ENSTA IP PARIS

UNIVERSITY OF BIRMINGHAM

Research Project (PRe)

University of Birmingham, School of Computer Sciences

Academic year : 2020-2021

Robotic Grasping: Improving data quality for Deep Learning of robust grasp poses

---

Confidentiality statement

Non confidential and publishable on the Internet report

Author : Nathan FAGANELLO **School year :** 2022

Academic Tutor (ENSTA Paris) : Gianni FRANCHI

Supervisor : Mohan SRIDHARAN

Internship carried out from 24/05/2021 **to** 06/08/2021

Host organization : University of Birmingham, School of Computer Sciences
Edgbaston, B152TT, Birmingham, Great Britain

Acknowledgements

During this research project, many people helped me and participated, in one way or another, in my work by supporting me, taking an interest in what I was doing or helping me.

First of all, I would like to thank Mr Mohan SRIDHARAN, my supervisor, for allowing me to do my internship in his team.

I would also like to thank especially Martin RUDORFER, researcher in Mr. SRIDHARAN's team, who accompanied me during most of my internship. I really appreciated his kindness, his availability and the time he gave me. He was very attentive and knew how to give me the right advice, the right lines of thought to always try to go further in the project.

Finally, I think it is very important to thank my tutor at ENSTA, Mr. Gianni FRANCHI, who, during the first difficult weeks of this project, took the time to help me and gave me precious advice.

Résumé

La saisie d'objets par des robots autonomes est toujours une problématique importante en robotique. De nombreuses méthodes ont été tentées mais aujourd'hui, l'apprentissage, comme dans tous les domaine, est le plus populaire et suscite le plus de recherches.

Afin d'entraîner tous ces algorithmes d'apprentissage, il est nécessaire d'avoir des jeux de données à plusieurs milliers d'entrées, présentant de la diversité et labellisés de manière cohérente avec la problématique donnée. Dans le cadre de cette étude, l'objectif est de créer un jeu de données permettant de d'entraîner des algorithmes de prédiction de saisies robustes.

Pour ce faire, j'ai travaillé sur la création aléatoire de scènes comprenant des objets placés aléatoirement ainsi que sur la création de position de saisies aléatoirement ou non. Egalement, j'ai mis en place des métriques d'analyse de saisies permettant de mesurer la qualité d'une saisie selon le critère de robustesse.

Finalement, je propose un outil de mise en place et de simulation de scènes et de saisies d'objets ainsi que des outils d'analyse de saisies. Malheureusement, ce dernier point n'est pas parfaitement au point et requiert encore des recherches pour qu'il soit parfaitement opérationnel.

Mots-clés

Deep learning, saisie, robotique

Abstract

The grasping of objects by autonomous robots is still an important issue in robotics. Many methods have been tried, but today, learning, as in all fields, is the most popular and generates the most research.

In order to train all these learning algorithms, it is necessary to have datasets with several thousand grasps, presenting diversity and labelled in a way consistent with the given problem. In the context of this study, the objective is to create a dataset allowing to train robust grasp prediction algorithms.

To do this, I worked on the random creation of scenes including randomly placed objects as well as on the creation of random or non-random grasp positions. I also set up metrics for the analysis of grasps in order to measure their quality according to the robustness criterion.

Finally, I propose a tool for setting up and simulating scenes and object grasp as well as grasp analysis tools. Unfortunately, this last point is not perfectly developed and still requires research so that it is perfectly operational.

Keywords

Deep learning, grasping, robotic

Contents

List of Figures

Chapter 1

Introduction

In the framework of my studies at ENSTA Paris, I have the opportunity to realize a research internship. As a computer science student, my project is to study more precisely A.I. and computer vision. In that context, I contacted M. SRIDHARAN, reader at the School of Computer Sciences, University of Birmingham, who accepted to take me as an intersh in his team. The internship lasted 11 weeks and was carried out completely remotely. I was supervised by M.SRIDHARAN, my internship supervisor and one of the researcher in his team, M.RUDORFER. To manage the remote course of the internship, we had weekly meetings, every Friday, and sometimes, when needed, others calls in the week. The rest of the time, we exchanged by mails.

M.SRIDHARAN made me work on M.RUDORFER's project : building a toolkit for grasp prediction, consisting of tools to build the data and deep learning methods. As an intern coming in the team, in a project which is relatively at its beginning, my mission was :

Setting up the tools for creating the associated datasets and setting up the analysis metrics to label the data

Before diving into the details of what have been done, we will first see, in the first chapter, the general context of the project i.e its important definitions and the background. It will allows the reader to understand what is the purpose, not only of the internship but also of the general project and moreover, it will explain the technical context of it. After that, we will detail step by step all the work. Then, the report follows the chronological order of the different steps but at the same time, follows the order of required stages. That's why we will start by explaining how the scene creation is working. This step consists in placing objects on a plane with different requirements. After that, we will see how to sample grasps for one object in the scene. It implies understanding how to characterise a grasp and then, how to build it and simulate it in a random scene. With these two steps, we will normally have the tools to build the data we need but, finally, we have to label it. The last part will be dedicated to the analysis of the grasps and the score metrics we need to classify grasps. We will finish the report by analysing and discussing the results and suggesting the next steps of the project.

You can find the codes on my github page : https://github.com/Nathan-Faganello/PRe

Chapter 2

Work description

Before more technical explanations of what I did in this project, I think it is important to remind the reader the scientific context in which the work was done. Then, after having given some important definitions for the understanding of this report, we will talk about the scientific context of this project and related works on the topic before finally describe the starting point of my work.

2.1 Preliminary definitions

In order to understand what follows, I think it is necessary to recall some definitions that will be essential:

- **an object pose** : The pose of an object corresponds to its position and orientation in space. Generally, it is characterised by a transformation matrix of $\mathbb{R}^{4\times4}$ which represents the translation and rotation of the object with respect to the origin of the coordinate system. Thus, we find the following structure:

$$\begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix}$$

with R $\in \mathbb{R}^{3\times3}$: a rotation matrix and T $\in \mathbb{R}^{3\times1}$: a translation vector

More specifically, we talk about **grasp pose** when it concerns the gripper pose. In our project, we work on 6 degrees of freedom (6-DOF) grasp poses which means we can translate and rotate the gripper along the three axes of space. The method to construct it will be presented in the in the chapter 4.

- **a mesh** : A mesh is a geometric structure that allows the representation of objects / shapes in 3D as illustrated in the figure 2.1. The most commonly used structure is the Triangle Mesh structure, in which interconnected triangles describe the surface of objects. This data structure is composed of :

- vertices
- faces : sets of three vertices that creates a triangle

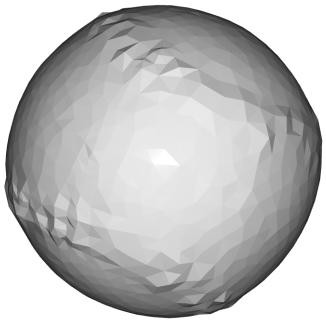

Figure 2.1: Mesh visualisation

- **a scene** : A scene is a set of meshes put together in a single environment. In our toolkit, a scene is characterised by: moving objects, which can be manipulated during simulations, fixed background objects and by a view, i.e. a camera placement which freezes the scene.

- **a gripper** : a gripper is the hand of the robot, i.e. the part intended for gripping/handling objects. In our case, we will use a two-finger model: the robotiq 2f-85 (see figure 2.2 and 2.3)

Figure 2.2: Real gripper 2f-85

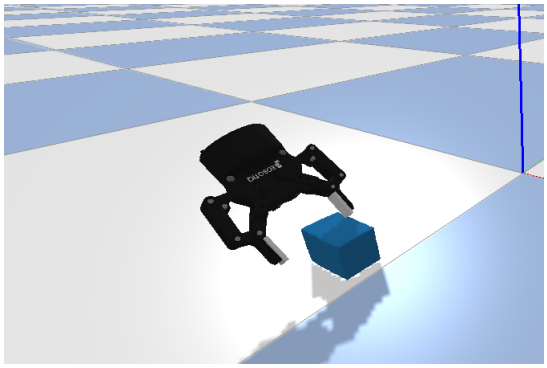

Figure 2.3: Gripper during the simulation

- **a force-closure grasp** : a force-closure grasp is a grasp that can resist any wrench applied to the grasped object. Force-closure is most of the time used as a criterion to evaluate the quality of a grasp.

- **a friction cone** : the friction cone appears in contacts with friction. It is defined by a normal $\vec{n}$ to the point of contact and an angle α, linked to the coefficient of friction μ of the object by the relation: $\mu = tan(\alpha)$. The friction cone is a fundamental notion because, for a given point on an object, it delimits the set of acceptable directions of forces for there to be adhesion (or not) between the two objects. In our case, this represents the set of possible directions at each point of contact, so that the object will grip the gripper and not fall. It is essential to determine is we have a force-closure grasp or not. You can see a simple representation in the figure 2.4.

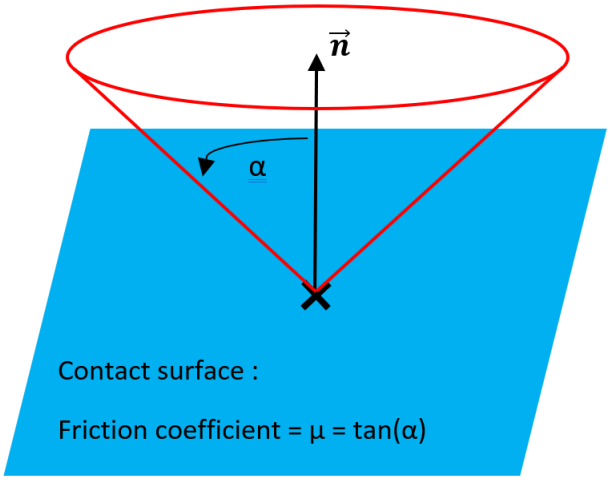

Figure 2.4: Visualisation of a friction cone

2.2 Scientific background

In robotics, grasping objects has been a fundamental problem in the discipline since its inception and still is today. Although it is natural and unconscious in humans, determining how to grasp a new object is a real challenge for a robot because of the complexity of the mathematics behind and because of the imprecision of the robot sensors and actuators. A lot of research is dedicated to the development of autonomous robots in unknown and dynamic environments as the applications are numerous, in different fields such as industry, for robots moving objects in factories, for robots intervening in risky areas and in the rescue of humans or even for household robots.

Robotic grasp planning is a subject that brings together two different disciplines: robotics and computer vision. A grasp consists of several steps : approach the gripper, grasp the object and lift it. Assuming the approach motion and the lifting are simple, the main challenge of this topic is to predict an effective grasp pose to optimally position the gripper despite placement errors to perform a given task (here lifting an object) from an image or a point cloud. Several approaches exist to reach this objective, depending on whether or not the grasped object model is available or not.

In the first case, the first step is to find the best grasp poses for objects in an offline-stage based on analytic metrics and simulation. To be effective, it requires a very large dataset or a very specific object set to work on. But the main problem is to determine the object model. Many methods exists, from very analytic to learning methods, described and compared by Y.Sahillioğlu [15]. This is a very complicated problem because, as simple it is for humans, understanding similarities in the shape structures of the object is not mathematically intuitive. But once you have passed this step, you only have to recognize the object pose and deal with collisions to pick the best grasp in your dataset.

In the second case, you don't try to know which object you are grasping but only try to find a suitable grasp pose. Having this lack of information is penalizing as you can't use analytic metrics. Then, Deep learning methods with very large datasets have become very interesting and promising as machine learning methods have improved a lot the last few years and they only require annotated grasps as training data. Kilian KLEEBERGER et al. [7] have referenced, described and compared a lot of them. Currently, many teams are working on this topic. Most of them intent to build very large-scale annotated grasp datasets and a associated network.

The first important part is learning models. A reference in robotic grasp planning is the Dex-Net programm [11]. They have set up a Grasp Quality-CNN (GQ-CNN) which predicts a robustness score (or a probability of success under uncertainty) from a grasp pose and a depth image of the scene. The idea was to learn a robustness function over many grasps, objects and points of view from a basic binary metric as "lifting success or not" or "the grasp is in force closure or not". They have reached very satisfying result with 80% success rate on their predictions on novel objects. In continuity, C.WU et al. [19] have proposed a new network, called Grasp Proposal Network (GPNet) which returns a list of different successfull grasps for an object placed on a plane from an point cloud of the scene. Their solution is innovative as they introduce a new feature to create new grasps : "anchors of grasp centers". Once again, they have succeeded in improving grasp prediction and have outperformed DexNet 2.0. Many other works are interesting such as the PointNetGPD [8] which improves the point cloud analysis to better predict grasp or the work of Y.Zhou et al. [20] which has generalized learning to non-parallel jaw gripper.

The second important part is the creation of enough data, properly labelled and properly formatted. All (or almost all) the teams who are working on grasp planning with deep learning methods put a lot of energy into building large, coherent and diversified datasets. It represents one of the biggest part of the work as results of learning algorithms can be drastically different with different datasets. However, gathering all this information from the real world is very time consuming and tedious. To save time, researchers turned to simulators. They allow large quantities of data to be generated much more quickly while keeping control on the properties of the various scenes : object positions, friction coefficients, etc. A lot of research is dedicated to the the sim-to-real gap. Indeed, results learned from a simulated environnement do not directly transfer and work in a real one. It is still beneficial to proceed like that. Different datasets are well-know for scenes creations and grasp planning. In our projet, we use the ShapeNetSem [16] and the YCB [2] object datasets for scene creations. They contain a lot of different object models with key properties. An inspiring dataset for learning grasp prediction methods is ACRONYM [5]. They have computed millions of grasps on different objects and in different contexts (cluttered scenes, ...). They have shown that, with their datasets, learning algorithms were performing better and generalizing better to unseen objects.

2.3 Development of the "Burg-Toolkit

M.Rudorfer, a researcher in the team of M.SRIDHARAN, my internship supervisor, is currently working on the development of a toolkit for dataset creation. It will allow the creation, the simulation and evaluation of thousands grasps in more or less complex scenes. Then, it will be used for deep-learning methods as discussed in the above section. In the framework of my internship, I helped in the design of the toolkit by participating in the implementation of different modules that we will detail as the report progresses.

The starting point of my work was a simulator, capable of performing a series of grasps for a single object. Many tools had already been put in place, notably to generate valid grasps and to manipulate the different types of data available to us (point clouds, meshes, quaternions, etc.).

For a more detailed description, I worked on Windows and everything was coded in Python. The simulation part was run with PyBullet. PyBullet is simulation module dedicated to robotics. It allows us to represents objects, robots, grippers as we need, control each joints etc. For our work, the simulation environment was realistic, with gravity.

Chapter 3

Creation of random scenes

The first step to create a grasp planning dataset is to build realistic scenes in which we will simulate the grasps. In our case, we wanted to build simple realistic scenes : objects placed on a plane in stable poses and without any collisions. As explained in the previous chapter, this is a very important step because it has to build a big diversity so that deep learning methods can have better results.

In this chapter, we will see how objects are randomly placed in the scene and how collisions are managed.

The code for that part is in the file *./burg − toolkit/scripts/scene_creation.py*

3.1 Random positions and rotations of the objets

In order to create as much diversity in the dataset as possible, it is necessary to place objects randomly in space but also in different orientations. For this purpose, the implementation of several tools is necessary. First of all, it is very important to find stable poses for the objects. Without this, they may move during initialisation and create significant uncertainties when evaluating the grasps.

A pose is stable if, when the object is released, it does not move. To determine such a position, one needs to know certain physical properties of the object such as its shape (the mesh) and the distribution of mass in the object (which we assume to be uniform in our project). An ObjectType class was created to store all this information. Once this was done, I used the `compute_stable_poses()` function of the trimesh module. From a mesh and information on the distribution of masses (position of the centre of mass in particular), this function returns a list of poses, with for each one a probability of appearance, sorted by decreasing order of probability. At first, I decided to systematically choose the most probable pose for a given object. However, this drastically limited the diversity of the scenes created, as the objects always had the same orientation. To remedy this, I decided to choose the pose of each object according to the probability law defined by the function

`compute_stable_poses()`. Thus, for example, if for a given object, we have 4 poses whose probabilities of appearance are respectively 0.6, 0.2, 0.15 and 0.05, pose 1 will be chosen in 60% of the cases etc.

Then, it is necessary to place the objects randomly in the scene. Randomness can be introduced according to different criteria : x/y position, x/y/z rotation. By computing et selecting randomly stable poses, I didn't need to had more randomness on rotation. The last parameter I had to work on what the placement on the plane i.e. the x,y coordinates. C.EPPNER et al. [5] chose a Gaussian probability model, centred around the origin. In all my tests, I opted for a uniform model, centred around the origin. Each method has its advantages and disadvantages. The Gaussian model allows objects to be more centred and closer together, but requires a longer runtime when increasing the number of objects to find positions where the objects do not collide. One solution would be to vary the standard deviation according to the number of objects. On the other hand, the uniform model allows a shorter search time for positions but sometimes creates very sparse scenes. My solution was to vary the size of the definition intervals to make sure that the objects are relatively close to each other.

I finally obtained a script which (a) allows the user to choose the objects he wants to see in the scene and (b) places the objects randomly in terms of position and rotation. You can see the result on the figure 3.1. At this stage of the process, objects can be placed on top of each others, which is not acceptable. This is why we need collision management.

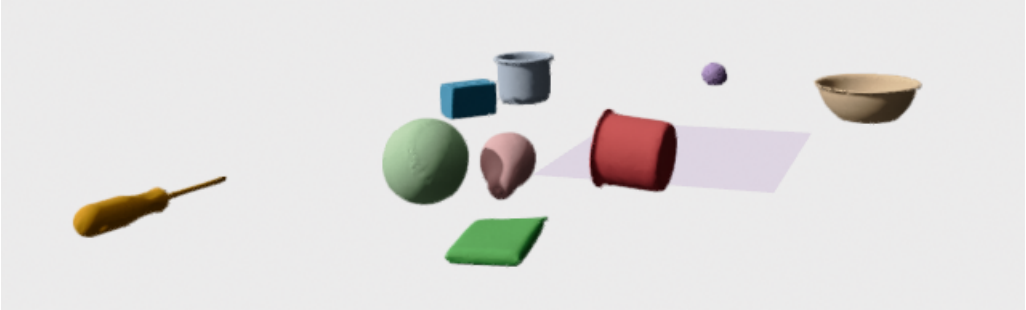

Figure 3.1: Scene exemple

3.2 Collision management

The main constraint is to avoid collisions between meshes in the scene. There is a well-developed module on Linux and MacOS, called Python-fcl, which can handle this problem efficiently. Unfortunately, we were working under Windows so we had to find another solution.

The first idea was to work on the bounding boxes of meshes. The bounding box of an object is the smallest "box" (usually a rectangle in 2D, cuboid in 3D) containing the object inside. More specifically, I tried to exploit the AxisAlignedBoundingBox (aabb) class of the open3D module. Their particularity is that each face of the box is orthogonal to one of the vectors of the base. In the case of an orthogonal base, this is particularly interesting since it allows to have the coordinates of its edges / vertices in a very simple way. Thus, working on this class has a certain interest: to represent a volume in a fast and light way in memory.

The idea was as follows: for each axis, test whether the ranges of each object intersect. If they intersect on all three axes, then the two objects can be considered as overlapping.

The idea is quite simple but unfortunately, I couldn't get it to work as shown in the figure 3.2. Indeed, I found many cases of objects overlapping. First, as I noticed it was mostly happening on objects with relatively complex shapes, I thought is was linked to the lack of precision of aabbs : the orthogonality constraint prevents the bounding box from being very faithful to the object. But this idea is not relevant. Actually, objects are by definitions fully contained in their aabbs (even if the aabb is not faithful to the object). Therefore logically, if aabbs of two different objects do not collide, objects don't collide. In fact, this intrinsic inaccuracy should have tended to push objects apart rather than overlap them. The problem is surely coming from my implementation. After many analyses of my code, I haven't found where was the error : the condition tests seemed to be right.

But we cannot accept such errors when placing objects because, when the simulators are set up, this would imply shocks between the objects and therefore displacements of certain objects that would introduce errors that would completely distort the planning of the seizures. Therefore, another solution had to be found. After some research, I noticed that the trimesh module has a function `is_intersecting()`. This function directly tests the tests between two meshes. It is faster and more efficient. It should be noted, however, that in some rare cases, objects have been found to overlap very slightly.

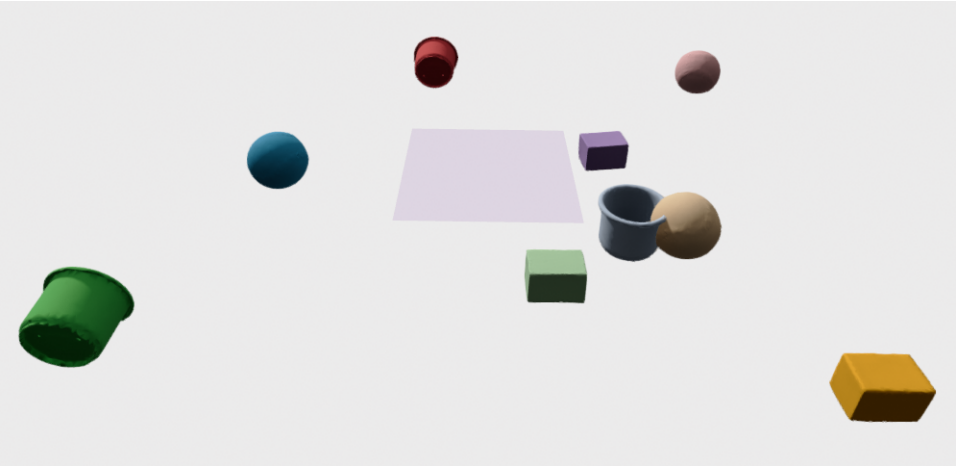

Figure 3.2: Scene with collisions

I'll conclude this chapter with the simulation part. Indeed, at the beginning, only a simulator for a single object was built and only the initialisation of the SceneGraspSimulator was coded. Based on the single object simulator, I modified slightly the way of storing the object data to make it functional : the main differences are that I added two different dictionnaries to store background objects and movable objects and so, modified the initialisation of the simulation environment.

Chapter 4

Grasp sampling

The purpose of the toolkit is, among other things, to provide data for grasp prediction algorithms. Thus, simply placing objects in a scene is not enough: gripper poses must be sampled to be then tested. More precisely, we will talk about "pre-grasp" poses. They define the state of the gripper during the "closing of the finger tips" stage i.e during the execution of the grasp.

The objective of this chapter is to explain how this is done by first explaining the theoretical approach and then, its implementation.

The code for that part is in the file $./burg - toolkit/burg - toolkit/sampling.py$ and $./burg - toolkit/scripts/perturbations.py$. Then, the grasp simulation can be found in $./burg - toolkit/scripts/grasps_simulation.py$

4.1 Antipodal grasps

There are many kinds of grippers with different numbers of fingers on it. Depending on its type, finding grasp poses for a given gripper can become very difficult. Very commonly and in our case, we use parallel-jaw gripper which means gripper with two fingers which will close by translating one to the other to remain parallel as illustrated in the figure 4.1.

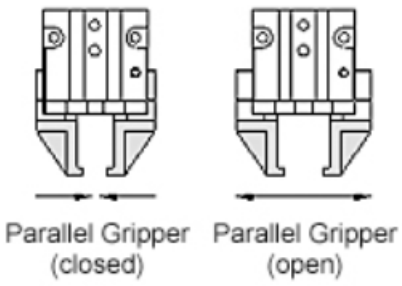

Figure 4.1: Parallel-jaw gripper (from Omega.co.uk)

Once we have decided to work with two-finger gripper, there are different ways of sampling grasps as described by C.EPPNER et al. [4]. But whatever method is used, successfull grasps always have to respect the antipodal constraint (except if some form of form-closure is used (e.g. thread one finger through the handle and let the mug hang from that)) so that they are force-closure grasps.

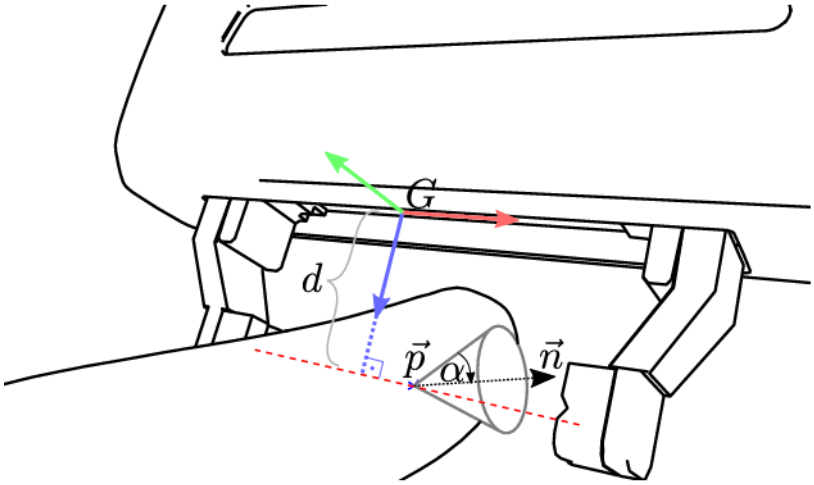

Figure 4.2: Antipodal grasp (de C.EPPNER [4])

The antipodal constraint applied to a pair of points and is defined by V-D. Nguyen [12] as follows :

"A pair of point contacts with friction is antipodal if and only if the line connecting the contact points lies inside both friction cones"

With this first definition in mind, we can easily define an antipodal grasp, illustrated in the figure 4.2, as A.TEN PAS et al. [13]:

"An antipodal grasp is a gripper pose $\in \mathbb{R}^{4\times4}$ *for which the two associated contact points respect the antipodal constraint".*

The antipodal constraint is anecessary and sufficient condition to have effective grasp and has the advantage to be fairly simple and fast to verify. Moreover, it gives a protocol to sample grasp that we will now detail.

4.2 Sampling method

Before starting, it is important to specify that the position of the gripper in our work is defined by the position of the point at the centre of the segment defined by the two finger tips. More

It exists different manners of sampling antipodal grasps. For exemple, R.DIANKOV [3] suggests to cast rays from the bounding box of the object to the object. After this step, you have a set of potential contact points for which you can compute the normals and then, the friction cone which gives you all the posible directions in which you could find a second contact point to form an antipodal grasp. This strategy is quite interesting as it allows him to generalize its sampling methods to grippers other than 2-fingers parallel-jaw grippers.

Our strategy is different. First, we sample thousands of potential first contact points on the object. For each of them, its normal and the associated friction cone (defined by a normal, a contact point and an angle α directly related to the friction coefficient of the object) are computed. Then, we emit rays within the friction cone and the intersections between rays and the mesh gives us potential second contact points. At this stage, it is very important to check that the ray also lies in the second contact point friction to satisfy the antipodal constraint. Once this step is passed, you can properly defined the grasp pose.

First, we define the position of the gripper by computing the center of the segment defined by the two finger tips. Then, we define as "x-axis" the line passing through the two contact points and orientate it so that the z-axis points upward with the right hand rule. Also, to find the final z-axis of the pose (i.e the approach vector of the gripper), we compute the cross-product between the x-axis and the world z-axis (i.e the z-axis of the coordinate system) and then rotate it just enough to keep it pointing upwards. Finally, we obtain the y-axis by doing the cross product between our x-axis and z-axis. At this stage, we have : the coordinates of the center point which allows us to translate our gripper to the right place and three new axes, linked to the final grasp pose which allows us to build the rotation matrix as a coordinate system change.

This technique is very effective in generating a large number of grasps. Indeed, it is very easy, starting from a mesh and its associated point cloud, to select potential first contact points. Once the first point of contact has been selected, it is also relatively simple to generate rays in the friction cone and find the intersection with the mesh. In the context of this project where we have to generate thousands of grasps to create a consistent dataset, this speed is very advantageous.

The antipodal grasp sampling is a feature that was implemented by Mr. Rudorfer before I arrived. I was therefore able to take advantage of it for this first section before having to modify it later on. Indeed, to build the grasp quality metric, I needed information about

normals, approach vectors and contact points. All these are computed during the grasp sampling but were not registered. Then, I modify the code to keep everything in memory.

Once again, we will talk about the simulation part and the adaptation to a scene environment as a conclusion. The gripper control was working well and nothing had to be modified at this level. However, for the SingleObjectGraspSimulator, the object was always placed at the origin of the coordinate system, it wasn't really adapted for a scene. Then, I added gripper translation to place it correctly, at the right spot in the scene so that the grasp is valid.

Chapter 5

Metrics for grasp analysis : definitions and implementation

The simple creation of scenes and grasps is not sufficient for our dataset. It is necessary to put a label on each of the grasps so that they can be used in our learning algorithm.

To choose the right label, it is important to think about how we want to use our data. Our algorithm will need to be able to predict an efficient and most robust grasp. In other words, we want our algorithm to predict a grasp that is capable of performing the desired task (in this case, lifting an object) and that is not sensitive to small errors in gripper placement. This last point is very important: during a simulation, it is possible to place all objects and the gripper at exact coordinates and, if necessary, to know the exact position of any point. However, in the real world, all this is subject to uncertainties. For example, when grasping, it is very likely that the gripper will be placed a few millimetres or even centimetres away from its intended location. The robustness of the grasp ensures that these small changes in input do not result in large changes in output.

With these constraints in mind, we decided to proceed as follows:

1. Generate hundreds of entries for a certain object in the scene, in a random pose.
2. Score each seizure on a scale of 0 to 5 for :
   $collision\_with\_ground = 0$ / $collision\_with\_target = 1$ / $collision\_with\_clutter = 2$ / $no\_contact\_established = 3$ / $slipped\_during\_lifting = 4$ / $success = 5$
3. Retain only successful entries and, for each of them, calculate a robustness measure

The code for that part is in the file $./burg - toolkit/scripts/quality.py$

5.1 Grasp quality measure : the ϵ-metric

There are many ways to measure the quality of an entry. Carlos Rubert et al.[14] have referenced and compared some of them. Each of them is based on a particular quantity linked to the input: input matrix, distribution of contact points, etc. For our project, we have decided to focus on a central measure in input analysis, used in many simulators/predictors such as GraspIt [17]: the ϵ metric

The ϵ metric is a measure that gives an indication of the maximum wrench that a grip can withstand. Let us give the mathematical definition of this metric before detailing and explaining it.

J. WEISZ and P.K.ALLEN [18] put it this way:

For a wrench set C $\subseteq \mathbb{R}^6$:

$$\epsilon_{GWS}(C) = max_{\epsilon}[B(\epsilon) \subseteq convexhull(C)] \tag{5.1}$$

with B(ϵ) the open ball of radius ϵ:

$$B(\epsilon) = [x \in \mathbb{R}^6 / ||x||^2 < \epsilon] \tag{5.2}$$

Also, a force-closure condition is that the origin of the wrench set (i.e the point where wrenches applied, generally the center of masses) is included in the convexhull of the wrench set [10]

First of all, let's give the definitions of each element involved in this definition, which we will find later:

- a wrench C: a wrench C is a vector $\in \mathbb{R}^6$ such as $C = (F_x\ F_y\ F_z\ \tau_x\ \tau_y\ \tau_z)$ with F_i and τ_i the applied forces and torques.

- the convexhull of an object set : the smallest convex set containing the set of objects. Mathematically, this is written :

  Let E be a set of objects. The convexhull of E, denoted Conv(E) is the intersection of all convex sets containing E :

  $Conv(E) = \bigcap_i C_i$ for all C, convex sets containing E

- the Grasp Wrench Space (GWS) : The GWS of a grip represents the total forces that can be exerted by the grip through the points of contact with the object. This GWS will vary according to the different contact models used: contact without friction, with friction, etc. In our case, we will work with contact with friction. The formalisation of this notion is given by C.BORST et al. [1] as follows:

Note the forces applied at the different contact points $\omega = \begin{pmatrix} F_i \\ \tau_i \end{pmatrix}$ with F_i the force and τ_i the associated torque. We can then gather all the forces that can be admitted at a contact point in the "Cone Wrench Space" (CWS):

$$CWS_i = \{w_i \,|(w_i = \omega) \wedge (||F_i - (F_i \cdot n_i)n_i|| \leq -\mu|F_i \cdot n_i| \wedge ||F_i|| \leq -\mu|F_i \cdot n_i|)\} \quad (5.3)$$

Finally, the GWS, the set of forces that can be transmitted by the gripper, is defined as the set of forces that the grasp can transmit through its various contact points:

$$GWS = \{w \mid w = \sum_{i=0}^{n} w_i \wedge w_i \in CWS_i \,\} \quad (5.4)$$

with n, the number of contact points

The ϵ metric measures the quality of an grasp by estimating the "force closure" of the input, i.e. the capacity to resist external efforts. C.Ferrari and J.Canny [6] formalised and developed it, always making the parallel with the computer implementation [6]. To calculate it, the first step is to determine the GWS of the input. In our case, we have two fingers and therefore two contact points, but let's generalise.

5.1.1 Method to calculate the ϵ-metric

Let us suppose that we have k fingers i.e. k points of contact. At each point of contact, whose normal n_i will be noted, a force F_i is applied. The grip can only be stable (resist external forces) if the forces F_i are within the cone of friction and therefore respect the constraint (3.3). Once this inequality is verified, we try to discretise the friction cone in order to express the forces F_i as a function of the extremal vectors of the friction cone. Generally, it is approximated by a pyramid with N sides as in figure 5.1.

Thus, we can express the forces F_i in the friction cones as linear combinations of the cone boundary vectors. We then obtain, for each force :

$$F_i = \sum_{i=0}^{N} \alpha_{i,j} f_{i,j} \quad , \quad \alpha_{i,j} \geq 0 \quad (5.5)$$

We can then re-express each ω effort as :

$$\omega_i = \sum_{i=0}^{N} \alpha_{i,j} \omega_{i,j} \quad , \quad \alpha_{i,j} \geq 0 \quad (5.6)$$

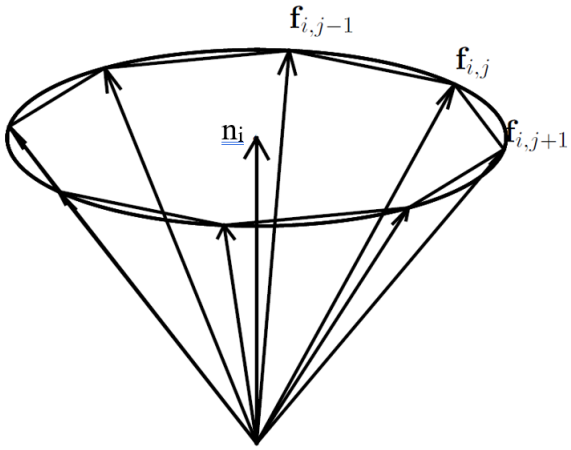

Figure 5.1: approximated friction cone (de S.Liu [9])

The grasp matrix G can then be defined by the equation :

$$\omega = G \times x \tag{5.7}$$

with ω containing all the wrenches ω_i et x containing all the $\alpha_{i,j}$

We then calculate the convexhull of G from which we can test if the origin is included is the convexhull and apply the formula (3.1) to find the radius of the largest ball that fits in. Ferrari and Canny gives the interpretation of this formula as "the magnitude of the minimum norm wrench that will break the object free of the grasp". Then, the bigger is the ϵ-metric, the better it is.

Encountered problems : Computing the convexhull of a set of objects is a difficult task and is still a research subject of computational geometry. To compute it, I used an already coded function of the module scipy.spatial : `ConvexHull()`. After the implementation of the whole above method, I faced a compilation problem returned by the ConvexHull method : a lot of information was returned and, among it, errors on input dimensions and geometry problems. More precisely, it advertised me that maybe one of my inputs didn't have the same number of dimensions and that the initial simplex was flat ("Qhull precision errors"). After several checks, I didn't find any problems concerning the dimensions of the wrench vector. The second error is more difficult to debug. However, Scipy suggests different modes of execution to try to solve different problem. With the suggested "Qj" option, which joggles a bit the input i.e "adds a random number in $[-n, n]$ for each dimension, n being at most $0.01 \times maximum\_width\ of\ the\ input$, it compiles and gives me result that we will analyze in the chapter 6. However, this solution is debatable. Indeed, that method introduces imprecision to the hundredth/thousandth in the input and so, in the results. To me, it is acceptable for what it is used for. As we will see in the next section, the results are used to calculate a probability and we are only interested in whether they exceed a given threshold. Even if, in a very few cases, it can disturb the results, in the vast majority of the grasp, it won't change the final outcome of the program.

5.1.2 Improvement of the ϵ-metric : the probability of force closure

The ϵ-metric is quite simple but is still used in a lot of different simulators. However, it's not enough to meet our objectives : measure robustness under uncertainty. To do so, WEISZ et al. [18] suggests a new measure, based on the above one : the probability of force closure.

The idea is to introduce the uncertainty of the gripper pose in the grasp quality measure by computing "noised" grasp pose to see how it reacts. It is a really important measure to transfer simulated data to real world data.

The probability of force closure is mathecally defined as follows :

For a grasp P,

$$\mathbb{P}(P_{fc}) = \mathbb{P}(\epsilon_{GWS} > \delta) \tag{5.8}$$

with δ a choosen threshold.

Weisz et al. has shown that choosing a grasp with this measure leads to a force closure grasp in 19% more cases than choosing a grasp with the ϵ-metric only. This method is quite heavy computationnaly because it requires to compute a large noised grasp set to build the probability. It might not be use for a real time quality measure. However, in our project, we can afford to spend a bit more time to build a effective metric as all the learning methods will depend on it.

To build the probability of force closure, we need a noised grasp set. There are different approaches : moving the object of moving the grasp. One or the other, the important is to introduce a shift between the gripper and the object. Weisz et al. [18] decided to create a uncertainty model based on the object, playing with three parameters : the position coordinates x and y and the rotation angle along the z axis, θ. This model is quite simple but requires to modify for each grasp the position of the object. You can keep in memory all the scenes with the different positions of the object or do one test, move the object and start again. Both of these options are heavy and on top of that don't really fit with our method. Then, we decided to directly sample noised grasp.

In order to do that, I decided to re-use the sampling method implemented by M.Rudorfer to adapt it to my problem.

The initial method follows the antipodal sampling scheme we explained in the part 2.2.1. Basically, it first samples thousands of potential first contact points. For each contact points, rays are sent within the friction cone and intersections with the mesh are registered as second contact points. After some filters, grasp poses are computed by choosing an orientation.

The main difference between a basic grasp sample and a noised grasp sample is that we have a reference grasp pose in the second one i.e we have two contact points and an orientation to take into account. So we need to partially remove the random parts of the algorithm to only create grasp poses that looks like the reference one. Let's explain how I did it.

Given a grasp pose G, we have from the first sampling its two contact points, its normal and its finger approach vector. I kept one of the two contact point as a reference. Then, I randomly sampled points on the mesh and, with a basic filter based on the distance between the reference point and the candidates, I only kept the close enough and sample new points

till I have N first contact points. Then, everything is the same until the grasp pose creation. Indeed, we need to have grasp only in a certain "orientation" range. During the first sample, grasp pose rotation matrix always have the same scheme and it's not modified during the all simulation process. So I intended to retrieve the orientation angle by doing the inverse operation on one of the coefficient of the matrix. With this coefficient, I just had to randomly choose orientation in a interval centered around it and I had my noised grasp and so, my noised grasp set.

However, I had weird results that suggested that different problems appeared. First, I realized the sampling method wasn't adapted. Indeed, it is made to create only force closure grasps i.e grasps that have a good ϵ measure. It is a problem because every grasp has a score above the threshold and the probability is always equal to 1, which is not relevant. A possible solution would have been to simulate all the grasps to actually see successfull grasp and failure ones. But it's too heavy and it would slow the whole process down.

To solve that problem, I decided to use a simpler approach. Instead of using the antipodal sampling method to build a noised graspset, I basically inserted random rotations and translations in a certain range to the reference grasp pose, without considering antipodal constraints. It is faster to compute and moreover, it is easier to control how we modify the reference grasp. However, that method poses another problem : finding the information we need to compute the convexhull, i.e the contact points, the normals and the approach vectors. Unfortunately, I had not enough time to think about how to do that and it's still a problem to solve. However, I think this is an fairly simple think to do. From that improvement, I hope we could get more diversified ϵ measures among a grasp set and so, have more usable results.

Moreover, another bad aspect of the first approach is that the random sampling around the reference grasp is not the best approach we can have. The idea would be to sample grasps further and further away from the reference grasp along the different position and rotation parameters until we fall below a certain threshold of the ϵ-metric. Like that, we could build a robustness score by measuring approximately how much the grasp can be disturb.

Chapter 6

Results

In this chapter, we will gather all the results I obtain during the internship, from the scene creation to the metrics result. The idea is to discuss for each part of the project what has been succesfully done and what is not and to suggest ways to improve it.

We will follow the same organization as the report itself, starting with the scene creation, then the grasp sampling and finally, the metrics building.

6.1 Scene creation

The scene creation is quite a successfull part. From an object library, the user can choose which objects he wants to have in the scene and they are placed randomly, in stable poses and without collision on a horizontal plane as you can see in the figure 6.1.

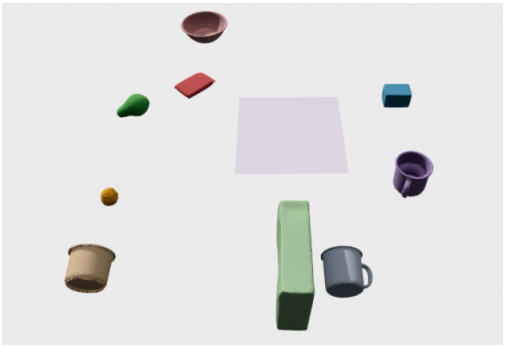

Figure 6.1: Scene exemple

Several improvements could be done :

- Work more with background objects : the first step of the work was to create simple scenes, with objects placed on the ground plane. To build more diversed scenes, it could be interesting to play with background objects as in [5]. I think the best idea would be to place the background objects according to the surface on which we want the movable objects to rest. Then, movable objects would still be placed on the ground plane and the background object would be placed so that it's not colliding with object and one of its surface supports objects. It implies rotate it and moving it along x,y and also z.

- Work more with the camera : for the moment, all the scenes are stored from the same point of view. If we remember that our future deep learning algorithm will take as

inputs images/point clouds, an effective way to increase the number of data is to store different images for the same scenes. Indeed, it will be effective in term of computation time because it won't be necessary to recreate scenes and all the things it implies. You can see on the figure 6.2. what it would look like.

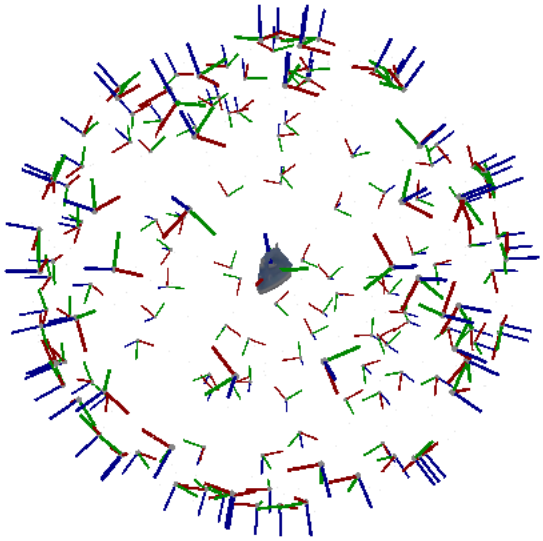

Figure 6.2: Camera points of view sampling

- automate the whole process : Currently, at the begging of each run, the user has to choose which objects he wants to see in the scene and to choose which object he wants to grab. It is quite convenient for the tests to see if everything is going well etc but, when it comes to generate thousands of scenes, we can't loose this time. It should be quite easy to select randomly objects in the library, select randomly an object to grasp in the selected list. It would allow us to gain a lot of time and efforts.

6.2 Grasp Sampling

There are two parts for the grasp sampling. First, the random sampling to build an initial grasp set for an object. Implemented by M.Rudorfer, it's a successfull part and it allows us to build diversed grasp set as shown in the figure 6.3. However, it is important to mention that a majority of the sampled grasps are not relevant because of collisions with the ground, the object or the clutter. The percentage is very fluctuating with each test but, on average, more than 50% of the sampled grasps are unusable.

The second part is the grasp sampling around a reference grasp. The building of a graspset is correct and functional as you can see in the figure 6.4. However, it is still not possible to gather all the associated information we need for the grasp evaluation.

Once again, there are several things to improve :

- Implement the methods to gather all the information needed for the grasp analysis.
- Width adaptation of the gripper : for the moment, the gripper we use has a predetermined width. This is limitating a lot the application because it is often so enough

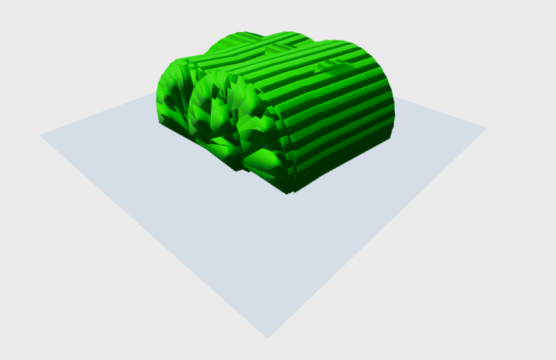

Figure 6.3: Visualization of a grasp set for a foam brick

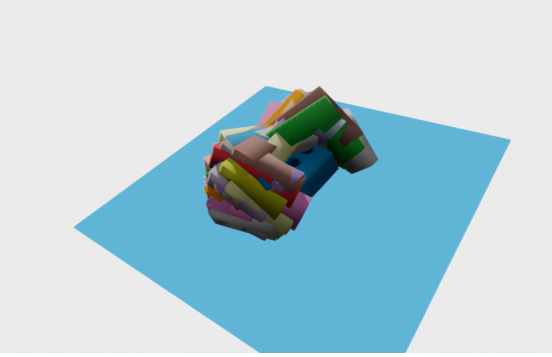

Figure 6.4: Visualization of a noised grasp set for a foam brick

to grasp many objects and more, it is not enough to grasp one object from all the sides. Then, it would be a big improvement if we could adapt the width of the gripper to the length between the contact points for exemple.

6.3 Metrics for grasp analysis

That part of the project has been the most difficult one as it was completely new to me. However, we now have different tools to :

- Compute the ϵ-metric of grasps
- Compute a simple robustness score by calculating the average simulation score of the noised grasp set
- Compute the probability of force closure

It is really difficult to evaluate the quality of the created data without testing it as an input to a machine learning algorithm. However, we can still give some information.

First, for each successful grasp, at the end, the program return a list with : [the grasp, the basic robustness score, the epsilon metric]. It looks like the figure 6.5 .

```
[<burg_toolkit.grasp.Grasp object at 0x000001F491C19310>, 1.6666666666666667, 25.009152957144007]
```

Figure 6.5: Exemple of results

the computation time. Let's analyse the chart given by the figure 6.6, obtained by doing an average, for each number of grasps, on 5 simulations (except for N = 10000 because of the very long simulation time (∼40 min)) :

The grasp sampling method has been done with the parameter $n\_orientations = 10$. This parameter acts during the grasp sampling method. When you have find two contact points to build a grasp, it decided how many grasp you will build from them by rotating the gripper.

We can firstly see that sampling small or huge numbers of grasps doesn't have a real impact on the ratio $successfull\_grasp/simulation\_time$. So, basically, the more you sample grasps, the more you have successfull ones but the longer it is. Actually, it seems quite

Number of grasps sampled	Number of successfull grasps	Simulation time	Successfull grasps per second
10	3	4,79	0,626304802
50	35,33	17,08	2,068501171
100	85,47	57,66	1,482310094
1000	243.52	320,5	0,748829953
10000	5252	2485,79	2,112809208

Figure 6.6: Charts of different results

random. It's probably linked to the fact that, depending on the first contact point you select, you will be able to find more or less successfull grasp. Then it is out of our control. Therefore, the setting of the different parameters should be according to a different criterion than the "brute" efficiency

We should then remember the purpose of our dataset : feeding a machine learning algorithm. I think the most important criteria are : diversity of scenes and grasps and density of grasps per object. The first criterion tells us to select a small number of sampled grasp to have a small simulation time and so, to be able, in a relatively short time, to have a lot of different scenes and grasped objects. The second criterion tells us the opposite : sample a lot of grasps for one object in order to be able to find the very best grasps. As we have to build a big dataset, it's in our interest not to have too much computation time for a data. According to me, the better compromise should be around 500. We can expect to have around 150 successfull grasps for 2 or 3 minutes simulation time. Thinking that we are only sampling grasps from above the objects, objects that are quite small, I think it is enough to have a high grasp density.

The last parameter I evoqued but didn't talk about is "n_orientation". For small number of sampled grasp, having it too big can absolutely break down the performance of the sampling because, if the two contact points are not able produce effective grasp (because of collisions etc). In any case, I think it is interesting to keep it relatively low to force diversity in the selected contact points and so in the grasps. Moreover, as we are grasping only from above, we can rotate the gripper only in $[-\pi, \pi]$, and it's really rare to see successfull grasps throughout the range.

The last thing that can influence computation time is the number of grasp sampled to compute the robustness score and the probability of force closure. For each successfull grasp, we need to sample a number N of grasps, simulate each of them and calculate the ϵ-metric of each one. Once again, it is important to have enough grasps to build a significative probability but we can't afford to spend too much time to do it. So keeping N around 100 / 150 seems reasonable.

grasp scores : it is pretty hard to analyze the grasp scores directly. Especially results from the ϵ-metric. On average, grasps for which we calculate that metric have a score of 24. However, I am not able to know give an analysis of these results. Moreover, it is hard to find a threshold for the probability of force closure if we don't have an idea of what's the score of a good grasp and of a bad one. I think it is because of an error in the inputs of the ϵ-metric algorithm. Indeed, instead of computing the convexhull of G (as defined in the section 5.1.1), I computed the convexhull of W.

The main remaining task is therefore to work again on the grasp matrix to calculate it properly. After that step, I think it will be much easier to compare ϵ scores between them and to use it for ce probability of force closure.

Chapter 7

Discussion

During this project, we have worked on building a complete new dataset for machine learning models. It includes creating new scenes, sampling grasps for objects and evaluating grasps.

First, one interesting difference between our work and many others is the inclusion of gravity in our simulation. Indeed, in the ACRONYM dataset [5], they have build all their grasps and tested it without gravity. It has forced them to create a new test for grasps than lifting the object. They suggest shaking in every direction. They have shown that final grasp results are not different than with the gravity. However, it seems reasonable to consider gravity as the data will be treated as real-world like. The result of the machine learning algorithm trained with the sim data will be transferred to the real world. Logically, it's seems normal to consider gravity .

With that first difference comes the approach vectors. ACRONYM [5] and Y.ZHOU et al. [20] have worked with a sampling along a complete sphere around the object. In our work, we only consider grasps from above the objects. Because we consider objects lying on a ground plane and with gravity, we can't imagine doing anything else. Indeed, it would for exemple imply grasping the object through the ground. Once again, it seems reasonable to create realistic situations to make easier the transfer to real situations. Moreover, it is probably a time saver or a density improvement : with only the grasps from above, on reduce the number of grasps you need to test and so, the computation time or, you can dedicate more grasps for a smaller region and so, with the same simulation time, having the better grasp density.

The scene creation is quite a standard part. For the moment, we have limited it to objects on a simple ground plane and it could be enough. In the ACRONYM model [5], they have worked on background objects so they have more complex and realistic scenes. This is an interesting feature as it allows them to create diversified clutter and more diversity in their dataset. However, it is not a absolutely necessary tool to have. Indeed, the Dex-Net data set [11] is only composed of single objects on a table i.e a simple plane. In their case, it is interesting to see that they have introduced randomness around the object by adding probability models for the friction coefficient and the poses but they always place the object at the same spot. That approach is interesting has it allows them to keep control on what they are doing, being able to choose what kind of distribution model they want to try etc... C.WU et al. [19] have also worked with a single object on a table but their method is very different. To create random scenes, they simply randomly place the object above the table

at a random position and with a random rotation and just drop it. Then, they let physics do its work and wait for the object to find a stable pose. Once it has, they save the image. This method is clearly wilder. It has the advantage of creating a lot of diversity in the object poses and on top of that, to be sure to have realistic poses. However, it is longer to compute as you need to simulate every single drop and to wait the object to find its stable poses.

Finally, we have decided to analyse our grasps with the ϵ-metric and more precisely, the probability of force closure. Actually, the simple ϵ-metric, even if it very common to find it ([11], [17]), has several defaults. As explained by R.DIAKOV in his thesis [3], it is very frequent to find grasps with a very high ϵ score but which are very fragile i.e which are not robust at all. Introducing the probability of force closure as suggested by J.WEISZ [18] was then necessary to be sure to build a quality labelling for the dataset. Indeed, the probability of force closure is made to take into account noise in the model (in our project, in the gripper placement). By calculating it, it ensures that we have a solid grasp, not only in its reference pose but also with small placement errors. Once again, this is a very important feature because, in the real world, noised is introduced (a) by the sensors when acquiring the image of the scene and (b) by the actuators when placing the gripper. It is impossible to control perfectly the placement of the robot and the gripper so, to have efficient grasping, we need to take it as a parameter.

Chapter 8

Timeline of the project

This chapter is made to describe to the reader how the internship was organised. You can find the timeline in the figure 8.1. The organization of the internship actually followed the organization of this report.

The beginning was a bit tough. The first weeks, I worked on articles to immerse myself in the context and the problematic. However, the first articles I read, especially about 3D shape matching, were not very useful for what I did after. I think I lost a bit of time that could have been useful at the end to finish properly my work. After a few weeks, M.SRIDHARAN introduced me to one of the researcher of his team, M.RUDORFER. From that moment, he presented me PyBullet, the burg-toolkit and I started working on the basis to be then able to do the project : getting started with PyBullet and the burg toolkit by trying simple features of the modules etc... After a few weeks learning how to manipulate the different already implemented functions, I really started working on the problematic of the project, starting with the scene creations and then grasp sampling and finally the metrics. It took me approximately 3 weeks before I got something effective and usable for the scene creation as I wasn't really understand what I had to do exactly and because all the modules were new to me. Then, I worked faster for the two last parts because I didn't much time left. Especially for the implementation of the metrics : I managed to do it in a bit more than a week. It was a really short time and it lead me to some of the final errors But, with the weekly meetings and the advice of M.SRIDHARAN and M.RUDORFER, I tried to always go further to achieve the objective of the internship.

I believe that the conduct of the course completely remotely has made the organization a bit more complicated and had slowed down the progression of the intership. Indeed, with only teleconference meetings and mails to talk and discuss about the different results, problems and difficulties, it is not easy to be very effective. Moreover, I faced personal issues which made me loose several days of work.

Figure 8.1: Timeline of the internship

Chapter 9

Conlusion

This internship has been an opportunity for me to participate and to contribute to a large-scale project. As a beginning student in A.I., it taught me many things. First, it introduced me to a new research field I didn't know at all. The intersection between robotics and computer vision was a completely new topic to me and it forced me to adapt and learn to produce something. Learning on its own from reearch articles is not an easy task and requires a lot of rigour and time. We don't face this problem as long as we are students and have classes for everything we need to know. However, it is a big part of being an engineer and it is very rewarding to have become aware of that work. Moreover, as I worked at a foreign university, it allowed me to pratice my english : participate to meeting fully in english, etc . Even if it's not the main purpose of that internship, I think it is important to say that is was a real effort and exercice and it helped me to improve.

Participating to M.RUDORFER's project was a real challenge as I had to understand and to become comfortable with what he already did. At the beginning of the internship, I thought I would do machine learning but I finally worked to prepare that stage. By creating scenes, sampling grasps and evaluating them, I helped M.RUDORFER to give more meaningfulness to their grasp annotations to improve their dataset quality. At the end, we have tools to create data and labelled it even if some improvements have to be done. I hope that will be useful for the next steps of the project.

The internship is now over but there is still a lot to do. First, as I suggested in the above chapters, there are many improvements to be done. It will allow the data annotation to be complete, coherent with the objective and ready for use. After that, and it will probably be a long task, it will be necessary to properly create the dataset, i.e execute thousands of simulations to samples thousands of grasps, on many different objects, in many different poses and save all results. Once that step will be finished, it will be possible to work on deep learning models and train them before open-source the project so it could help and be used by other researchers : to build and train their models or to build autonomous robots.

Bibliography

[1] Christoph BORST. "Grasp planning : How to choose a suitable task wrench space". In: (2004).

[2] Berk CALLI. "Benchmarking in Manipulation Research : the YCB Object and model set and benchmarkinf protocols". In: (2017).

[3] Rosen DIANKOV. "Automated Construction of Robotic Manipulation Programs". In: (2010).

[4] Clemens EPPNER. "A Billion Ways to Grasp : an evaluation of grasp sampling schemes on a dense, physics-based grasp data set". In: (2019).

[5] Clemens EPPNER. "ACRONYM : a large-scale grasp dataset based on simulation". In: (2020).

[6] Carlo FERRARI et John CANNY. "Planning optimal grasps". In: (1992).

[7] Kilian KLEEBERGER. "A survey on learning-based Robotic Grasping". In: (2020).

[8] Hongzhuo LIANG. "PointNetGPD : Detecting Grasp Configurations from point sets". In: (2019).

[9] Shuo LIU. "Partial convex hull algorithms for efficient grasp quality evaluation". In: (2016).

[10] Richard M.MURRAY. "A mathematical Introduction to Robotic Manipulation". In: (1994).

[11] Jeffrey MAHLER. "Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics". In: (2017).

[12] Van-Duc NGUYEN. "Constructing Force-Closure Grasps". In: (1986).

[13] Andres TEN PAS. "Using Geometry to detect grasps in 3D point clouds". In: (2015).

[14] Carlos RUBERT. "Characterisation of Grasp Quality Metrics". In: (2018).

[15] Yusuf SAHILLIOGLU. "Recent advances in shape correspondence". In: (2019).

[16] Manolis SAVVA. "Semantically-Enriched 3D models for common-sense knowledge". In: (2015).

[17] Andrew T.MILLER. "GraspIt! : a versatile simulator for grasp analysis". In: (2004).

[18] Jonathan WEISZ and Peter K. ALLEN. "Pose error robust grasping from Contact Wrench Space metrics". In: (2012).

[19] Chaozheng WU. "Grasp proposal networks : an end-to-end solution for visual learning of robotic grasps". In: ().

[20] Yilun ZHOU. "6DOF Grasp Planning by Optimizing a Deep Learning Scoring Function". In: (2019).