Learning autonomous grasping strategies for a care robot

A Machine Learning approach

Enrico Liscio

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Learning autonomous grasping strategies for a care robot

A Machine Learning approach

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Enrico Liscio

September 18, 2017

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by Heemskerk Innovative Technology BV.
Their cooperation is hereby gratefully acknowledged.

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Learning autonomous grasping strategies for a care robot

by

Enrico Liscio

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: September 18, 2017

Supervisor(s):

ir. Nicky Mol

prof.dr.ir. R. Babuška

Reader(s):

prof.dr.ir. Hans Hellendoorn

prof.dr.ir. Martijn Wisse

ir. Tim de Bruin

Abstract

Autonomous grasping is a key requisite for the autonomy of robots. However, grasping of unknown objects in domestic environments is difficult due to the presence of unpredictability and clutter.
In this paper, a novel algorithm capable of finding an unobstructed grasping pose on unknown regular object shapes in cluttered environments is proposed. At first, an Axis Aligned Bounding Box (AABB) is fitted around the target object point cluster, proposing a limited set of possible grasps. A Support Vector Machine (SVM) is trained to take the decision of the preferred grasp via supervised Machine Learning. Such decision is based on the distribution of obstacles around the target object, inferred by sampling the OctoMap of the scene.
Four different versions of the algorithm have been trained, tested and validated with simulation and real data, with known and unknown target object shapes, in more than 120 different scenarios. The best performing algorithm version returned an unobstructed grasping position in more than 90% of the cases.
Thanks to the the use of the OctoMap representation, significant results are achieved with a limited set of training examples, with a 10 to 10.000 fold reduction of training examples amount with respect to comparable state-of-the-art algorithms. The algorithm also proved itself capable to generalise to novel object shapes and from simulation to real data.

Contents

List of Figures

List of Tables

Acknowledgements

At first, I would like to thank my daily supervisor, Nicky Mol. His support was key for the development and success of the project. His constant challenging of my choices and range of interesting questions helped me in steering the thesis to the results you can read now. He did what a proper supervisor should do: not telling me what to do, but supporting me and asking the right questions to lead me to thoroughly think every step. And even if he does not want to admit it, I know he has nightmares at night, dreaming of me appearing behind his back to ask him another question.

A special thanks goes to my other supervisors, professor Babuška and Cock Heemskerk, for all the inputs and suggestions during our meetings. They were all insightful and helpful in looking at the project from a different angle. And especially thanks for making possible my research in Heemskerk Innovative Technology.

Then, I need to mention Dimitris. In the first year we worked on all projects together, and in the second year we had our thesis in the same company. Despite having different projects, we helped each others during the whole year, always confronting on all decisions taken in both projects. His presence was key in integrating in the company environment, alternating serious moments to small talks. It has been a pleasure to have you as buddy in these two years, malako.

Last but not least, Kamila. Thank you for your patience in always listening to my babbling about the thesis. Thank you for being there in the moments of stress. Thank you for editing and modifying the images in this report. Without you, I would have done everything with Paint. You brought my report and my life from a Paint to a Photoshop level.

Delft, University of Technology — Enrico Liscio
September 18, 2017

"Limits, like fears, are often just an illusion."
— *M. J. Jordan*

Chapter 1

Introduction

1-1 Motivation

According to data from the United Nations Department of Economic and Social Affairs [1], the number of elderly - those aged 60 years or over - has increased substantially in recent years in most countries, and the growth is projected to accelerate in the coming decades. Globally, the number is expected to more than double by 2050, as visible in Figure 1-1. Especially in developed countries, it is estimated a rapid decrease in the ratio between working age people and elderly. In Japan, for instance, the ratio is expected to reach 2.1:1 by 2025 (Figure 1-2). This proportion directly results in less care professionals that have to take care of more geriatric patients.

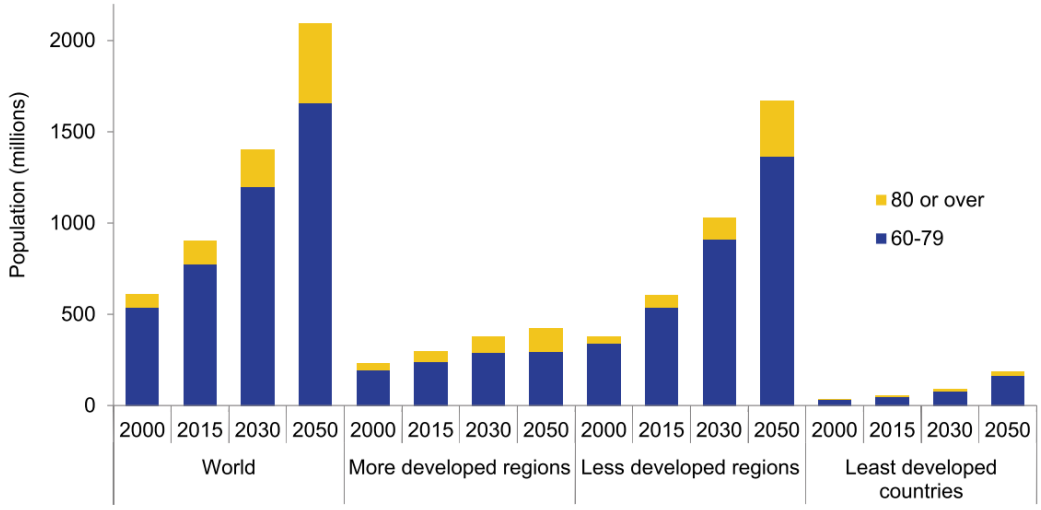


Figure 1-1: Estimated world elderly population growth in the years 2000-2050, divided by country development (from [1]).

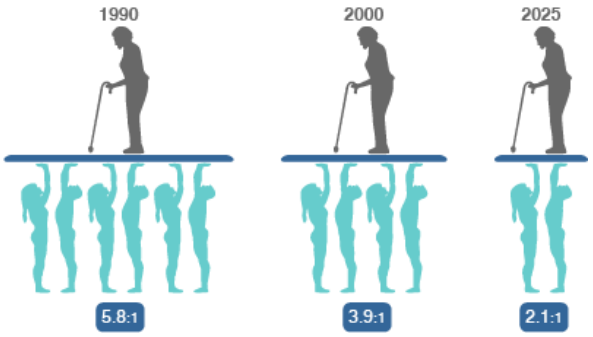


Figure 1-2: Measured and estimated workers to pensioners ratio in Japan[1].

To secure high-quality independent living, domestic care robots can integrate the care professional personnel, by assisting clients in performing their Activities of Daily Living (ADL's, [2]). These activities involve basic actions as opening a door or pouring water in a glass. Entrusting these activities to care robots could alleviate the workload of professionals, allowing them to focus on their primary task of caregiving.

In the recent years, robots proved themselves capable of offering concrete help in industrial environments, by autonomously patrolling aisles and performing pick-and-place tasks [3, 4]. Nevertheless, the use of uncluttered, structured and shielded environments constitute the robots to operate within safe boundaries, often without interactions with human beings. On the other hand, in domestic applications, care robots are destined to operate in an unstructured and cluttered environment, facing tremendous new challenges.

To explore the viability of the aforementioned solution, Heemskerk Innovative Technology (HiT) joined the SACRO project[2], where it is proposed the use of Marco, a semi-autonomous service robot based on the TIAGO platform[3], shown in Figure 1-3. The robot is equipped with a differential drive base, an Aus Xtion Pro Live camera, a 1 Degree of Freedom (DoF) torso, 7 DoF arm equipped with a 6 DoF force/torque sensor at the wrist, and 1 DoF parallel gripper. At any time, a human operator can take over the control of the robot by means of teleoperation, with the use of a coactive interface [5]. Such control can be remotely exerted on drive base, torso and arm, relying on information coming from camera, haptic feedback, and interactions through microphone and screen. In HiT vision, the desired assistance will not be achieved soon by aiming for domestic care robots with fully autonomous capabilities, due to the formidable challenges posed by the unstructured nature of domestic environments in which the robot has to operate. Therefore, a (remote) operator is essential to monitor the robot and assist when its limited autonomous capabilities fall short. In an early stage, the robot will be mostly teleoperated, in order to earn the trust of care receivers and to prove reliability. The automation will be slowly introduced in later stages, accordingly to the implemented technological improvements.

[1] http://news.bbc.co.uk/1/hi/world/asia-pacific/7101663.stm

[2] The Semi-Autonomous Care Robot (SACRO) project was started in April 2015 with the goal provide a robust, low cost, mature solution for the care market that is both manufacturable and functional. It is the result of the collaboration between Heemskerk Innovative Technology, PAL Robotics and Eurostars Eureka.

[3] http://tiago.pal-robotics.com/

Figure 1-3: Marco robot. On the bottom, the black differential drive base. Then, the 1 DoF lifting torso and the 7 DoF arm, ending with a 1 DoF underactuated gripper. On top, the head contains an Asus Xtion Pro Live camera.

For the robot to be able to support care professionals, it needs to be able to autonomously perform a series of physical tasks, like setting the table, or fetching objects, which require the robot to manipulate objects by grasping them. Grasping an object is a complex problem which presents numerous challenges for a robot. The complete grasping pipeline can be divided in the following elementary tasks, as adapted from [6] and visualised in Figure 1-4:

1. **Object detection**: the starting point of a grasping procedure is the localisation of the target object in the environment. The detection is generally performed on the data stream coming from one or more cameras [7].

2. **Determination of the grasping pose**: a grasping position is determined on the target object. Such position corresponds to a 6 DoF final pose for the gripper on the target object [8, 9, 10].

3. **Perception of the environment**: the environment is perceived by a laser range finder or stereo camera, in order to infer the presence of possible obstacles which may obstruct the arm movement [11].

4. **Obstacle-aware path planning**: a plan for the arm movement is computed, in order to reach the desired target position avoiding collisions with obstacles. Such path can be optimal according to pre-defined criteria (e.g. time) or learned with the use of Machine Learning techniques [12, 13].

5. **Control**: the arm is controlled to reach the final destination following the planned path. Visual and haptic feedback can help in correcting the trajectory in case of errors [14, 15, 16].

The outlined grasping pipeline presents several distinct challenges which have to be addressed in order to achieve an efficient grasping. In particular, in domestic environments, as opposed to industrial environments, deciding the optimal grasping pose becomes more challenging

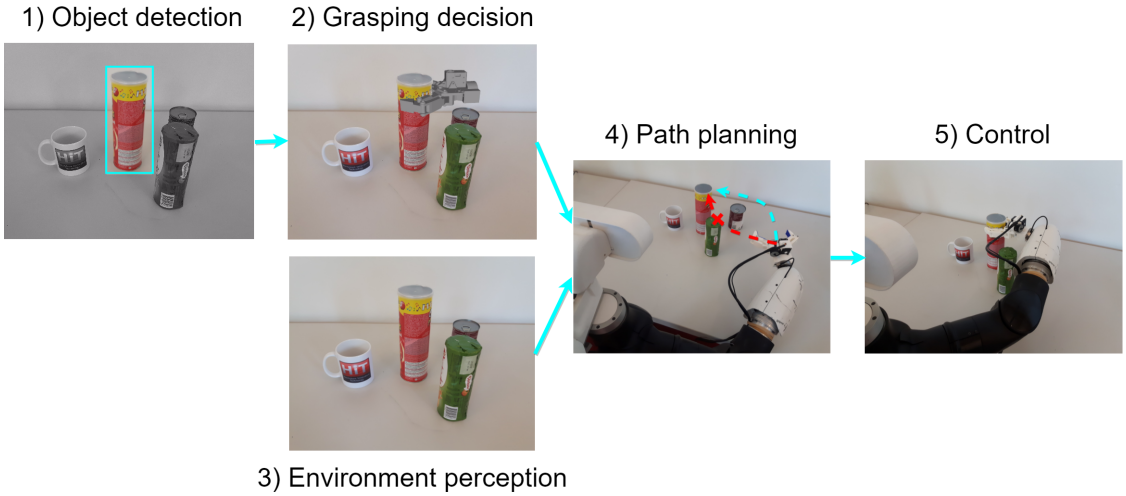


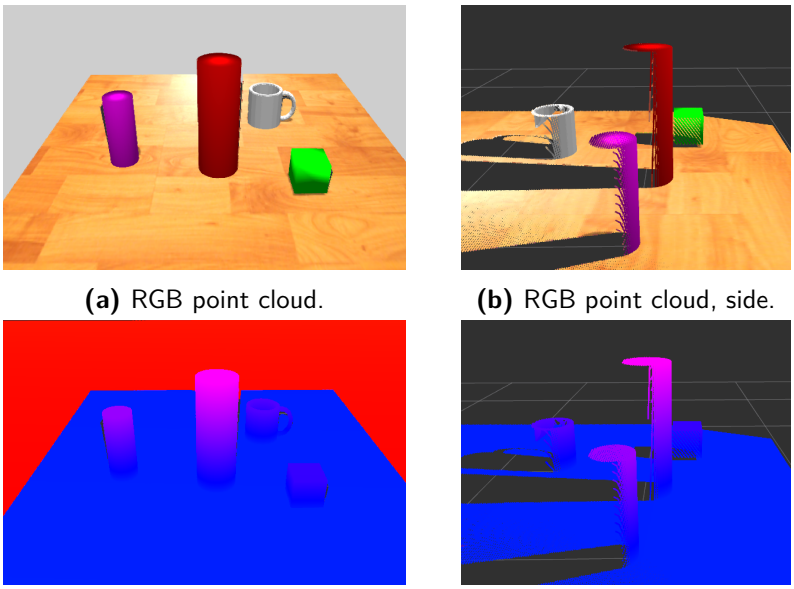
Figure 1-4: The proposed grasping pipeline: 1) the target object is detected in the scene; 2) a grasping pose is selected in front of the surface of the object; 3) the environment surrounding the target object is perceived; 4) a collision-free path is selected in order to place the end effector in the desired position; 5) the arm joints are controlled in order to follow the selected trajectory (adapted from [6]).

since the object to be grasped is often different in size and shape and often the near vicinity of this object is cluttered with other objects, introducing constraints on feasible grasping poses. Furthermore, in typical industrial settings multiple cameras and objects 3D models are available, offering a complete information on the scene. In domestic environments, mounting multiple cameras is usually judged to be too invasive, and due to the unpredictability of the environment, the 3D model of the target object is rarely available. Hence, care robots are typically equipped with a single RGBD camera mounted on the head of the robot, and can rely only on the information coming from that camera for deciding the grasping pose.
This difference in available information, in addition to the multiple limitations and complications introduced in such an environment, constitute the grasping pose decision in a domestic environment to be a formidable challenge.

1-2 Background

1-2-1 Point Cloud Library

Typically, care robots are equipped with an RGBD camera, such as Asus Xtion Pro Live and Microsoft Kinect [17, 18]. These cameras return an information composed of an RGB stream with aligned depth map, namely RGBD information. The raw camera data are commonly processed with the use of libraries as Point Cloud Library [19], returning a 3D representation of the scene in *point clouds*. Each point composing this cloud has a 3D coordinate with respect to the camera frame, with available an aligned RGB map. Figure 1-5 shows the point cloud of a simple simulation scene, with aligned RGB map and z-axis heat-map (i.e. with the colour varying with the height of each point). For the sake of comprehensibility, in this report all point clouds are represented with aligned RGB map.

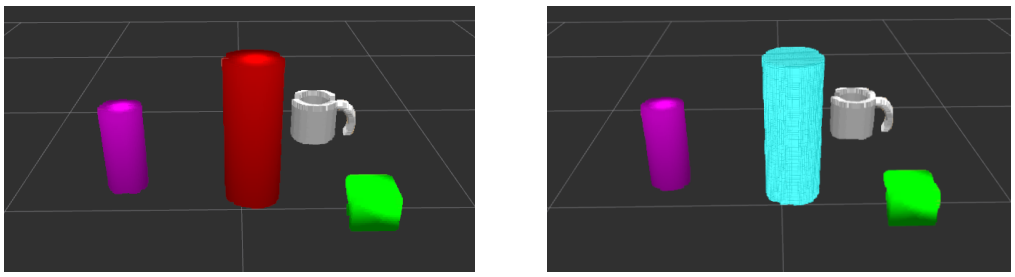


(a) RGB point cloud. **(b)** RGB point cloud, side.

(c) z-axis heat-map point cloud. **(d)** z-axis-heat map point cloud, side.

Figure 1-5: Point cloud representation of a simulation scene, with RGB and z-axis heat-map.

With the use of this representation, it is possible to recognise and isolate groups of points, referred to as point *clusters*. Figure 1-6 shows an example, where the point clusters of the objects on the table top are isolated, and consequently the point cluster of the red cylinder is selected (represented in light blue). Numerous algorithms are publicly available for processing point clouds [20]. Appendix A offers an overview of the algorithm used for isolating a desired object point cluster.



(a) Table top objects. **(b)** Table top objects, red cylinder selected.

Figure 1-6: In the point cloud in Figure 1-5, the table top objects clusters are isolated, and the red cylinder is selected (and displayed in light blue).

1-2-2 OctoMap representation

The OctoMap representation of a scene is based on an octree structure, where a probabilistic occupancy estimation is performed to ensure updatability and to cope with sensor noise [11].

An octree is a hierarchical data structure for spatial subdivision in 3D [21]. Each node in an octree represents the space contained in a cubic volume, named *voxel*. This volume is

recursively subdivided into eight subvolumes until a given minimum voxel size is reached, as shown in Figure 1-7. This minimum voxel size determines the *resolution* of the octree. In robotic applications, octrees can be used to model the *occupancy information* of a volume. If a certain volume is measured as occupied, the corresponding node in the octree is initialized.

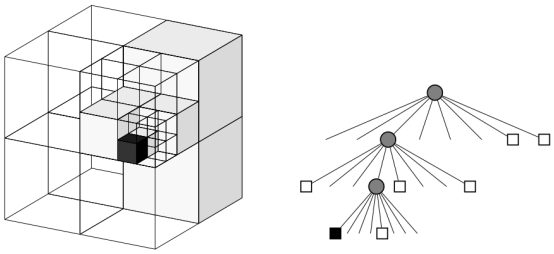
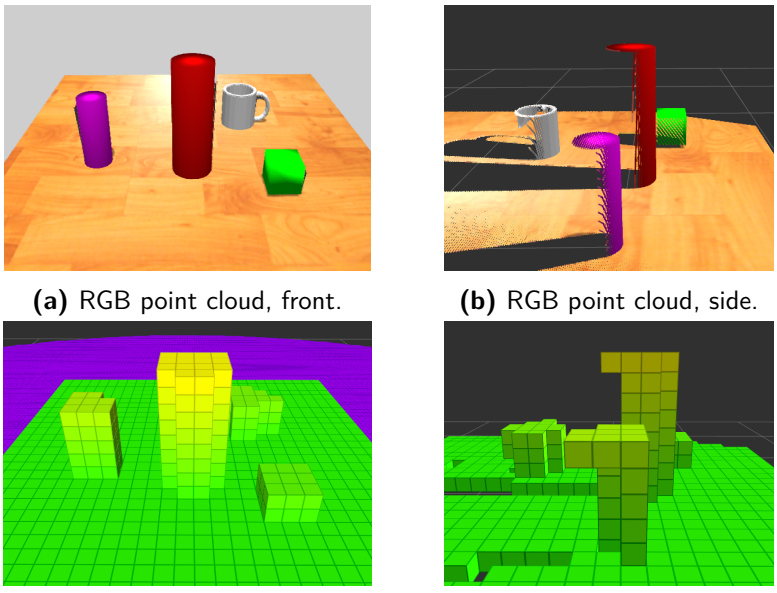


Figure 1-7: Example of an octree storing free (shaded white) and occupied (black) cells, and the corresponding tree structure (from [11]).

In robotic systems, one typically has to cope with sensor noise and temporarily or permanently changing environments. In such cases, a discrete occupancy label is not sufficient: instead, occupancy is modelled probabilistically. The probability of a node being occupied does not depend only on the current measurement, but also on past measurements: any change in the state of a node requires a certain number of observations before being registered. A forgetting factor is introduced in order to ensure the confidence in the map to remain bounded, and consequently the model of the environment updatable.

Figure 1-8 shows a scene with its RGB point cloud and OctoMap representation. For the sake of visibility, only the occupied voxels are represented, with colour varying with the height.



(a) RGB point cloud, front. **(b)** RGB point cloud, side.

(c) OctoMap representation, front. **(d)** OctoMap representation, side.

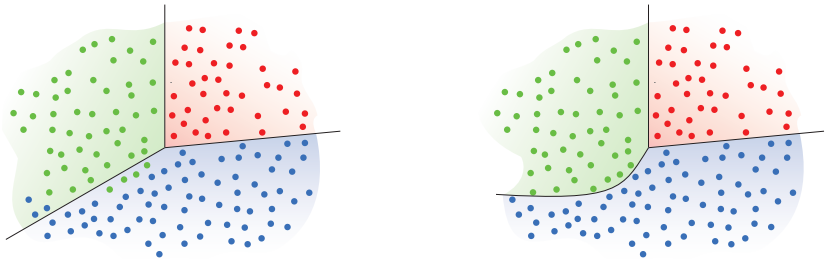
Figure 1-8: RGB point cloud and OctoMap representation of a simple scene. In the OctoMap, only occupied voxels are represented, with colour varying with the height. The purple voxels in the back represent the floor lying behind and below the table.

1-2-3 Machine Learning introduction

In its most basic form, Machine Learning aims at learning an input-to-output map from a set of training examples [22]. $x \in \mathbb{R}^n$ is defined as the set of available inputs, commonly referred to as *features*. Here, x^j is not used to represent the exponentiation, but each exponent represents one of the n elements used to describe the n-dimensional x input ($j = 1 : n$). Then, $h_\theta(x)$ is defined as a parametric function of x, with parameters θ. This function is referred to as *hypothesis*. A simple example of hypothesis is the linear combination of features and parameters: $h_\theta = \theta_1 x^1 + \theta_2 x^2 + \cdots + \theta_n x^n = \theta^T x$, where $x, \theta \in \mathbb{R}^n$. To each input x is associated an output y, referred to as *label*. The goal is to compute the best set of θ parameters so that $y \approx h_\theta(x)$. In order to do so, m examples of (x, y) pairs are provided. The θ parameters are computed to constitute $h_\theta(x)$ the best map from each input example x_i to its output y_i, for all the example pairs ($i = 1 : m$). In mathematical terms, it translates in the minimisation of the following $J(\theta)$ cost function:

$$\min_\theta J(\theta), \quad J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x_i) - y_i\right)^2 \tag{1-1}$$

With the use of more complex hypothesis such as sigmoid functions, it is possible to perform *classification*. Differently from above, the hypothesis no more represents an approximation of the desired output, but the probability to belong to a specific *class* represented by the label y. When the number of classes is larger than 2, such process is defined *multi-class* classification. Figure 1-9 shows a graphical representation of a multi-class classification, with a 2-dimensional feature space and 3 classes. The classification boundaries are defined using a training set composed of m $(x_1, y_1) \ldots (x_m, y_m)$ pairs. When a new x input is proposed, it is classified accordingly to the generated boundaries. Depending on the employed hypothesis function, the boundaries between classes can be linear or nonlinear. As simplistically shown in Figure 1-9, a nonlinear classification is typically more powerful and capable of dealing with complex input-output relations. Support Vector Machines (SVM) with Radial Basis Function (RBF) Kernel [23], Convolutional Neural Networks (CNN, [24]) and their Deep Learning extension [25, 26] are examples of nonlinear multi-class classifiers.



(a) Linear boundaries. **(b)** Nonlinear boundaries.

Figure 1-9: Set of 2-dimensional points classified in 3 classes (each one represented with a different colour). In the linear classification, some green and blue points lie in the wrong class, despite the best attempt at linearly dividing the classes. In the nonlinear classification, all the points are correctly classified.

1-2-4 Related work

Successful algorithms presented in literature are hardly adaptable to the outlined framework, either because they rely on the knowledge of the 3D model of the target object [4, 27, 28], or because they try to reconstruct it with the use of multiple cameras [29, 30]. A detailed literature survey of similar algorithms can be found in [31]. Algorithms that only rely on the information received from a single RGBD camera are here explored.

Hsiao et al. [14] proposed the creation of a bounding box around the target object point cluster, with z-axis in the vertical gravity direction and x- and y-axis along the axes of minimum and maximum variance of the cluster (computed through Principal Component Analysis). Possible grasps are searched orthogonally to the faces of the box, ranked according to geometrical and physical criteria, as the distance from the estimated centre of mass. Jiang et al. [10] represented the grasping position as a 2D oriented rectangle modeling the physical size of the gripper; the decision of the optimal rectangle for a given RGBD image is based on a score function learnt by a two-step cascade Support Vector Machine (SVM). The algorithm was further improved by Lenz et al. [32] with the use of Deep Learning and manual labels. Despite the success, both methods did not take into account the surrounding environment in the decision of the grasping position.

Ten Pas et al. [33] proposed an algorithm capable of grasping objects in a dense clutter. A possible grasp is individuated where the hand is in a collision-free position and part of the object surface is contained between the two fingers of the gripper. An SVM, trained with 6.500 images of objects, uses the point cloud of the scene to return a set of preferred grasping position. Pinto et al. [15] and Levine et al. [16] similarly trained a Convolutional Neural Network (CNN) based on RGBD information, with a self-supervised training built on trial and error: force sensors on the finger tips could autonomously determine the success of a grasp. Respectively, 50.000 and 800.000 attempts were used as training data. Inpired by the success of the mentioned methods, Mahler et al. [34] created the Dex-Net 2.0, a CNN capable of achieving a 99% grasping success rate on singulated objects, learning a grasp robustness function with the use of 6,7 million point clouds. As noticeable, these methods require a consistently large amount of training examples, difficult to collect in a brief period of time.

Finally, recent studies by Balasubramanian et al. [35, 36] show how human intuition consistently leads to successful robotic grasps. They asked humans to interact physically with a robot arm and hand, carefully moving and guiding the robot into the grasping pose, while the robot's configuration was recorded. Statistical and physical analysis demonstrated how humans can successfully guide robots towards efficient grasps. Nevertheless, human labels may be expensive to acquire for large data sets.

Two different components can be distilled from the presented literature. First, the inclination to employ Machine Learning algorithms to predict grasps. Two main reasons support this choice: it provides a degree of generalisation capable to react to novel object shapes, and its training on data allows improvement over time [10, 15, 16, 32, 34, 37]. Second, a trade-off is required between the amount of hand-coded criteria and the quantity of training data needed. On one hand, the robot can be free to explore any grasping option with an entirely self-supervised learning [15, 16]; though, a large training set is needed to face the curse of dimensionality [38]. On the other hand, the grasping options can be limited by hand-coded geometrical and physical criteria (such as proximity to the centre of mass or surface

orthogonality [10, 32, 33]), restraining the range of grasping possibilities but requiring a smaller amount of training data. In the latter case, human intuition and guidance can be used in order to improve grasp efficiency.

1-3 Problem description

Autonomous grasping pose strategies have been successfully devised in the recent years, though mainly for industrial applications [4, 27, 28, 29, 30]. Domestic applications have only recently been approached, and hence new challenges need to be faced. The focus of this project is on the decision of the grasping position, without involving the arm movement. It is assumed that the object detection is separately performed, and that the target object location is available. The following problem is presented:

> **Clutter in the near vicinity of an object to be grasped, limits the set of possible grasping poses.**

In order to fit in the grasping pipeline presented in section 1-1, the decision of the grasping pose should use as input the processed data stream coming from one single RGBD camera (with available target object location) and output a grasping pose for the gripper. The returned grasping pose is assumed to be in range of the workspace of the gripper, and is used as input for arm motion planning.

1-4 Goal

In order to approach the problem presented in section 1-3, the following goal is defined:

> **Conceive an algorithm capable of finding an unobstructed grasping pose on simple novel object shapes, in an environment which may present clutter in the close vicinity of the object to be grasped.**

This goal translates in devising an algorithm which uses as input the processed information obtained from the robot camera (i.e. OctoMap and point cloud, where the target object segmentation is available) and outputs an unobstructed grasping pose for the gripper. Figure 1-10 offers a visual representation of the presented framework. It is requested for the algorithm to be able to generalise and adapt to previously unseen object shapes. In particular, the algorithm should be able to find a feasible grasp on the object despite the presence of clutter in the close vicinity of the object to be grasped. In fact, the goal is to find a grasping pose which is unobstructed and safely distant from neighbouring obstacles, in order to prevent unwanted collisions.

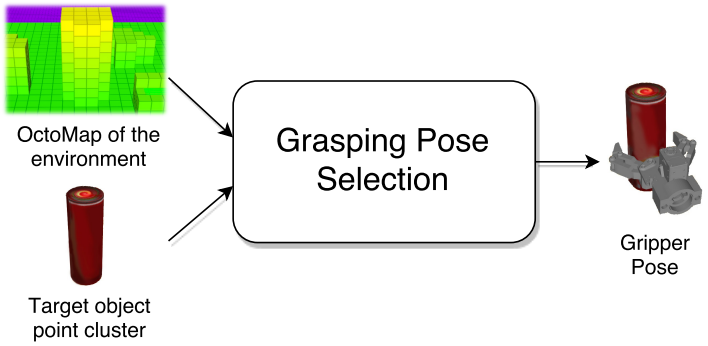


Figure 1-10: The algorithm to be devised is here referred to as 'Grasping pose selection'. Its inputs are target object point cluster and OctoMap representations of the environment. The output is a 6 DoF gripper pose.

1-5 Approach

In order to fulfil the goal of adapting to novel object shapes mentioned in section 1-4, Machine Learning emerges as preferred approach [10, 15, 16, 32, 34, 37]. Furthermore, human guidance was proven to be helpful in guiding robots towards a higher grasp success [35, 36]. Nevertheless, it is not possible to assume to have a large training set of human labels available: a reasonable assumption is to have several hundreds examples. Hence, as opposed to the (raw) point cloud representation of the environment, it is proposed the use of the OctoMap [11], a more compact 3D representation of the scene, believed to carry sufficient information for the grasping decision. This restricted information on the scene can help in downsizing the Machine Learning problem, offering a limited but insightful feature set to the Machine Learning algorithm. Hence, the following approach is proposed:

> **Utilise the OctoMap representation of the environment in the training of a supervised Machine Learning algorithm with the use of a limited amount of human labelled training examples, in order to find an unobstructed grasping pose on a desired object.**

In particular, a Support vector Machine (SVM, [23]) with RBF Kernel is proposed as Machine Learning algorithm. In order to return an unobstructed grasping position on novel object shapes, the SVM should rely on the knowledge of target object and surrounding environment. In order to generalise to new object shapes, the same representation is used for all possible target objects: an Axis Aligned Bounding Box (AABB, [39]) is fitted around the target point cluster, and grasps are searched on the faces of the box. The approximation of the object shape with an AABB constitutes only simple shapes (cylinders, parallelepipeds, cubes) to be truthfully represented, but offers a consistent representation of the target, adapting its dimensions to the object size. Hence, the three dimensions of the AABB are used as features for the SVM.
In order to infer the distribution of obstacles in the near vicinity of the object to be grasped, an insightful volume of OctoMap surrounding the target object is sampled: the returned occupancy information completes the feature set of the SVM.

In order to prove the feasibility of the suggested approach, 800 training examples were collected in a simulation environment, with several target objects and different obstacles distributions. The algorithm was tested in the simulation environment with the same target objects used in the training set plus previously unseen object shapes, proving adaptability to novel shapes. Furthermore, the algorithm was also tested with real camera data, showing extensibility and robustness to noisy and inaccurate data, in spite of the training set being collected in the simulation environment.

1-6 Assumptions

The following assumptions are made in this project:

- The target object detection and segmentation are separately performed, according to the guidelines presented in Appendix A.
- The target object shape is simple and symmetric, as cube, cylinder or parallelepiped.
- The target object is lying on a flat surface. The obstacles are instead not assumed so: they may be lying above the scene as in case of shelves, for instance.
- There is always at least one unobstructed grasping pose on the target object.
- The returned grasping pose is in range of the gripper workspace.

Chapter 2

Method

This chapter provides a detailed overview of the proposed algorithm, aimed at returning feasible grasping poses on novel object shapes in cluttered environments. The input is the point cloud (where the target object cluster has previously been detected and isolated) and OctoMap of the scene. The returned output is a full 6 DoF grasping pose for the gripper. It is composed of four steps, as depicted in Figure 2-1:

1. An Axis Aligned Bounding Box (AABB, [39]) is created around the target object point cloud, and grasps are only searched on its faces. This decision allows to reduce the grasping options (section 2-1).
2. An insightful portion of space surrounding the target is sampled, in order to infer the distribution of obstacles. The use of the OctoMap [11] of the scene permits to reduce dimensionality with respect to the use of the full point cloud (section 2-2).
3. A supervised Machine Learning algorithm is employed to return an unobstructed grasping choice. Specifically, a Support Vector Machine (SVM) with Radial Basis Function (RBF) Kernel [23] is used (section 2-3).
4. The gripper grasping pose and orientation are computed in order to be in front of the selected AABB face and line (section 2-4).

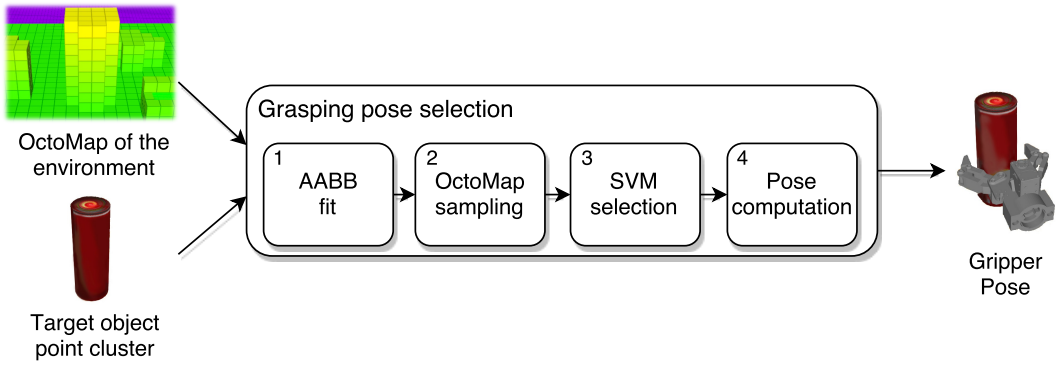
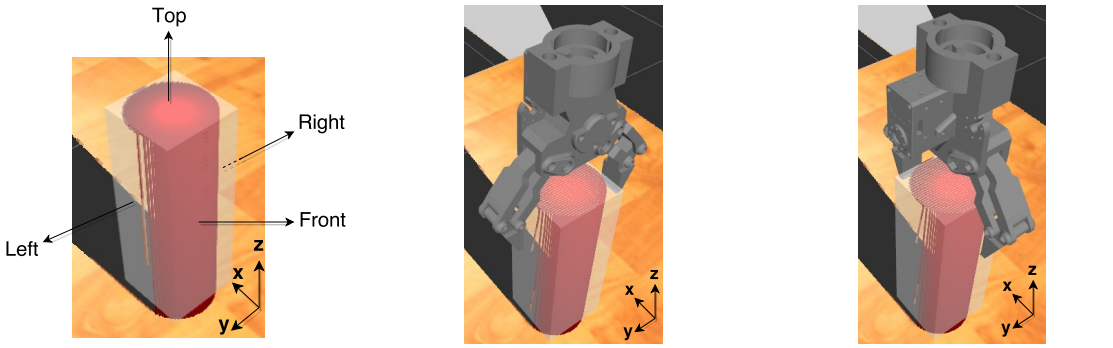


Figure 2-1: Visual overview of the proposed approach: 1) An Axis Aligned Bounding Box is fitted around the target object; 2) The surrounding space around the target object is sampled; 3) An SVM chooses a grasping option; 4) The gripper pose is computed.

2-1 Axis Aligned Bounding Box

Given the target object point cluster, an Axis Aligned Bounding Box (AABB) is fitted around it [39]. An AABB is the smallest box, aligned with a reference frame, containing all the points of a cluster. In order to compute it, the points of the cluster with smallest and largest coordinate are selected along each axis of the frame, representing the bounds of the box in that direction. In this case, the chosen reference frame is the one on the base of the robot, used as reference for all the other frames of the robot, camera included. Thus, the dimensions of the box vary accordingly to the target object shape, but the orientation is invariant. In particular, the z-axis is always oriented with the vertical gravity axes, and it is possible to define left, front, right, top, back and bottom face of the box with respect to the robot's gaze.

Being the target object assumed to be lying on a flat surface (which can be referred to as a table, for simplicity), possible grasps are searched on right, front, left and top face. The bottom face is excluded being in contact with the table, and the back due to its non-visibility. Furthermore, the top face can be approached in two orthogonal directions, not possible for the other faces due to the presence of the table below the object. In order to simplify the notation, the two approach directions are represented as two different faces, named 'top-parallel' (when the fingers opening direction is aligned with the AABB y-axis) and 'top-perpendicular' (when the fingers opening direction is aligned with the AABB x-axis). Fig. 2-2 offers a visual interpretation of the AABB and of the mentioned faces. Hence, in total, the gripper can approach the box on five different faces.

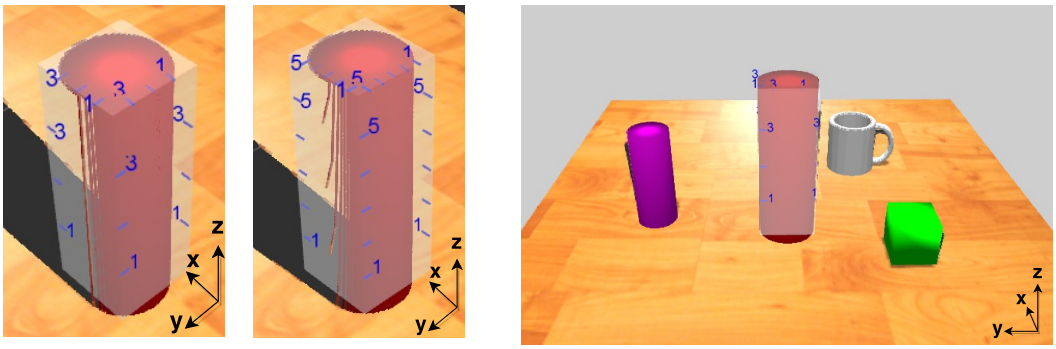


(a) AABB faces. **(b)** Top-parallel grasp. **(c)** Top-perpendicular grasp.

Figure 2-2: A visualisation of the AABB around a red cylinder. The top-parallel and top-perpendicular grasps are shown.

Each face is divided into l steps, generated proportionally to the dimension of the respective face in order to equally divide it. These steps are referred to as *lines*. The gripper can be placed in front of each one of the generated lines: hence, $5l$ grasping options are presented for each object. A visual interpretation is offered in Fig. 2-3. It is a discretisation of the grasping choices available on each face, leading to a restriction on the available grasping options.

The proposed representation significantly reduces the grasping options, offering $5l$ analogous choices for each objects. The representation does not take into account the physics of the object, but just the geometry. It is hardly applicable to complex shapes, but suitable to represent simple shapes as cylinders or parallelepipeds. Nevertheless, this simplification aims at restraining the size of the problem for the Machine Learning algorithm presented in section 2-3: each grasping option represents a class for the algorithm, resulting in $5l$ classes.



(a) AABB, $l = 3$. **(b)** AABB, $l = 5$. **(c)** AABB in a scene, $l = 3$.

Figure 2-3: On the left, examples of AABB with $l = 3$ and $l = 5$. On the right, an AABB is generated around the target object in a simple scene, with $l = 3$.

2-2 Space Sampling

A knowledge of the environment is essential for determining an unobstructed grasping pose. In fact, especially in cluttered environments, it is believed that the feasibility of the grasp logically precedes the grip stability on the object: there is no point in judging the potential stability of a grasp, if such grasp is not feasible due to obstruction. Hence, the space surrounding the gripper should be searched for areas where to safely locate the gripper. Given this consideration, a volume of space surrounding the target object is sampled; precisely, a semi-sphere centred at the base of the AABB, as shown in Figure 2-4. This portion of space is considered of interest, being the object assumed to be supported by a flat surface.

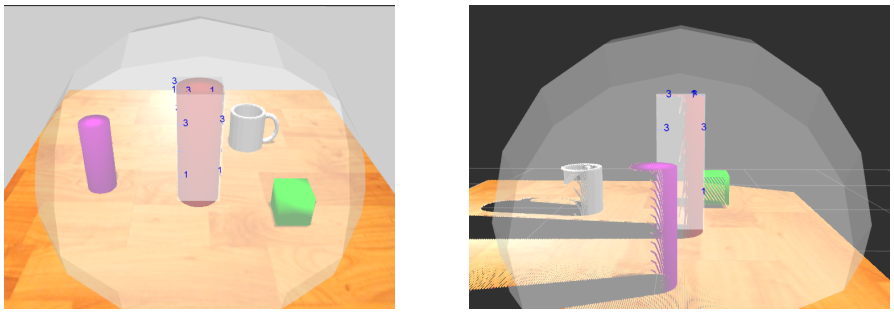


Figure 2-4: The interesting semi-spherical volume, with 30 cm radius, around the target object (highlighted with its AABB).

In order to sample the volume, the OctoMap of the scene is used [11]. Refer to subsection 1-2-2 for an overview on OctoMap. A distribution of P 3D points is generated, in order to sample the volume of the aforementioned semi-sphere. In this project, the following simplification is proposed: if a point falls in a voxel with occupancy probability larger than 66%, it returns an occupancy information of 1; if the occupancy probability is smaller than 33%, it returns 0; in any other case, it returns 0,5. In particular, other cases can be represented by moving objects (not considered in this frame) or by occluded areas behind large objects.

Ideally, each voxel included in the semi-sphere should be sampled by a point, in order to collect a complete occupancy information. For instance, an OctoMap with 3 cm resolution would lead to a total of 2.256 points within a semi-sphere with 30 cm radius, as shown in Figure 2-5. This amount is considerably limited with respect to the information contained in a standard RGBD image: an image with 480x360 resolution contains 480·360=172.800 values. Nevertheless, the amount is still believed to be too large for the goal of training the Machine Learning algorithm with few hundreds manually labelled examples. In addition, as visible in Figure 2-5, this distribution is judged to be unnecessarily dense: it would provide a very accurate representation of the scene, which is not strictly needed in order to judge areas of grasping feasibility. A sparser distribution could still provide sufficient information on the scene, helping in downsizing the Machine Learning problem. Two options are explored in order to scale it down, detailed in the following subsections.

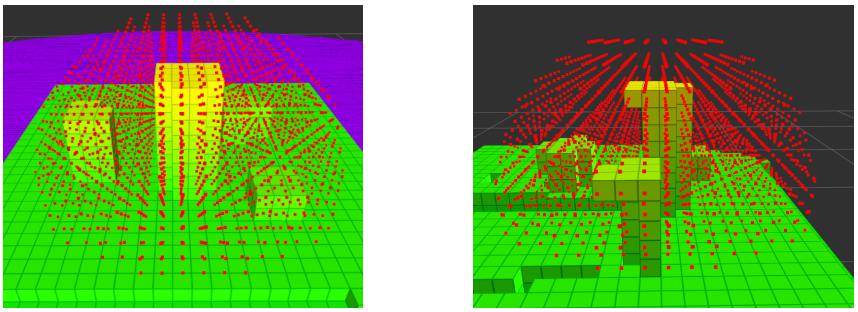


Figure 2-5: With respect to the scene presented in Figure 2-4, all the voxels of the 3 cm resolution OctoMap present in semi-spherical volume are sampled (the sampling points are shown in red).

2-2-1 Spiral Sampling

One option for reducing dimensionality is to maintain a high OctoMap resolution, but without sampling all the voxels lying within the semi-sphere. Instead, a sparser distribution of points is generated, trying to cover the volume as homogeneously as possible. Each points will fall in a particular voxel, returning its occupancy information as previously described.
Points are generated according to the so-called Vogel's method [40]. This method was first introduced by Helmut Vogel in 1979 for describing the distribution of seeds in the sunflower head. Its peculiarity is the ability to cover the surface of a circle in an evenly spread fashion, starting with the first point generated in the centre and expanding in a spiral fashion towards the circumference. Algorithm 1 shows the generation of ν points on the surface of a unit circle centred at the origin:

Algorithm 1 Golden spiral in 2D

function GoldenSpiral(ν)

$$\alpha = \pi\left(3 - \sqrt{5}\right)$$
$$\vartheta = \alpha \cdot linspace(1,\ \nu,\ \nu)$$
$$r = \sqrt{\frac{linspace(1, \nu, \nu)}{n}}$$
$$\text{points[x]} = r\cos(\vartheta)$$
$$\text{points[y]} = r\sin(\vartheta)$$

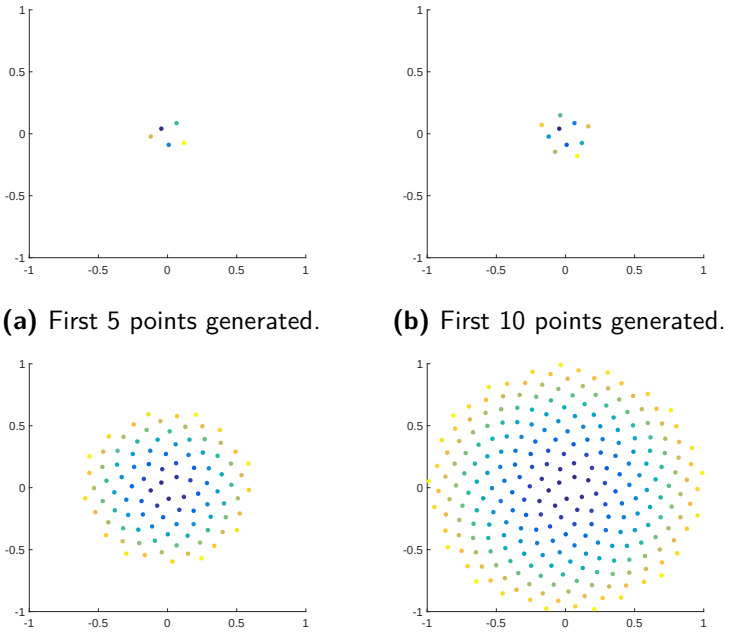
Where α is the golden angle ($\approx 137.5°$).

Figure 2-6 shows the generation of $\nu = 256$ points on a unit circle centred at the origin. In Subfigure 2-6a the spiral pattern can be appreciated: each newly generated point has an increasing distance from the centre, and a golden angle angular distance from the previous.

Adapted in a 3D version, Algorithm 2 demonstrates how to generate ν points on the surface of a unit sphere centred at the origin.

Algorithm 3 shows the extension of the algorithm to a semi-sphere with arbitrary centre and radius.

In total, P points are generated on the surfaces of equally spaced concentric semi-spheres (referred to as *layers*), in order to cover the targeted volume. Figure 2-7 shows one layer and a section of a multi-layer distribution.



(a) First 5 points generated. **(b)** First 10 points generated.

(c) First 100 points generated. **(d)** Total 256 points generated.

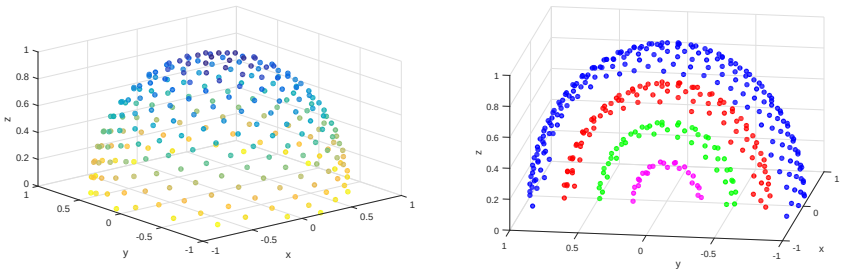
Figure 2-6: A set of 256 points is generated with the Vogel's method on a unit circle centred at the origin. In each subfigure, a portion of points is displayed. In each portion, the points color transitions from blue (first generated) to yellow (last generated).

Algorithm 2 Golden spiral on a unit spherical surface centred at the origin

function GOLDENSPIRAL(ν)
$\alpha = \pi\left(3 - \sqrt{5}\right)$
$\vartheta = \alpha \cdot linspace(1,\ \nu,\ \nu)$
$z = linspace(1 - \frac{1}{\nu},\ \frac{1}{\nu} - 1,\ \nu)$
$r = \sqrt{1 - z \cdot z}$
$\text{points[x]} = r\cos(\vartheta)$
$\text{points[y]} = r\sin(\vartheta)$
$\text{points[z]} = z$

Algorithm 3 Golden spiral on a semi-spherical surface with specific centre and radius

function GOLDENSPIRAL(ν, $centre$, $radius$)
$\alpha = \pi\left(3 - \sqrt{5}\right)$
$\vartheta = \alpha \cdot linspace(1,\ \nu,\ \nu)$
$z = linspace(1 - \frac{1}{\nu},\ \frac{1}{\nu} - 1,\ \nu)$
$r = \sqrt{1 - z \cdot z}$
if $z \geq 0$ **then**
$\text{points[x]} = radius \cdot r \cdot \cos(\vartheta) + centre[\text{x}]$
$\text{points[y]} = radius \cdot r \cdot \sin(\vartheta) + centre[\text{y}]$
$\text{points[z]} = radius \cdot z + centre[\text{z}]$



(a) One layer. **(b)** Section of four layers.

Figure 2-7: A layer and a section of a multi-layer spiral sampling. On the right, each layer is represented with a different colour.

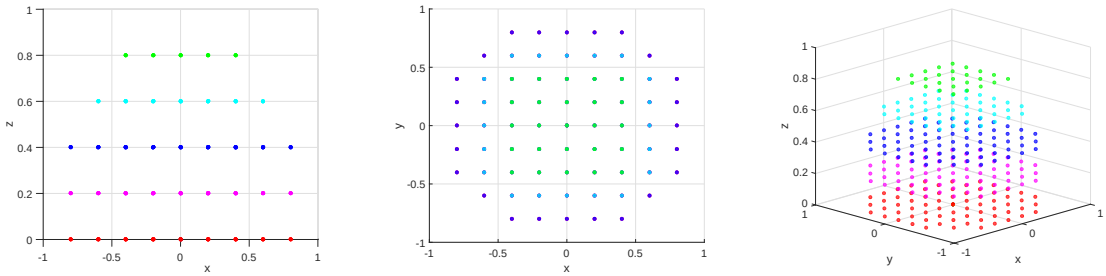
2-2-2 Grid Sampling

An alternative method to reduce the dimension of the feature space is to decrease the OctoMap resolution: using a lower resolution, less voxels are contained inside the semi-spherical volume, and each one can be sampled. Approximately, the amount of voxels decreases cubically with the OctoMap resolution. Hence, fewer points are generated in a 3D grid pattern, with each one falling in a different voxel. Algorithm 4 shows the generation of a grid pattern with s resolution inside a semi-sphere with arbitrary centre and radius.

Figure 2-8 shows a grid with s =0,2 resolution generated in a unit semi-sphere centred at the origin.

Algorithm 4 Grid in a semi-spherical volume

```
function GRID(centre, radius, s)
    point3d p
    for p[z]=centre[z] to centre[z]+radius step s do
        for p[y]=centre[y]-radius to centre[y]+radius step s do
            for p[x]=centre[x]-radius to centre[x]+radius step s do
                if (p[x])^2+(p[y])^2+(p[z])^2 ≤ radius^2 then
                    points[i]=p
```



(a) Side view. **(b)** Top view. **(c)** 3D side view.

Figure 2-8: Grid sampling performed inside a unit semi-sphere centred at the origin. The colour varies along with the z height.

2-3 Machine Learning Algorithm

A Machine Learning algorithm is employed with the aim to return an unobstructed grasping choice. In particular, the occupancy information returned by the space sampling (section 2-2) and the three dimensions of the AABB are proposed as features. The output is a choice of face and line on the AABB, hence leading to $5l$ classes. As multi-class Machine Learning algorithm, Support Vector Machines (SVM) with RBF Kernel and Artificial Neural Networks (ANN) are considered. Resulting from the empirical experiment performed by Caruana et al. [41], a brief comparison between the two algorithms is offered in Table 2-1.

The proposed problem is expected to be nonlinear, but not heavily: it is expected to have clusters of similar training examples leading to a similar choice of class. Using an example, a group of obstacles located on the right of the target object leads to a grasp on the left of the target; a different but analogous distribution of obstacles on the right should lead to a similar result. Hence, a group of similar obstacles distributions leads to the same choice. The Gaussian RBF Kernel is suitable for dealing with similar situations. Furthermore, SVM's provide a certainty of optimum convergence and an understanding of the internal behaviour. Finally, the presence of only two parameters helps in understanding the effect of each of them in the class selection. In conclusion, the use of a multi-class SVM with RBF Kernel is proposed for this project.

A mathematical overview of SVM's is offered in subsection 2-3-1 and Appendix B. Two different combinations of the algorithm are proposed in subsections 2-3-2 and 2-3-3.

Table 2-1: The table shows how SVM's and ANN's compare in different categories.

	SVM with RBF	ANN
Nonlinear Classification	Good	Excellent
Convergence to Global Optimum	Ensured	Not Ensured
Knowledge of Internal Behaviour	Good	Poor
Number of Tunable Parameters	2	Variable

2-3-1 Support Vector Machines

Logistic Regression

Before diving in the mathematical background behind Support Vector Machines, a brief introduction to the basic classification method, named *logistic regression*, is needed. Refer to subsection 1-2-3 for the terminology used in the following paragraphs. In its basic form, logistic regression has to return a choice between two classes, referred as binary 0 or 1. It uses the following sigmoid function as hypothesis (shown in Figure 2-9):

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{2-1}$$

This sigmoid function returns values between 0 and 1, which can be interpreted as probability scores: $\theta^T x \geq 0 \implies h_\theta(x) \geq 0.5 \implies y = 1$; $\theta^T x < 0 \implies h_\theta(x) < 0.5 \implies y = 0$.

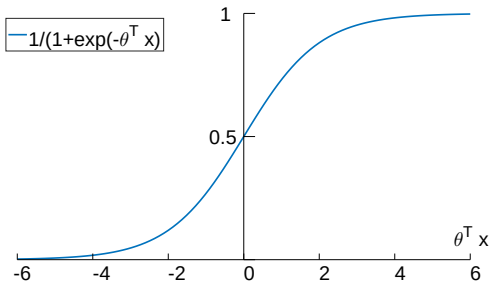


Figure 2-9: The sigmoid function used in logistic regression.

In order to have a convex optimisation problem, the cost function $J(\theta)$ is recast as follows:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}\left(h_\theta(x_i), y_i\right) \tag{2-2}$$

$$\text{Cost} = \begin{cases} -log(h_\theta(x)) & \text{if } y_i = 1 \\ -log(1 - h_\theta(x)) & \text{if } y_i = 0 \end{cases} \tag{2-3}$$

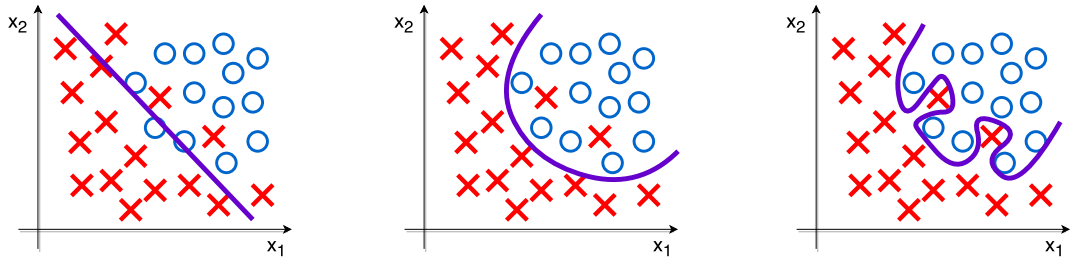
$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left[(y_i) \left(-log(h_\theta(x))\right) + (1 - y_i) \left(-log(1 - h_\theta(x))\right) \right] \tag{2-4}$$

The optimisation problem is typically solved with gradient descent or stochastic gradient descent. In order to guarantee a rapid convergence to the global optimum, a feature scaling is suggested, i.e. having the feature magnitudes lying in a specific limited range. Suggested ranges are $-1 \leq x_i \leq 1$ or $0 \leq x_i \leq 1$ [42].

For expanding to multi-class classification, the one-vs-rest approach is employed: with k classes, each class is classified versus all the remaining $k-1$ classes with a binary classification, and this process is repeated k times. On a new input x, to make a prediction, it is selected the class i that maximises the related $h_\theta^i(x)$ hypothesis.

Bias and overfitting

The cost function in 2-2 only aims at minimising the misclassification errors. The optimisation is heavily influenced by the complexity of hypothesis $h_\theta(x)$. In Equation 2-1 just the linear combination of x and θ is proposed, but it can be replaced with a high order polynomial in x, for instance, to lead to a nonlinear classification. A simple function may return a poor class separation due to its intrinsic limitation, whereas a more complex function may lead to an unrealistic division of the classes. Figure 2-10 shows an example. These two opposed problems are respectively referred to as *bias* and *overfitting* (where the latter is commonly referred to as more *variance* in the classification).



(a) Linear division (bias). **(b)** Smooth nonlinear division. **(c)** Nonlinear division (overfitting).

Figure 2-10: Examples of bias and overfitting. A linear separation of classes (left) can return a poor classification, leading to bias. A highly nonlinear separation of the classes (right) can lead to an unrealistic division of classes aiming at minimising misclassification errors, causing overfitting. A smooth separation of the classes (centre) can represent a better classification, despite the presence of *outliers*, i.e. few misclassified cases which do not truthfully represent the class division.

Regularisation

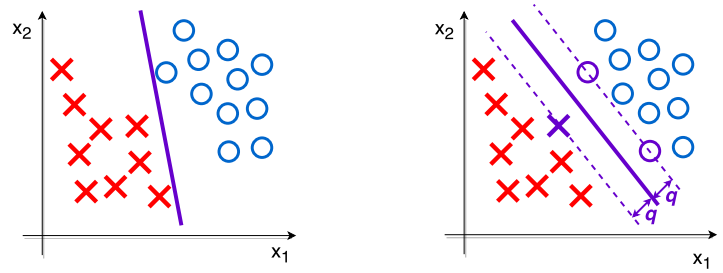
A small magnitude of the parameters θ helps in limiting high nonlinearities, hence preventing overfitting. In order to achieve so, a *regularisation* is commonly performed. The $J(\theta)$ cost function in Equation 2-2 is modified as follows:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}\left(h_\theta(x_i), y_i\right) + \lambda \sum_{j=1}^{n} \theta_j^2 \tag{2-5}$$

Where λ is the regularisation parameter. A large λ enhances the penalty on the magnitude of the parameters θ, limiting nonlinearity; consequently, the bias is accentuated and the variance reduced. Viceversa, a small λ leads to larger variance and smaller bias.

Large margin classification

The cost function in Equation 2-5 penalises misclassifications and parameters magnitude. Though, it does not consider the quality of the classification margin. Figure 2-12a shows an example of linearly classified point distributions: it fulfils all the requirements of the proposed cost function. Nevertheless, it is intuitive to judge the separation in Figure 2-12b as better. The latter is indeed a *large margin classification*, representative of a Support Vector Machine (SVM). In the following paragraphs, an intuitive approach to SVM is offered; for the complete and accurate mathematical explanation, refer to Appendix B.



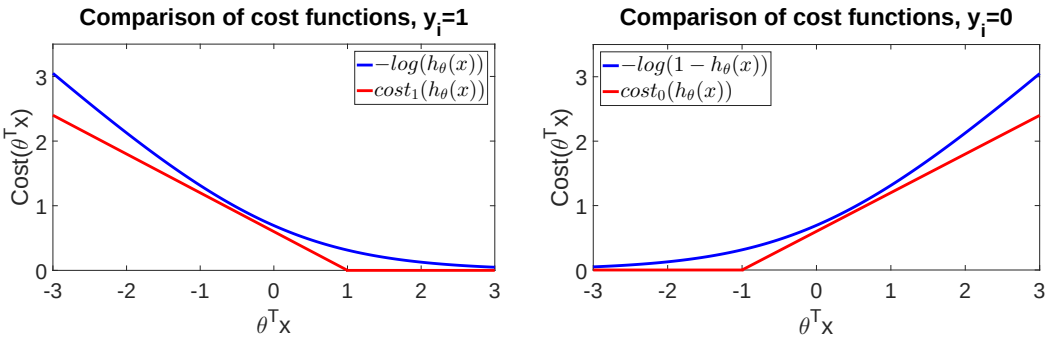
(a) Logistic regression. **(b)** Large margin classification.

Figure 2-11: Both presented examples correctly classify the points distribution with a linear decision boundary. Nevertheless, the large margin classifier maximises the distance q between the support vectors (points denoted in purple) and the division boundary.

In order to achieve this result, the Cost presented in Equation 2-3 is modified as follows:

$$\text{Cost} = \begin{cases} cost_1(\theta^T x) & \text{if } y_i = 1 \\ cost_0(\theta^T x) & \text{if } y_i = 0 \end{cases} \tag{2-6}$$

With $cost_1(\theta^T x)$ and $cost_0(\theta^T x)$ shown in Figure 2-12.



(a) Cost($\theta^T x_i$), $y_i = 1$. **(b)** Cost($\theta^T x_i$), $y_i = 0$.

Figure 2-12: The new cost functions (red, Equation 2-6) introduced in order to achieve large margin classification, compared with the linear regression cost functions (blue, Equation 2-3).

With the use of this new Cost, the $J(\theta)$ function in Equation 2-5 is recast as follows:

$$J(\theta) = C \sum_{i=1}^{m} \Big[(y_i) cost_1(\theta^T x_i) + (1 - y_i) cost_0(\theta^T x_i) \Big] + \sum_{j=1}^{n} \theta_j^2 \tag{2-7}$$

Here C has an effect opposed to λ: a large C enhances the penalty on the classification errors, causing the classification boundaries to be more prone to overfitting. Viceversa, a small C puts more emphasis on the parameters magnitude and hence leads to more bias.
This cost function not only pushes $\theta^T x_i$ to be ≥ 0 when $y_i = 1$ (which was the goal of the logistic regression, to have $h_\theta(x) \geq 0.5$), but pushes it to be $\theta^T x_i \geq 1$ to minimise such cost, hence forcing x_i to have a larger distance from the classification boundary. In fact, the effect of the new cost function is to maximise the distance q between the classification boundary and the closest points of neighbouring classes, as shown in Figure 2-12b: only these points are used for the definition of the boundaries, and are referred to as *support vectors*.

Kernel trick

The hypothesis and cost function presented so far led to a large margin classification with linear boundaries. Nevertheless, better performances can be achieved with the use of nonlinear classification boundaries. With this purpose, the use of a nonlinear kernel is proposed. The feature space is mapped to a higher dimensional space with the use of a nonlinear function. Several functions can be used, but for the scope of this project only the Gaussian Radial Basis Function (RBF) is presented. Each x_i is mapped through the following ϕ function:

$$\phi_i = exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2} \right) = exp\left(-\gamma \|x - x_i\|^2 \right) \tag{2-8}$$

With $\gamma = 1/2\sigma^2$. The cost function in Equation 2-7 becomes as follows:

$$J(\theta) = C \sum_{i=1}^{m} \text{Cost}\left(h_\theta(\phi_i), y_i\right) + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2 \tag{2-9}$$

In practice, the large-margin linear classification is performed on the space obtained from the nonlinear function application on the feature space. Mapping back to the original feature space, such classification is denoted with nonlinear boundaries. Refer to Appendix B for additional details.

The parameter γ is the inverse of σ^2 in a Gaussian function: the smaller the σ^2 is (and hence the larger γ is), the smaller the Gaussian curve width is, leading to more abrupt changes in the classification decision, and hence to more variance. Viceversa, a smaller γ will lead to lower variance and higher bias.

C and γ represent the two tuning parameters which can be used to act on bias and variance of the returned SVM model.

2-3-2 Single SVM

In order to approach the aforementioned problem, the simplest option is to use a single multi-class SVM, using the complete features set and returning as output the choice of one of the $5l$ classes, i.e. a selection of one of the $5l$ lines generated on the faces of the AABB. Figure 2-13 offers a schematic representation, and subsection 3-2-1 an insight on the selection of the P points.

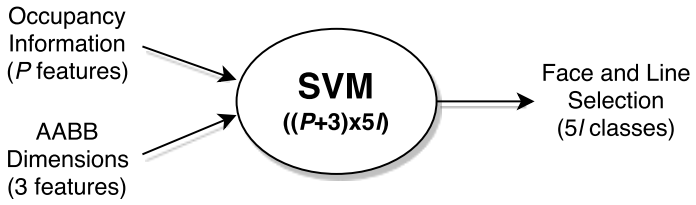


Figure 2-13: A schematic representation of the Single SVM implementation. One multi-class SVM is trained: on the left, the features are occupancy information (P points) and AABB dimensions; on the right, the classes are the lines generated on the faces of the AABB. Inside the SVM ellipse it is indicated the amount of features and classes, as (*features* x *classes*).

2-3-3 Cascade SVM

Alternatively, two multi-class SVM's can be employed in a cascade fashion. The first SVM uses the complete features set and returns the choice of one of the five faces of the AABB. The second SVM employs as features the AABB dimensions, the selection taken by the previous SVM and the occupancy information of just a portion of p points located in front of the face selected by the first SVM, judged to be sufficient for the line selection (Figure 2-14). Subsection 3-2-2 offers an insight on the selection of the p points.

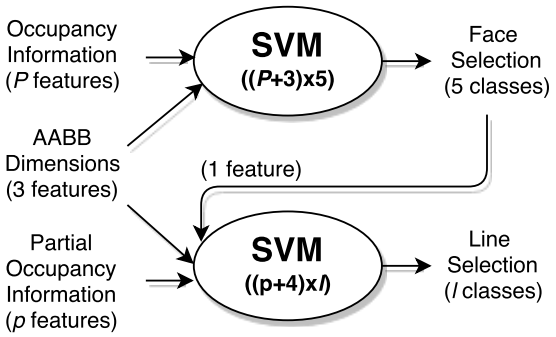
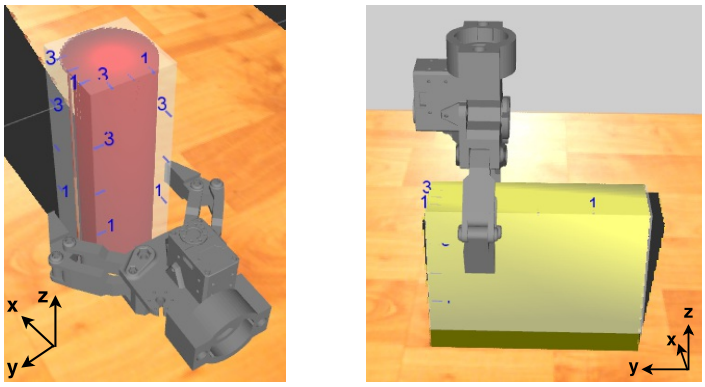


Figure 2-14: A schematic representation of the Cascade SVM implementation. The first part uses AABB dimensions and occupancy information as features, returning the selection of one of the five faces of the AABB. The second uses AABB dimensions, partial occupancy information and face selection as features, to return the choice of one of the three lines on the selected face. Inside the SVM ellipse it is indicated the amount of features and classes, as (*features* x *classes*).

2-4 Pose Computation

Given a selection of face and line, the gripper pose is computed so that the gripper centre is at 10 cm from the chosen face, orthogonal to the face. Just a 6D grasping pose computed; no arm movement is performed. A visualisation is offered in Figure 2-15.



(a) Front face, line 1. **(b)** Top-perpendicular face, line 3.

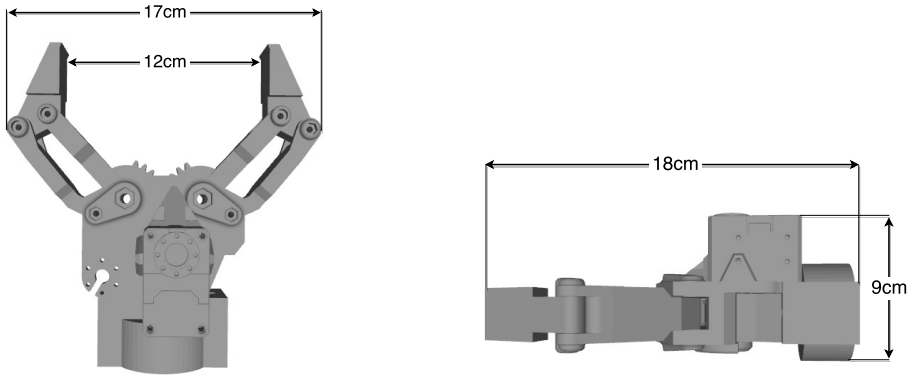
Figure 2-15: The gripper pose is computed accordingly to the selected face and line.

Chapter 3

Experimental Validation

3-1 Experimental Setup

The collection of the training set is performed in a simulation environment, whereas the experiments are conducted both in simulation and on the robotic platform. The employed simulation environment is the Gazebo platform [43], run on a personal laptop with processor Intel Core i5-6300HQ, 2.3 GHz clock frequency. Both in simulation and real experiments, the PAL Robotics Tiago platform is employed. The robot uses ROS as communication middleware [44]. It is equipped with an Asus Xtion Pro Live RGBD camera, and the included MoveIt! library [45] performs an online OctoMap computation of the visible environment. The parallel gripper has a 12 cm maximal opening between the fingers, as illustrated in Figure 3-1.



(a) Top view. **(b)** Side view.

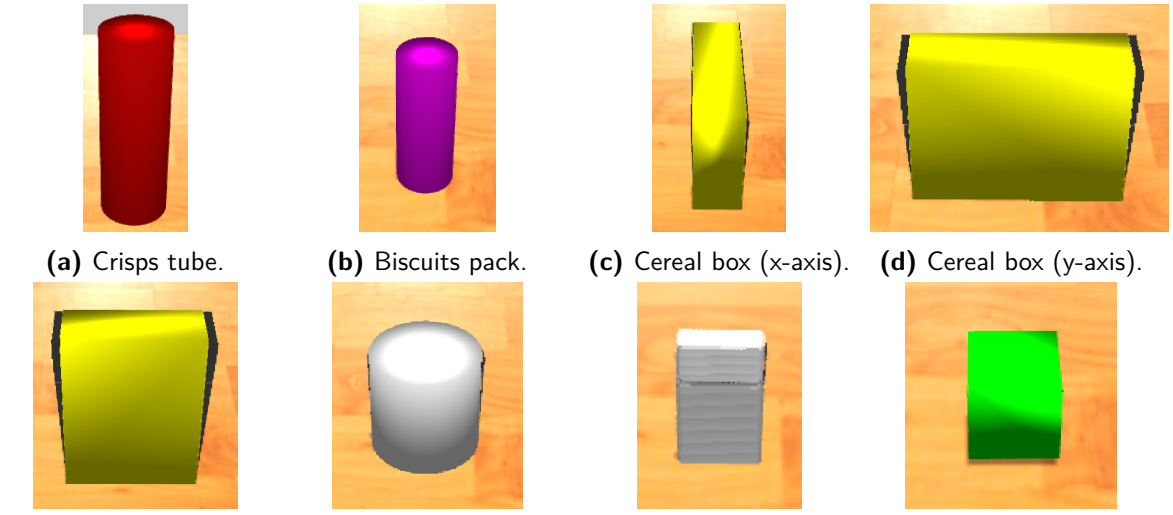
Figure 3-1: The gripper used in the experimental setup, shown in full opening.

In the simulation environment, the target objects set is composed of the following objects:

- Crisps tube: cylinder, 4 cm radius, 23 cm height;
- Biscuits pack: cylinder, 2,9 cm radius, 18 cm height;
- Cereal box: parallelepiped, 20 cm width, 15 cm height, 4 cm depth;
- Beans can: cylinder, 5 cm radius, 10 cm height;
- Cigarettes pack: parallelepiped, 5 cm width, 2 cm height, 8 cm depth;
- Green cube: cube, 6 cm edge.

Where width, height and depth are respectively defined as dimension along y-, z- and x-axis. In addition, the cereal box is proposed with three different orientations (i.e. its larger dimension oriented along the three different reference axes), considered as three different objects due to the difference in grasping feasibility on the faces. Hence, in total 8 different target objects are defined, shown in Figure 3-2.

The obstacles set is composed of the same objects present in the target set, plus a mug and a 'floating lamp' (a simple table lamp on which the gravity is deactivated, so to be steadily floating in mid-air).



(a) Crisps tube. **(b)** Biscuits pack. **(c)** Cereal box (x-axis). **(d)** Cereal box (y-axis).

(e) Cereal box (z-axis). **(f)** Beans can. **(g)** Cigarettes pack. **(h)** Green cube.

Figure 3-2: The eight target objects available in the simulation environment.

For the robotic experiment, the target set is composed of:

- Crisps tube: cylinder, 3,75 cm radius, 23 cm height;
- Biscuits pack: cylinder, 2,5 cm radius, 18 cm height;
- Cereal box: parallelepiped, 22 cm width, 19 cm height, 4,5 cm depth;
- Chickpeas can: cylinder, 2,5 cm radius, 11 cm height.

The cereal box was proposed with largest dimension oriented along x- and y-axis, for a total of 5 target objects, shown in Figure 3-3. In addition, a mug and a wooden bar (lying above the scene) are used as obstacles.

(a) Crisps tube. **(b)** Biscuits pack. **(c)** Cereal box (y-). **(d)** Cereal box (x-). **(e)** Chickpeas can.

Figure 3-3: The five objects used in the real testing set.

Both in simulation and robotic experiments, the target objects were lying on an 80 cm high table, always placed at the centre of the table. Axis Aligned Bounding Box (AABB) computation and consequent OctoMap sampling are independent on the object position, hence for simplicity their position was unvaried. As illustrated in Figure 3-4, the robot was fully lift and with the gaze directed to the table top, capable of seeing the top of the targets.

Figure 3-4: Example of the robotic setup during the robotic tests. An analogous setup was used for the simulations tests.

3-2 Selection of design parameters

At first, the number of lines generated on each AABB face is defined at $l = 3$ (section 2-1). This choice is driven by the trade-off between amount of grasping options and resizing of the Machine Learning problem. A large l would offer a broader variety of grasping options, but at the same time increase the complexity in the decision taken by the SVM. It is believed that 3 lines represent a simple but effective choice for proving the validity of the method.

In order to be able to compare spiral and grid sampling, the selections of semi-spherical volume dimensions, P points in the semi-spherical volume and p points in the portions use by the cascade combinations for the line selection, should be identical or similar:

- The volume is centred 5 cm above the base of the target object AABB (so to avoid sampling the voxels representing the table), with a 30 cm radius. This semi-sphere is expected to cover the volume interesting for the grasping pose decision, being the length of gripper plus attached link 22 cm.
- The number of P points is identical for both samplings (subsection 3-2-1).
- The number of p points is similar and comparable in all portions (subsection 3-2-2).

Moreover, with the use of the simulation environment on the mentioned personal laptop, a 3 cm OctoMap resolution corresponds to the highest resolution which can guarantee online computations: higher resolutions cause the OctoMap computation to be too slow to be computed and shown in real-time.

3-2-1 Selection of the P points for complete space sampling

Spiral Sampling

In the spiral sampling, a 3 cm OctoMap resolution is employed. Points are generated on the surfaces of four concentric layers. Each layer has the following radius (r_i) and number of generated points (n_i):

1. First layer: $r_1 = 7,5$ cm, $n_1 = 6$
2. Second layer: $r_2 = 15$ cm, $n_2 = 25$
3. Third layer: $r_3 = 22,5$ cm, $n_3 = 55$
4. Fourth layer: $r_4 = 30$ cm, $n_4 = 98$

The radius of the layers increases linearly, whereas the number of points increases quadratically with the dimension of the radius, in order to maintain a homogeneous distribution of points (refer to subsection 2-2-1 and Algorithm 3 for additional detail).

In total, 184 points are generated in the semi-spherical volume, with a slightly larger density around the centre: precisely, the distance between neighbouring points is on average 7,2 cm; the smallest distance is 6,6 cm, the largest 7,5 cm. This distribution is considered to be a good trade-off between amount of points and sparsity. Figure 3-5 shows the generated sampling from different angles.

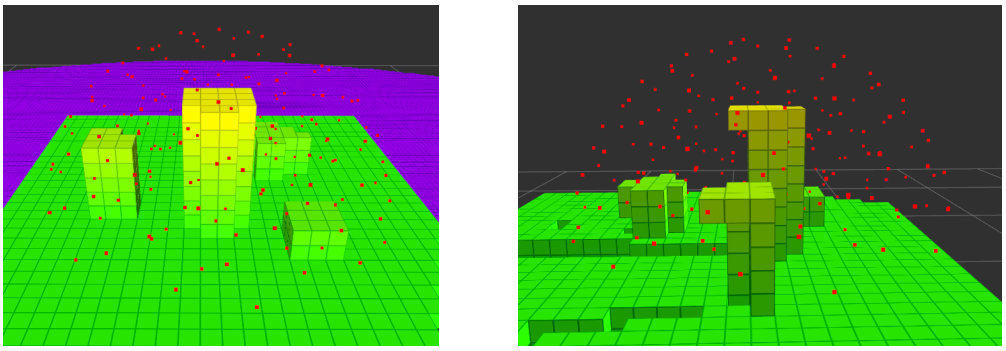


Figure 3-5: Front and side view of the spiral sampling performed on a 3 cm resolution OctoMap. The 184 sampling points are denoted in red.

Grid Sampling

For the grid sampling, the OctoMap resolution is decreased to 7 cm, as shown in Figure 3-6. In this way, only 184 voxels fit inside the semi-spherical volume. Hence, 184 points are generated in a grid pattern overlaying the voxel distribution, as depicted in Figure 3-7.

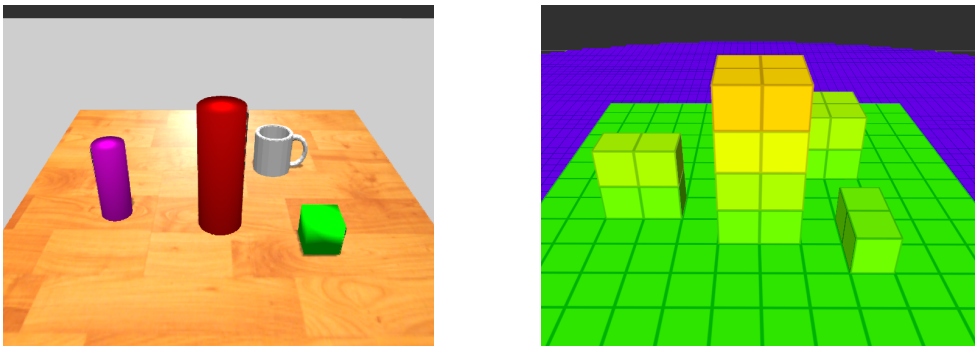


Figure 3-6: The same simulation scene is represented with RGB point cloud and 7 cm resolution OctoMap representation.

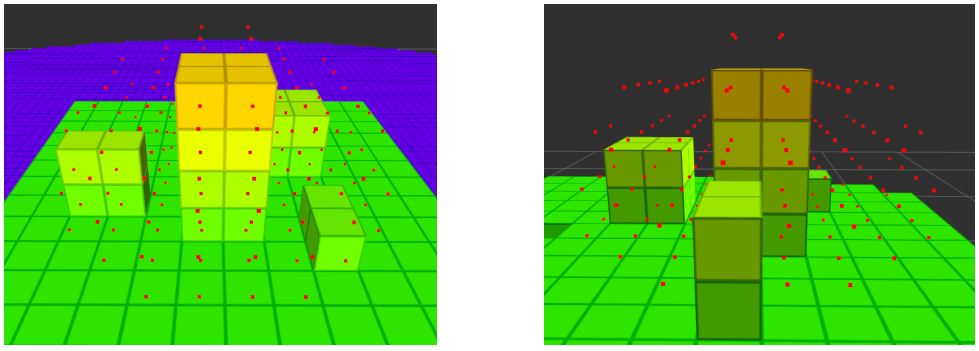


Figure 3-7: Front and side view of the grid sampling performed on a 7 cm resolution OctoMap. As noticeable, the points distribution (in red) overlays the voxel representation.

Comparison

Both methods generate 184 points within a semi-spherical volume with 30 cm radius. Their geometrical distribution is comparable, as proven by the neighbouring points distance. Nevertheless, they differ in three aspects: OctoMap resolution, points distribution homogeneity and OctoMap coverage. One one hand, the spiral sampling inherently returns a more homogeneous and diverse distribution of points, and its higher OctoMap resolution offers a more accurate representation of the scene. On the other hand, the grid sampling allows the complete coverage of the targeted volume, since each present voxel is sampled by a point. Table 3-1 shows a schematic overview of the differences.

Table 3-1: The table shows how spiral and grid sampling compare in the three categories: OctoMap resolution, distribution homogeneity and OctoMap coverage.

	Spiral Sampling	Grid Sampling
High OctoMap Resolution	✓	✗
Homogeneous Distribution	✓	✗
Complete OctoMap Coverage	✗	✓

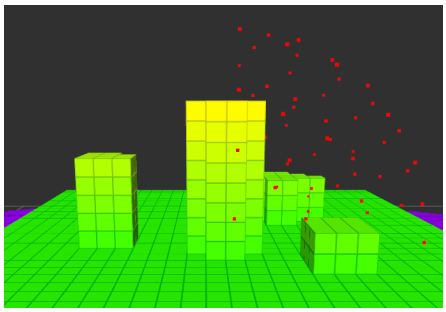
3-2-2 Selection of the p points for partial space sampling

The portion of p points is located in front of the face selected by the first Support Vector Machine (SVM) in the cascade combination (subsection 2-3-3). Precisely, this portion is selected with width of 30 cm, reasonably larger than the gripper width (17 cm).

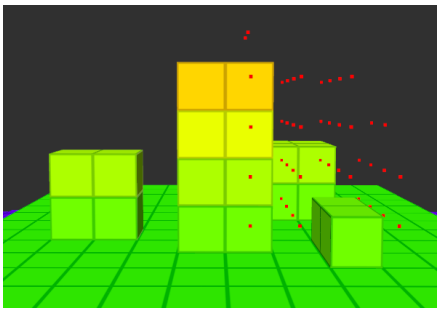
Two different portions of points are cut, differing if the chosen face is on the side (right, front, left) or on top (top-parallel, top-perpendicular). Figures 3-8 and 3-9 show examples for right and top-perpendicular face. These two examples are used for defining the geometrical bounds of the portion; for the other faces, the portion is simply rotated with respect to the vertical axes located at the centre c of the distribution. For the right face, a point lies in the portion if its x and y coordinates are $(c(x)-15) \leq x \leq (c(x)+15)$ and $y \leq c(y)$. For the top-perpendicular face, a point lies in the portion if its x coordinate is $(c(x)-15) \leq x \leq (c(x)+15)$. Given these boundaries, a different amount of points falls in the portion according to the type of sampling and face selection. Table 3-2 shows the amount of p points for each combination, leading leads to a total of four different SVM's to be trained with four different features maps.

Table 3-2: The table shows the amount of points (p) present in each portion, according to the selected face and sampling method.

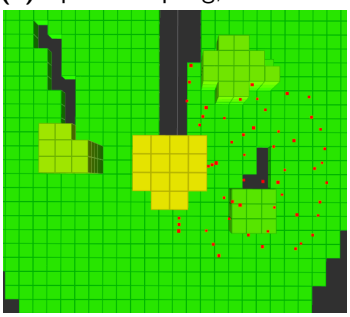
	Spiral Sampling	Grid Sampling
Side Face	57	60
Top Face	118	120



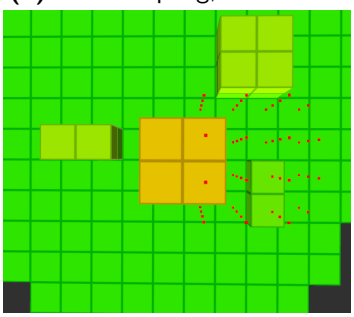
(a) Spiral sampling, front view.



(b) Grid sampling, front view.

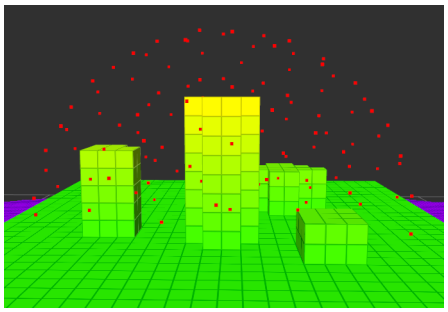


(c) Spiral sampling, top view.

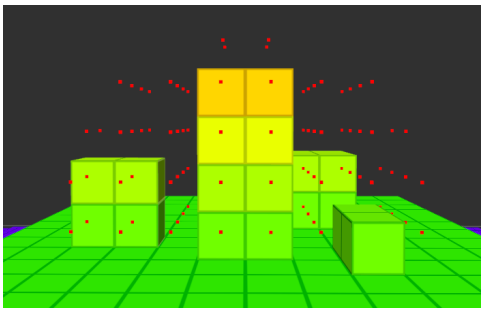


(d) Grid sampling, top view.

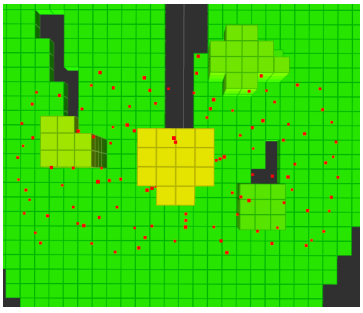
Figure 3-8: The portion of points used by the second part of the cascade SVM to choose the line, with a selection of the right face.



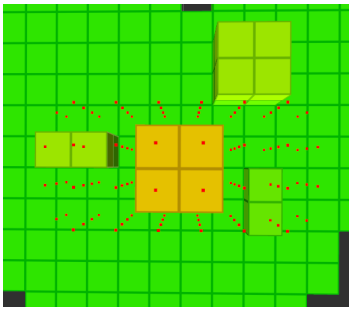
(a) Spiral sampling, front view.



(b) Grid sampling, front view.



(c) Spiral sampling, top view.



(d) Grid sampling, top view.

Figure 3-9: The portion of points used by the second part of the cascade SVM to choose the line, with a selection of the top-perpendicular face.

3-3 Training Phase

Figure 3-10 offers an overview of the 6 algorithm versions and 8 different SVM's to be trained, generated by the choice of sampling, SVM configuration and face. A different face choice leads to an implementation which is different in practice, but similar in concept. Hence, it is possible to simplify the distinction and consider only 4 different versions of the algorithm, as outlined in Table 3-3. The versions are referred to as 'single spiral', 'single grid', 'cascade spiral' and 'cascade grid'.

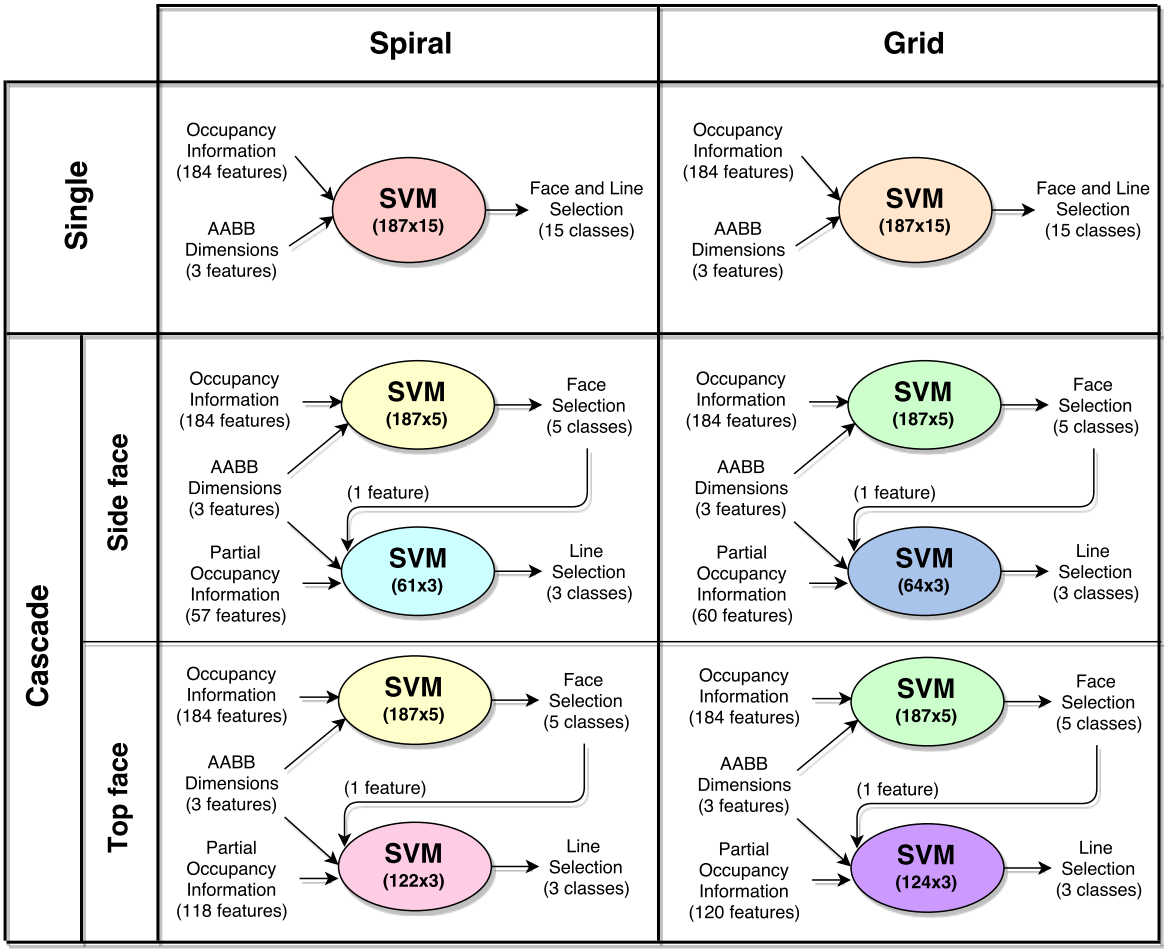


Figure 3-10: Overview of the 6 different version of the algorithm. Columns represent different sampling types, rows represent different SVM combinations. Each different SVM is highlighted with a different colour, for a total of 8 different SVM's to be trained.

Table 3-3: A recap the four different algorithm versions, resulting from different space sampling and SVM combination.

	Spiral Sampling	**Grid Sampling**
Single SVM	Single Spiral	Single Grid
Cascade SVM	Cascade Spiral	Cascade Grid

3-3-1 Collection of the training set

The training set was entirely collected in the simulation environment. Four objects were used as targets: crisps tube, biscuits pack and cereal box aligned with x- and y-axis.
The distribution of obstacles was randomly generated for each new training example. In particular, a maximum of four obstacles was located on the table within a 50x50 cm area centred at the target location. The lamps were spawn with x and y positions within the mentioned boundaries, and with a height with respect to the table surface varying between 20 cm and 30 cm. The choice of employed obstacles was randomly taken in each new scenario.

For each of the four target objects, 200 different obstacle distributions were generated, leading to a total of 800 different *scenarios*, i.e. 800 training examples. In each scenario, one of the 15 grasping options on the target was manually selected by a single human operator. The decision was taken simply based on a sight of the RGB point cloud of the scene, with AABB generated on the target object. Two examples are proposed in Figure 3-11. No gripper was spawn, no movement was performed, no mathematical criterion was used in the choice. This type of decision adheres to real case scenarios, where the operator has to decide with just a simple look on the scene.

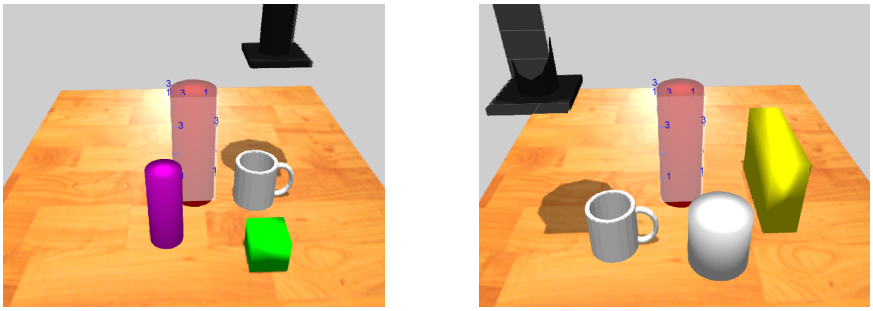


Figure 3-11: Examples of training set scenarios. The target object lies in the centre, denoted with its AABB. In both, floating lamps are present over the scene.

3-3-2 SVM models generation

The 8 SVM models were trained with the use of the LIBSVM library [42]. In order to facilitate the model computation, the features values were scaled to the interval $[0, 1]$ (as mentioned in subsection 2-3-1). Hence, some adjustments were performed on the features as follows:

- The occupancy information is a set of points with values $\in \{0; 0,5; 1\}$, as explained in section 2-2.
- The three AABB dimensions are normalised with respect to the maximal AABB edge dimension, returning values $\in [0, 1]$.
- The choice of the face is represented with the following values: 0 for the left face; 0,5 for the front face; 1 for the right face; 0 for the top-parallel face; 1 for the top-perpendicular face. The selection of side or top face leads to the use of different SVM's, hence it is possible to use the same feature value for different face selections.

For simplicity of notation, the resulting classes (corresponding to the grasping options) are labelled with numbers. The five faces of the AABB are labelled as follows: 1 - left; 2 - front; 3 - right; 4 - top-parallel; 5 - top-perpendicular. The lines are listed from 1 to 3. In the case of the single SVM implementation, the 15 options are labelled from 1 to 15 according to the following criterion: $((\text{face\_number-1}) \cdot 3) + \text{line\_number}$. For instance, class 8 corresponds to the selection of right face, line 2. Table 3-4 offers an overview of all the 15 classes.

Table 3-4: Classes labels corresponding to the grasping options.

		Face				
		Left	Front	Right	Top$\parallel$	Top$\perp$
Line	*1*	1	4	7	10	13
	2	2	5	8	11	14
	3	3	6	9	12	15

In order to train the SVM models, [42] suggests to perform a 5-fold cross-validation on the training set. The training set is divided in 5 subsets of equal size, and in turn one subset is tested using the classifier model trained on the remaining 4 sets. This cross-validation returns an accuracy percentage, measure of how the generated model classifies the examples present in the selected testing set. A coarse grid-search for the two tuning parameters (C and γ, refer to section 2-3-1) is performed, resulting in a heat-map representing the amplitude of the cross-validation accuracy of the tested parameters couples, as in the example shown in Figure 3-12.

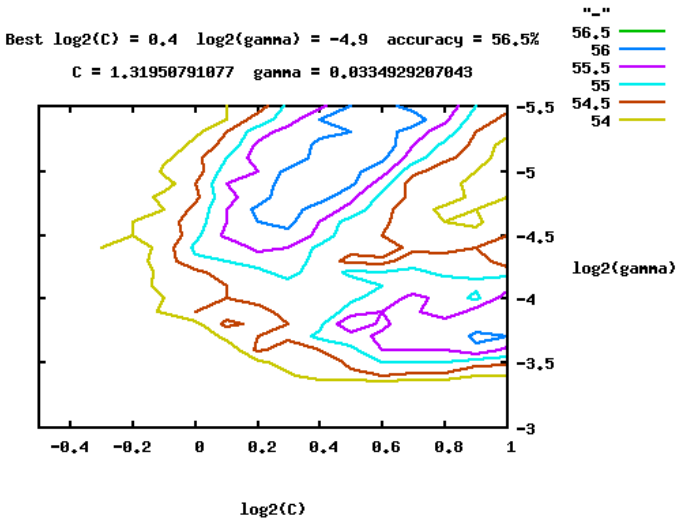
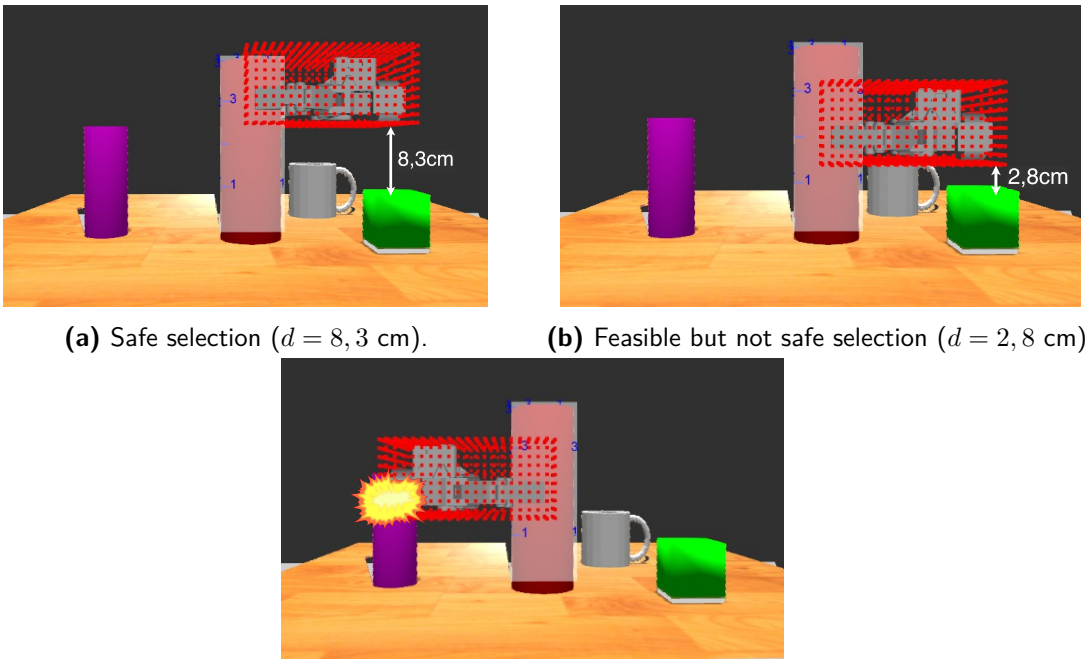


Figure 3-12: Heat-map of the single grid SVM model, with outlined the C and γ couple returning the highest accuracy.

3-4 Performance Metric

In each testing scenario, a target is defined and the gripper is spawned according to the guidelines presented in Chapter 2. In each scenario, the shortest Euclidean distance d between the gripper bounding box (with the dimensions presented in section 3-1) and the closest obstacle point cloud is computed. Figure 3-13 shows the bounding box and the distance from the closest obstacle. Such geometrical metric is intended to be a measure of *feasibility* and *safety* of the returned pose.



(a) Safe selection ($d = 8,3$ cm). **(b)** Feasible but not safe selection ($d = 2,8$ cm).

(c) Not feasible selection.

Figure 3-13: Three grasping options are presented for the same simulation scenario. The gripper bounding box is represented with red points, and the distance d from the closest obstacle (green cube) is shown. In the bottom image, the gripper is overlapping with the purple cylinder, hence the distance d is not shown.

In particular, if $d < 2$ cm, the grasp is judged to be infeasible; if $d \geq 2$ cm, it is judged feasible; if $d \geq 6$ cm, it is judged feasible and safe (with the thresholds heuristically defined). This evaluation causes each returned gripper pose to be classified in one of three categories, for simplicity defined 'red' ($d < 2$ cm), 'yellow' (2 cm $\leq d < 6$ cm) and 'green' ($d \geq 6$ cm). The result is rewarded with a score of 0 if it falls in the red category; a score quadratically increasing between 0 and 1 if in the yellow; a score of 1 if in the green. Figure 3-14 offers a clear visualisation of the described evaluation function. With the use of the evaluation metric, the decisions in Figure 3-13 can be evaluated: in Subfigure 3-13a, $d = 8,3$ cm leads to a score of 1; in Subfigure 3-13b, $d = 2,8$ cm leads to a score of 0,04 (out of 1); in Subfigure 3-13c, $d = 0$ cm leads to a score of 0.

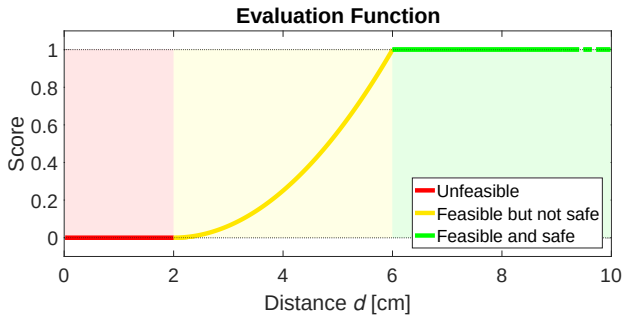


Figure 3-14: The evaluation function devised to judge feasibility and safety of a returned grasping pose. Functions components and background are coloured accordingly to the relative categories.

As mentioned in section 1-6, in each scenario there is at least one feasible grasp. Though, there may not be a safe grasp: in the cases in which the largest possible distance between gripper and closest obstacle is $\leq$ 6 cm, the threshold between feasible and safe is moved to such largest possible distance.

This function is intended to classify and compare the decisions taken by the four proposed combinations. Other functions (linear, exponential) and thresholds between classification categories (5 cm, 6 cm) were considered. Using different evaluation functions, the absolute results and classification distribution changed, but the relative ranking and comparison among the four combinations were unchanged. Hence, a quadratic function with thresholds at 2 cm and 6 cm was considered a strict and fair evaluation criterion, but not excessively strict as an exponential would have been. In fact, grasping choices with distance d around 2 cm are penalised, but choices with d approaching 6 cm are evaluated with a good score.

In every tested scenario, just one grasping option is returned by each algorithm combination: therefore, for each combination, it is possible to compute the percentage of decisions falling in any of the three categories, in addition to the average score resulting by the evaluation function assessment on the complete testing set.

3-5 Experiments

The four trained combinations of the algorithm were equally tested in the simulation environment with the goal to compare their performances (subsection 3-5-1). Consequently, according to the results of the simulation experiments, additional tests were run on the robotic platform, in order to confirm the guidelines deducted from the simulation results (subsection 3-5-2). The setup was analogous in simulated and robotic tests, as detailed subsection 3-1.

3-5-1 Simulation experiments

The four combinations were tested in a simulated environment using Gazebo, under the conditions described in section 3-1. All the 8 proposed target objects were used as targets (the four used in the training set plus four novel), so to test adaptability to novel object shapes or orientations. For each target, 15 random obstacles distributions were generated, for a total of 120 different testing scenarios. Each combination was tested with the exact same testing set, returning grasp decision distribution and average score.

3-5-2 Real world experiments

In the real world tests with the robotic setup, the employed target objects are the five mentioned in section 3-1. Initially, 10 *ad hoc* scenarios were generated to compare the difference in behaviour between single and cascade combinations emerged from the simulation tests results. Subsequently, the cascade grid combination was tested in 42 scenarios.

Differently from the simulation tests, the real camera inherently carries noise and inaccuracies: despite robot and objects being unmoved, the generated OctoMap can vary over time. A different distribution of voxels leads to a different feature map, and may lead to a different grasping decision. The OctoMap probabilistic generation has a good noise rejection (subsection 1-2-2), but nonetheless variations may still occur. In order to limit this influence, only half of the point cloud was used for its generation: one every two points was discarded. This allowed the usage of a more stable point cloud, more rapid to process and less prone to carry noise. Nevertheless, in every scene each algorithm was run 10 times, in order to observe noise effects.

Chapter 4

Results

4-1 Training phase results

Figure 4-1 shows the distribution of the grasping decisions taken by the human operator in the training set (subsection 3-3-1), evaluated according to the criterion presented in section 3-4. The grasping feasibility average score (resulting from the function in section 3-4) is 0.93.

The single Support Vector Machine (SVM) and the first halves of the cascade SVM's are trained with 800 training examples. The amount for the cascade SVM's second halves varies accordingly with the amount of selections of side or top faces: precisely, a side face was selected 408 times, and a top face 392. A distribution of the classes in the training set is shown in Figure 4-2. As noticeable, the distribution is not homogeneous.

A finer grid-search of the tuning parameters (subsections 2-3-1 and 3-3-2) was then performed on the region with higher accuracy intensity, opting for the couple which would return high accuracy and larger γ values (i.e. higher variance), in order to counterbalance the distribution inhomogenity, limiting the expected bias. Appendix B-1 provides a list of selected parameters.

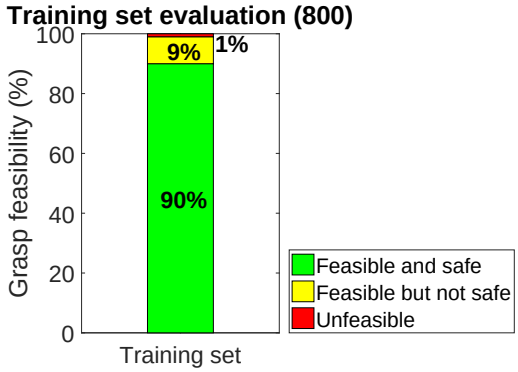
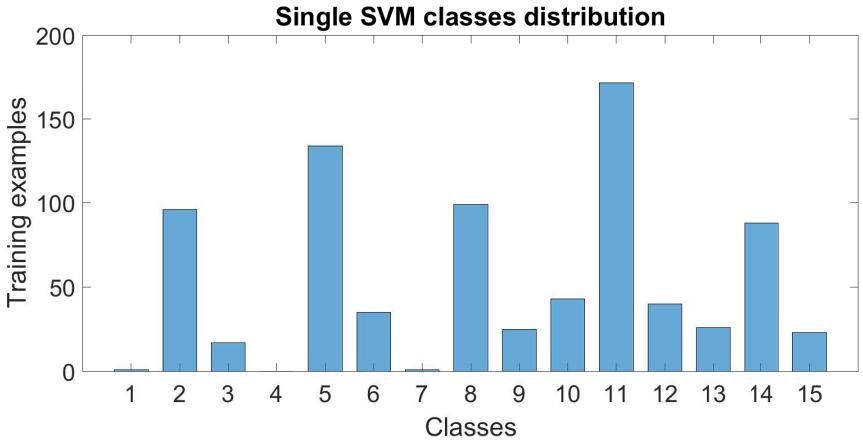
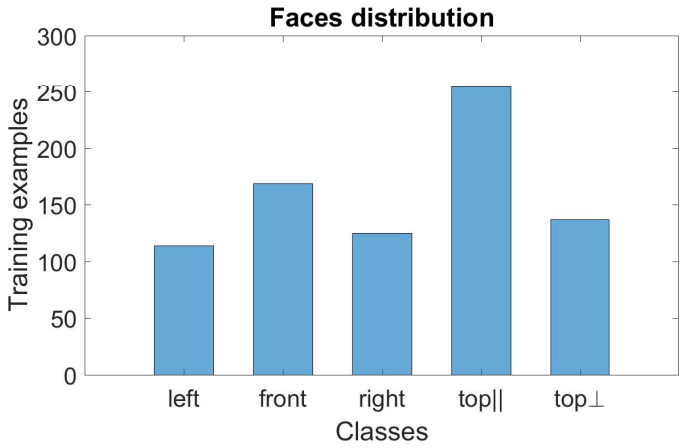


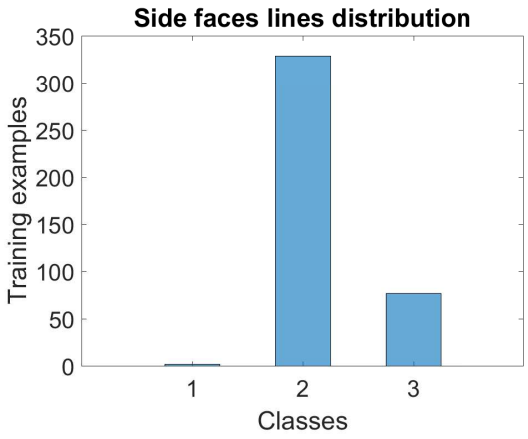
Figure 4-1: The category distribution resulting from the evaluation of the training set with the devised function. In parenthesis in the title, the amount of samples (i.e. scenarios).



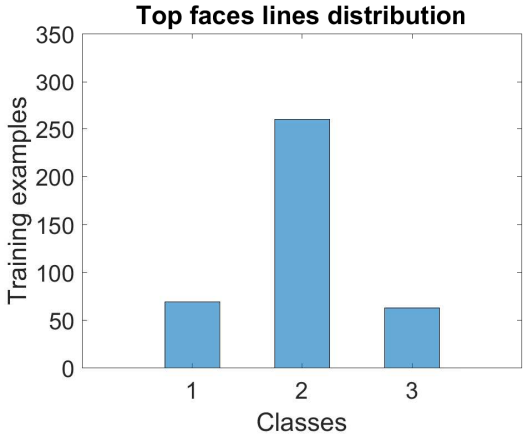
(a) Overall distribution (single SVM).



(b) Distribution of face selections.



(c) Distribution of side faces line selections.



(d) Distribution of top faces line selections.

Figure 4-2: Distribution of classes in the training set. The distribution of the 15 classes is subdivided in selections of faces and lines.

4-2 Simulation experiments results

In each of the 120 simulation testing scenario, one grasping decision was taken by each algorithm version. This decision was evaluated according to the guidelines presented in section 3-4. Figure 4-3 shows the distribution of grasping decisions in the overall testing set. The evaluation of the training set is proposed again, in order to compare the results with it. Figure 4-4 offers a separation of the evaluated results in known and novel object shapes (each composed of 60 scenarios). Finally, Table 4-1 offers a comparison of the average results with known, novel and total target objects sets, result of the use of the evaluation function proposed in section 3-4. The latter results can be used as a single score for comparing the performances of the four combinations.

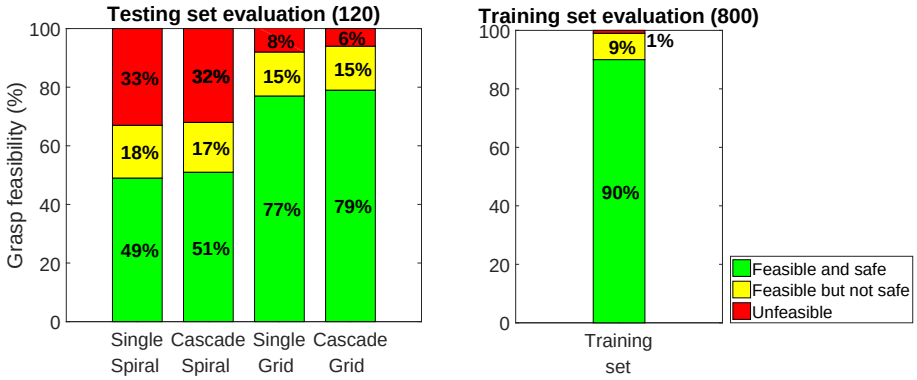
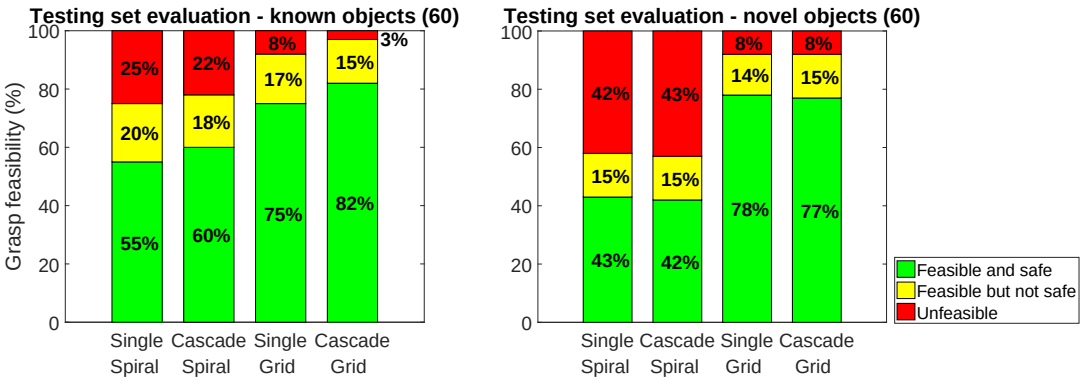


Figure 4-3: The categories distribution in the testing set, for the four different combinations. On the right, the evaluation of the training set is proposed as ground truth. In parenthesis in the title, the amount of samples (i.e. scenarios).



(a) Known objects.

(b) Novel objects.

Figure 4-4: The categories distribution in the testing set, divided in distributions with known and novel object shapes. In parenthesis in the title, the amount of samples (i.e. scenarios).

Table 4-1: Average score resulting from the evaluation with the function proposed in section 3-4, for each of the four combinations with the four objects used in the training, the four novel objects, and the total testing set (composed of known and novel objects).

	Single Spiral	Cascade Spiral	Single Grid	Cascade Grid
Known Objects (60)	0,60	0,64	0,78	0,84
Novel Objects (60)	0,50	0,48	0,84	0,84
Total testing set (120)	0,55	0,56	0,81	0,84

The average score encapsulates the quadratic evaluation in the yellow zone: not all yellow grasps are the same, but their score increases accordingly with their distance from obstacles. A clear example is provided by the comparison between single and cascade grid with novel object shapes: the grasps distribution (Figure 4-4b) may lead to the conclusion that the single grid is slightly better than the cascade, but their average score (Table 4-1) is exactly the same. This means that the yellow grasps returned by the cascade grid were on average better than those returned by the single grid.

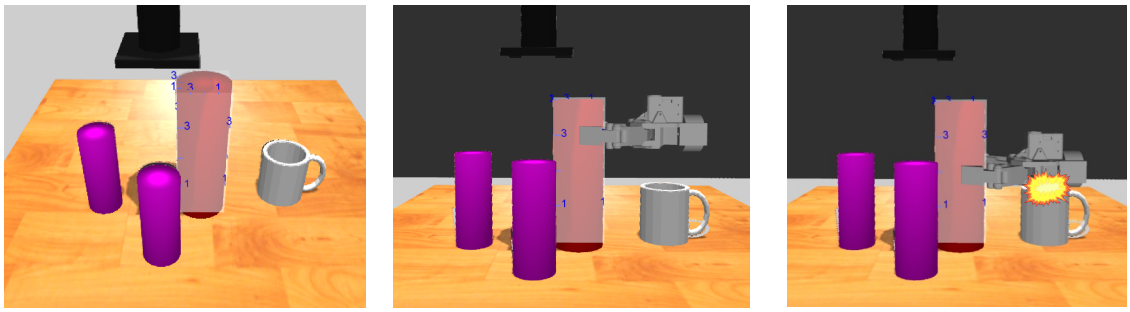
Several results can be observed in the presented figures:

- The grid sampling led to better results than the spiral sampling;
- Spiral combinations results heavily degraded with novel object shapes;
- Single and cascade combinations (of the same sampling method) behaved comparably with novel objects and in the overall training set;
- Cascade combinations returned better results with known objects.

Analysing the results in more detail, it emerged that single and cascade implementation mostly returned similar grasping decisions. Precisely, single and cascade spiral differed in 8% of the cases, single and cascade grid in 12% of the cases. Dividing these percentages between known and novel objects leads to interesting results: single and cascade spiral differed in 7% of the cases (4 out of 60) with known objects, and in 8% (5 out of 60) with novel. Single and cascade grid differed in 17% (10 out of 60) with known, and in 7% (4 out of 60) with novel.

Out-of-bias *need* scenarios

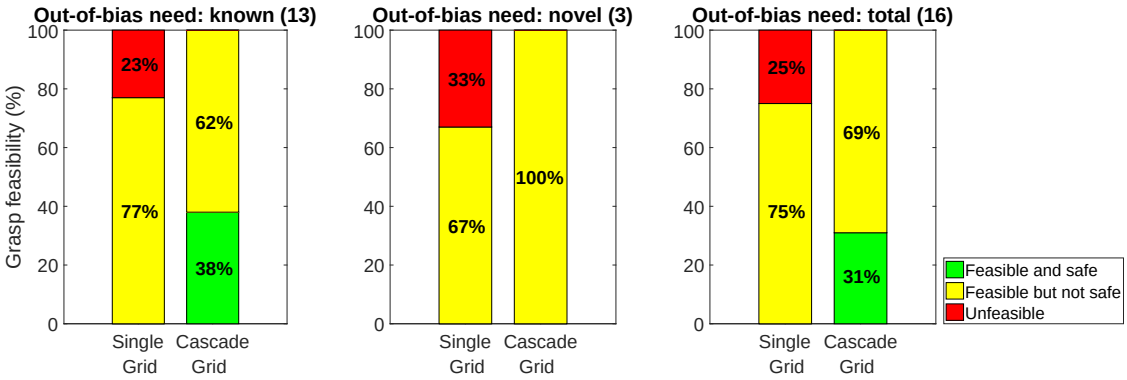
In particular, a difference in the selections taken by single and cascade implementation was found in cases in which an out-of-bias line selection was needed to return a safe position. Precisely, the *need* of an out-of-bias line selection is present when the scene is so cluttered that there is only one partially unobstructed face on the Axis Aligned Bounding Box (AABB), and in front of that face there is an object with the right dimensions so that an out-of-bias selection is better than a bias option (where the bias option is to go for a central line grasp). Figure 4-5 offers an example. In these cases, the cascade implementation returned safer positions than the single: the single could select the correct face, but not the out-of-bias line selection. This difference did not emerge in the spiral combinations, though.



(a) Scene with AABB on target. **(b)** Cascade: right face, line 3. **(c)** Single: right face, line 2.

Figure 4-5: Example of a cluttered scene and related choices of cascade and single grid. Due to the presence of the lamp and table top obstacles, the only feasible grasping position ($d \geq 2$ cm) is on the right face, line 3.

Nevertheless, such extremely specific cases rarely occurred in the testing scenarios, as a result of the random generation of the obstacles distributions. In fact, these cases occurred 13 times with known object shapes, and only 3 with novel. Figure 4-6 shown the resulting distribution of grasps, and Table 4-2 shows the average scores. Only a comparison between single and cascade grid is offered, since the spiral did not prove ability to return out-of-bias line selections. Noticeably, the cascade grid never returned an unfeasible grasping position.



(a) Known objects. **(b)** Novel objects. **(c)** Overall testing set.

Figure 4-6: The grasp quality distribution in out-of-bias *need* scenarios. The results are in percentage values, and the amount of samples indicated in parenthesis in the title.

Table 4-2: Average score for single and double grid in the out-of-bias *need* line selection, divided in known, novel and total object shapes. In parenthesis, the sample size for each category.

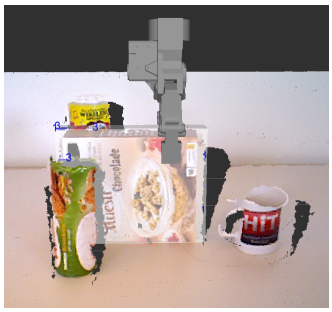
	Single Grid	Cascade Grid
Known Objects (13)	0,15	0,48
Novel Objects (3)	0,27	0,32
Total (16)	0,17	0,45

4-3 Real world experiments results

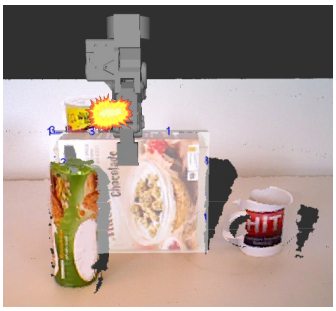
Only the grid combinations were tested with the robotic setup. The single and cascade grid combinations returned identical grasping decisions in 88% of the simulation test scenarios, as mentioned in previous section. Hence, it was considered redundant to test both methods with the robotic setup. Instead, 10 *ad hoc* scenarios were generated with the intent to compare the two implementations in cases where an out-of-bias selection is needed in order to return a safe grasping position, as discussed in the previous section. Two examples are shown in Figure 4-7, with the grasping position returned by single and cascade grid.

(a) Cascade grid: left face, line 3.

(b) Single grid: left face, line 2.



(c) Cascade grid: top-$\perp$ face, line 1.



(d) Single grid: top-$\perp$ face, line 2.

Figure 4-7: Real life scenarios devised to compare single and cascade implementation in specific cases. On the left, the decisions taken by the cascade grid; on the right, the decisions taken by the single grid. In the top example, a wooden bar is present on the top-right corner of the image, which is lying above the scene.

As explained in section 3-5, in order to test noise robustness the algorithm was run 10 times for each scenario. No decision variations were detected. The compared results are proposed in Figure 4-8. The average score was 0,87 for the cascade grid, 0,42 for the single.

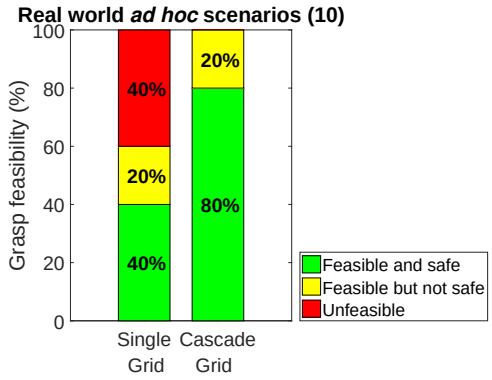


Figure 4-8: Comparison of results between single and cascade grid in the 10 *ad hoc* scenarios.

Summing together these *ad hoc* scenarios with the analogous scenarios examined in the previous section (Figure 4-6 and Table 4-2), the out-of-bias need scenarios can be grouped in a batch with a bigger sample size, precisely 26 samples. Figure 4-9 and Table 4-3 offer an overview of the comparison results.

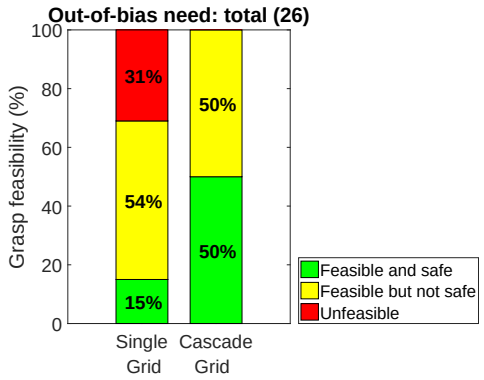
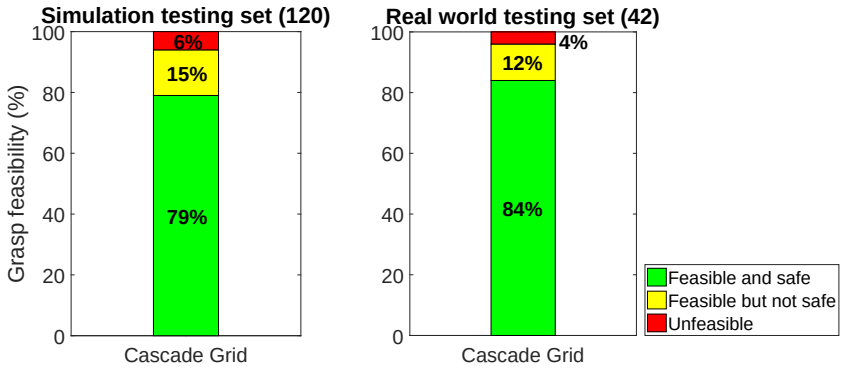


Figure 4-9: Comparison of results between single and cascade grid in the 26 out-of-bias need scenarios, composed of 16 simulation scenarios and 10 real world scenarios.

Table 4-3: Average score for single and double grid in the out-of-bias need line selection, divided in simulation, real world and total scenarios. In parenthesis, the sample size for each category.

	Single Grid	Cascade Grid
Simulation (16)	0,17	0,45
Real World (10)	0,42	0,87
Total (26)	0,27	0,61

Only the cascade grid was tested in the 42 robotic scenarios, for a total of 420 runs of the algorithm. The results are presented in Figure 4-10b. In Figure 4-10a the cascade grid simulation results are proposed for a comparison. The average score in the real world tests was 0,86, as opposed to the 0,84 in the simulation tests.
Moreover, noise influenced only 2 scenarios (out of 42). These scenarios will be singularly analysed in the following chapter.



(a) Cascade grid, simulation tests. **(b)** Cascade grid, real world tests.

Figure 4-10: Comparison between simulation and real world tests for the cascade grid combination.

Chapter 5

Discussion

5-1 Discussion on the training phase results

Common features emerged in the decisions taken by the human operator (section 4-1, Figure 4-2). In particular, on objects whose dimensions allowed full range of choice, front and top-parallel faces were preferred: they were considered safer, since a grasp on the other three faces would cause the gripper to have a finger on the blind face of the Axis Aligned Bounding Box (AABB). If an obstacle was present in front of the mentioned faces, it was observed that one of the unobstructed faces was chosen. Typically, it can be observed that grasps around the estimated centre of mass were preferred (lines 2), resulting from human experience and intuition. In case of presence of obstacles, this bias selection was avoided and a different line was selected, depending on the generated scenario. Hence, a certain bias towards the most commonly selected options was expected in the Support Vector Machine (SVM) models, both for single and cascade implementation.

The human was preferring unobstructed poses selections, but with no information on the resulting gripper pose. Thus, in some cases the bias to opt for grasps around the centre of mass led to decisions which were not safe according to the devised evaluation function. For this reason, the graph in Figure 4-1 does not report only safe choices. Nevertheless, this distribution should be considered as ground truth for the test results.

5-2 Discussion on the experiments results

5-2-1 The grid sampling outperforms the spiral sampling

This difference is caused by the spiral sampling lack of complete OctoMap sampling (see the comparison in subsection 3-2-1). This problem has three major consequences, responsible for the performances difference:

- The incomplete sampling of the OctoMap may cause to not perceive the presence of small obstacles: it was the case for the green cube, for instance (see section 3-1). In some cases, its small dimensions (6 cm edge) constituted it to not be sampled, lying in the space between sampling points (which is between 6,6 and 7,5 cm). On the other hand, thanks to its complete sampling of the scene, the grid sampling had full information even in the presence of small objects.
- The spiral sampling led to an information not aware of the size and/or position of the obstacles. Figure 5-1 shows an example. Again, the lack of complete sampling caused these situations, where the space between neighbouring points was unknown. The grid sampling did not leave any unknown information between sampling points, avoiding these inconvenient situations.
- The incomplete sampling caused problems in detecting the size of the target object. For the same reason as above (missing information between sampling points), the SVM was not capable to generalise to the size of the new targets, proposing grasps on faces larger than the gripper opening. This problem was especially evident in the cereal box, where top-parallel grasp were often proposed (Figure 5-2). The use of the AABB dimensions as features was not sufficient to overcome this problem. On the other hand, the grid over-representation of the scene led to more conservative choices, always proposing grasps on faces whose dimensions could fit between the gripper fingers.

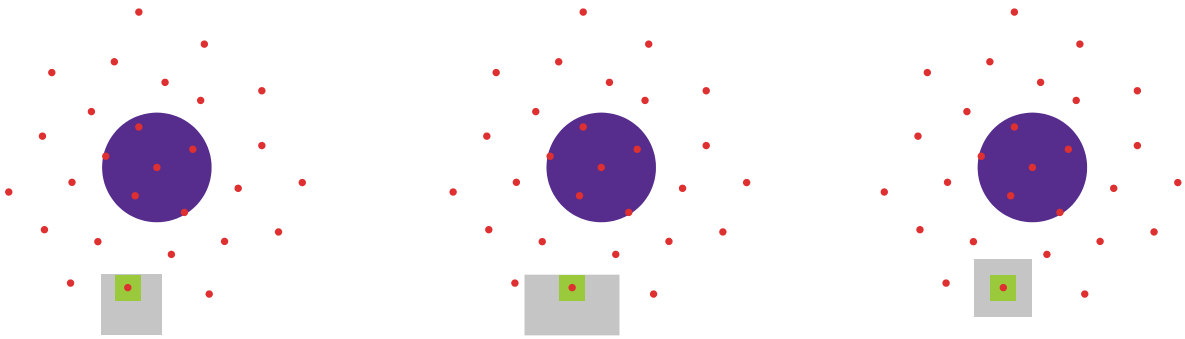
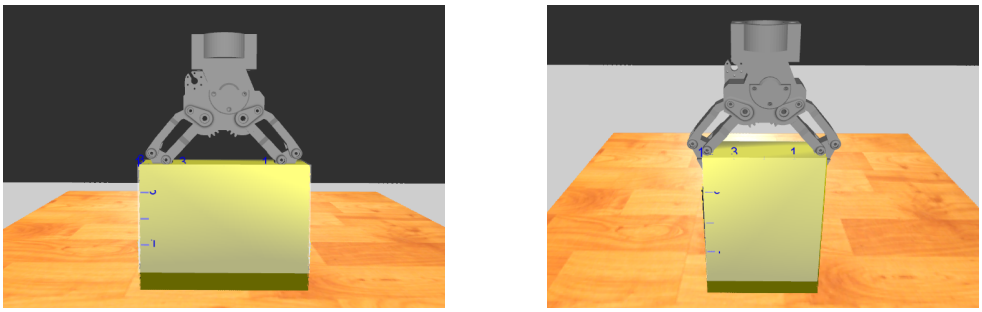


Figure 5-1: Top view of an example of the spiral sampling lack of dimension and position inference. The target object is represented in blue, the obstacle in gray, the sampling points in red, the voxel in green. In the three cases shown in the figure, the returned occupancy information is the same, but the dimension and position of the object consistently vary.



(a) Cereal box (y-axis). **(b)** Cereal box (z-axis).

Figure 5-2: Top-parallel grasp on unfeasible faces of the AABB.

5-2-2 The cascade combinations perform better than the single in specific scenarios

As shown both in simulation and real world experiments, in out-of-bias need line selection scenarios, the cascade implementation returned safer positions than the single: the single could select the correct face, but not the out-of-bias line selection. This failure is caused by the classes distribution for the single SVM implementation (Figure 4-2a): such distribution led to a high bias towards central lines selections, which could hardly be overcome by an increased SVM model variance. On the other hand, the cascade implementation has the second half solely dedicated to the line selection, and hence more capable to vary the line selection.
This difference did not emerge in the spiral combinations due to the inability to quantify the dimensions of obstacles (subsection 5-2-1).

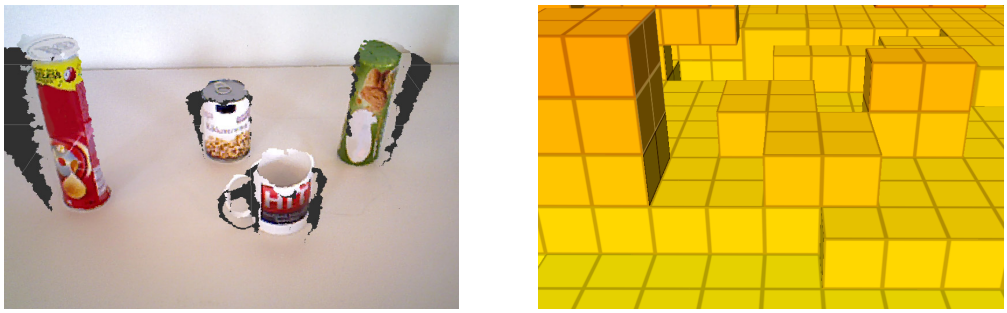
As mentioned, such extremely specific cases rarely occurred in the testing scenarios, due to the random generation of the obstacles distributions. In addition, the dimension of the target has to be large enough for a different line selection to make a difference in the safety of the grasping position. In the simulation testing set, the known targets were on average bigger than the novel targets, and hence a larger behavioural difference was noticed with the known target object shapes, where the cascade grid returned better results with respect to the single.

5-2-3 Real world results are comparable with simulation results

This result is mainly generated by two factors:

- The OctoMap representation proves to be robust against noise, and leads to representations which are remarkably similar to those presented in the simulation environment. Figure 5-3 offers a visualisation. As noticeable, the OctoMap is adherent to the real scene: the table is over-represented but not sampled, being the centre of the semi-sphere located 5 cm above the centre of the AABB. Furthermore, the filtering of the input point cloud aided in maintaining stability of the representation over time (subsection 3-5-2).
- Nevertheless, sometimes some voxels were changing their occupancy information over time. This problem occurred especially at the borders of the objects, where the depth

sensor has accentuated uncertainties, and in distant regions, where the depth sensor degrades its quality. The uncertainties at the objects borders were mitigated by the filtering of the input point cloud (subsection 3-5-2), and the distant regions were out of the sampling volume (as the back wall in Figure 5-3). In fact, these uncertainties led to only two grasping decisions varying over time, in especially cluttered scenarios. This result is an indicator of the robustness of the SVM models, capable of returning a correct decision even in presence of a slightly changed feature map.



(a) Point cloud. **(b)** OctoMap representation.

Figure 5-3: The same real world scene is represented with its point cloud and OctoMap.

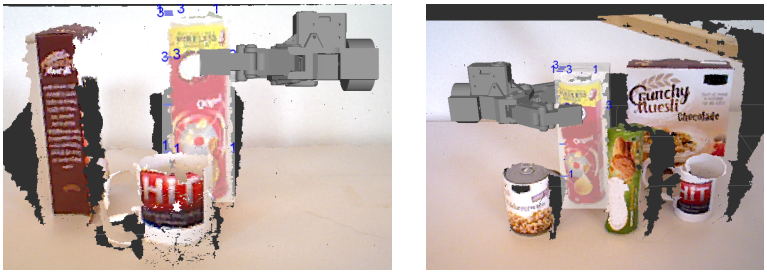
5-2-4 Noise and uncertainties caused variability in two scenarios

The two scenarios are explored in the following paragraphs, in order to analyse how noise affected the decision.

Figure 5-4 shows the first insightful case. Due to the presence of the biscuits pack on the left and the wooden bar on top, the only partially unobstructed face was the right face. Though, the mug was lying in front of it, causing the only feasible and safe grasping position to be on the right face, top line. Indeed, the algorithm returned this selection 4 times. In the other six runs of the algorithm, noise and inaccuracies caused the wooden bar to be over-represented: the space between mug and wooden bar is rather small, and the misrepresentation of just one voxel is enough for the algorithm to judge it too little. Hence, in these cases, the algorithm opted for a left face grasp (since the bar was lying behind the cereal box in that area, being placed diagonally on top of the scene), or even for a top-perpendicular grasp (which is the bias option for this orientation of the cereal box, the selection in which of the algorithm falls back if the scene is over-cluttered).
This situation was peculiar, due to the small height of the target object. With taller objects (as the crisps tube), similar problems did not occur, since the unobstructed space on the chosen face was larger and hence the selection more resistant to OctoMap perturbations. Figure 5-5 shows two insightful examples.

(a) Right face, line 3 (4). **(b)** Left face, line 3 (4). **(c)** Top-$\perp$ face, line 2 (2).

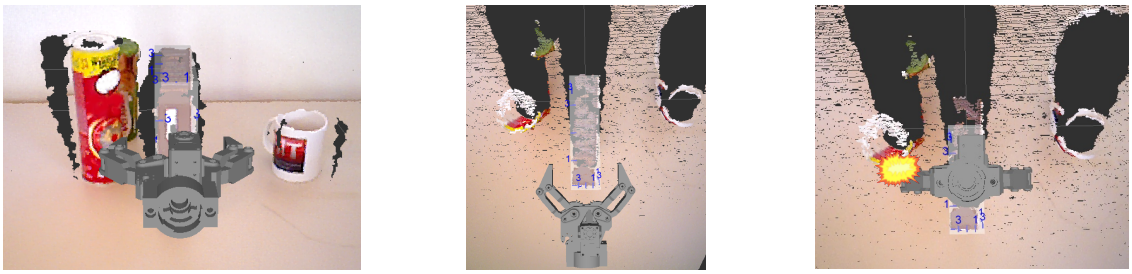
Figure 5-4: A testing scenario where the grasping decision varied over time. In each subfigure it is indicated which face and line were selected, and in parenthesis how many times (out of 10).



(a) Right face, line 3. **(b)** Left face, line 3.

Figure 5-5: Two testing scenarios where the grasping decision did not vary, due to the height of the target object. In each subfigure it is indicated which face and line were selected.

Figure 5-6 shows the second case in which the decision varied over time. The algorithm correctly returned a safe decision on the front face 8 times out of 10. In the other two cases, the dimensions of the AABB on the target object were wrong: the top view in Figure 5-6c clearly shows how the cereal box was represented shorter than its actual dimensions. This problem was not due to the OctoMap representation, but to the object segmentation: the sparsity of cereal box top points led the object segmentation algorithm to believe that it was shorter than it actually was. Hence, due to the shorter dimensions, the algorithm believed that the crisps tube was lying safely behind the cereal box, and then opted for the preferred bias selection on the top face.

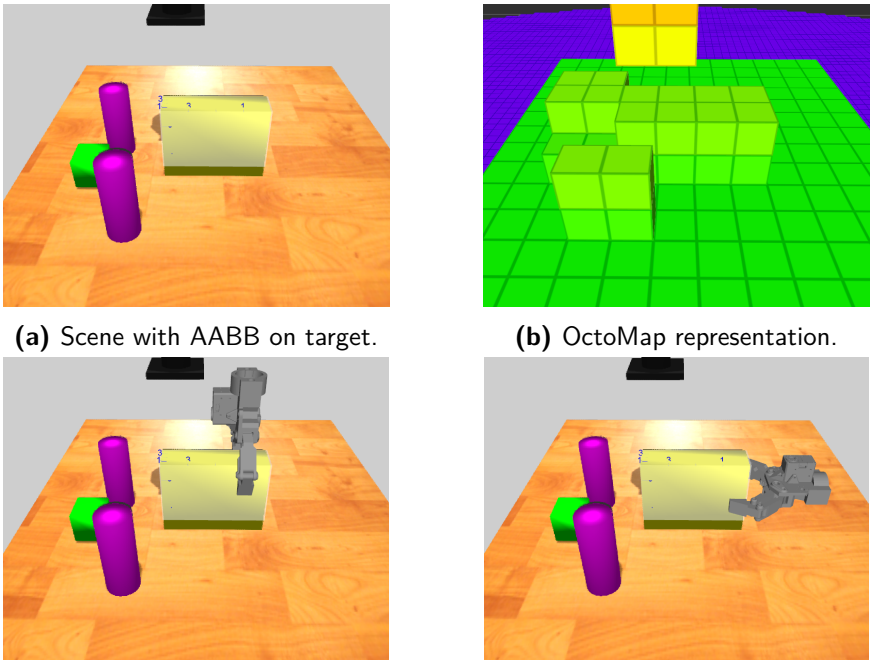


(a) Front face, line 2 (front view, 8). **(b)** Front face, line 2 (top view, 8). **(c)** Top-$\parallel$ face, line 2 (2).

Figure 5-6: A testing scenario where the grasping decision varied over time. In each subfigure it is indicated which face and line were selected, and in parenthesis how many times (out of 10).

5-2-5 Downsides of the grid sampling

A major problem that caused errors in the grid combinations selections was bias. This problem mostly affected the single grid due to the aforementioned difficulty in opting for out-of-bias line selection, but nevertheless affected also the cascade grid. Figure 5-7 shows an example. In the training set, the top-perpendicular was the most commonly selected face on the cereal box (y-axis), hence creating a bias on it. In this case, though, the presence of the lamp constituted the safest decision to be on the right face, as correctly selected by the single grid. Nevertheless, the lamp was only partially occluding the face, not completely: hence, the cascade grid fell in the bias face selection, for then opting for an out-of-bias line selection, which was still not as safe as a right face selection.



(a) Scene with AABB on target.

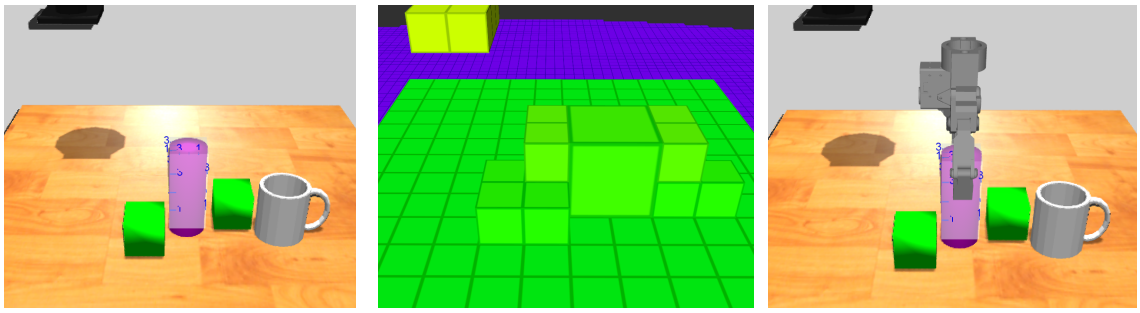
(b) OctoMap representation.

(c) Cascade grid: top-$\perp$ face, line 1.

(d) Single grid: right face, line 2.

Figure 5-7: Example where bias affected the cascade grid combination. At the top, the scene is represented with point cloud and OctoMap. At the bottom, the decisions taken by single and cascade grid are shown.

Furthermore, the grid sampling inherently carries an important flaw: the over-representation of the scene. This aspect can help in leading to more conservative and hence less risky decisions (see subsection 5-2-1), but in some cases it can prevent from detecting feasible grasps. Figure 5-8 shows an example. In this case, the over-representation of the scene guided the SVM to believe that the top-parallel grasp was not feasible, opting for the top-perpendicular face.
In fact, this flaw caused the variations over time presented in the previous subsection, where a single voxel of difference (caused by noise) made a difference in the evaluation of the feasibility of the grasp.



(a) Scene with AABB on target. **(b)** OctoMap representation. **(c)** Grid: top-$\perp$ face, line 2.

Figure 5-8: Example where the low OctoMap resolution over-represented the scene, and related grasping pose choice (for both single and cascade grid). In the central figure, the larger voxel represents a 'parent' voxel, whose 'children' have the same occupancy information.

Chapter 6

Conclusions

In this thesis, a novel autonomous grasping algorithm for domestic cluttered environments, capable of returning an unobstructed grasping pose on novel simple object shapes with the use of a supervised Machine Learning algorithm (a Support Vector Machine (SVM) with Radial Basis Function (RBF) Kernel), is proposed and validated. The main conclusions of this research are:

- The use of the OctoMap sampling can remarkably reduce the amount of features still maintaining a sufficient information on the scene, allowing the use of a supervised Machine Learning algorithm trained with a training set size with a 10 to 10.000 fold decrease with respect to state-of-the-art methods [15, 16, 33, 34].
- The results show that the algorithm returned an unobstructed grasping pose in more than 90% of situations, safely distant from obstacles in more than 80% of cases, both in simulation and real testing scenarios.
- The Axis Aligned Bounding Box (AABB) representation of the target object constitutes the algorithm capable to adapt and generalise to new object shapes and dimensions: in the simulation tests with novel object shapes, the percentage of returned unobstructed grasping poses was still above 90%.
- The OctoMap representation of the scene offers robustness in extending from simulated to real data, as proven by the similarity in results between simulation and real life experiments.
- The comparison between grid and spiral sampling proved a low OctoMap resolution with complete sampling to be preferable with respect to a higher resolution with incomplete sampling, with more than a 20% increase in quality of grasping selections.
- A low OctoMap resolution leads to a coarse representation of the scene, which may constitute some unobstructed grasping poses to not be detected, especially with the use of real noisy data.

6-1 Future Work

The proposed method has shown to be effective, returning an unobstructed grasping pose in more than 90% of situations. Nevertheless, several improvements are proposed:

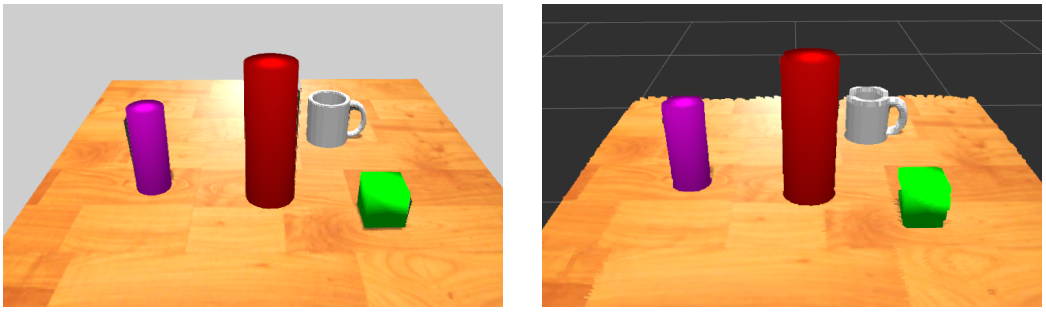
- Involve human teleoperators in the collection of the training set. As mentioned in the Introduction, the robot will be mainly teleoperated in its first operational stage. Hence, the operators' choices can be collected as valuable on-field training examples. Two options are proposed. On one hand, the operators can perform the grasp by means of teleoperation, and their decision will be associated to the most similar grasping pose on the AABB of the target object. Alternatively, the operators can be proposed with the AABB on the target object, decide a grasping pose and let the arm path planner reach it. The latter option can be also proposed to patients, to let them acquire familiarity with the robot. The automation can be later introduced when sufficient confidence in the autonomous decisions is registered.
- Benefit from the human operators' support. In some cases, the environment can be so cluttered than no unobstructed grasping pose is present on the object. In that case, the robot can rely on the assistance of the operators to carefully move impediments. This idea can be translated with a 'face 6'. 5 possible faces on the AABB are available for grasps: if the robot judges all of them to be cluttered, it can select the 6^{th} option, which becomes a request for assistance from the operator.
- Conduct more experiments with a higher OctoMap resolution, to verify how the SVM performs with a similar amount of training set, but increased feature set dimension. Detailed performances comparisons can show a more precise correspondence between OctoMap resolution and amount of training examples needed to achieve desirable results.
- Increase the OctoMap resolution in order to avoid problems of coarse representation of the scene due to the low resolution. To avoid to cubically increase the number of features, the denser space sampling can be processed with the use of Deep Learning techniques [25]. In particular, a 3D convolutional moving filter can be applied [46], with consequent 3D pooling. In this way, a smaller processed feature set can be fed to the SVM.
- Use a multi bounding box approach in order to be able to deal with more complex object shapes, as proposed by Huebner et al. [47]. With this approach, for instance, a mug can be represented by two different bounding boxes, one for the body and one for the handle, leading a more truthful representation of its shape. Following the proposed decomposition of SVM, a further SVM can be added on top of the existing ones, responsible for the box selection. Furthermore, the bounding box can be freed from the bound of the robot's reference frame, and oriented along with the object cluster axes of maximum variance [14].
- Involve arm reachability analysis in the selection of the grasping pose [48], so to propose gripper poses reachable by the robot's arm.

Appendix A

Object detection

The object detection and segmentation is performed online on the data stream recorded by the robot's camera, analogously in simulation and real environment. The computations are performed on the Point Cloud representation of the scene: refer to subsection 1-2-1 for an introduction to Point Cloud and its terminology [19].

Filtering is performed on the original point cloud, reducing its dimension in order to ensure online computations. At first, a passthrough filter is applied: only the points with Euclidean distance between 0,4 m and 1,6 m from the camera frame are maintained. In this way, the floor points are removed, and only the table top points are kept. Then, a downsampling is applied. A 3D voxel grid is generated over the input point cloud data: in each voxel, all the points present are approximated (i.e., *downsampled*) with their geometrical centre. In this project, the downsampling resolution (i.e., the dimension of the voxels) is set to 3 mm. Figure A-1 shows the point cloud obtained after the application of passthrough filter and downsampling.



(a) Original point cloud. **(b)** Filtered point cloud.

Figure A-1: The original point cloud (left) is filtered with the use of passthrough filter and downsampling.

In the filtered point cloud, the main plane (i.e., the table) is located with the use of Random Sample Consensus segmentation (RANSAC, [49]): all the points that support a plane model

are grouped and isolated. In this way, the filtered point cloud is divided into *table* and *nontable* point clusters. In both clusters, outliers are removed using statistical trimming of those points too far from their neighbours [50]. Figure A-2 shows the resulting *nontable* point cluster, composed of the table top objects.

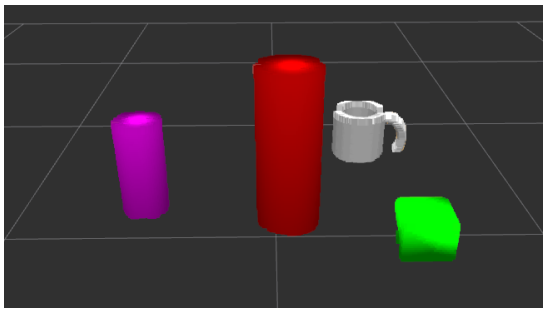


Figure A-2: *nontable* point cloud, obtained after the table segmentation and the outliers removal.

As last step, a single object cluster is selected from the *nontable* point cluster. With this purpose, the *nontable* cluster is separated in several smaller clusters with the use of the Euclidian Cluster Extraction [51]. In particular, points with Euclidean distance smaller than 1,5 cm are considered to be neighbours, and grouped in the same cluster. In this way, the target object point cluster can be manually selected and used as input for the algorithm presented in Chapter 2. In Figure A-3 the red cylinder point cluster is selected and displayed in light blue.

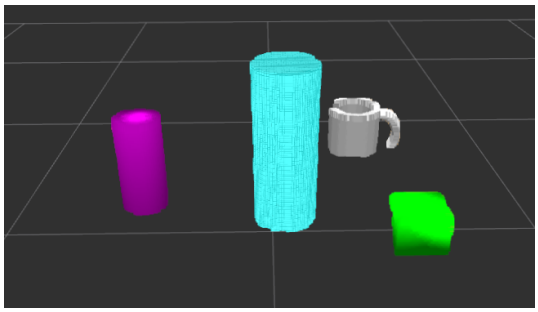


Figure A-3: The red cylinder point cluster is selected and represented in light blue.

Appendix B

Support Vector Machines

A Support Vector Machine (SVM) is a supervised learning model used for classification analysis [23]. Refer to subsection 1-2-3 for an introduction to the terms used in this chapter. Given a set of training examples, each marked as belonging to one or the other of two categories (here indicated as 1 and -1), an SVM training algorithm builds a model that assigns new examples to one category or the other. More precisely, it constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class, since in general the larger the margin the lower the generalisation error of the classifier. Figure B-1 shows an intuitive example.

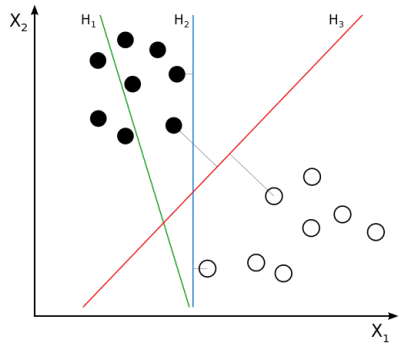


Figure B-1: Division of a 2-dimensional feature set in 2 classes. H1 does not separate the classes. H2 does, but only with a small margin. H3 separates them with the maximum margin.

It is given a training set of m points of the form $(x_1, y_1), \ldots, (x_m, y_m)$, where the y_i are either 1 or -1, indicating the class to which the point x_i belongs, with each $x_i \in \mathbb{R}^n$. The goal is to find the maximum-margin hyperplane that divides the group of points x_i for which $y_i = 1$ from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point x_i from either group is maximized. Figure B-2 shows an example of maximum-margin hyperplane dividing two classes.

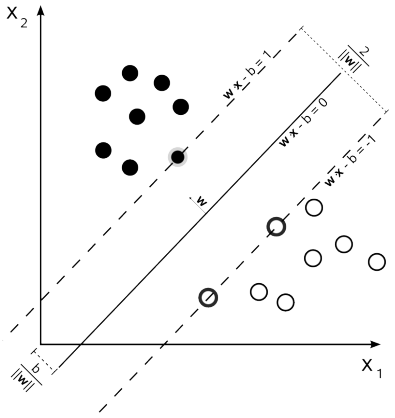


Figure B-2: Division of a 2-dimensional feature set in 2 classes with the maximum-margin plane $w \cdot x - b = 0$.

An hyperplane can be written as the set of points x satisfying:

$$w \cdot x - b = 0 \tag{B-1}$$

where w is the normal vector to the hyperplane. The parameter $\frac{b}{\|w\|}$ determines the offset of the hyperplane from the origin along the normal vector w.

If the training data are linearly separable, two parallel hyperplanes can be selected to separate the two classes of data, so that the distance between them is as large as possible. The region bounded by these two hyperplanes is called the *margin*, and the maximum-margin hyperplane is the hyperplane that lies halfway between them. These hyperplanes can be described by the equations:

$$w \cdot x - b = 1 \qquad \text{and} \qquad w \cdot x - b = -1 \tag{B-2}$$

Geometrically, the distance between these two hyperplanes is $\frac{2}{\|w\|}$, so the maximisation of the distance between the planes becomes a minimisation of $\|w\|$. In order to prevent data points from falling into the margin, the following constraint is added for each (x_i, y_i):

$$w \cdot x_i - b \geq 1, \text{ if } y_i = 1 \qquad \text{or} \qquad w \cdot x_i - b \leq -1, \text{ if } y_i = -1 \tag{B-3}$$

These constraints state that each data point must lie on the correct side of the margin. They can be rewritten as follows:

$$y_i(w \cdot x_i - b) \geq 1, \qquad \text{for } i = 1, \ldots, m \tag{B-4}$$

Hence, the following optimisation problem is introduced:

$$\begin{aligned} &\min_{w} \quad \|w\|^2 \\ &\text{s.t.} \quad y_i(w \cdot x_i - b) \geq 1, \qquad \text{for } i = 1, \ldots, m \end{aligned} \tag{B-5}$$

This is a quadratic optimisation problem subject to linear constraints, hence convex. The w and b that solve the problem determine the classifier. The maximum-margin hyperplane is completely determined by those x_i which lie nearest to it. These x_i are called *support vectors*.

Soft margin classification

To extend SVM to cases in which the data are not linearly separable, the following *hinge loss* function is introduced:

$$\max(0,\ 1 - y_i(w \cdot x_i - b)) \tag{B-6}$$

This function is 0 if the constraint in Equation B-4 is satisfied, in other words, if x_i lies on the correct side of the margin. For data on the wrong side of the margin, the function's value is proportional to the distance from the margin. The optimisation problem in Equation B-5 becomes:

$$\min_w \left[\frac{1}{n} \sum_{i=1}^{m} \max(0,\ 1 - y_i(w \cdot x_i - b)) \right] + \zeta \|w\|^2 \tag{B-7}$$

where the parameter ζ determines the trade-off between increasing the margin size and ensuring that the x_i lie on the correct side of the margin. Its influence on the optimisation problem is comparable to λ (presented in subsection 2-3-1), and hence opposed to C. The hard constraint present in Equation B-5 is here replaced with a soft constraint inside the cost function, which has now become similar to Equation 2-7 in subsection 2-3-1.

Kernel application

The algorithm presented so far performs classification with a linear hyperplane. In order to achieve nonlinear classification, the kernel trick can be employed [52]. In particular, each input vector is transformed to a higher dimensional feature vector through the transformation:

$$\phi : \mathbb{R}^n \to \mathbb{R}^N \tag{B-8}$$

mapping x to $\phi(x)$. The classification function $f(x)$ is then transformed into:

$$f(x) = w \cdot \phi(x) + b \tag{B-9}$$

As demonstrated in [23], it is possible to recast w as follows:

$$w = \sum_{i=1}^{m} y_i \alpha_i x_i \tag{B-10}$$

where $\alpha_{i,\dots,m}$ are positive parameters. With the use of the ϕ transformation, Equation B-10 is transformed into:

$$w = \sum_{i=1}^{m} y_i \alpha_i \phi(x_i) \tag{B-11}$$

Hence, Equation B-9 is recast as follows:

$$f(x) = w \cdot \phi(x) + b = \sum_{i=1}^{m} y_i \alpha_i \phi(x) \cdot \phi(x_i) + b \tag{B-12}$$

The kernel function $K(x, x_i)$ is introduced to replace the dot product between transformation functions, which can be computationally expensive:

$$K(x, x_i) = \phi(x) \cdot \phi(x_i) \tag{B-13}$$

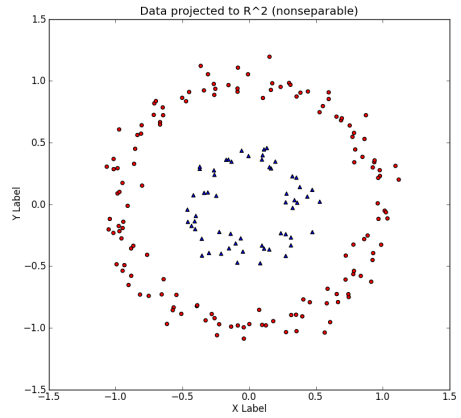
Equation B-12 is recast in:

$$f(x) = w \cdot \phi(x) + b = \sum_{i=1}^{m} y_i \alpha_i K(x, x_i) + b \tag{B-14}$$

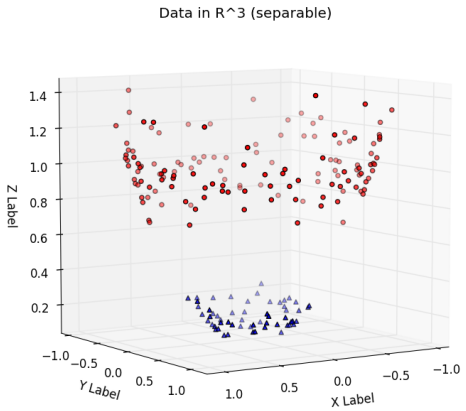
Any K function that satisfies Mercer's theorem [52] guarantees the existence of a ϕ function that meets the condition in Equation B-13. In this way, the mapping ϕ is not explicitly defined, but just the kernel function K is. All the emerging dot products are replaced with a more computationally efficient kernel function. One commonly used kernel function is the Gaussian Radial Basis Function (RBF):

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right) \tag{B-15}$$

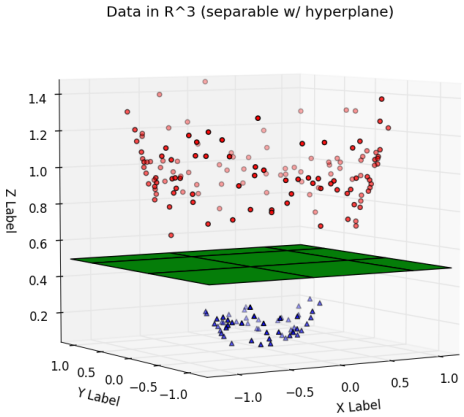
Where the effect of γ is explained in subsection 2-3-1. Figure shows the application of a simple kernel in order to achieve nonlinear separation.



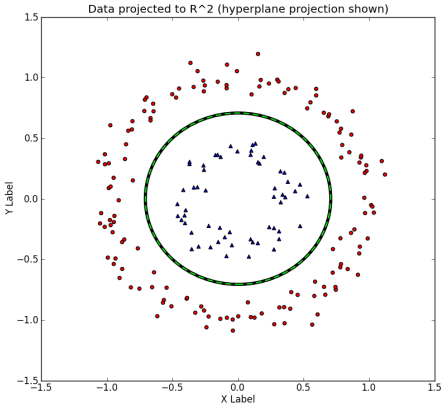
(a) Input data in $\mathbb{R}^2$.



(b) Data mapped in $\mathbb{R}^3$ through ϕ.



(c) Data linearly separated in $\mathbb{R}^3$.



(d) Nonlinear boundary in the original $\mathbb{R}^2$ space.

Figure B-3: SVM with $\phi((x_1, x_2)) = (x_1, x_2, x_1^2 + x_2^2)$ (and hence $\phi : \mathbb{R}^2 \to \mathbb{R}^3$) and thus $K(x, y) = \phi(x) \cdot \phi(y) = x \cdot y + \|x\|^2 \|y\|^2$. The 2-dimensional training points are mapped to a 3-dimensional space where a separating linear hyperplane can be easily found[1].

[1] http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

B-1 Tuning parameters selection

Figure B-4 shows an in-depth overview of the 8 SVM's used in the 4 algorithm combinations, as previously shown in Figure 3-10. Refer to sections 2-3, 3-3 and 4-1 for additional details.

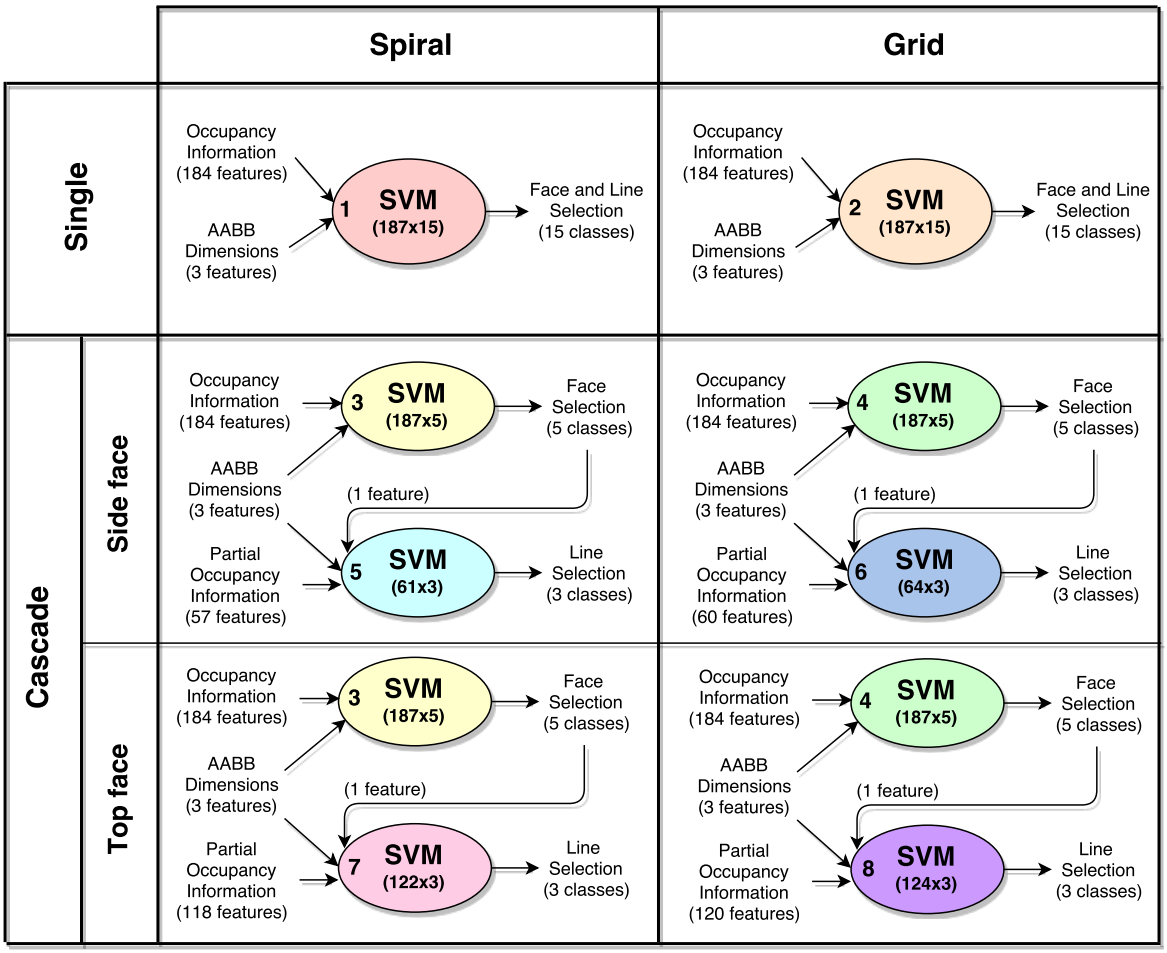


Figure B-4: Overview of the 6 different version of the algorithm. Columns represent different sampling types, rows represent different SVM combinations. Each different SVM is highlighted with a different colour and number, for a total of 8 different SVM's to be trained.

In the following page, a list of the selected parameters couples (C and γ) is offered, in addition to the resulting total amount of support vectors for the specified SVM model, denoted as $sv\#$. Refer to the aforementioned sections, and in particular to subsection 3-3-2 for additional explanations on the tuning procedure.

1. Single spiral, 187x15 SVM, 800 training examples:

$$C = 3,3636 \qquad \gamma = 0,1387 \qquad sv\# = 796$$

2. Single grid, 187x15 SVM, 800 training examples:

$$C = 1,3195 \qquad \gamma = 0,0035 \qquad sv\# = 793$$

3. Cascade spiral - first half, 187x5 SVM, 800 training examples:

$$C = 10,1967 \qquad \gamma = 0,0490 \qquad sv\# = 720$$

4. Cascade grid - first half, 187x5 SVM, 800 training examples:

$$C = 1,2311 \qquad \gamma = 0,0427 \qquad sv\# = 725$$

5. Cascade spiral - second half, 61x3 SVM, 408 training examples:

$$C = 1,7411 \qquad \gamma = 0,2871 \qquad sv\# = 232$$

6. Cascade grid - second half, 64x3 SVM, 408 training examples:

$$C = 174,8532 \qquad \gamma = 0,0229 \qquad sv\# = 151$$

7. Cascade spiral - second half, 122x3 SVM, 392 training examples:

$$C = 50,2134 \qquad \gamma = 0,0156 \qquad sv\# = 229$$

8. Cascade grid - second half, 124x3 SVM, 392 training examples:

$$C = 3,6050 \qquad \gamma = 0,0490 \qquad sv\# = 266$$

Appendix C

ROS Implementation

The algorithm presented in this report is implemented with the use of the ROS framework [44]. Quoting the ROS wiki[1]:

> *"ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management."*

In its simplest form, ROS is a collections of simple agents (named *nodes*) communicating between each others through messages. Each node *subscribes* to specific *topics* in order to receive the desired messages, processes the received data and *publishes* other messages. The online exchange of messages between the nodes allows the robot to operate in real-time. Nodes belonging to the same family are commonly grouped in *packages*. In this project, a package dedicated to autonomous grasping pose selection was created. An overview of the nodes included in the package is presented in Figure C-1.

/segment_table and */cluster_extraction* compose the object segmentation covered in Appendix A, subscribing to the point cloud obtained from the camera and returning the target object point cluster. In the *autonomous_grasping* package, at first the *aabb_computation* node computes and publishes the Axis Aligned Bounding Box (AABB) of the target object, as an Rviz marker in order to be displayed on a screen[2]. The *space_sampling* node subscribes to */aabb_marker* and */octomap* (published by the *move_group* node, core of the MoveIt! software [45]). It prints the needed features (i.e. AABB dimensions and occupancy information, refer to subsection 3-3-2 for additional details) on the *features.txt* file. This file is fed to the Support Vector Machine model (using the C file from LIBSVM [42]), returning the grasping decision in the *decision.txt* file. The latter file is read by the *pose_computation* node, which returns a 6D pose for the gripper. For the sake of visualisation, this pose is used for spawning the gripper marker in Rviz.

[1] http://wiki.ros.org/ROS/Introduction
[2] http://wiki.ros.org/rviz

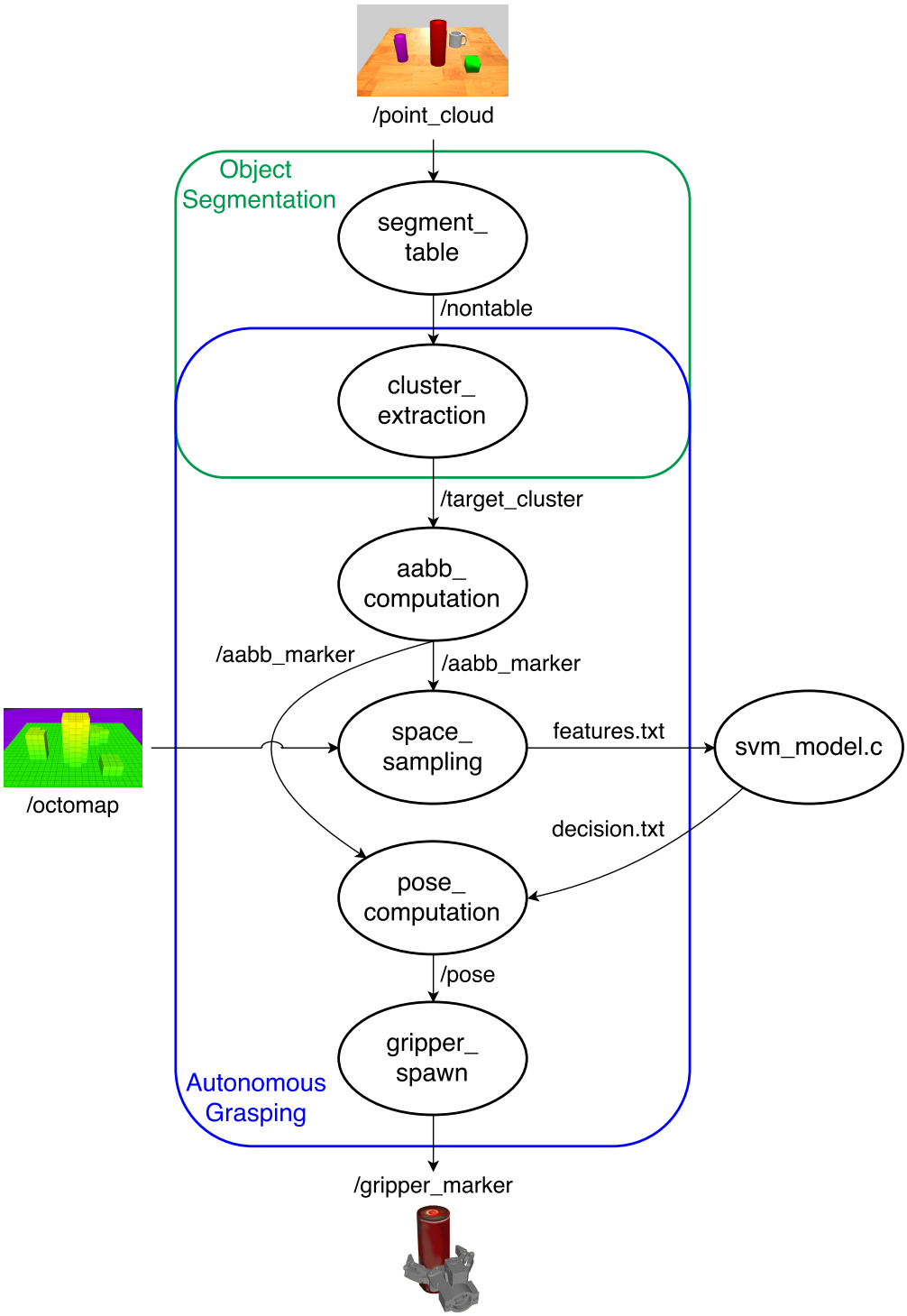


Figure C-1: Graphical overview of the practical implementation. The ROS nodes are contained inside ellipses, the ROS messages are located on the arrows (with the nomenclature */ros_message*). The created *autonomous_grasping* package is highlighted in blue.

Bibliography

[1] "World population prospects: the 2015 revision." Department of Economic and Social Affairs - Population Division, United Nations, 2015.

[2] H. McHugh Pendleton and W. Schultz-Krohn, *Pedretti's Occupational Therapy: Practice Skills for Physical Dysfunction*. Elsevier Health Sciences, 2013.

[3] S. B. Mane and S. Vhanale, "Real Time Obstacle Detection For Mobile Robot Navigation Using Stereo Vision," in *International Conference on Computing, Analytics and Security Trends (CAST)*, pp. 637–642, 2016.

[4] C. Hernandez, M. Bharatheesha, W. Ko, H. Gaiser, J. Tan, K. van Deurzen, M. de Vries, B. Van Mil, J. van Egmond, R. Burger, M. Morariu, J. Ju, X. Gerrmann, R. Ensing, J. Van Frankenhuyzen, and M. Wisse, "Team Delft's Robot Winner of the Amazon Picking Challenge 2016," pp. 1–13, 2016.

[5] M. Johnson, J. M. Bradshaw, P. J. Feltovich, C. M. Jonker, M. B. Van Riemsdijk, and M. Sierhuis, "Coactive Design: Designing Support for Interdependence in Joint Activity," *Journal of Human-Robot Interaction*, vol. 3, no. 1, p. 43, 2014.

[6] A. Saxena, L. Wong, M. Quigley, and A. Y. Ng, "A vision-based system for grasping novel objects in cluttered environments," *Springer Tracts in Advanced Robotics*, vol. 66, no. STAR, pp. 337–348, 2010.

[7] X. Zhang, Y. Yee-Hong, Z. Han, H. Wang, and C. Gao, "Object Class Detection: A Survey," *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, 2013.

[8] A. Saxena, L. L. S. Wong, and A. Y. Ng, "Learning Grasp Strategies with Partial Shape Information," *Aaai*, vol. 3, no. 2, pp. 1491–1494, 2008.

[9] Q. V. Le, D. Kamm, A. F. Kara, and A. Y. Ng, "Learning to grasp objects with multiple contact points," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5062–5069, 2010.

[10] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from RGBD images: Learning using a new rectangle representation," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3304–3311, 2011.

[11] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, vol. 16, pp. 403–412, 2010.

[12] A. D. Dragan, K. C. T. Lee, and S. S. Srinivasa, "Legibility and predictability of robot motion," *ACM/IEEE International Conference on Human-Robot Interaction*, pp. 301–308, 2013.

[13] J. Kober, M. Gienger, and J. J. Steil, "Learning Movement Primitives for Force Interaction Tasks," in *International Conference on Robotics and Automation (ICRA)*, vol. 3, pp. 3192–3199, 2015.

[14] K. Hsiao, S. Chitta, M. Ciocarlie, and E. G. Jones, "Contact-reactive grasping of objects with partial shape information," *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pp. 1228–1235, 2010.

[15] L. Pinto and A. Gupta, "Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours," *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[16] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, pp. 1–16, 2017.

[17] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE Multimedia*, vol. 19, no. 2, pp. 4–10, 2012.

[18] H. Haggag, M. Hossny, D. Filippidis, D. Creighton, S. Nahavandi, and V. Puri, "Measuring Depth Accuracy in RGBD Cameras," in *7th International Conference on Signal Processing and Communication Systems (ICSPCS)*, 2013.

[19] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–4, 2011.

[20] [Online] Available: http://pointclouds.org/documentation/.

[21] J. Wilhelms and A. V. A. N. Gelder, "Octrees for Faster Isosurface Generation," *ACM Transactions on Graphics (TOG)*, vol. 11, no. July, 1992.

[22] T. M. Mitchell, *Machine Learning*. McGraw Hill, 1997.

[23] C. Cortes and V. Vapnik, "Support Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," in *Proceedings of the IEEE*, vol. 86, 1998.

[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. May, pp. 436–444, 2015.

[26] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[27] W. S. Howard and V. Kumar, "On the stability of grasped objects," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 6, pp. 904–917, 1996.

[28] A. Miller, S. Knoop, H. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, pp. 1824–1829, 2003.

[29] Q. Lei and M. Wisse, "Unknown object grasping using force balance exploration on a partial point cloud," *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2015.

[30] V.-D. Nguyen, "Constructing Stable Grasps in 3D," *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, pp. 234–239, 1987.

[31] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3D object grasp synthesis algorithms," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 326–336, 2012.

[32] I. Lenz, H. Lee, and A. Saxena, "Deep Learning for Detecting Robotic Grasps," *Robotics: Science and Systems*, 2013.

[33] A. ten Pas and R. Platt, "Using Geometry to Detect Grasp Poses in 3D Point Clouds," in *Int'l Symposium on Robotics Research (ISRR)*, 2015.

[34] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, and X. Liu, "Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics," *arXiv:1703.09312*, 2017.

[35] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka, "Human-guided grasp measures improve grasp robustness on physical robot," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2294–2301, 2010.

[36] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka, "Physical Human Interactive Guidance: Identifying Grasping Principles From Human-Planned Grasps," in *IEEE Transactions on Robotics*, vol. 28, pp. 899–910, 2012.

[37] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic Grasping of Novel Objects using Vision," *International Journal of Robotics Research*, vol. 27, pp. 157–173, 2008.

[38] R. E. Bellman, *Dynamic programming*. Princeton University Press, 1957.

[39] [Online] Available: http://pointclouds.org/documentation/tutorials/moment_of_inertia.php.

[40] H. Vogel, "A better way to construct the sunflower head," *Mathematical Biosciences*, vol. 189, pp. 179–189, 1979.

[41] R. Caruana and A. Niculescu-Mizil, "An Empirical Comparison of Supervised Learning Algorithms," in *ICML '06 Proceedings of the 23rd international conference on Machine learning*, pp. 161–168, 2006.

[42] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[43] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems. (IROS 2004)*, 2004.

[44] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS : an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[45] Ioan A. Sucan and Sachin Chitta, "MoveIt!", [Online] Available: http://moveit.ros.org.

[46] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 4489–4497, 2015.

[47] K. Huebner, S. Ruthotto, and D. Kragic, "Minimum Volume Bounding Box Decomposition for Shape Approximation in Robot Grasping," *2008 IEEE International Conference on Robotics and Automation*, pp. 1628–1633, 2008.

[48] O. Porges, T. Stouraitis, C. Borst, and M. A. Roa, "Reachability and capability analysis for manipulation tasks," in *ROBOT2013: First Iberian Robotics Conference*, pp. 703–718, Springer, 2014.

[49] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, 1981.

[50] G. H. John, "Robust Decision Trees: Removing Outliers from Databases," *KDD*, no. August, pp. 174–179, 1995.

[51] R. B. Rusu, *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

[52] M. Aizerman, "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and remote control*, vol. 25, pp. 821–837, 1964.

Glossary

List of Acronyms

ADL	Activity of Daily Living
SACRO	Semi-Autonomous Care Robot
HiT	Heemskerk Innovative Technology
DoF	Degree of Freedom
ROS	Robot Operating System
RGB	Red Green Blue
RGBD	Red Green Blue Depth
SVM	Support Vector Machine
CNN	Convolutional Neural Network
AABB	Axis Aligned Bounding Box
RBF	Radial Basis Function
ANN	Artificial Neural Network
RANSAC	Random Sample Consensus

List of Symbols

α	Set of parameters used to recast w in the classification hyperplane
γ	Tuning parameter acting on the variance of the Gaussian function in the Radial Basis Function (RBF) Kernel of a Support Vector Machine (SVM)
λ	Regularisation parameter introduced in the logistic regression cost function
ν	Amount of points generated with the Golden Spiral algorithm
$\phi(x)$	Function used to map to a higher dimension with the Kernel trick in an SVM
σ^2	Variance in a Gaussian function
θ	Set of parameters in a Machine Learning problem
ζ	Soft-margin parameter introduced in the SVM cost function, with effect similar to λ, and hence opposed to C
b	Second parameter of the hyperplane generated by the SVM
C	Regularisation parameter (or soft-margin parameter, depending on the interpretation) introduced in the SVM cost function
c	Centre of the point distribution generated in order to sample the OctoMap
d	Distance between gripper bounding box and closest obstacle
$f(x)$	Classification function used to classify a new input with the boundaries computed by the SVM
$h_\theta(x)$	Hypothesis function of x used for regression or classification in a Machine Learning problem, with parameters θ
$J(\theta)$	Cost function to minimise with respect to θ in a Machine Learning problem
$K(x, x_i)$	Kernel function used for the Kernel trick in SVM
l	Number of lines in which each face of the AABB is subdivided
m	Amount of training examples available in a Machine Learning problem
n	Dimension of input (feature) set in a Machine Learning problem
P	Total amount of points generated by the space sampling method
p	Number of points belonging to a portion for the second part of the cascade SVM implementation
q	Distance between classification boundary and support vectors
s	Resolution of the grid pattern generated with the grid sampling method
w	Parameter of the hyperplane generated by the SVM, vector perpendicular to the hyperplane itself
x	Set of n-dimensional inputs (features) in a Machine Learning problem
x^j	The j^{th} parameter of the n-dimensional input in a Machine Learning problem $(j = 1 : n)$
x_i	The i^{th} training example n-dimensional input in a Machine Learning problem $(i = 1 : m)$
y	Set of outputs (labels) in a Machine Learning problem
y_i	The i^{th} training example output in a Machine Learning problem $(i = 1 : m)$