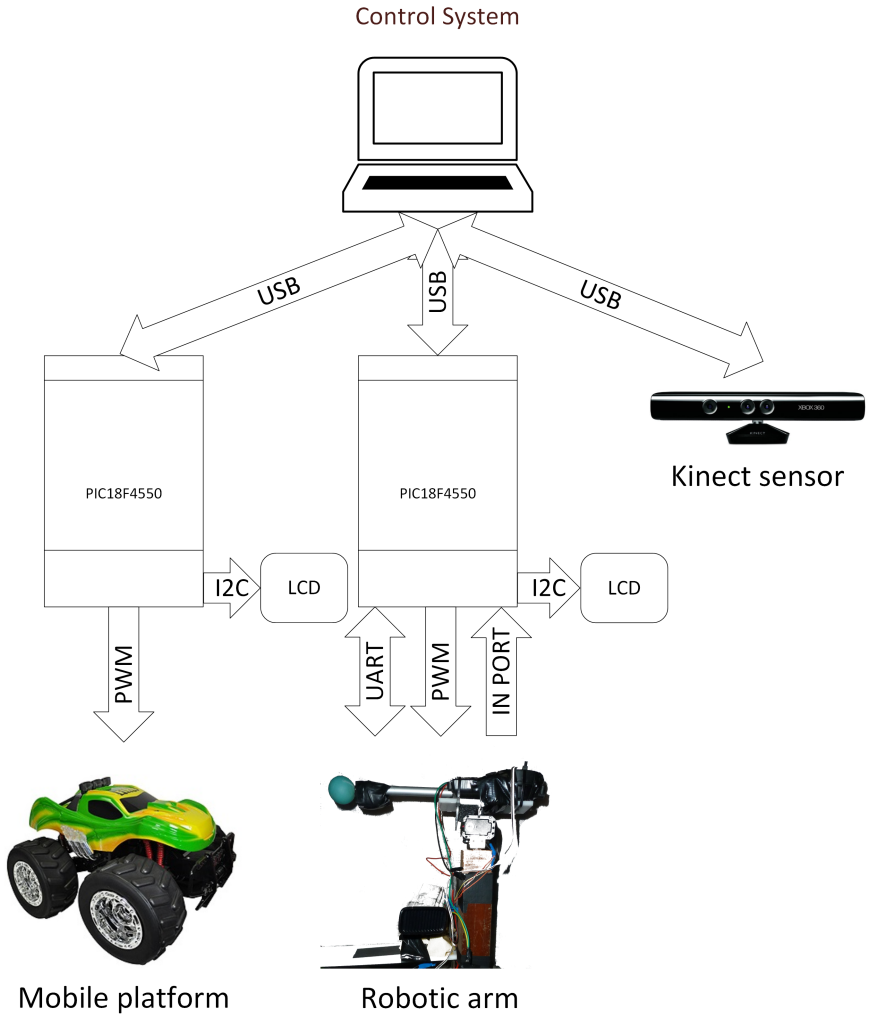

Figure 5.3: Layers and communications of the experimental setup.

computer to control the system. Figure 5.3 shows the architecture of the prototype system, as well as the communications between different components, that will be analyzed in later sections of this document.

5.2 Telescopic Antenna

As explained in the previous chapter, the hardware must be capable of interacting with the environment surrounding it. To do so, a linear actuator that extends a telescopic probe operating as an extensible arm was used to provide a simple and straightforward mean of interaction. This created a simple method for retraction and increased range of interaction. The telescopic probe is powered by 12V DC tension provided by the microcontroller card, and by changing the Pulse-width modulation (PWM) is possible to control the extention or retraction of the probe. The probe controller consists in an electric motor and a control signal. The electric motor is powered when moving the probe, and the control signal changes the direction; when a signal is provided the probe rises and when no signal is provided the probe retracts, to stop at the desired position, the electric motor power is cut off.

Figure 5.4: Detail of the antenna tip used to be able to be tracked by the sensor.

To control how much the probe is stretched a tachometer was created with a phototransistor and an encoder. An Omron EE-SF5/SF5-B phototransistor was used and a 11.25 degree rate encoder, allowing 16 transitions per rotation. The encoder was placed on the gear that makes the probe rise or retract, and with the phototransistor the microcontroller detects the transitions and therefor movement. Each transition of the encoder translates into a 0.01 m movement in the stretching of the probe. If the probe gets stuck while rising it is detected by the microcontroller so it always has an accurate position of the arm end effector. The phototransistor was mounted according to the specification in the datasheet and the resistors used are shown in Figure 5.5, as well as the encoder used. Figure 5.6 shows the prototype robot assembly. This allows the arm to stretch to a maximum of 1 m.

For purposes of calibration, the probe end effector had to be adapted with a recognizable shape in order to apply the RANSAC algorithm. A small rubber ball with approximately 0.03 m, r_{int}, of radius was attached to the tip of the antenna and it proved sufficient to turn the end effector into something recognizable by the sensor. The ball also allows for a bigger radius of interaction, being much more effective than a simple antenna tip as explained in the previous chapter. Figure 5.4 shows a close plan of the rubber ball used.

The robot's arm axis consists of two rotary joints, and for that two Dynamixel MX-28 actuators were used in a custom configuration and the communication with the microcontroller was made by UART. These actuators were chosen because of their small size, provided enough torque for the weight of the probe and size of the robot, the implemented PID control system for stabilization and, more importantly, they are quite accurate and provide reliable feedback about how much torque they are generating and the direction of it. This is important in order to determine if they are stuck, and for determining if they are interacting with an object that it potentially and obstacle or not. The actuators were connected in a way to transform the robotic arm into a Stanford arm, meaning that the arms forward kinematics is simply the spherical coordinate system. The stretch of the probe corresponds to the radius of the spherical coordinate system and both actuators correspond to the θ_1 and θ_2 (See Figure 5.7). This allowed for a faster and simpler implementation, without sacrificing the interaction capabilities for this kind of robot. The robot's arm frame of reference $\mathcal{A}$ (see Section 4.1) was placed at the beginning of the antenna, and through the direct kinematic and inverse kinematic equations the robot is able to control the end effector position. The actuators and the probe were placed on the

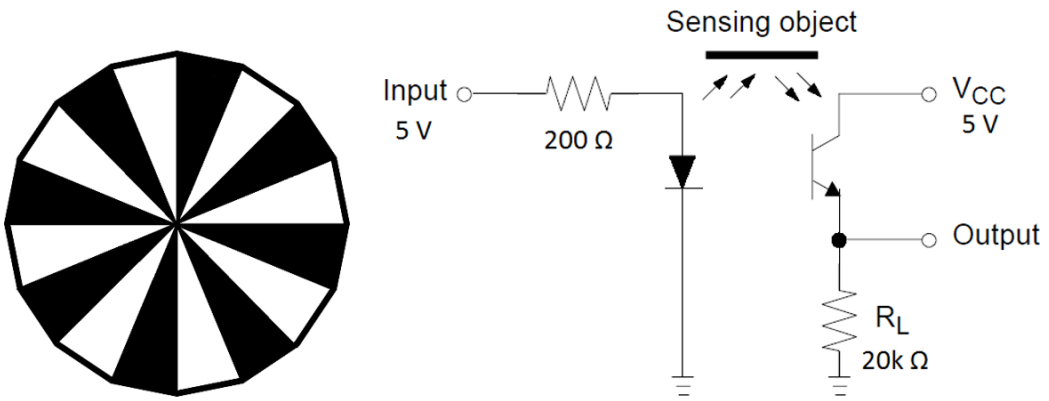

Figure 5.5: Installation schematics of the phototransistor and encoder. On the left, the 16 transition encoder used is represented while on the right the installation schematics are shown.

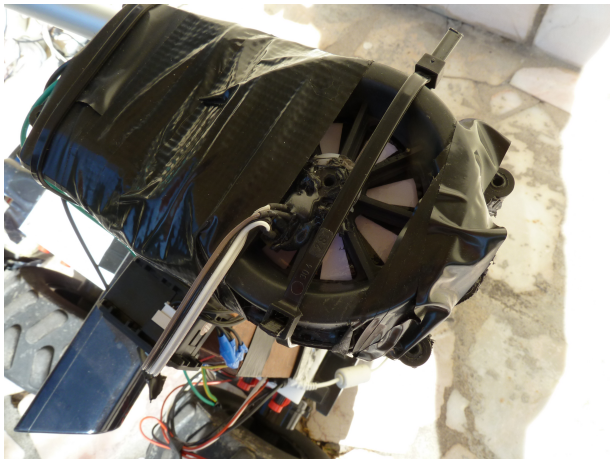

Figure 5.6: Detail of the phototransistor and the encoder setup in the prototype system.

end of a pole, so it did not interfere with the cloud sensor. The robotic arm provides an interaction radius of approximately 1 m, and 180 degrees of freedom in both axis.

5.3 Depth Sensor

A Microsoft Kinect sensor was used for the acquisition of the point cloud. This sensor was chosen mostly because it is very inexpensive, easy to obtain, light weight and has big support from the community. The Kinect sensor has been used for vegetation characterization, as shown by Azzari et al. (2013), so it is known to be capable of capturing the information needed for this system. Because of the characteristics of the Kinect sensor, the robot does not function under direct sun light. Other limitation introduced by Kinect is the inability to detect points closer than 0.5 m from the sensor (Khoshelham, 2011). Because of this, the sensor had to be put further from the front of the vehicle in order to correctly capture the cloud. The Kinect is also equipped with a 640 x 480 pixel VGA camera but it was not used in this implementation, only for demonstration purposes. The sensor was placed at a height where the sensor range was not affected by the mobile platform structure.

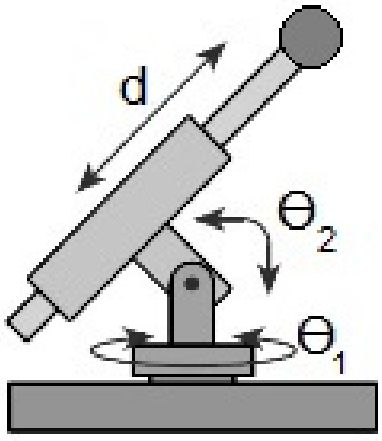

Figure 5.7: Robotic arm coordinate schematic. As seen in the figure, the arm's kinematics can be approached by the spherical coordinate system.

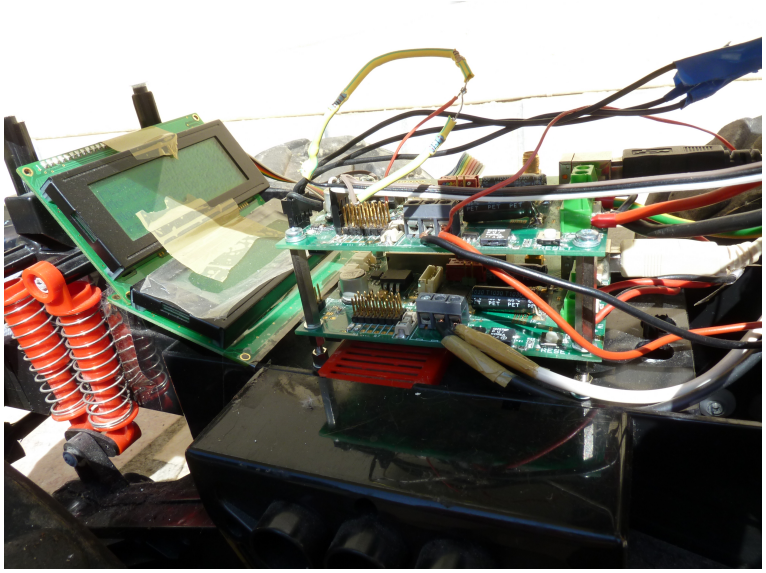

Figure 5.8: Microcontrollers location and connections.

5.4 Mobile platform

For mobility, a Fast Lane Wild Fire RC Monster Truck was adapted in order to accommodate the sensor and the telescopic antenna. By connecting the DC motor directly to the microcontroller, it is possible to directly control the speed and direction generated by them, by providing the necessary PWM signal. The platform was big enough to fit the sensors and the telescopic antenna, but heavy modifications were needed to increase the sturdiness and adequately house the microcontrollers. The car microcontroller receives speed and direction from the software and applies the necessary signal to the DC motors. To change direction, a differential steering method is applied, allowing the car to rotate on its center. Figure 5.8 shows in detail the microcontrollers assembled to the mobile platform frame while Figure 5.9 shows the prototype interacting with the environment.

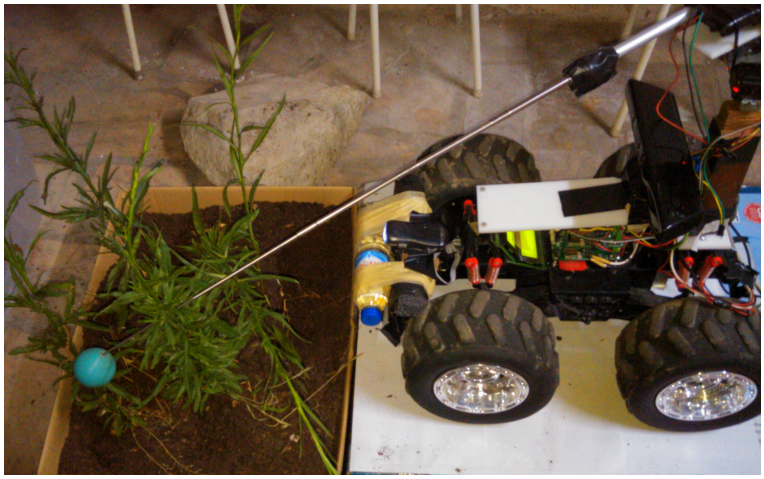

Figure 5.9: The robot prototype with its arm stretched to its maximum range.

Chapter 6

Experimental results

This chapter presents the model parametrization and testing of the proposed model. The prototype used is described in Chapter 5. The robot was assembled and the various parameters implemented from the model explained in Chapter 4 are defined in Section 6.1. Later, the experimental results obtained in controlled and field tests are presented in 6.2.

6.1 Model Parametrization

In this section the model parametrization for the field tests is shown. It is divided into 4 chapters, one for each of the main components of the model.

6.1.1 Calibration Parameters

As described in Chapter 4, the first step in the system is to find the relation between the sensor and the robot arm. To do so, a manual set of points were pre-configured, based on the position relation between the arm and the sensor, in order to estimate the 3D rigid transform matrix, M. The $\mathcal{A}$ referential at the tip of the antenna, A, and the sensor z axis is aligned with it, so the chosen points were picked to provide variation in position in every axis of the sensor frame of reference, $\mathcal{C}$, so the estimation has better results. Five points were picked, n_p = 5 (see Table 6.1). A time lapse image of the procedure is shown in Figure 6.1. The ball is positioned in front of the sensor in order to be tracked.

In the implemented model, it is possible to estimate the maximum depth the arm end effector can be found at, and since both frame of references are aligned in the z axis, the Cl_{bkg} and Cl_n are filtered in the z axis to eliminate any points further than the maximum reach of the arm, which is about 1 meter in order to provide faster results. After some testing, the octree point cloud change detector had more consistent results when initialized at a resolution of 0.08 because the of the resolution of the sensor used, and this created the cloud Cl_f.

Point	$X_{\mathcal{A}}$	$Y_{\mathcal{A}}$	$Z_{\mathcal{A}}$
1	-0.3	0.1	0.6
2	-0.3	0.2	0.6
3	0	0	0.8
4	0.1	0.1	0.8
5	-0.2	0.1	0.8

Table 6.1: Coordinates used for calibration, from the arm reference axis $\mathcal{A}$.

Figure 6.1: Time lapse image of the calibration procedure. The used antenna coordinates are shown in Table 6.1.

The small rubber ball at the tip of the antenna was put to be able to be detected by the system. To estimate the Cl_f cloud point normal, a tree search method in a K search of 50 was used, after it was proven it provided consistent results after experimentation. These cloud normals were used with the RANSAC model of a sphere, with a maximum of 1000 iterations, 0.03 m of distance threshold and a radius limit between 0.01 m and 0.07 m. With these settings the ball is successfully found even when varying the background and sensor orientation. To estimate the arm end effector position, the centroid of the found spherical model is calculated and it is stored in the appropriate vector for later compute the M matrix.

6.1.2 Object Evaluation Parameters

The system has an object storing and comparing methodology which needs to be initialized (see Section 4.4.1). For this part of the system, the voxel size used was $v_x = v_y = v_z = 0.01$. For the histogram **H**, after some testing and taking into account the resolution of the sensor, a grid of $x_{\mathbf{H}} = 16$ by $y_{\mathbf{H}} = 14$ was used. This provides a surface area small enough to be able to determine the difference and similarities between two clouds, but at the same time big enough to deal with height shifts in the capture point and still provide fast processing.

The next parameter on the memory evaluation procedure is the score generation from comparing the clusters from the histograms. $N^j_{\mathbf{H}}$(**H'**), score generated by the number of adjacent bins occupied in line j, separated from each other by an empty bin in histogram **H**, $N^j_{\mathbf{H}}$ has for υ, a value dependent on the robot model, equal to 2. This value was a result of experimentation and is related to the sensor and range of operation used. $W^j_{\mathbf{H}}$(**H'**), the score generated by the comparing the size (in number of bins) of the largest cluster found in a line j, ω, a value dependent on the robot model required for the $W^j_{\mathbf{H}}$ score, is this implementation of this model is 5 (see Section 4.4.1). The scoring generated by density of points per bin $\rho^j_{\mathbf{H}}$ comparison, $\rho^j_{\mathbf{H}}$(**H'**); and the scoring generated by the number of maximum points in a cluster, $P^j_{\mathbf{H}}$, comparison in each line, $P^j_{\mathbf{H}}$(**H'**) were adapted to the experimental conditions. After some experiments, by applying a 3 tier evaluation the desired results were obtained. Figure 6.2 shows the scores attributed for the factor of difference of the same parameter between two objects.

The similarity of each line (see Equation 4.8), provides the final score with some parameters, and the sum of these parameters must be equal to 1 (see Equation 4.9). For the implementation, an equal weight was given for parameters α_S, β_S, δ_S and γ_S, meaning that all have a weight of 0.25. The number of neighbors used for this system was $n = 5$.

After the memory evaluation, the system proceeds to the interaction point evaluation, and that is the next point part to be parametrized. The parameters for this section were chosen based on the robot platform characteristics. For the calculation of the interaction points, a bigger voxel size was used, i.e., $v_x = v_y = v_z = 0.04$. For the height evaluation $C_h(P)$ (see Equation 4.11), the height off-

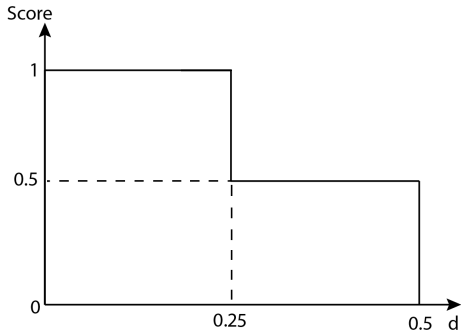

Figure 6.2: Scoring system based on 3 tiers. In this system, Score represents either the $\rho^j_{\mathbf{H}}(\mathbf{H'})$ or $P^j_{\mathbf{H}}(\mathbf{H'})$ value. The d can be either Δ_ρ or Δ_P, respectively. If the ratio is lower than 0.25, the score attributed is one, if it is between 0.25 and 0.5 the score is 0.5, otherwise the score is 0.

set, μ = 0, and σ = 0.8 was chosen, which is the same σ used for the distance to centroid evaluation, $C_d(P)$, in Equation 4.12. According to the voxel size, the τ value to find the minimum neighbor radius, r_n, was 0.03 m, meaning that the r_n chosen was 0.1 m. These values are closely related to the robots dimensions and arm characteristics and were obtained experimentally. As shown in Section 5.2, the r_{int} value used was 0.05 m.

For the total interaction score, the weight of each parameter had to be taken into account. Respecting Equation 4.17, the α_C value, weight of parameter $C_h(P)$, used was 0.3. β_C, weight of parameter $C_d(P)$, was set to 0.2. The final value, γ_C, weight of parameter $C_\rho(P)$, was defined as 0.5. These values can be tuned depending on the environment.

6.1.3 Interaction and Environment Crossing Parameters

The only value required to provide for the interaction generation points is the confidence in the memory factor α (see Equation 4.21). This is an important value which allows to change the dynamics of the system, so three values for α were tested; $\alpha = 0.5$; $\alpha = 1$ and $\alpha = 1.5$. This will show how fast the system becomes on interacting, or how careful it is.

For the environment crossing section, the voxel size used is $v_x = v_y = v_z = 0.01$, and the change threshold, α_ρ, is 0.03. Tests showed that this value provided enough robustness to noise in the sensor while it is still able to detect changes in the surroundings.

6.2 Test results

In this section, the tests performed with the system are explained and the results are discussed. Three tests were made with the system, being (1) classification accuracy from haptic interactions, (2) classification accuracy from learning and (3) environment change detection. All tests were made with the parametrization shown in Section 6.1, and the calibration procedure was omitted because it

Figure 6.3: Objects used for classification accuracy analysis in a controlled environment, as seen from the robot with its depth sensor (data set 1). The yellow rectangles represent the objects' bounding boxes. (a) Wall (non-traversable); (b) Rock (non-traversable); (c) Big plant (non-traversable); (d) Shrub (traversable); (e) Small shrub (traversable); (f) Tall plants (traversable); (g) Tall Plant (traversable); (h) Vertical logs (non-traversable); (i) Horizontal logs (non-traversable).

always produces the same results, because of the constant physical model.

6.2.1 Classification Accuracy from Haptic Interactions

This test was performed in order to evaluate the quality of the interactions provided by the system. The robot was put in a controlled environment where it would move forward until an object is found. After that, the interaction points are computed, interacted with and the object is labeled about being traversable or not. By using different thresholds for interaction, it is possible to judge if the quality of points it generates is enough to accurately learn about its traversability. A higher threshold provides fewer points of interaction, while a smaller threshold provides more points. A set of nine objects was used for this experiment, four traversable and five obstacles. The objects used are shown in Figure 6.3, which were captured before the start of the interaction. These objects are representative of objects that might be found in an outdoor environment.

Figure 6.4 shows the clouds captured by the robot and evaluated. All points created by evaluation are shown, and at this point no filtering to those points was done. In case (c) the cloud is corrupted because of the Kinect characteristics; the object got too close to the sensor and it was not able to capture those points, but the interaction was still successful. This shows the robot is robust for unpredictable input problems.

After the evaluation, the system proceeded to interact with the object. On the first run, it was told to interact with every interaction point generated, and on the second run a threshold was given. On the first test the system successfully identified the obstacles and traversable objects. For the second

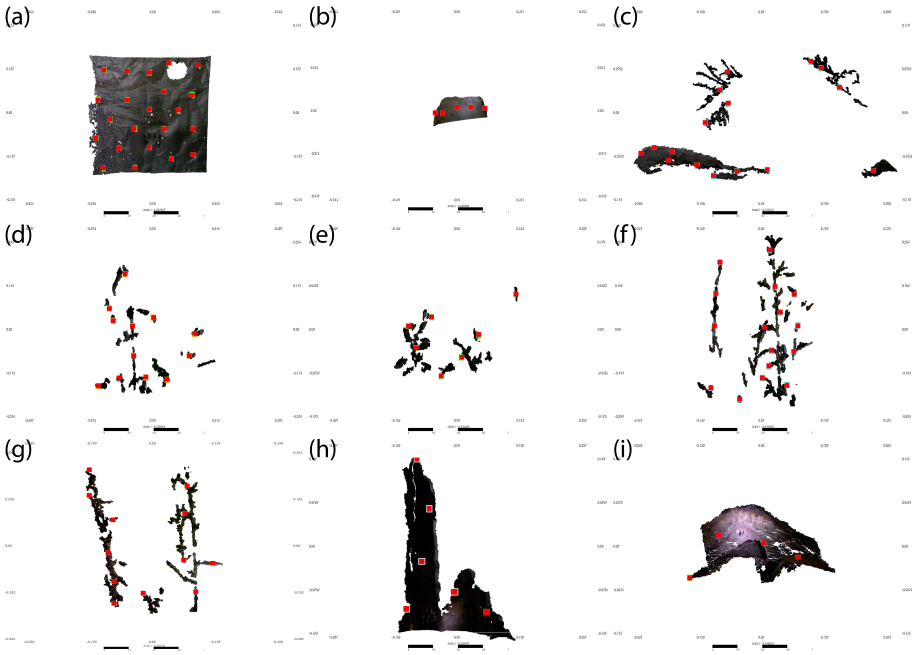

Figure 6.4: Object point clouds captured. The red points are interaction points suggested by the system.

Score	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)
$[0.9, 1.0]$	3	0	0	0	0	0	0	0	0
$[0.7, 0.9[$	5	3	2	1	3	2	2	4	2
$[0.5, 0.7[$	11	2	14	11	4	14	10	2	2

Table 6.2: Number of points selected for haptic interaction within a given score interval.

test, a lower number of interaction points was used. In Table 6.2 is shown that every object had at least one point where the score was higher than 0.7, so on the second test the system interacted only with points that scored higher that 0.7 and still identified every object correctly. This shows this system is capable of generating an adequate interaction methodology for unknown objects even when using a smaller number of points. Figure 6.5 depicts a typical haptic interaction. Figure 6.6 shows the example of two objects with different scores for interaction.

6.2.2 Classification Accuracy from Learning

To test the dynamics of the memory evaluation system, in conjunction of the previous test objects, two more objects were used with different appearances in order to test the system's ability to generalize and learn from the experience. The robot approached these objects from various angles and clouds were captured to generate the scoring results for later comparisons. Figures 6.7 and 6.8 show the objects tested (data set 2) as an image and with a frequency histogram example. After experimentation, to the first figure the label traversable was applied while to the second one it was considered not traversable. This is important to evaluate the memory evaluation dynamics. Three tests were performed to evaluate the classification accuracy. On the first test (see Section 6.2.2.1), the goal was to evaluate the system's ability to recognize previous seen objects, while on the second test (see Section 6.2.2.2) the main focus was to verify the system's ability to generalize what it has learned from previous experiences. Finally, on the third test (see Section 6.2.2.3), the speed and the caution of the system was tested by adjusting the α factor.

Figure 6.5: Typical haptic interaction execution. The arm stretches to hit the first interaction point and then follows the plan by adjusting the stretch level accordingly. Note that this behaviours results in a scanning pattern that bends traversable obstacles.

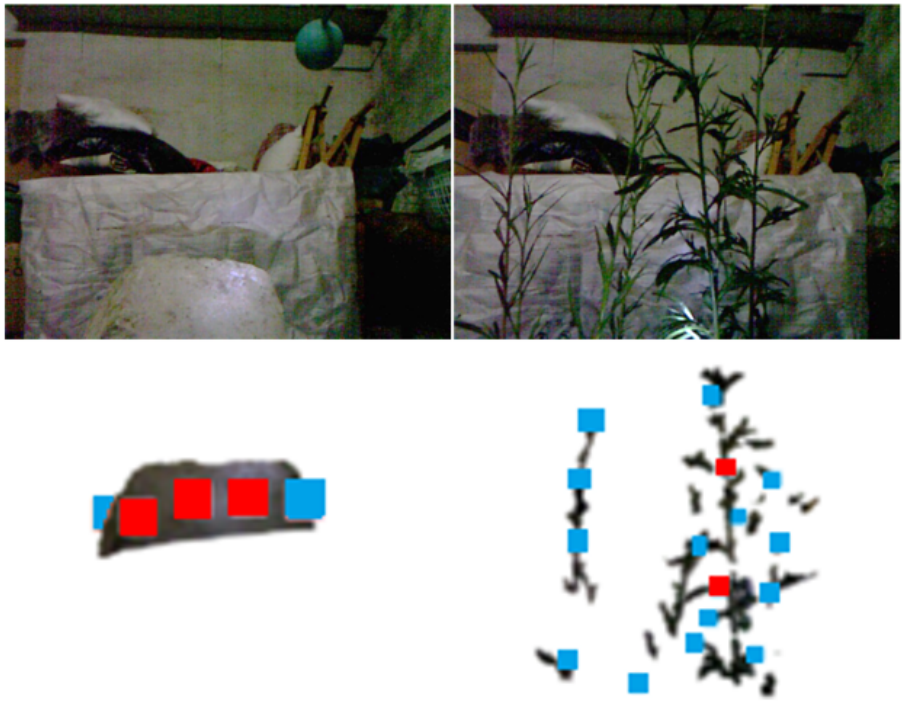

Figure 6.6: Different interaction points suggested by the system for two object. The blue points represent points that scores lower that 0.7 while red points represent points that scored over 0.7.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	7	0	0	2	0	0	0	0	0	0	0	0	0
0	0	0	26	8	0	26	0	12	0	0	0	0	0	0	0
0	0	0	0	0	18	26	0	8	10	0	0	0	0	0	0
0	0	0	0	0	12	40	2	0	10	9	0	0	0	0	0
0	0	0	0	0	0	28	10	0	0	12	5	0	0	0	0
0	0	0	0	0	0	0	38	0	0	3	45	5	0	0	0
0	2	19	3	0	0	0	24	8	0	0	23	18	1	0	0
0	0	0	18	7	0	0	21	16	2	0	14	2	17	0	0
0	0	0	0	14	7	0	16	29	13	0	17	5	5	11	0
0	0	0	0	0	15	9	1	27	19	7	22	7	0	0	0
0	0	0	0	0	0	13	9	42	22	22	16	22	0	0	0
0	0	0	0	0	0	0	3	6	18	18	19	29	0	0	0

Figure 6.7: First object used for the memory evaluation test (1). On the left, a image of the object is shown and on the right an example frequency histogram is presented. By observing the histogram, it can be seen that the histogram has many empty bins and the frequency of points is often low.

0	0	1	3	0	0	0	0	0	0	0	0	0	0	0	0
0	0	7	13	0	0	0	0	0	0	0	0	0	0	0	0
0	0	35	34	6	8	3	0	0	0	0	0	0	0	0	0
0	0	60	58	0	34	11	13	0	0	0	0	0	0	0	0
0	0	21	10	24	55	30	7	0	0	0	6	0	0	0	0
0	0	0	13	19	35	85	45	32	0	16	54	5	0	0	0
0	0	8	30	19	49	60	21	4	0	36	71	11	0	0	0
0	0	16	11	33	50	49	7	38	17	41	29	1	0	0	0
0	0	59	16	48	25	38	61	42	26	14	46	12	5	0	0
0	0	0	1	26	57	44	61	65	29	23	2	5	0	0	13
0	0	21	37	49	56	82	65	40	33	24	28	38	0	0	9
0	0	14	0	35	44	60	32	69	22	0	42	38	13	0	10
0	0	0	0	0	4	4	4	7	9	0	4	1	0	0	24
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6.8: Second object used for the memory evaluation test (2). On the left, a image of the object is shown and on the right an example frequency histogram is presented. By observing the histogram, on this second object the empty bins in the middle of the object are few. The point frequency is usually higher than on object 1.

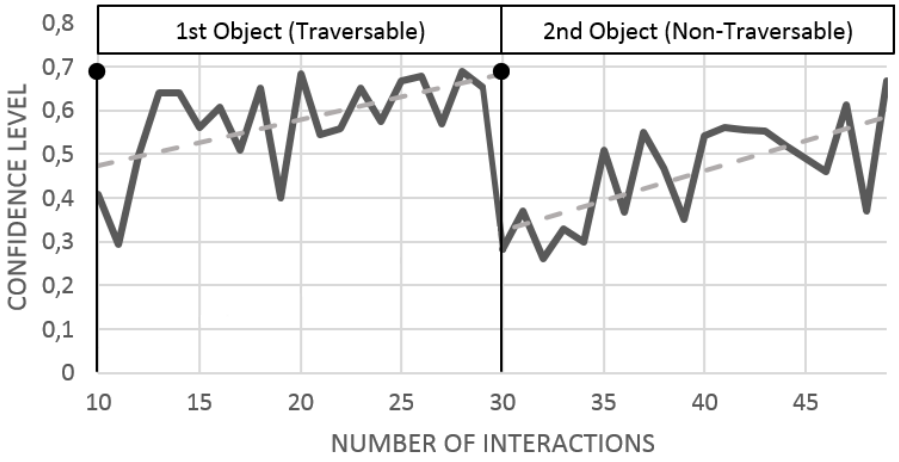

Figure 6.9: Classification confidence when progressively incorporating two new objects into memory. The two interrupted lines represent the linear regression for the confidence level before and after meeting the second object.

6.2.2.1 Object Recognition

The goal of this test is to detect a new object encounter, and learn by repeating the encounter. For this test, the robot approached each object twenty times from different angles. The database was initialized with the data set 1 cloud (see Figure 6.3). The first step for the object recognition is the frequency histogram creation and comparison.

Figure 6.9 shows the confidence level of the database knowledge on the new object after each encounter. The confidence level slowly rises and stabilizes after the 1st object is encountered a few times. The system reaches a confidence level of about 60% and after encountering the 2nd object, at the 30th iteration, it looses the confidence level and, again, slowly rises until it reaches the same value. This shows the methods of object identification developed are capable of recognizing the same object, and recognizing when it finds something new.

To verify if the confidence of the database increases while the samples increase, the same clouds were incrementally introduced to the memory, but this time in a random order. As seen in Figure 6.10, the trend line from the confidence increases over time, suggesting that as more encounters are made, less interactions with the environment will occur.

6.2.2.2 Object Generalization

After checking the learning dynamics for known and new objects, the capabilities of generalization were tested. This test aims to see if from different clouds in the database, the system can make an educated decision of what a new unknown object might be, obstacle or not. To assess the robustness of the object descriptor (see Section 4.4.1.1) and the memory recalling process (see Section 4.4.1.2), a leave-one-out cross-validation analysis was undertaken based on the 9 objects. The principle used is to leave one of the objects out of the training set and then classify it based on the training set, which has been hand-labelled. As depicted in Fig. 6.11, the system produced a correct traversable/non-traversable classification $67\,\%$ of the times for $k = 1$ and $78\,\%$ of the times for $k = 3$. This an

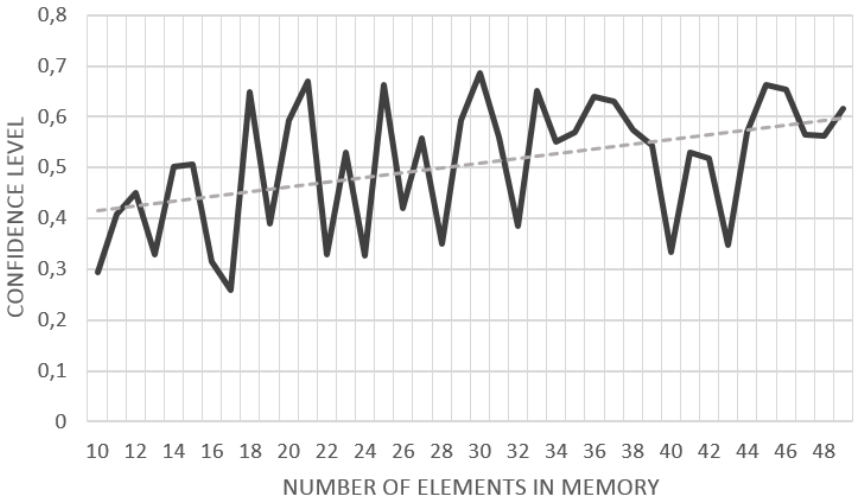

Figure 6.10: Results on the random introduction to memory test. The dotted trend line shows the system's confidence increases over time.

k = 1		Predicted	
		Traversable	Non-Traversable
Actual	Traversable	2	2
	Non-Traversable	1	4

k = 3		Predicted	
		Traversable	Non-Traversable
Actual	Traversable	3	1
	Non-Traversable	1	4

Figure 6.11: Confusion matrix obtained from leave-one-out cross-validation.

interesting result given the lack of redundancy present in the data set. That is, for $k = 3$, the system recognizes the objects based on their intra- and inter-class resemblance.

To further test the object generalization, the data set 2 was loaded into the database, and every cloud from data set 1 was put through cross-validation. Table 6.3 shows the obtained results after this testing procedure. This test was also used to test the blending of data between vegetation and non vegetation objects. After analyzing the results, the system made the correct guess with a good confidence level in clouds containing similar objects to the ones in objects 1 and 2. Clouds containing vegetation ((c), (d), (f) and (g)) got the correct label attributed, and specially the latter ones got good confidence levels (0.48, 0.49 and 0.60). A visual and empiric test has the same results, since they appear similar to the human eye. The first cloud (a) also got a good result. The density was very similar to the object 2 representing an obstacle, so it also obtained a good result. The other clouds had a very little number of points, so the system had difficulties to generalize the result. A larger data set could help to obtain better confidence levels and results, but similar clouds were correctly identified.

Object	Traversable	Confidence Level	System Guess
(a)	No	0.42	Not Traversable
(b)	No	0.37	Traversable
(c)	No	0.33	Not Traversable
(d)	Yes	0.65	Traversable
(e)	Yes	0.48	Traversable
(f)	Yes	0.49	Traversable
(g)	Yes	0.60	Traversable
(h)	No	0.44	Traversable
(i)	No	0.23	Traversable

Table 6.3: Classification of objects in data set 1 given knowledge about objects in data set 2 with $k = 5$. Mis-classified objects: (b), (h), and (i).

Database position	Confidence Level	System Guess
6	0.38	Traversable
10	0.54	Traversable
13	0.57	Traversable
16	0.64	Traversable
18	0.67	Traversable

Table 6.4: Confidence levels and systems guess between same object encounters.

6.2.2.3 Impact of alpha on the system interaction and speed

As seen in Section 4.4.3, the α factor has a direct impact on the number of interactions and speed of the system. For α values lower than 1, the system takes a more careful approach while for values higher than 1, the system has higher confidence on what it has learned and takes less time in a possible haptic evaluation of an object. For this test, only one object was used (1) and all twenty clouds were loaded into the database. After each cloud was loaded, the confidence level of the new cloud was computed as well as the system guess about its traversability. Table 6.4 shows the results for five clouds randomly picked. With these results, three α values were chosen to determine the impact on the interaction methodology of the system. Figure 6.12 shows the results of this test.

As seen in Figure 6.12, the higher the α value, the more the system trusts on the database, rather than on the interaction points quality. The α value of 0.5 is quite cautious and it continuously interacts with the object, even when it has a good confidence level in the database. For α value of 1.5, a little confidence is enough to trust the database. For clouds 13, 16 and 18 for α value of 1.5 the system decided not to interact and it would just trust the database, and as seen in Table 6.4, the system would try to traverse the object.

6.2.3 Environment Change Detection

To determine the system's ability to traverse obstacles, three different objects were approached with different characteristics. Adding to that, on the first object two different situations were tested. On the first situation, the robot meets the object and proceeds to traverse it, and opposite to the object there is open space, on the second situation a large dense non traversable obstacle was placed in the middle of the object. First the robot approached the objects in order to interact with them using the arm, and after that evaluation it proceeded to traverse them. All the interactions with the arm

Figure 6.12: Impact of different α on the number of interactions.

Ps_ρ	P_ρ	$\|\delta_\rho\|$	α_ρ	Stop
0.055	0.055	0	0.03	No
	0.036	0.019		No
	0.042	0.013		No
	0.041	0.014		No
	0	0.055		Yes

Table 6.5: Results for traversing object (a) to open environment.

resulted in a traversable evaluation, so that aspect is not reported in this test.

Object 1, shown in Figure 6.13 (a), is a small shrub with very thin branches. Figure 6.14 shows the robot proceeding to traverse the obstacle. As seen in Tables 6.5 and 6.6, the system successfully detected changes in the environment either to an open space or against a wall. As seen in the tables, the environment change difference, $|\delta_\rho|$, was kept under the threshold while the object was being traversed, and when it was it cleared, i.e., the robot found open space or a different denser environment, the robot stopped. The object was fragile enough for the robot to traverse completely through it and the results support the validity of the model. With the Object 2, as seen in Figure 6.13 (b), the robot platform did not have enough torque to traverse through the object. The object was denser that Object 1, as can be seen by the values of Ps_ρ, density change first input cloud value, and P_ρ, density change input cloud value. By analyzing Table 6.7 it can be easily detected that the robot is not progressing by seeing repeated values of δ_ρ, environment change difference. This information could be used in future works to detect that the robot is stuck in the environment without consulting other sensors. Finally, with Object 3, a small curtain like leafs attached to flimsy branches shown in Figure 6.13 (c), the robot quickly traversed through it, as can be seen by the small amount of data collected in Table 6.8.

The results of this test show that the environment crossing model is capable of detecting changes in the tested environments. Some attention is required with the sensor, because a small leaf, per example, could easily blind it and lead the system to believe that the environment changed while in fact is just blind.

(a) (b) (c)

Figure 6.13: Traversability test subjects. (a) - Small shrub; (b) - Flimsy canes; (c) - Twigs with thin leafs.

Figure 6.14: Traversability test in action.

Ps_ρ	P_ρ	$\|\delta_\rho\|$	α_ρ	Stop
0.043	0.043	0	0.03	No
	0.032	0.011		No
	0.034	0.009		No
	0.064	0.021		No
	0.074	0.031		Yes

Table 6.6: Results for traversing object (a) to closed environment.

Ps_ρ	P_ρ	$\|\delta_\rho\|$	α_ρ	Stop
	0.015	0		No
	0.035	0.020		No
	0.037	0.022		No
	0.033	0.018		No
	0.023	0.008		No
	0.022	0.007		No
0.015	0.020	0.005	0.03	No
	0.025	0.010		No
	0.026	0.011		No
	0.037	0.022		No
	0.037	0.022		No
	0.037	0.022		No
	0.044	0.029		No

Table 6.7: Results for traversing object (b). Since the robot got stuck, the system continuously tried to detect change in the environment, but it never happened.

Ps_ρ	P_ρ	$\|\delta_\rho\|$	α_ρ	Stop
	0.059	0		No
0.059	0.045	0.014	0.03	No
	0	0.059		Yes

Table 6.8: Results for traversing object (c).

Chapter 7

Conclusions and Future Work

In this chapter, a review of the results and a critical look on the model and the developed work is given. Some directions for future research are also approached.

7.1 Conclusions

A ground vehicle capable of exploiting haptic cues to learn navigation affordances from depth cues was presented and validated on a set of field trials, offering a solution to the problem of identifying traversable objects in natural unstructured environments. For this purpose the system has implemented a calibration procedure to assess the relation between the sensor and the interaction method, allowing for action possibilities to the environment. When encountering an object, the system generates a haptic interaction plan based on previous encounters and the object's geometry analysis. If the outcome of said interaction or the confidence in the memory evaluation results in a traversable evaluation, the system has an environment change detector to recognize when the object was traversed. These evaluations are constantly improving through a self-supervised learning mechanism, where all previous encounters are considered when generating a new haptic interaction plan.

For the haptic interaction, a low-cost pan-tilt telescopic antenna was used, whereas for distal sensory feedback the robot uses a low cost depth sensor. The plan creation is based on two factors, the memory recall system and the object geometry evaluation. For the memory recall, the system implements a comparison method from point cloud frequency histogram characteristics of previous found objects and the new encountered object. On this implementation, the depth descriptors used are based on point density, continuity and cluster size. The interaction point classification system prioritizes the centroid of the object, denser areas of the object and the height that the vehicle can traverse. By combining the two, an interacion methodology is created. If the system concludes that the object is traversable, the environment change detector indicates when the object was traversed by analyzing the point cloud data density shift.

The system was put through several tests, each one validating a different part of the system. To

test the quality of the haptic interaction points, the system was put against nine different objects in order to generate the best points of interaction. Even by providing a score threshold that would still give interaction points to all objects, the system managed to successfully perceive the traversability. A new set of objects was presented to verify the system's ability to recognize an object previously encountered, generalize the object's characteristics to a new object and to verify the memory confidence factor change impact on the system's speed and caution when analyzing an object. The tests results evidenced the confidence rise as the database increased, affecting the interaction speed to every value of the memory confidence factor. Finally, the environment change detector was tested by making the robot cross three different objects, one of which in two situations; open space and against a denser obstacle. In every situation the robot detected the change of environment, or in one case the lack of it (the robot got stuck), suggesting that it would be easy to implement a verification to check if the environment crossing was not successful and correct the previously given label.

The functioning principle of the system is inspired by the affordance theory of James J. Gibson, where he creates a link between the ability of a subject through its actions to the features of the environment, and states that to learn an affordance the agent needs to interact with the environment. The presented results tend to indicate that haptic interaction allows for the robot to learn about traversability and allied to a descriptor system provide the necessary inputs to create a self-supervised learning robot. The implementation of haptic feedback reduces the danger of robot damage because it allows to assess traversability in a controlled and non hazardous manner. The simplicity of the proposed system allows its application in small sized robots, which are useful tools for domains like search & rescue.

7.2 Future Work

The main point of future work would be to further test the proposed model with different hardware and environment configurations. It would further prove the robustness of the system and allow to identify points were improvement is needed. The hardware used to validate the model was sufficient, but a mobile platform with higher torque and autonomy through batteries would allow for more tests to be performed.

Even though the proposed and implemented model successfully accomplished the main goal, every building block of the system can be further improved. A different interaction method not based in binary output could be used to calculate economics of the traversability, and use that information to make a decision based on the system state and the urgency of the mission. This aspect could also be propagated to the labels applied to the objects to give a score that informs more than traversable or not, but into something like traversable but with a high cost.

Regarding to the evaluation methods, a simple addition of a camera to allow for an appearance score based on images would be beneficial for the refinement of the evaluation. Objects may appear structurally similar and even a simple evaluation, like color, could detect differences unseen by point clouds. The Kinect sensor has the ability to capture images but this was not implemented on the system due to the inability of the Kinect to work under strong light sources, and the captured images were too dark to work with. It would also be interesting to compute the skeletal frame of the object

to, based on the point cloud data, detect possible weak spots and interact with them, opposed to the metrics used on this dissertation.

A more challenging but more rewarding point of improvement would be the abolition of the hard coded parameters and create a model where the robot platform auto tunes these values based on the interaction and further self-exploration with the available sensors (as used in the calibration process in this model), to create a totally autonomous and self-aware platform.

7.3 Dissemination

Some of the concepts covered in this dissertation can be additionally viewed in the following publication, co-authored by the author:

• Baleia, J., Santana, P., and Barata, J. (2014). Self-Supervised Learning of Depth-Based Navigation Affordances from Haptic Cues. *Proceedings of the 2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC'2014)*.

Bibliography

Ahissar, E. and Knutsen, P. M. (2008). Object localization with whiskers. *Biological cybernetics*, 98(6):449–458.

Anderson, S. R., Pearson, M. J., Pipe, A., Prescott, T., Dean, P., and Porrill, J. (2010). Adaptive cancelation of self-generated sensory signals in a whisking robot. *Robotics, IEEE Transactions on*, 26(6):1065–1076.

Azzari, G., Goulden, M. L., and Rusu, R. B. (2013). Rapid characterization of vegetation structure with a microsoft kinect sensor. *Sensors*, 13(2):2384–2398.

Bajracharya, M., Howard, A., Matthies, L. H., Tang, B., and Turmon, M. (2009). Autonomous off-road navigation with end-to-end learning for the lagr program. *Journal of Field Robotics*, 26(1):3–25.

Chemero, A. (2003). An outline of a theory of affordances. *Ecological psychology*, 15(2):181–195.

Dahlkamp, H., Kaehler, A., Stavens, D., Thrun, S., and Bradski, G. R. (2006). Self-supervised monocular road detection in desert terrain. In *Robotics: science and systems*, volume 38.

Dang, T. and Hoffmann, C. (2005). Fast object hypotheses generation using 3d position and 3d motion. In *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, pages 56–56. IEEE.

Detry, R., Baseski, E., Popovic, M., Touati, Y., Kruger, N., Kroemer, O., Peters, J., and Piater, J. (2009). Learning object-specific grasp affordance densities. In *IEEE International Conference on Development and Learning (ICDL)*, pages 1–7. IEEE.

Edsinger-Gonzales, A. (2005). Design of a compliant and force sensing hand for a humanoid robot. Technical report, DTIC Document.

Eggert, D. W., Lorusso, A., and Fisher, R. B. (1997). Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications*, 9(5-6):272–290.

Fend, M. (2005). Whisker-based texture discrimination on a mobile robot. In *Advances in Artificial Life*, pages 302–311. Springer.

Ferrando, N., Gosalvez, M., Cerdá, J., Gadea, R., and Sato, K. (2011). Octree-based, gpu implementation of a continuous cellular automaton for the simulation of complex, evolving surfaces. *Computer Physics Communications*, 182(3):628–640.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Franklin, J. and Hiernaux, P. (1991). Estimating foliage and woody biomass in sahelian and sudanian woodlands using a remote sensing model. *International Journal of Remote Sensing*, 12(6):1387–1404.

Gibson, E. J. (2000). Perceptual learning in development: Some basic concepts. *Ecological Psychology*, 12(4):295–302.

Gibson, J. J. (1977). The concept of affordances. *Perceiving, acting, and knowing*, pages 67–82.

Gibson, J. J. (1979). The theory of affordances. pages 127–136.

Grodecki, J. (2001). Ikonos stereo feature extraction-rpc approach. In *Proc. ASPRS Annual Conference, St. Louis*, pages 23–27.

Halme, M. and Tomppo, E. (2001). Improving the accuracy of multisource forest inventory estimates to reducing plot location errorÙa multicriteria approach. *Remote Sensing of Environment*, 78(3):321–327.

Haralick, R. M., Joo, H., Lee, D., Zhuang, S., Vaidya, V. G., and Kim, M. B. (1989). Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1426–1446.

Held, R., Ostrovsky, Y., de Gelder, B., Gandhi, T., Ganesh, S., Mathur, U., and Sinha, P. (2011). The newly sighted fail to match seen with felt. *Nature neuroscience*, 14(5):551–553.

Horton, T. E., Chakraborty, A., and St. Amant, R. (2012). Affordances for robots: a brief survey. *AVANT. Pismo Awangardy Filozoficzno-Naukowej*, (2):70–84.

Jones, K. S. (2003). What is an affordance? *Ecological psychology*, 15(2):107–114.

Jung, D. and Zelinsky, A. (1996). Whisker based mobile robot navigation. In *Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, volume 2, pages 497–504. IEEE.

Khoshelham, K. (2011). Accuracy analysis of kinect depth data. In *ISPRS workshop laser scanning*, volume 38, page 1.

Kim, D. and Möller, R. (2007). Biomimetic whiskers for shape recognition. *Robotics and Autonomous Systems*, 55(3):229–243.

Kim, D., Sun, J., Oh, S. M., Rehg, J. M., and Bobick, A. F. (2006). Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 518–525. IEEE.

Lacey, S., Hall, J., and Sathian, K. (2010). Are surface properties integrated into visuohaptic object representations? *European Journal of Neuroscience*, 31(10):1882–1888.

Lalonde, J.-F., Vandapel, N., Huber, D. F., and Hebert, M. (2006). Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of field robotics*, 23(10):839–861.

Langer, D., Rosenblatt, J., and Hebert, M. (1994). A behavior-based system for off-road navigation. *Robotics and Automation, IEEE Transactions on*, 10(6):776–783.

Lederman, S. and Klatzky, R. (2009). Haptic perception: A tutorial. *Attention, Perception, & Psychophysics*, 71(7):1439–1459.

Lederman, S. J. and Klatzky, R. L. (1987). Hand movements: A window into haptic object recognition. *Cognitive psychology*, 19(3):342–368.

Lefsky, M., Cohen, W., and Spies, T. (2001). An evaluation of alternate remote sensing products for forest inventory, monitoring, and mapping of douglas-fir forests in western oregon. *Canadian Journal of Forest Research*, 31(1):78–87.

Linsen, L. (2001). *Point cloud representation*. Univ., Fak. für Informatik, Bibliothek.

Locke, J. (1700). *An essay concerning human understanding*.

Lu, D. (2006). The potential and challenge of remote sensing-based biomass estimation. *International journal of remote sensing*, 27(7):1297–1328.

Manduchi, R., Castano, A., Talukder, A., and Matthies, L. (2005). Obstacle detection and terrain classification for autonomous off-road navigation. *Autonomous robots*, 18(1):81–102.

Meagher, D. (1982). Geometric modeling using octree encoding. *Computer graphics and image processing*, 19(2):129–147.

Montesano, L., Lopes, M., Bernardino, A., and Santos-Victor, J. (2008). Learning object affordances: from sensory–motor coordination to imitation. *Robotics, IEEE Transactions on*, 24(1):15–26.

Moorthy, I., Miller, J. R., Berni, J. A. J., Zarco-Tejada, P., Hu, B., and Chen, J. (2011). Field characterization of olive (*Olea europaea* l.) tree crown architecture using terrestrial laser scanning data. *Agricultural and Forest Meteorology*, 151(2):204–214.

Norman, D. A. (2002). *The design of everyday things*. Basic books.

Papadakis, P. (2013). Terrain traversability analysis methods for unmanned ground vehicles: A survey. *Engineering Applications of Artificial Intelligence*.

Pfeifer, R., Lungarella, M., and Iida, F. (2007). Self-organization, embodiment, and biologically inspired robotics. *science*, 318(5853):1088–1093.

Phillips, F., Egan, E., and Perry, B. (2009). Perceptual equivalence between vision and touch is complexity dependent. *Acta psychologica*, 132(3):259–266.

Popescu, S. C., Wynne, R. H., and Nelson, R. F. (2003). Measuring individual tree crown diameter with lidar and assessing its influence on estimating forest volume and biomass. *Canadian journal of remote sensing*, 29(5):564–577.

Pratt, G. A. and Williamson, M. M. (1995). Series elastic actuators. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 1, pages 399–406. IEEE.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *Proceedings of the IEEE ICRA Workshop on Open Source Software*, volume 3.

ROSWiki (2014). Ros concepts. *http://wiki.ros.org/ROS/Concepts*.

Russell, R. A. (1992). Using tactile whiskers to measure surface contours. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 1295–1299. IEEE.

Rusu, R. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4.

Sader, S. A., Waide, R. B., Lawrence, W. T., and Joyce, A. T. (1989). Tropical forest biomass and successional age class relationships to a vegetation index derived from landsat tm data. *Remote Sensing of Environment*, 28:143–198.

Şahin, E., Çakmak, M., Doğar, M. R., Uğur, E., and Üçoluk, G. (2007). To afford or not to afford: A new formalization of affordances toward affordance-based robot control. *Adaptive Behavior*, 15(4):447–472.

Santana, P., Guedes, M., Correia, L., and Barata, J. (2011). Stereo-based all-terrain obstacle detection using visual saliency. *Journal of Field Robotics*, 28(2):241–263.

Santana, P., Santos, C., Chaínho, D., Correia, L., and Barata, J. (2010). Predicting affordances from gist. In *From Animals to Animats 11*, pages 325–334. Springer.

Scholz, G. R. and Rahn, C. D. (2004). Profile sensing with an actuated whisker. *Robotics and Automation, IEEE Transactions on*, 20(1):124–127.

Schwenkler, J. (2013). Do things look the way they feel? *Analysis*, 73(1):86–96.

Shin, D., Sardellitti, I., Park, Y.-L., Khatib, O., and Cutkosky, M. (2010). Design and control of a bio-inspired human-friendly robot. *The International Journal of Robotics Research*, 29(5):571–584.

Sinapov, J. and Stoytchev, A. (2008). Detecting the functional similarities between tools using a hierarchical representation of outcomes. In *Development and Learning, 2008. ICDL 2008. 7th IEEE International Conference on*, pages 91–96. IEEE.

Stoffregen, T. A. (2003). Affordances as properties of the animal-environment system. *Ecological Psychology*, 15(2):115–134.

Tanaka, K. (1993). Neuronal mechanisms of object recognition. *Science*.

Toutin, T. and Cheng, P. (2002). Quickbird–a milestone for high-resolution mapping. *Earth Observation Magazine*, 11(4):14–18.

Treuhaft, R. N., Law, B. E., and Asner, G. P. (2004). Forest attributes from radar interferometric structure and its fusion with optical remote sensing. *BioScience*, 54(6):561–571.

Turvey, M. (1992). Affordances and prospective control: An outline of the ontology. *Ecological psychology*, 4(3):173–187.

Ugur, E., Dogar, M. R., Cakmak, M., and Sahin, E. (2007). The learning and use of traversability affordance using range images on a mobile robot. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1721–1726. IEEE.

Uğur, E. and Şahin, E. (2010). Traversability: A case study for learning and perceiving affordances in robots. *Adaptive Behavior*, 18(3-4):258–284.

Vanderborght, B., Albu-Schaeffer, A., Bicchi, A., Burdet, E., Caldwell, D., Carloni, R., Catalano, M., Eiberger, O., Friedl, W., Ganesh, G., et al. (2013). Variable impedance actuators: a review. *Robotics and Autonomous Systems*, 61(12):1601–1614.

Vincent, S. (1913). The tactile hair of the white rat. *Journal of comparative neurology*, 23(1):1–34.

Vincent, S. B. (1915). The white rat and the maze problem: The introduction of a tactual control. *Journal of Animal Behavior*, 5(3):175.

Wellington, C., Courville, A., and Stentz, A. T. (2006). A generative model of terrain for autonomous navigation in vegetation. *The International Journal of Robotics Research*, 25(12):1287–1304.

Wijaya, J. A. and Russell, R. A. (2002). Object exploration using whisker sensors. In *Australasian Conference on Robotics and Automation*.

Wurm, K. M., Kretzschmar, H., Kümmerle, R., Stachniss, C., and Burgard, W. (2012). Identifying vegetation from laser data in structured outdoor environments. *Robotics and Autonomous Systems*.

Zheng, D., Rademacher, J., Chen, J., Crow, T., Bresee, M., Le Moine, J., and Ryu, S.-R. (2004). Estimating aboveground biomass using landsat 7 etm+ data across a managed landscape in northern wisconsin, usa. *Remote Sensing of Environment*, 93(3):402–411.

Zinn, M., Khatib, O., Roth, B., and Salisbury, J. K. (2004). Playing it safe [human-friendly robots]. *Robotics & Automation Magazine, IEEE*, 11(2):12–21.

Zirbes, R. (2014). Scientific visualization: Volume surface rendering. *http://johnrichie.com/V2/richie/isosurface/volume.html*.