SAPIENZA
UNIVERSITÀ DI ROMA

Human-Guided Grasp Planning

Master Degree in Artificial Intelligence and Robotics

Candidate

Vianello Lorenzo

Thesis Advisor

Prof. Giuseppe Oriolo

Co-Advisor

Dr. Serena Ivaldi

Academic Year 2018/2019

Thesis defended on 25 October 2019
in front of a Board of Examiners composed by:

Prof. Daniele Nardi (chairman)
Prof. Ioannis Chatzigiannakis
Prof. Alessandro De Luca
Prof. Fabrizio D'Amore
Prof. Giuseppe Oriolo

Human-Guided Grasp Planning
Master's thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: vianello.1805889@studenti.uniroma1.it

Abstract

This thesis presents recent works in autonomous grasp planning and human demonstration; moreover it presents an implementation of a new method which can be used to transmit knowledge from a human expert to a grasp planning algorithm. I implemented this method during a six months internship in INRIA (National Institute for Research in Computer Science and Automation) in Nancy (FR).

Autonoumus grasp planning algorithms are usually based on Analytical or Deep Learning approaches, both these methods present limitations. Several attempts have been done to transfer human experiences in such a way to fill some of the gaps in these methods. But human imitation requires a large amount of training making the operator's job expensive.

For this reason we implemented a method for human's knowledge transfer based on some state-of-the-art methods such as: Dex-Net, Mask-RCNN, Variational-AutoEncoder (VAE) and Gaussian Processes (GP).

We tested our method on three experiments with increasing difficulty: grasp classification, grasp evaluation and grasp correction. The results we obtained are promising and show that the proposed method can be used to transfer knowledge with a minimum training set.

Contents

1 Introduction 1

2 HEAP Project 2

3 State of the Art 4
3.1 Grasp planning . . . 4
3.2 Dexnet . . . 6
3.2.1 DexNet 1.0 . . . 6
3.2.2 DexNet 2.0 [2] . . . 9
3.2.3 Other DexNet Algorithms . . . 10
3.3 Human in the loop . . . 10
3.4 Mask-RCNN . . . 12
3.5 Variational Autoencoder . . . 14
3.5.1 Autoencoder (AE) . . . 14
3.5.2 Variational Autoencoders (VAEs) . . . 14
3.6 Gaussian Process . . . 16
3.7 Franka collaborative robot . . . 18

4 Methods 19
4.1 Grasp selection pipeline . . . 19
4.2 Mask-RCNN for object selection . . . 20
4.3 Human demonstration autonomous grasp planning . . . 21
4.4 Variational Autoencoder (VAE) for represent grasps . . . 21
4.4.1 Grasp representation . . . 21
4.4.2 Variational Autoencoder . . . 22
4.4.3 VAE with Intrinsic Classification (VAE-ic) . . . 23
4.5 Gaussian Process for best grasp selection . . . 23
4.5.1 Classification of grasps . . . 24
4.5.2 Regression on human grasp evaluation . . . 24
4.5.3 Regression on human correction . . . 24
4.6 Task execution: . . . 25

5 Experiments 26
5.1 The setup . . . 26
5.1.1 Robot Setup and Initialization . . . 27
5.1.2 The gripper . . . 27
5.1.3 The environment . . . 28
5.1.4 Camera setup and Extrinsic calibration w.r.t. Robot Frame . . . 28
5.2 DexNet . . . 30
5.3 Mask RCNN . . . 31
5.4 Variational Auto-Encoder(VAE) . . . 33

5.5 Gaussian Process (GP) for classification 37
5.6 GP for regression 39
5.6.1 Human grasp evaluation $Q_H(g)$ 39
5.6.2 Human grasp Transformation $T_H(g)$ 42
5.7 Task executions 44

6 Conclusions **46**

Bibliography **47**

Chapter 1

Introduction

Grasp planning is an active research area in robotics. Modern grasp planning methods include analytical and data-driven approaches.

In both cases it is difficult to integrate the human knowledge inside the planner because it is hard to model analytically. It can be modelled, in a data-driven approach, but it requires a huge amount of data. This thesis presents the literature in grasp planning and human teaching by demonstration, it also presents a novel method implemented during my internship to teach to the planner with minimum amount of data. This is the thesis structure:

In Chapter 2 is presented the HEAP project, a european project I worked on during the internship. My work, in fact, will contribute to a project managed by several institutions.

In Chapter 3 is presented the current state-of-the-art in grasp planning, human collaboration and others methods used in the thesis.

In Chapter 4 is presented the proposed method to select the best grasp between several proposed in a heap. The method uses Dex-Net's candidates and human suggestion.

Chapter 5 contains some experiments and the results obtained.

Chapter 2

HEAP Project

HEAP project is a european project inspired by the aim of solving problems in nuclear waste sorting. The aim of the project is to advance grasp planning when the 3D model of the object to grasp is not available and furthermore the object is irregular, articulated, often broken. The researcher has to consider situations in which the items may break during grasping and transportation. The planner can be fully autonomous or teleoperated by humans.

This kind of problem is challenging because it concerns several fields:

- It is necessary to adapt the robot structure to this task (robot design). We will see, for example, how the gripper has been chosen in such a way to reduce the damage on the grasped objects.
- Also the camera must be chosen adequately: some depth sensors have low resolution if they work with sunlight, several cameras cannot work in presence of radiations.
- The object to be grasped must be recognized, segmented and is necessary to estimate the best grasp in such a way to do not break it.
- Some human-in-the-loop and shared control techniques could be integrated, but always considering that the operator cannot undergo radiations.

Figure 2.1. Nuclear waste sorting.

My work does not want to solve all these problems but it is an attempt to integrate the human knowledge in the grasp planner in such a way to teach some object properties.

Chapter 3

State of the Art

3.1 Grasp planning

Most recent grasp planning methods can be classified into analytical and data-driven approaches[5].

Analytical methods use physical properties of the manipulated object (mass, shape, centre of mass, friction coefficient). These information are difficult to obtain with the classical robot sensors (visual or tactile sensors). Anyway, with a partial knowledge of the properties of the object, we can propose a set of grasps and use several metrics to evaluate them [6][18] [19]. Between them the Ferrari Canny [6] metric demonstrated its ability to predict grasp success; for this reason it is used also in the state-of-the-art method Dex-Net [2].

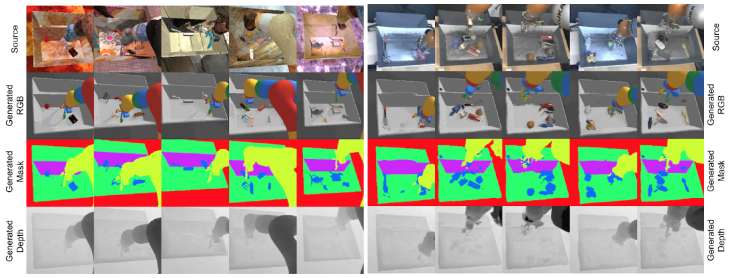
Analytical approaches typically require the task to be hard coded, using expert knowledge and understanding of the environment and robot capabilities. These approaches perform well, but they suffer when presenting cases outside of the scope of the developer's mind. For that reason recent works use more data-driven approaches [5] or techniques both analytical and data-driven [1] [2].

On the other hand, Data driven approaches use manipulation data to (1) sample potential grasps and (2) rank them according to a learned function.

Manipulation data may have different origins: they can be real pre-recorded robot grasps attempts like in Levine et al. [33] or in Kalashnikov et al.[21] where a set of robots tried six thousand grasp attempts to train a MDP.

In other works the training set comes from a simulated environment: this method is preferable because it is less expensive. For example James et al. [32] extended the work of Kalashnikov et al. [21] with a training based only on simulated data.

Figure 3.1. Kalashnikov et al. [21] work uses real examples (580K grasps attemps) to train an MDP model.



(a) Randomized-to-canonical samples. (b) Real-to-canonical samples.

Figure 3.2. James et al. [32] extended the Kalashnikov's work substituting the pre-recorded manipulation data with simulation data. Simulation data are generated using randomization techniques and then adapted to the real domain.

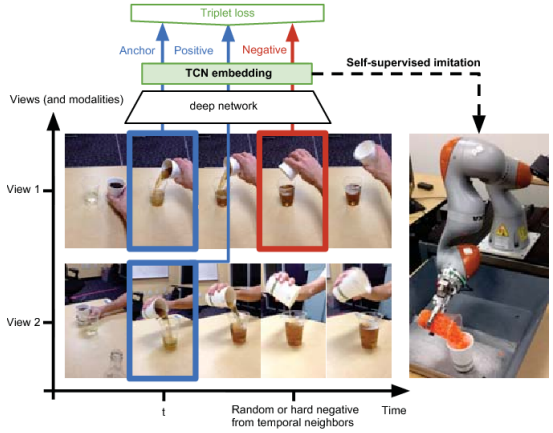


Figure 3.3. In Sermanet et al. [31] from multi-view videos the model encodes the task and then executes it using the robot.

Authors demonstrated that the efficiency of this new solution is comparable to the one trained only on real robot. Some domain adaptation techniques are necessary to transport in a real environment the information acquired in simulation. Other works present similar techniques, like in Mousavian et al.: there it is used a VAE trained purely in simulation to generate 6-DOF grasp poses from 3D point clouds of single objects observed by a depth camera [10].

Manipulation data can also come from human demonstration. Several works have been proposed based on human demonstration's data [15][16][30]. For example in Ficuciello et al. the robot uses a human manipulation dataset to reduce the space of the possible solutions[30]. In the same way in Sermanet et al. [31] the robot is able to understand and execute a task based on multi-view video of human demonstrations. From the results of this paper we can infer that human demonstration can substitute and support other data driven techniques.

So, resuming, analytical grasp planners require more human knowledge with respect to the data-driven approaches; they are strong in the environment for which they have been designed but they have limitations for generic applications. On the other hand data-driven approaches can draw from many sources (real, simulation, human demonstration) but they require a huge amount of data.

Many success-full grasp planners make use of both techniques. An example of this mixed approach is Dex-Net[1, 2, 3, 4], it uses both an analytical method

(to sample and evaluate antipodal grasps using Ferrari-Canny metric [6]) and a data-driven algorithm (to use grasp's datasets to predict the success of future grasps attempts).

In the next section I will present Dex-Net trying to concentrate more on the aspects relevant for our work.

3.2 Dexnet

DexNet is a state-of-the-art method for grasp planning. It is developed by Berkeley AUTOLAB. The authors presented it through a series of paper in which they consequentially explained its components. Given a mesh or a point-cloud, DexNet returns the best grasp. The grasp selection is based on pruning, scoring and ranking a set of antipodal grasps.

3.2.1 DexNet 1.0

Training phase

In this section I will present the Training phase of the algorithm proposed in DexNet 1.0 [1] for grasp selection:

The paper contributes to the formalization of a parallel-jaw grasp parametrization: in fact, the grasp is represented by a centroid x and an approach direction v. Knowing these grasp properties it is possible to define the gripper contact points c and two normals n:

$$c_i = x + (-1)^i(w/2 - t_i)v \qquad n_i = \Delta f(c_i)/\|\Delta f(c_i)\| \tag{3.1}$$

Each one of the parameters presents a degree of uncertainty modeled by a Gaussian $X \sim \mathcal{N}(0, \sigma^2)$.

A grasp is considered Antipodal if the normal vectors are collinear and in opposite direction. So for a mesh o_i we can collect a certain number of grasps sampling in the domain of the grasp parameters (x, v, σ) and then keep only the grasps which are antipodal checking the values of the normal n.

From these grasps are removed also those with a value of the Ferrari-Canny metric [6] under a given threshold.

The Ferrari-Canny metric ranks grasps based on their ability to resist an arbitrary disturbance wrench acting on the object being restrained.

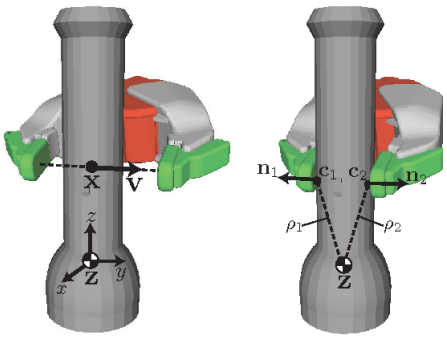


Figure 3.4. Grasp definition.

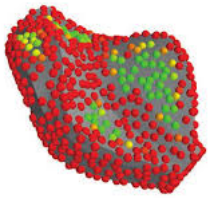


Figure 3.5. In DexNet 1.0 from the set of sampled grasps are removed those which are not antipodal and with Ferrari Canny metric over a given threshold.

This procedure has been iterated on a set of 13,252 meshes (DexNet 1.0 dataset, Fig. 3.6a). The authors used on the meshes some dimensional reduction tools, like PCA, to reduce the computational weight of this passage.

The result of this iterations is a dataset of 2.5 million of antipodal and robust (under the Ferrari-Canny criterium) grasps associated with the relative object of belonging.

At this point the algorithm creates two embedding: one to represent the grasps similarities, one to represent the objects similarities.

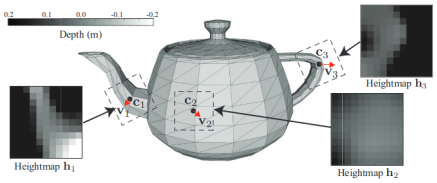
- **Grasp embedding** for encode grasps similarities: Each grasp can be represented by the 2 planes perpendicular to the normals n_i and tangent in the contact points c_i. The height-map of the signed distance function (SDF) of the mesh was used to represent these information (Fig. 3.6b).

  The grasp's features are the gradient d_i on the x and y component of the two height-maps h_i. So a grasp g belonging to and Object O, in the described feature space, can be represented as:

$$\eta(g, O) = (d_{1,x}, d_{1,y}, d_{2,x}, d_{2,y}) \tag{3.2}$$

- **Object embedding**: The authors used a Deep-Learning approach to build an object embedding which encodes objects similarities: Multi-View Convolutional Neural Network (MV-CNNs)(Fig. 3.7), the network takes 50 virtually rendered camera viewpoints discretized around a sphere and passes them in a in a CNN to create an embedding for 3D object models.

(a)



(b)

Figure 3.6. DexNet 1.0 dataset andGrasp height-map representation.

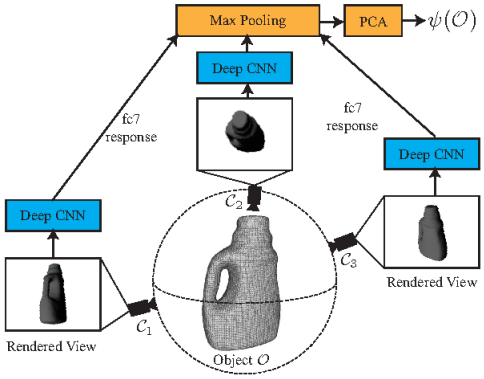


Figure 3.7. MV-CNNs architecture.

Given and object O, its vector representation is $\psi(O) \in R^{100}$. To navigate in this vector space the algorithm uses a KD-Tree structure.

Test phase

When the training phase is concluded, it is possible to test the algorithm on a new object, this is the procedure to extract the best grasp given a mesh:

From the mesh a set of Antipodal and robust (under Ferrari-Canny metric) grasps has been selected.
After that, a belief distribution of the force closure P_F is evaluated using the similarity to the 2.5 million grasps in dexnet 1.0 dataset and using the two embeddings presented before. To do that the Multi Armed Bandit (MAB) algorithm has been used. This belief is updated using Continuous Correlated Beta Processes(CCBPs), the result of this passage is a score for each grasp. The best grasp g_T^* is the one with maximum score. The psedocode explains the test phase of DexNet 1.0 algorithm:

Input: Object $\mathcal{O}$, Number of Candidate Grasps N_g, Number of Nearest Neighbors N_n, Dex-Net 1.0 Database $\mathcal{D}$, Features maps ψ and η, Maximum Iterations T, Prior beta shape α_0, β_0, Lower Bound Confidence p, Random Variables ν, ξ, and γ
Result: Estimate of the grasp with highest P_F, $\hat{\mathbf{g}}^*$
`// Generate candidate grasps and priors`
Γ = AntipodalGraspSample($\mathcal{O}, N_g$) ;
$\mathcal{A}_0 = \varnothing, \mathcal{B}_0 = \varnothing$;
for $\mathbf{g}_k \in \Gamma$ **do**
  `// Equations VI.1 and VI.2`
  $\alpha_{k,0}, \beta_{k,0}$ = ComputePriors($\mathcal{O}, \mathbf{g}_k, \mathcal{D}, N_n, \psi$);
  $\mathcal{A}_0 = \mathcal{A}_0 \cup \{\alpha_{k,0}\}, \mathcal{B}_0 = \mathcal{B}_0 \cup \{\beta_{k,0}\}$;
end
`// Run MAB to Evaluate Grasps`
for $t = 1, .., T$ **do**
  j = ThompsonSample($\mathcal{A}_{t-1}, \mathcal{B}_{t-1}$);
  $\hat{\nu}, \hat{\xi}, \hat{\gamma}$ = SampleRandomVariables(ν, ξ, γ);
  F_j = EvaluateForceClosure($\mathbf{g}_j, \mathcal{O}, \hat{\nu}, \hat{\xi}, \hat{\gamma}$);
  `// Equations VI.3 and VI.4`
  $\mathcal{A}_t, \mathcal{B}_t$ = UpdateBeta(j, F_j, Γ);
  $\mathbf{g}_t^*$ =MaxLowerConfidence($\mathcal{A}_t, \mathcal{B}_t, p$);
end
return $\mathbf{g}_T^*$;

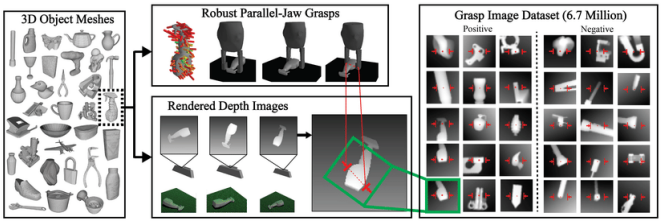


Figure 3.8. DexNet 2.0 pipeline for training Dataset generation.

3.2.2 DexNet 2.0 [2]

This paper extends the precedent work (DexNet 1.0) creating a dataset (DexNet 2.0 dataset) for planar parallel-jaw grasping. The planner has access to a single-view (2.5D) point cloud taken with a depth camera. The main idea is to predict the robustness function $Q \in [0, 1]$ training a network which uses as label E_Q, the robust epsilon quality. From the robustness function we can recover the success of the grasp, in fact the grasp success is:

$$S(u, x) = \begin{cases} 1 & \text{if } E_Q > \delta \text{ and } collfree(u, x) \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

where E_Q is the robust epsilon quality and *collfree(u,x)* indicates that the gripper does not collide with the gripper or the table.

The method is based on two main points: DexNet 2.0 dataset and grasp quality CNN (GQCNN).

- Dataset: the dataset is built using the DexNet 1.0 dataset containing 3D meshes; for each object a set of Robust parallel-jaw grasps has been proposed. Each grasp is evaluated using E_Q (robust epsilon quality).

  The point cloud of each object is then rendered while it is in a stable pose over the table. To each grasp is associated a depth image with center in the grasp position and orientation parallel to the grasp orientation (Fig. 3.8).

- GQCNN: The network uses a set of parameters Θ to represent a grasp robustness function Q_Θ. It takes as input the grasp candidate and processes in a CNN the image (x, y) and then concatenate the distance (z) to the resulting Representation (Fig. 3.9).

The algorithm has been tested both on a set of household object and on a set of adversarial generated objects: in both cases it showed a high success rate.

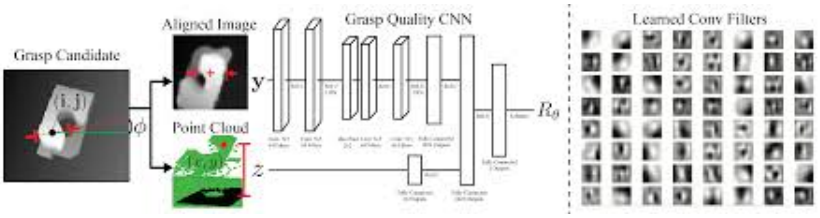


Figure 3.9. GQCNN architecture.

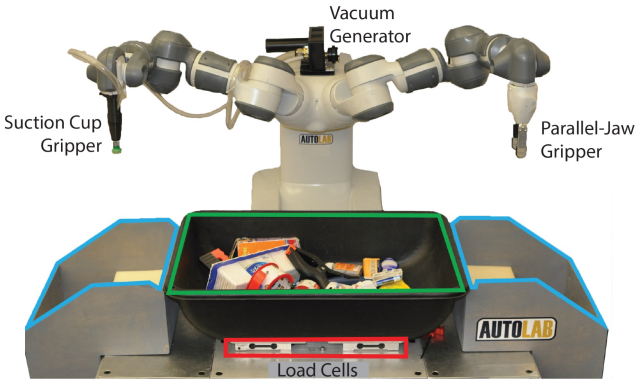


Figure 3.10. Dex-Net 4.0 has a bi-manual configuration with parallel-jaw and vacuum grippers.

3.2.3 Other DexNet Algorithms

Successive Algorithms are less correlated with the architecture proposed in this thesis, they specialize in vacuum gripper grasp evaluation (Dex-Net 3.0) [3] and bi-manual configuration with parallel-jaw and vacuum grippers (DexNet 4.0) [4] bringing to a configuration like the one in Fig. 3.10.

The limit of Dex-Net algorithm is that it observes only the geometric properties of the object to manipulate. For this reason it does not consider some physical characteristics of the object, like the material, center of mass, dynamic features.

Estimating these properties with the classical visual and tactile sensors is time and data demanding and difficult to implement. A common solution is to use demonstration to transfer human knowledge to the planner. In the next section are presented recent works in human demonstration and more in general human in the loop.

3.3 Human in the loop

In this section we will show a review of most recent human in the loop techniques to integrate human knowledge in a predictive model [20].

Classical supervised learning and reinforcement learning approaches require human supervision to define and associate evaluations and rewards to a training set. In many cases the definition of an evaluation function can be difficult, like in grasp planning case. For these cases a better solution is to use demonstration-based training (DBT) and inverse reinforcement techniques to bypass this kind of problem.

For example in Sermanet et al. [31] the planner uses human example to train a self-supervised imitation algorithm. In this way the robot is able to evaluate its own movement based only on the given examples.

Some other state-of-the-art methods have approached the interactive learning problem by mapping human information to rewards and iterating over them to compute better control policies [24][25].

In Griffith et al. is presented a more effective characterization of human feedback. In this paper Authors introduced Advise: an algorithm for estimating a human's Bayes optimal feedback policy and a technique for combining this with

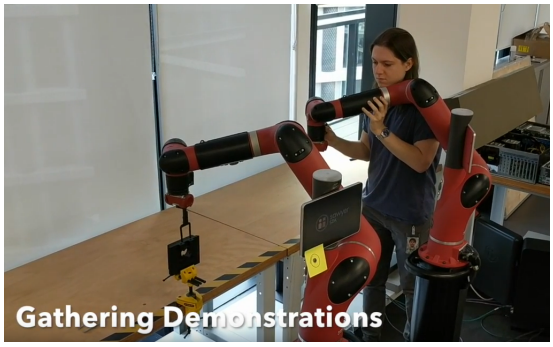


Figure 3.11. Vecerik et al. [28] use kinesthetical teaching and DDPG to execute manipulation tasks.

the policy formed from the agent's direct experience in the environment (Bayesian Q-Learning)[27].

Vecerik et al. proposed a model-free method based on Deep Deterministic Policy Gradient (DDPG) and structured to use demonstrations. Demonstrations are collected by a robot kinesthetically force-controlled by a human demonstrator. Results on four simulated insertion tasks show that DDPG from demonstrations out-performs DDPG, and does not require engineered rewards [28].

A different approach is proposed in Singh et al. [29]: in this paper, it is proposed an approach for removing the need for manual engineering of reward specifications by enabling a robot to learn from a modest number of examples of successful outcomes, followed by actively solicited queries. In the queries the robot shows to the user a state and asks for a label to determine whether that state represents successful completion of the task. Three main approaches have been tested, with increasing success rate(RL with Active Queriesor RAQ, Off-police VICE, Off-police VICE-RAQ).

In Losey et O'Malley it is presented a method based on learning from corrections[36]. The authors solved the inverse RL problem using Kalman filter, where human corrections are considered noisy observations. The paper contributes to the creation of a human-robot interaction (HRI) model based on human corrections and at same time showed as this kind of problem could be solved also with methods alternatives

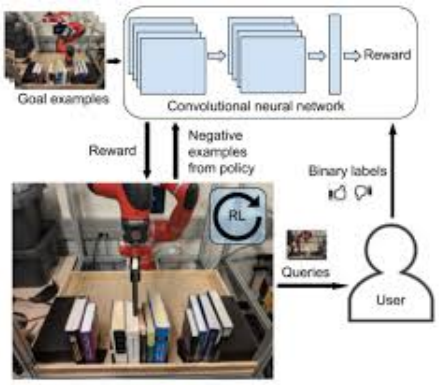


Figure 3.12. In Singh et al. [29] there is no need of reward specification. The planner learn by a set of examples of successfully outcomes and then from queries to the human operator.

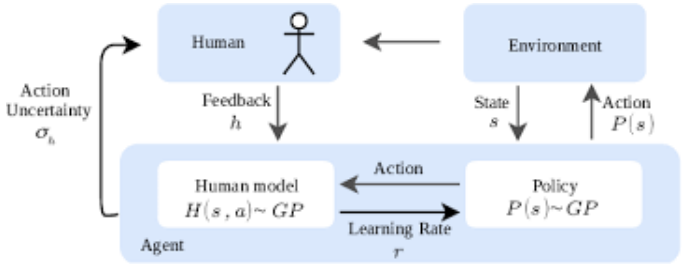


Figure 3.13. Schema of method presented in Wout et al. [23].

to the classical RL approaches.

Another different approach has been proposed in Daumé et Eisner[37]. They noticed that imitation learning from an Oracle can be hard in two ways: first, the learning policy space is far from the space that the oracle policy lies in, meaning that the learner only has limited learning ability. Second, the environment information known by the oracle cannot be sufficiently inferred from the state,meaning that the learner does not have access to good learning resources. To address this problem, they defined a coach in place of the oracle. To better instruct the learner, a coach should demonstrate actions that are not much worse than the oracle action but are easier to achieve within the learner's ability. Experiments demonstrated how the proposed method outperforms state-of-the-art imitation learning methods.

The solution proposed by Daumé et Eisner was further improved in Wout et al.[23], this paper introduces Gaussian Process Coach (GPC), where feature space engineering is avoided by employing Gaussian Processes. A schematic of the method is depicted in Fig. 3.13: the trainer observes the environment and the current policy and provides action corrections to advance the policy. These corrections trigger agent updates in order to take immediate effect on the policy. This process is repeated until convergence.

So, resuming, the human can interact in several way with the system (corrections, examples, evaluations, rewards). Several methods have been proposed to model the human feedback, some of them based on MDP, others using GP or Kalman filter.

In our work we attempt to model the human feedback in two phases: in a first phase human examples are provided to choose the best object to grasp, the method used to model the human policy is based on Mask-RCNN (next section). In a second phase the human provides an evaluation or a correction to improve the best grasp selection, the method used to model its behaviour is based on Variational Autoencoders (section 3.5) and Gaussian Processes (section 3.6).

3.4 Mask-RCNN

In this section I will present Mask-RCNN and its structure.

MaskRCNN [7] is a state-of-the-art machine learning tool for image segmentation. Image segmentation is a computer vision classical problem: it is the process of assigning a label to every pixel in an image, so pixels with the same label share certain characteristics. In our specific case we want to segment pixels which belong to the same object.

There are two stages of Mask-RCNN (Fig. 3.14). First, it generates proposals about the regions where there might be an object based on the input image. Second,

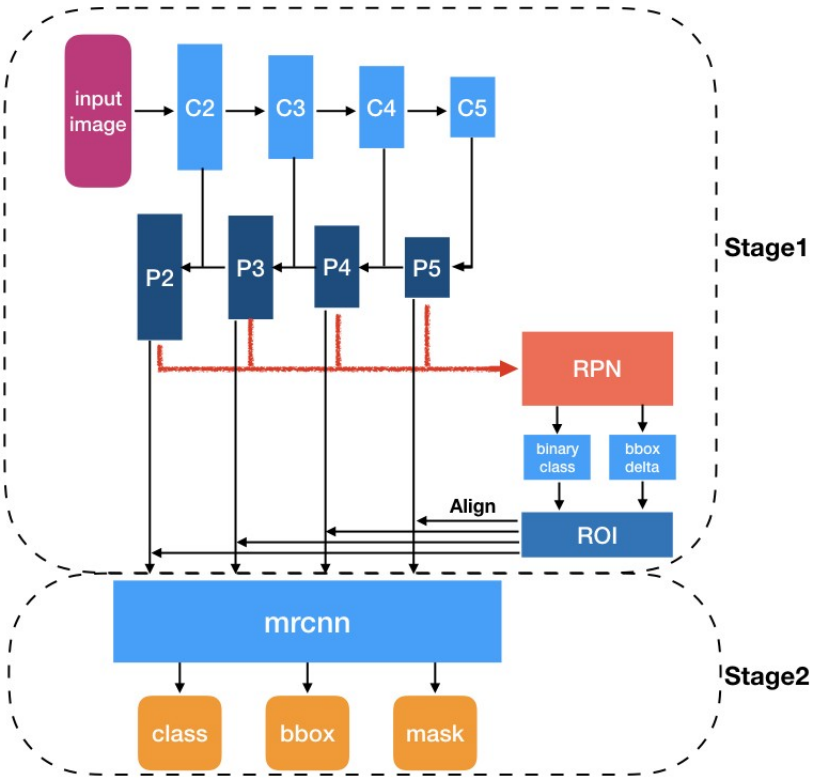


Figure 3.14. Mask RCNN architecture.

it predicts the class of the object, refines the bounding box and generates a mask in pixel level of the object based on the first stage proposal. Both stages are connected to the backbone structure.

A more detailed explanation of the network components follows:

- Backbone model: a standard convolutional neural network which extracts the features of the region of interest.
- Region Proposal Network (RPN): RPN scans each region and predicts whether or not an object is present. One of the great advantages of the RPN is that does not scan the actual image, the network scans the feature map, making it much faster.
- Region of Interest (ROI) Classification and Bounding Box: In this step the algorithm takes the regions of interest proposed by the RPN as inputs and outputs a classification (soft-max) and a bounding box (regressor).
- Segmentation Masks: In the final step, the algorithm takes as inputs the positive ROI regions and generates 28x28 pixel masks with float values as outputs. During inference, these masks are scaled up.

Several dataset are available for object segmentation, between them most commons are coco (Common Objects in Contest) dataset [8] and ImageNet[9].

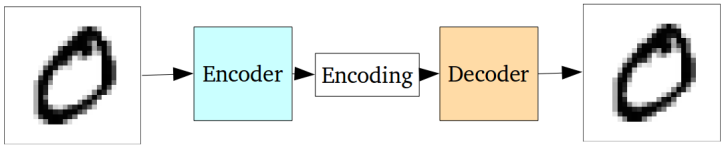


Figure 3.15. Autoencoder schema.

3.5 Variational Autoencoder

3.5.1 Autoencoder (AE)

AutoEncoders (AEs) are tools used for dimensional reduction. They map a high dimensional tensor in a more dense representation. AEs are based on two sequential network: the encoder and the decoder.

The *encoder* compresses the high dimensional data x (starting dimension $\|x\|$) to a lower dimensional data ($\|z\|$). The space of the compressed vectors z is called latent space. The encoder is typically referred to as a 'bottleneck' because it must learn an efficient compression of the data into this lower-dimensional space.

The *decoder* is another neural network, it takes as input the latent space z and returns a representation $\hat{x}$ with the same dimension of x; $\hat{x}$ represents the reconstruction of x.

The loss function is usually either the mean-squared error or cross-entropy between the output and the input; it is known as reconstruction loss. It penalizes the network for creating outputs different from the input.

Problem with AEs:

The fundamental problem with autoencoders, for generation, is that the latent space they convert their inputs to and where their encoded vectors lie, may not be continuous, or allow easy interpolation. For this reason Variationa AutoEncoders (VAEs) have been proposed.

3.5.2 Variational Autoencoders (VAEs)

VAEs have one fundamentally unique property that separates them from simple autoencoders: their latent spaces are, by design, continuous.

They make that by creating two vectors μ, σ which respresent the mean and the standard deviation. The latent vector of the input x is sampled in the region $\mathcal{N}(\mu, \sigma)$ so, in the training, we can have more z representing the same x.

Intuitively, the mean vector controls where the encoding of an input should be centered around, while the standard deviation controls the "area", how much from the mean the encoding can vary. As encodings are generated at random from anywhere inside the "circle" (the distribution), the decoder learns that not only is a single point in latent space referring to a sample of that class, but all nearby points refer to the same as well.

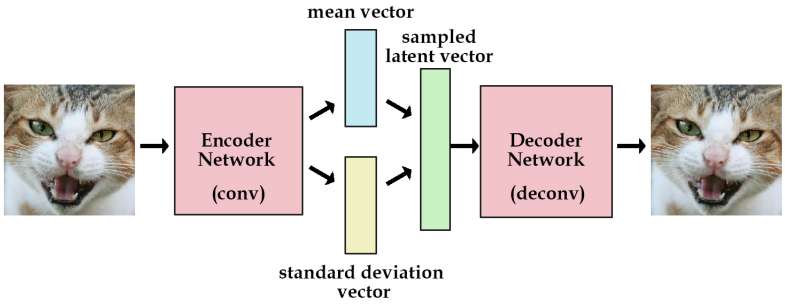


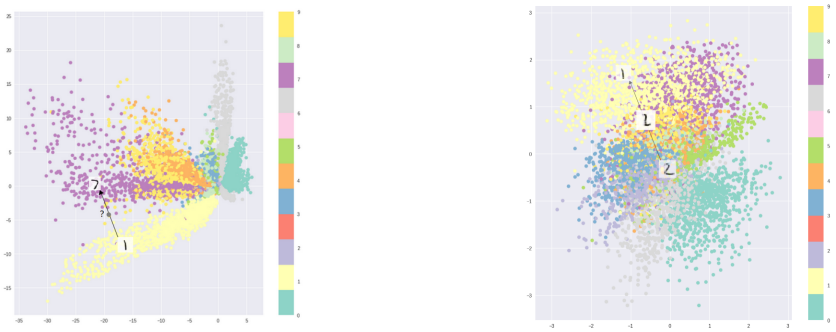
Figure 3.16. VAE schema.

What we ideally want are encodings, all of which are as close as possible to each other while still being distinct, allowing smooth interpolation, and enabling the construction of new samples.

In order to force this, it is introduced the Kullback–Leibler divergence (KL divergence) into the loss function. The KL divergence between two probability distributions simply measures how much they diverge from each other.

$$\sum_{i}^{n}(\sigma_i^2 + \mu_i^2 - log(\sigma_i) - 1) \tag{3.4}$$

VAE is also widely used with other task, for example they can be use in semi-supervised learning, noise reduction tools and generative network. In particular for generative networks they utilize the properties encoded in the latent space: for example, if we want to apply a feature to a given picture, we move its representation in the direction in which this kind of feature is encoded.



(a) Latent space for AE.

(b) Latent space for VAE.

Figure 3.17. Latent space for AE and VAE: in these plots we can visualize how each image of MNIST, handwritten digits database, is transformed in a latent space of dimension two, both for AE and VAE.

Figure 3.18. Image generation using VAE.

3.6 Gaussian Process

In this section, I will explain the main concepts behind Gaussian Process (GP) [12], GP is a state-of-the-art method for supervised regression.

Recall that in the simple linear regression setting, we have a dependent variable y that we assume can be modeled as a function of an independent variable x, i.e. $y = f(x)+\epsilon$ (where ϵ is the irreducible error) but we assume further that the function f defines a linear relationship and so we are trying to find the parameters θ_0 and θ_1 which define the intercept and slope of the line respectively, i.e. $y = \theta_0 + \theta_1 x + \epsilon$.

Classical approaches (like Bayesan one) try to find a distribution over the parameters θ_0 and θ_1. The problem of this kind of a formulation is that, if the target function $f(x)$ needs more parameters with respect to those defined, the reconstruction will never be adequate.

The GP approach, in contrast, is a non-parametric approach, it finds a distribution over the possible functions $f(x)$ which is consistent with the observed data. As with all Bayesian methods it begins with a prior distribution and updates this as data points are observed, producing the posterior distribution over functions.

Prior distribution:

Describing under another point of view, the idea is that we do not need to determine the minimum number of parameters necessary to reconstruct adequately the target function.

But of course we need a prior before we have seen any data. If we know the domain of x, for example $x \in [-5, 5]$ we can define different functions with different properties, like the smoothness.

There's a way to specify that smoothness: we use a co-variance matrix to ensure that values that are close together in input space will produce output values that

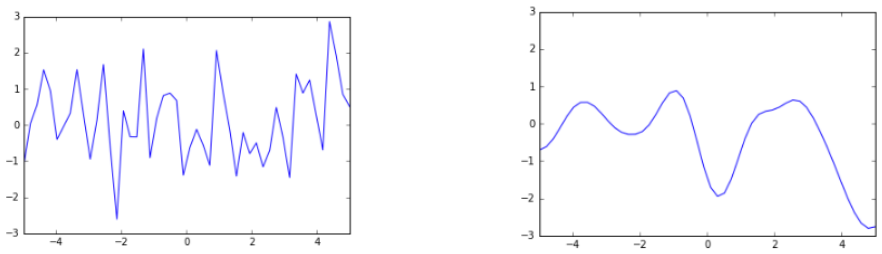


Figure 3.19. Examples of functions with different smoothness degree.

are close together. This co-variance matrix, along with a mean function to output the expected value of $f(x)$ defines a Gaussian Process.

Update:

One time we have defined the prior distribution we can update it, and calculate a posterior, using the Chain Rule:

$$P(A|B) = \frac{P(A,B)}{P(B)} \tag{3.5}$$

There are some points x for which we have observed the outcome $f(x)$ (denoted above as simply f). There are some points x_* for which we would like to estimate $f(x_*)$ (denoted above as f_*).

So we are trying to get the probability distribution $p(f_*|x_*, x, f)$ and we are assuming that f and f_* together are a jointly Gaussian so defined:

$$\begin{pmatrix} f \\ f_* \end{pmatrix} \sim N\left(\begin{pmatrix} \mu \\ \mu_* \end{pmatrix}, \begin{pmatrix} K & K_* \\ K_*^T & K_{**} \end{pmatrix}\right). \tag{3.6}$$

where: K is the matrix we get by applying the kernel function to our observed x values, i.e. the similarity of each observed x to each other observed x; K_* gets us the similarity of the training values to the test values whose output values we're trying to estimate; K_{**} gives the similarity of the test values to each other.

After some computation and applying Chain rule and Bayes rule what we end up with is an estimation of our distribution $f_*(x) \sim \mathcal{N}(\mu_*, \Sigma_*)$ which best approximates the target function $f(x)$.

The parameters of the distribution found in this way are the mean μ_* and the co-variance matrix Σ_*. An equivalent but computationally less expensive representation of that distribution is $f_*(x) \sim \mu_* + B_* \mathcal{N}(0,1)$, where B_* is a matrix such as $B_* B_*^T = \Sigma$ and can be recovered using Cholesky decomposition.

GP is a strong and use-full tool for regression because it can directly captures the model uncertainty and because in GP it is possible to add prior knowledge and specifications about the shape of the model by selecting different kernel functions (Fig. 3.20).

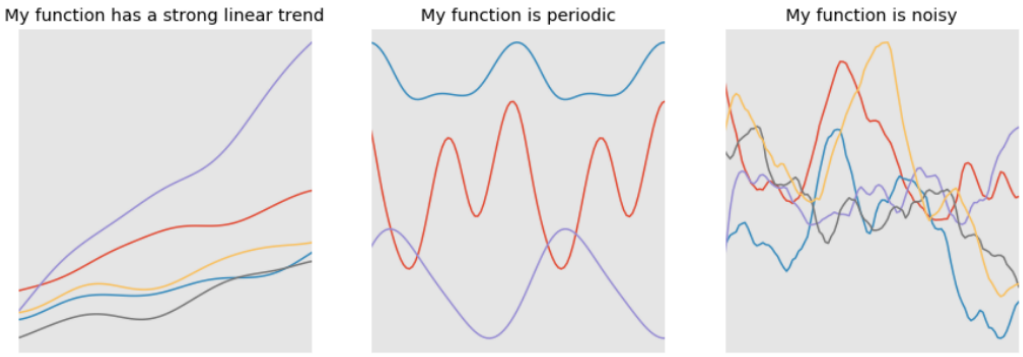


Figure 3.20. If some knowledge about target function's shape is available it is possible to integrate it in the prior.

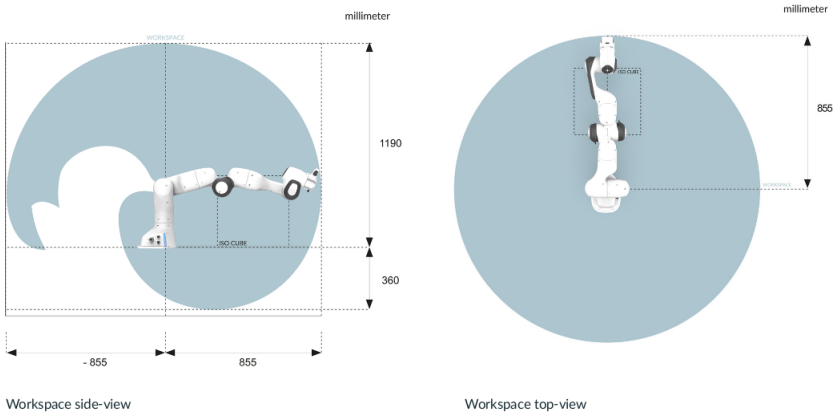


Figure 3.21. Franka work-space.

In our work we used GP to predict human's grasp evaluation with minimum size training set.

One time the object to grasp and the best grasp over that object have been defined the planner communicates to the Robot where to execute the grasp. In the next section I will explain the characteristics of the Franka Panda, the robot used for our experiments.

3.7 Franka collaborative robot

In this section I will briefly describe the Franka collaborative robot, namely the robot we work with.

Franka robot is a lightweight collaborative robot, it is a manipulator and its arm has 3 kg maximum payload. It is composed by 7 rotary joint and a parallel jaw gripper.

It can share an environment with a human operator thanks to its ability for collision detection, in fact it can detect a collision with a time that go from 2ms to 100 ms in the worst case. When a collision is detected the robot stops its movement.

The robot also offers a collaborative mode where it follows the teaching of the human operator. It is possible to access to this mode simply pressing a button located near the robot End-effector. One time the button is pressed on the robot acts an impedance control with minimum stiffness.

Chapter 4

Methods

4.1 Grasp selection pipeline

Our idea is to introduce human demonstration in two phases of the pipeline and then to model them so that, after a training phase, the robot can resume the execution in a autonomous way:

- Human operator selects the object to grasp;
- Human operator selects the best way to grasp the object;

In Fig. 4.1 is described how human operator interacts with the planner: after the camera takes the picture of the heap, the operator selects the object to be grasped, based on some criteria C; using the pictures of the heap and the object selection, Dex-Net proposes the grasps for that object; then, the operator selects the best grasp based on personal metric M.

We proposed two models, in our project, to represent the criteria C and the metric M, in that way, after a training phase, the pipeline can work also without human operator's suggestions.

In the following sections will be presented the two models and their architecture:

In particular, in section 4.2, the model for the object selection has been proposed, then, in section 4.3, the model for grasp selection metric is presented.

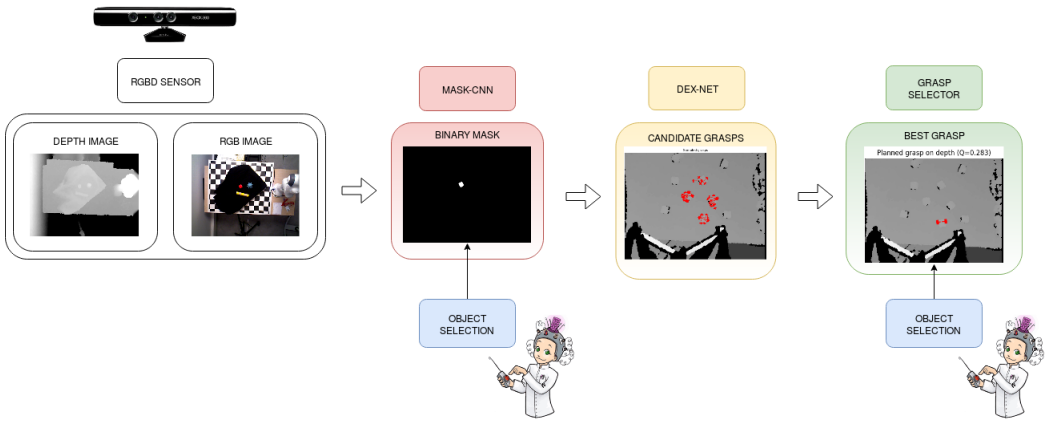
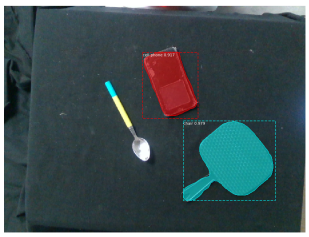
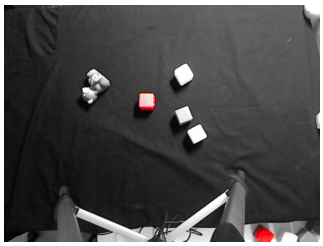


Figure 4.1. Pipeline.



(a) Mask-RCNN trained only with coco dataset.



(b) Mask-RCNN trained to recognize the red cubes.

Figure 4.2. Object recognition using MaskCNN.

4.2 Mask-RCNN for object selection

The first model proposed is based on Mask-RCNN [7]: this is a state of the art method for image segmentation. In our architecture Mask-RCNN will be used to model the human's object selection.

In a first training phase the operator selects the object to grasp. To do that, he takes the image returned by the RGBD sensor as input, and draws the mask of the desired object on it. So, the image and the mask are then passed to Mask-RCNN to train the model. The same image and mask can be passed to the grasp selector and so the pipeline is not interrupted in the training phase.

Mask-RCNN returns also the confidence of a proposed mask, that value can be used to require more examples from the human operator. On the other side we want to reduce as much as possible the human's presence in the pipeline, so we pre-trained the network with the COCO dataset.

COCO dataset contains several use-full classes of objects: persons, household object, foodstuffs. In this way some common objects can be segmented also without human teaching. On the other side, sometimes, the network proposes some mask but with wrong label: in these cases, minimum effort is required from the human operator.

To reduce as much as possible the human effort the Mask-RCNN's node is so composed:

1. Train Mask-RCNN with COCO dataset.
2. The RGB image i is passed to Mask-RCNN: it returns a proposed mask m. If no objects are segmented, the mask is empty.
3. Human operator evaluates the mask:
    - If the mask m is satisfactory then it is passed with the image i to Dex-Net. No training of Mask-RCNN is necessary.
    - If the proposed mask m is not good enough, the operator draws a new correct mask m'. The correct mask m' and the image i are passed to DexNet to resume the pipeline; m' and i are passed also to Mask-RCNN to be trained more.

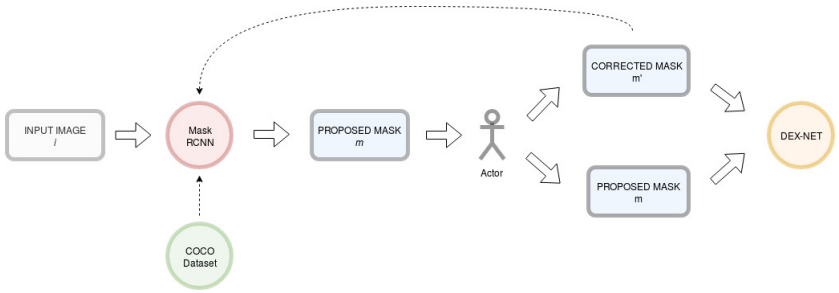


Figure 4.3. Mask-RCNN node with human in the loop.

4. The human evaluation and supervision can continue forever, or until a certain threshold of confidence.

Following this structure, when the network is able to recognize the object with high confidence, the human operator is no more necessary. In Fig. 4.3 the Mask-RCNN's node is schematize.

4.3 Human demonstration autonomous grasp planning

In the following sections, I will present how the human's grasp selection has been modeled.

The main obstacle to integrate human demonstration in a grasp planner is that the planner needs a huge dataset to learn. The main contribute of our work is, instead, the ability to transfer knowledge with a minimum size data set.

To do that we use a Variational autoencoder (VAE) to reduce the dimension of the grasp representation (section 4.4), then we use a Gaussian Process to do classification and regression on the latent space of the VAE (section 4.5).

4.4 Variational Autoencoder (VAE) for represent grasps

In this section, I will show how we used VAE to reduce the size of the grasp representation.

4.4.1 Grasp representation

Dex-Net algorithm (D) takes as input a point cloud image (X) of the heap and returns, as output, a list of N grasps (L):
$D : X \rightarrow L$ with $L_i = (p_i, \theta_i, w_i, q_i)$ for $i = 0 : N$ where p, θ, w and q are respectively position, orientation, width and grasp evaluation of each grasp.
In our configuration each grasp L_i is represented as a RGB and a depth image centered in the grasp center p and with x axis parallel to the grasp axis and with high and width selected constant (100x100).
The resulting dataset is composed by images of dimension 4x100x100 where the first three channels are those of the RGB image; the forth is for depth image as represented in Fig. 4.4.

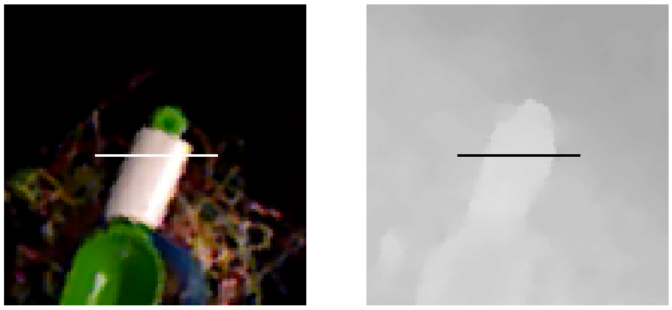


Figure 4.4. Each grasp is represented with a RGB and a depth picture centered in the grasp center and with orientation planar to grasp orientation. In these pictures the lines represents the grasp orientation (they are not present in the grasp representation, we show them only for a more easy visualization).

4.4.2 Variational Autoencoder

To reduce the dimension of our representation we built a VAE which maps high dimension images ($|x| = 4 * 100 * 100$) to a latent space with reduced dimension ($|z|$).

Following the work of [22] we processed in parallel the RGB image and the depth image, the structure of the resulting network is showed in Fig. 4.5.

The dataset used to train the VAE is composed by ten thousand grasp's images extracted by different objects positioned in different configurations. The dataset has been enlarged applying some data augmentation: the rotated images have been added to the training set of the VAE.

One weak point of an implementation like this, is that the VAE attempts to reconstruct the image without concentrating too much on the grasp information.

In the next section, I will present an extension of the VAE which tries to maximize the grasp information encoded in the latent space and, at same time, to reconstruct the image.

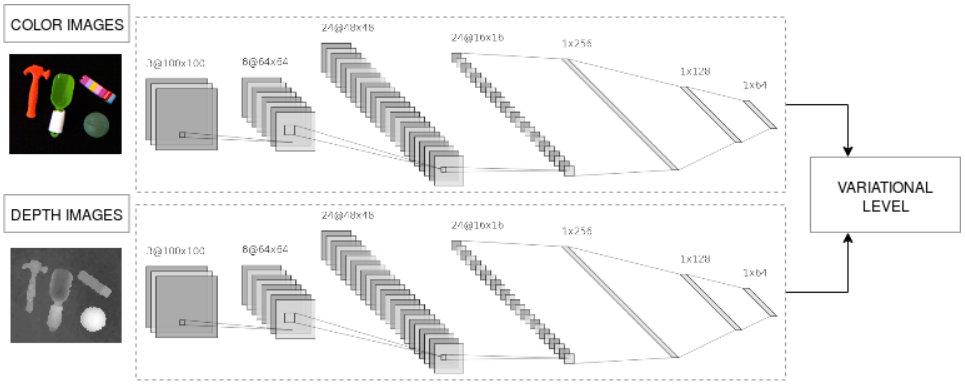


Figure 4.5. Architecture of the encoder. In this architecture it is possible to see how the RGB and the depth channels are processed in parallel. The two latent are then used together in the variational level.

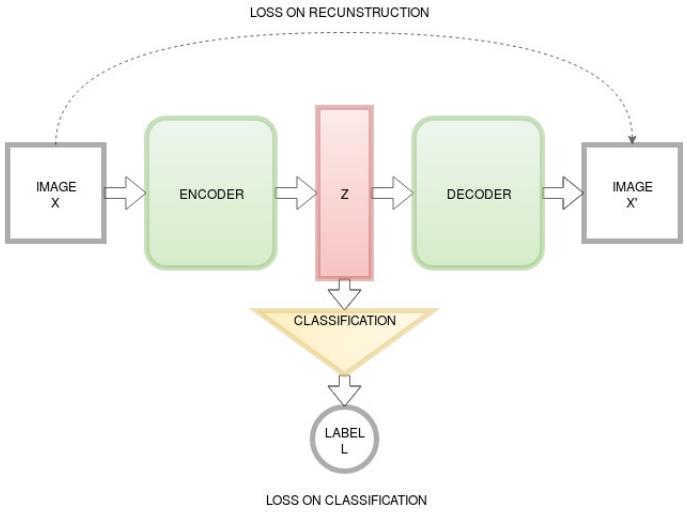


Figure 4.6. VAE-ic structure.

4.4.3 VAE with Intrinsic Classification (VAE-ic)

In this section we substitute the classical VAE with a structure we will call Variational Autoencoder with Intrinsic Classification (VAE-ic). Our will is to extract more specifically feature of the grasp in the latent space. This structure should concentrate less on the image reconstruction and more on grasp features.

To do that we use the same dataset of 10M images of grasps of the VAE. Our assumption is that each image in the dataset represents a good grasp, on the other side we hypothesize that the rotation of the image for an angle of 90 degrees does not represent a good grasp.

Using this assumption we extend the dataset of 10M images inserting the rotated images. The original images will be then labeled as positive grasps while the rotated ones as negatives grasps.

Then we modify the VAE's structure inserting at the end of the encoder a classification layer as showed in Fig. 4.6.

The aim of this architecture is to select z to maximize both the classification and the reconstruction. Doing that the z latent space will concentrate more on the features relevant to represent a grasp. To train the network we consider as loss function the weighted sum of the classification and the reconstruction loss functions.

$$L_{VAE-ic} = \alpha L_{Rec} + \beta L_{Cla} \tag{4.1}$$

where the parameters α and β are experimentally calculated. For completeness in the explanation, the simple VAE can be seen as a case of VAE-ic with $\beta = 0$ and $\alpha = 1$.

4.5 Gaussian Process for best grasp selection

Gaussian Process (GP) method has showed to be a strong tool for regression. In this work we want to associate a grasp to an evaluation function given by the human operator. For that reason we formalized our problem as a regression problem with target function the human's evaluation.

The limit of GP is the fact that it seems to work better for low dimension data, for that reason we implemented VAE to reduce grasp representation.

We evaluate our method with three experiments with increasing difficulty:

1. Human suggestion is a classification label $\hat{q} \in \{0, 1\}$. In this experiment the human operator divides the grasps he consider good (1 label) from those he consider bad (0 label).

2. Human suggestion is a evaluation function $\hat{q} \in [0, 1]$. In this experiment the human operator associates a score to each grasp from 0 to 1.

3. Human suggestion is a transformation to a better grasp $\hat{q} \in \mathcal{T}_2$, where $\mathcal{T}_2$ is the set of the transformations in $\mathcal{R}^2$ composed by a rotation and two translations on x and y. In this experiment the operator proposes a new grasp, the target function will be the distance between the proposed grasp and the corrected one.

4.5.1 Classification of grasps

Human operator evaluates each grasp as positive or negative, it does not consider middle cases. The GP is used as classifier to predict these labels.

4.5.2 Regression on human grasp evaluation

Human operator gives a score in the range $[0, 1]$ to each grasp; GP is used to map the latent space of the VA to the target function $\hat{q}$.

The limit for a solution of this kind is the fact that is not straightforward to associate a score to a grasp. A human operator assigns to each image a score, this kind of solution can not be considered statistically correct because it is subject to bias. I future works we will use the evaluation coming from several human operators.

4.5.3 Regression on human correction

Human operator designs a better grasp w.r.t. the proposed one [23]. The error function is the rotation $\Delta\theta$ and the translation $\Delta X = (\Delta x, \Delta y)^T$ necessary to transform the grasp proposed by Dex-Net to the one proposed by the human

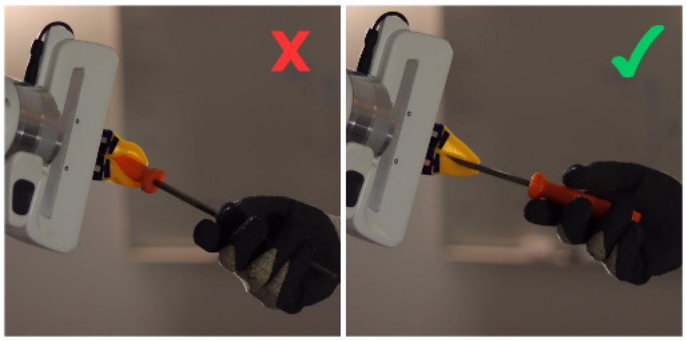


Figure 4.7. An example of good and bad labels for classification.

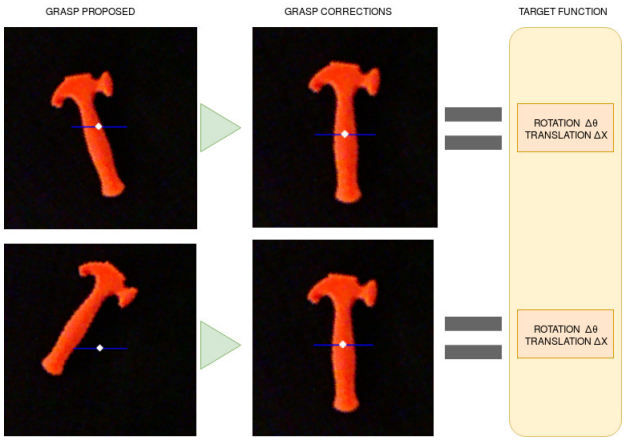


Figure 4.8. Target function is the transformation from the proposed grasp to the corrected one.

operator (Fig. 4.8). Also in this case the solution is subject to bias, for this reason we propose to use data from more than one operator for future experiments.

The GP uses the three label target functions $(\Delta\theta, \Delta x, \Delta y)$ to map the latent space of the VAE to the transformation target functions.

The three experiments represent three tasks with increasing difficulty, the purpose is to test what is the limit of our implementation.

4.6 Task execution:

In this section I will present how the robot acts one time the best grasp is selected.

Franka robot goes to the required position and tries to collect the desired object applying the desired grasp, this is the procedure it executes:

- Grasp approaching: The robot collocates the end effector (EE) over the grasp pose. The EE orientation is perpendicular to the table and facing down.
- Open the gripper.
- Descent trajectory execution: the robot executes a linear Cartesian path to descent while keeping the EE orientation invariant.
- Close the gripper.
- Ascent trajectory execution: the robot executes a linear Cartesian path to ascent while keeping the EE orientation invariant.
- Transport: the robot brings the object in a second location.
- Evaluation: the grasp is consider successfully if at the end of the procedure the object is in the second location.
- Comeback to initial position: the robot comes back to the initial position, where the position is chose in such a way that the robot is not inside the camera field of view.

Chapter 5

Experiments

5.1 The setup

The setup for experiments follows the HEAP project requirements (Chap 2): It is composed by a top and fixed camera which can visualize the whole heap of objects. The robot is a Franka robot with collision detection and possibility of interaction with human operators. The gripper is composed by 3D printed fingers, they have been designed by one of the partners of the project, their structure reduces the probability of break the manipulated object and augment the surface of contact. Finally, the objects to be grasped are household objects or games for children. We chose these objects because they are heterogeneous and they differ in material, shapes and colors. In Fig. 5.1 it is possible to see the setup for experiments.

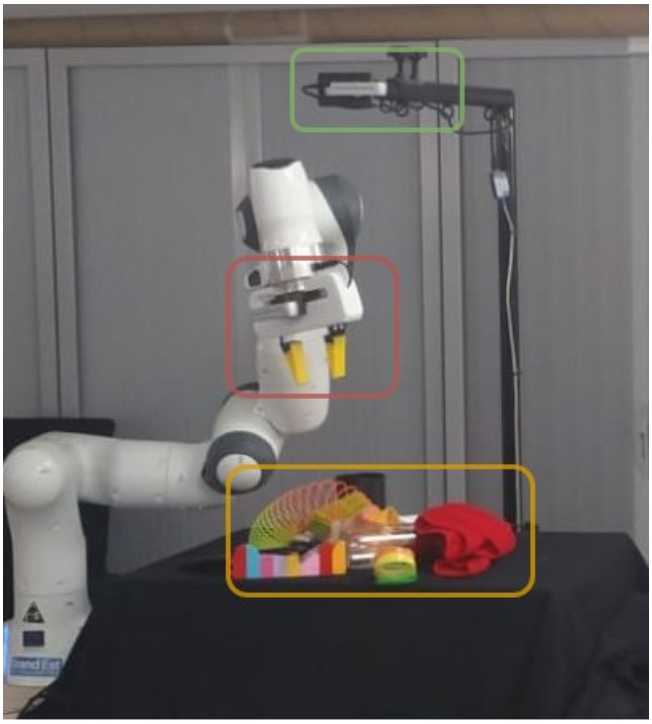


Figure 5.1. Robot camera setup: In green the Realsense RGB-D sensor; in orange the heap of objects on which the grasp is executed; in red the Lincoln Fin Ray fingers; on the left the Franka collaborative robot.

Figure 5.2. Franka Robot.

5.1.1 Robot Setup and Initialization

In this section I will present how the robot is initialized.

Franka robot is a manipulator with seven degree of freedom (Fig. 5.2). It is able to detect collisions and to follow teaching coming from a human operator. To set the parameters of its control (i.e. impedance, time of reaction, maximum acceleration) we can access to the robot interface in the ignition phase.

To control the robot we used MoveIt, a motion planning framework which communicates with the robot through Ros. MoveIt has an interface in both python and C++; we used python for our experiments. After the ignition phase the robot executes a set of movements to calibrate the fixed camera (section 5.1.4), during these movement on the gripper of the robot a landmark is applied.

After the calibration phase, the robot is set up in a configuration which allows the camera to see the full set of objects to grasp.

5.1.2 The gripper

Final task includes the manipulation of radioactive objects, since they might break the danger for the human operator is high. To reduce the probability that it happens in the Franka robot has been installed Lincoln Fin Ray fingers [35].

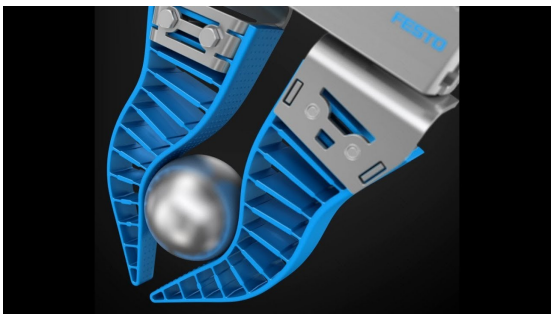


Figure 5.3. Lincoln Fin Ray fingers.

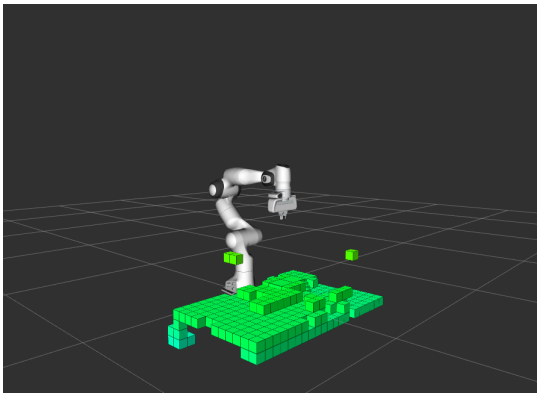


Figure 5.4. Visualization the environment recover from the Kinect sensor.

This kind of fingers has been created to manipulate breakable things (i.e. vegetables). They can increase the surface of contact and therefore the grasping ability.

5.1.3 The environment

The environment, in which the robot works, presents several fixed obstacles (like the table). To manipulate the objects in the heap, the motion planner needs to know the position of the obstacles in such a way to avoid them. To do that, the transformation from the camera frame to robot frame is calculated (section 5.1.4). Using this transformation, it is possible to detect the position of the obstacles in the scene.

In a first phase of the experiment, the position of the fixed obstacles has been recorded in such a way to avoid them during future movements (Fig. 5.4).

5.1.4 Camera setup and Extrinsic calibration w.r.t. Robot Frame

Several RGB-D sensors have been compared, at the end we chose to use Realsense d415 (Fig. 5.1).

Camera must be positioned in front of the table, in this way its z axis is perpendicular to the plane where the objects are located.

In this section I will describe the calibration procedure for the RGB-D sensor.

Intrinsic calibration:

In the first phase of our experiment we calculated the parameters of the camera: the focal length, the width and the height of the pixels.

This is done using the classical cheeseboard and the intrinsic calibration launch file in Ros package. The calibration must be done for both cameras (RGB and Depth). After that, we used another Ros launch file to do the registration between the two.

Extrinsic calibration:

In this second part we need to calculate the transformation from robot frame to camera frame in such a way to position the end effector (EE) in the correct grasp location.

Our approach is based on the estimation of the EE using a marker. After, we collected a set of points in the two reference frame, in this way, we can calculate the transformation matrix between them.

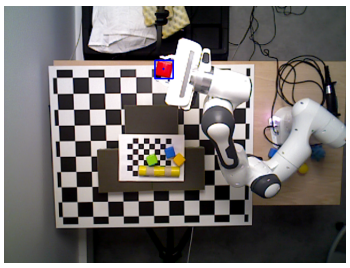


Figure 5.5. Identification of the end effector position using the red cube as marker.

The EE's position in robot frame (${}^R\vec{e} \in R^3$) is obtained using the MoveIT commander interface, while, the EE's position in camera frame (${}^C\vec{e} \in R^3$) is recovered by the Depth sensor;

Having a dataset of points in both reference frames, we used *Perspective-n-Point* to recover the translation ${}^Rt_C$ and the rotation ${}^Rr_C$ from the robot frame to camera frame.

Every time we have a candidate grasp (in the image plane), we perform the following procedure to recover its position in robot frame:

Each grasp is represented by its coordinate in image plane $[p_x, p_y]^T$ and its orientation θ with respect to the image x axis.

To recover the grasp position in camera frame ${}^Cg = [{}^cx, {}^cy, {}^cz]^T$, we used the map from the pixel position to the point cloud generated by the depth camera.

We observed the relations between the image frame and the camera frame (pinhole camera model in Fig. 5.6) and obtain ${}^co = [0, 0, \theta]$, the grasp orientation in camera frame.

Thus, the grasp position ${}^Rg$ and orientation ${}^Ro$ in robot frame, can be obtained by applying the relations between homogeneous transformations:

$$\begin{bmatrix} R_{RPY}({}^Ro) & {}^Rg \\ 0\ 0\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{RPY}({}^Rr_C) & {}^Rt_C \\ 0\ 0\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_{RPY}({}^Co) & {}^Cg \\ 0\ 0\ 0 & 1 \end{bmatrix} \tag{5.1}$$

Since the camera is 1.50m far from the object's location the extrinsic calibration parameters are subject to errors which affect the success of some grasp, for this reason, in future we will mount a second RGB-D sensor in the proximity of the end effector.

Once calculated, the parameters for intrinsic and extrinsic calibration are available

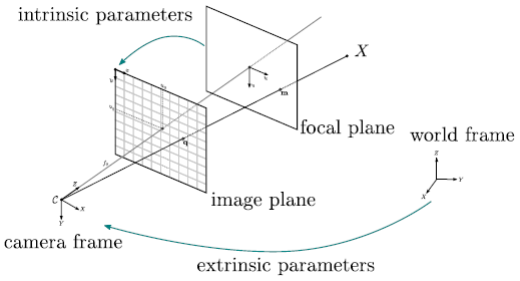


Figure 5.6. Pinhole camera model.

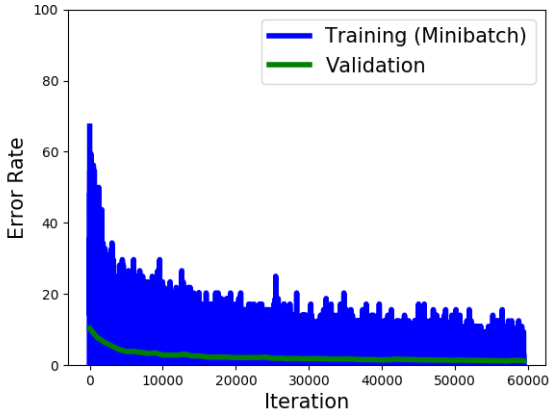


Figure 5.7. Training of DexNet 2.0.

for experiments. Anyway, these parameters have to be periodically updated in such a way to not increase too much the amount of error.

5.2 DexNet

As seen in section 3.2, DexNet is one of the state-of-the-art algorithms for grasp planning. The software is open source and I spent some time to configure it.

As first thing I trained the GQCNN network using the DexNet 2.0 dataset. In Fig. 5.7 it is possible to see the decrease of the error rate during the training.
Once GQCNN is trained, the network can be used to generate new grasps and evaluate them starting from a single-view point cloud. To do that is necessary to set in the planner some of the parameters of the environment like the distance camera table, the width of the gripper, the calibration matrices.

Given a depth image of the heap GQCNN returns a set of grasps; each grasp is described by the following parameters:

- Grasp Center position in pixel x and y.
- Orientation of the grasp w.r.t. the horizontal plane of the image.
- Gripper width.
- Evaluation of the grasp $Q(\mathrm{g})$.

Between the proposed grasps, Fig. 5.8a, the best grasp is the one with highest $Q()$, Fig. 5.8b.

(a) Proposed grasps.

(b) Best grasp is the one with highest evaluation $Q()$.

Figure 5.8. Example of Dex-Net proposed grasps.

DexNet is a very robust algorithm, but it has some limits:

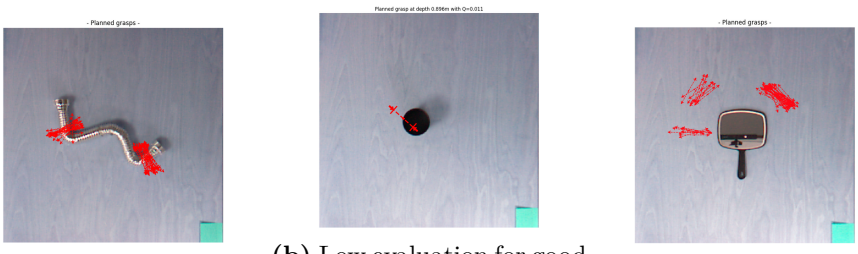
It uses only the depth information to predict the best grasp. In particular the planner does not consider information about the object material, dynamic model and any considerations about the future use. For example, it does not consider the CoM position of the object like in Fig. 5.9a, and has problem with reflective materials like in Fig. 5.9c.

The aim of our work is to extend the already success-full DexNet model inserting some considerations coming from human experience.

5.3 Mask RCNN

As told in section 3.4 Mask-RCNN is one of the state of the art method for image segmentation. We chose to use this architecture to model the human preference in object selecting.

Mask-RCNN is first of all trained with COCO dataset: using an implementation of this kind 80 class of objects can be identify and segmented. Between these classes we can find persons, animals, vehicles, household objects, foods. In Fig. 5.10 we can see how the network trained with this dataset can segment and identify some objects.



(a) Far from CoM.

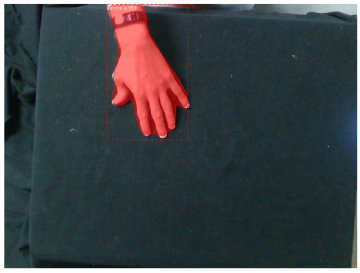
(b) Low evaluation for good grasps.

(c) Mirrors.

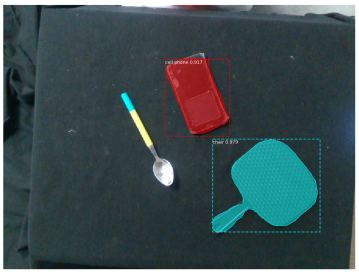
Figure 5.9. Example of proposed grasps.

(a) Mobile phone True positive.

(b) Spoon True Positive and toothbrush False Positive.



(c) Person True Positive.



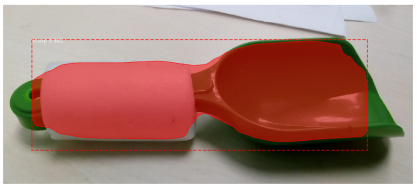
(d) Cell True Positive, mirror correct mask but wrong label, spoon not detected.

Figure 5.10. Examples of MaskRCNN trained with COCO.

The network is then retrained on a second dataset with images and mask of a given object. In Fig. 5.11 it is possible to see how the network can identify others objects.

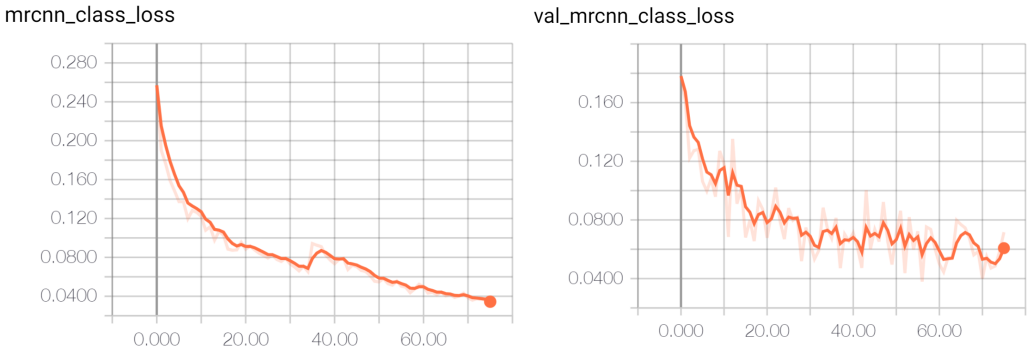
The loss function of Mask-RCNN is based on pixel pairwise differences between the predicted mask and the correct one. In Fig. 5.12 is plotted how the loss function decreases during the training, both for train and validation sets.

(a) Network retrained to select the red cube.



(b) Mask retrained to select the scoop.

Figure 5.11. Examples of Mask-RCNN trained with COCO and retrained to identify new objects.



(a) Train loss. **(b)** Validation loss.

Figure 5.12. Train and validation loss during the training of Mask-RCNN.

5.4 Variational Auto-Encoder(VAE)

In this section I will present some of the experiments done to reduce the dimension of the grasp representation using the VAE architecture.

We define each grasp as an RGB-D image with height and width of 100 pixels. The resulting dimension of the data entering in the encoder is 4x100x100.

The training set of the VAE is composed by ten thousand images. Each image is preprocessed before entering in the network: we removed from the depth image the values too close and too far and then we re-scale the remaining values; all the four channels are normalized.

The limit of the VAE is the need of a huge dataset for the training phase so data augmentation is applied: we enlarge the dataset with the images rotated for 90 degree and with the transpose images.

Our first approach is to use a simple CNN taking as input the 4 channels. This kind of solution introduces some artifacts in the depth image given by the influence of the RGB image. In Fig. 5.13 it is possible to visualize the artifacts introduced by an architecture of this kind.

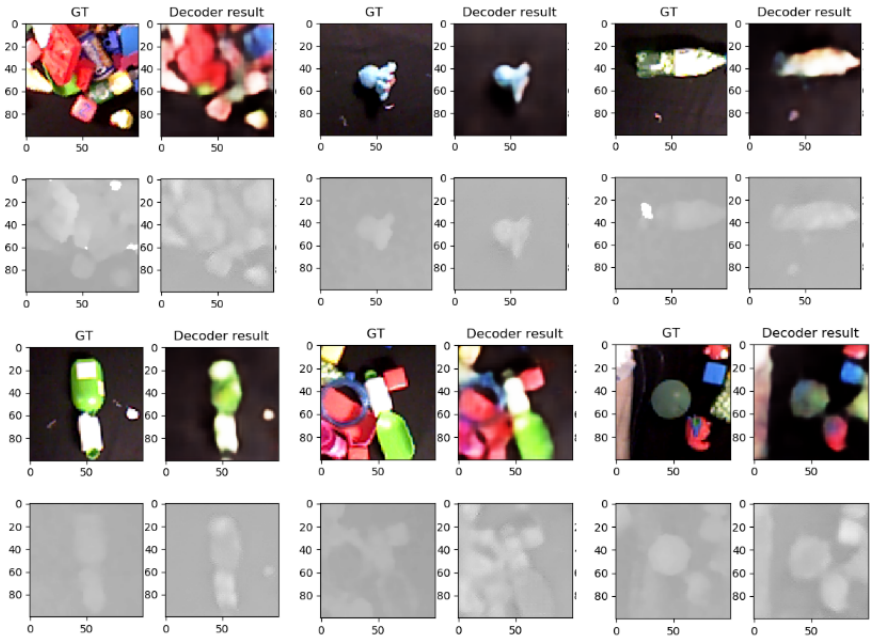


Figure 5.13. In the deth images can be observed the influence of the RGB image. In this way RGB and Depth images are no more indipendent.

For this reason, following the work of Ren and Lu[22], we processed in parallel the RGB image and the depth image. The structure of the resulting network is showed in Fig. 5.14.

Several configurations have been tested changing the latent space dimension. Obviously, with bigger latent space it is possible to encode more information, so the image out-coming from the decoder will be more similar to the starting one (Fig. 5.15). On the other hand, our task is to reduce the dimension for the grasp representation so that we can improve the Gaussian Process predictive power.

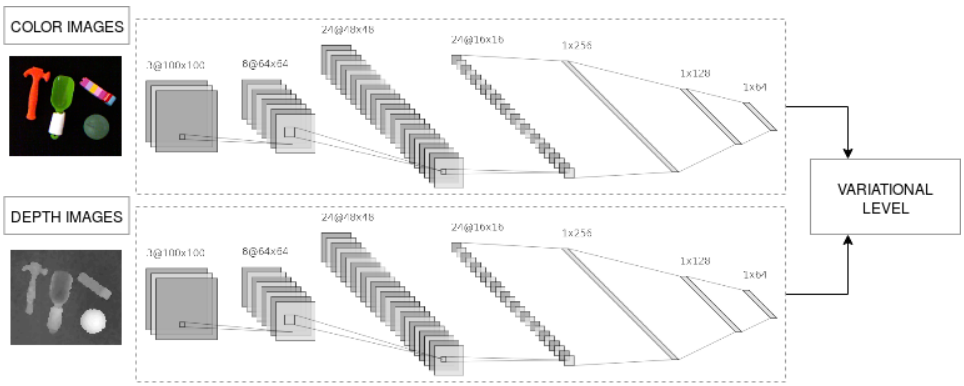


Figure 5.14. RGB and Depth information are processed in parallel in such a way to maintain the Independence of the information encoded.

Figure 5.15. Reconstruction of the RGB and the depth image changing the latent space dimension.

The dataset has been divided in training set and validation set.

The training set has been used to train the VAE, while the validation set is used to test the reconstruction.

In Fig. 5.16 is plotted the loss on the reconstruction while we train the network. The loss function $L(x, \hat{x})$ is define as:

$$L(x, \hat{x}) = BCE(x_{RGB}, \hat{x}_{RGB}) + \omega BCE(x_D, \hat{x}_D) + KL(\mu, \sigma) \tag{5.2}$$

where $BCE()$ is the binary crossentropy function, ω is a constant used to give the same weight to the reconstruction of Depth and RGB images, $KL()$ is the Kullback-Leibler divergence (eq. 3.4).

As saw in Fig. 5.15 the latent space dimension influences the reconstruction of the VAE; in Fig. 5.17 it is possible to see how the loss in the reconstruction and the latent space dimension are related.

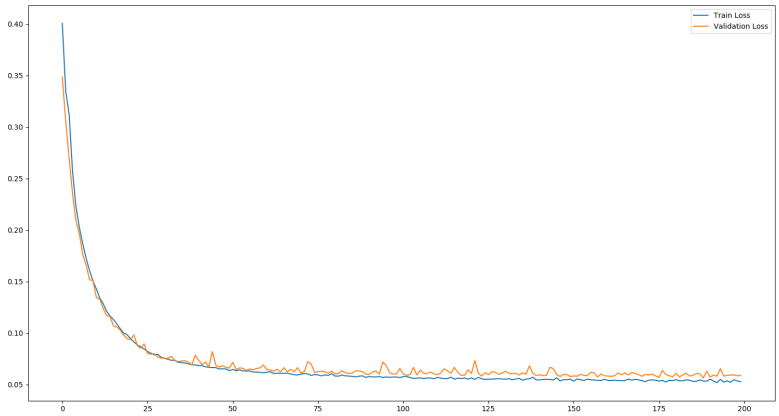


Figure 5.16. Loss of the VAE during training, both train set (blue plot) and validation set (orange plot).

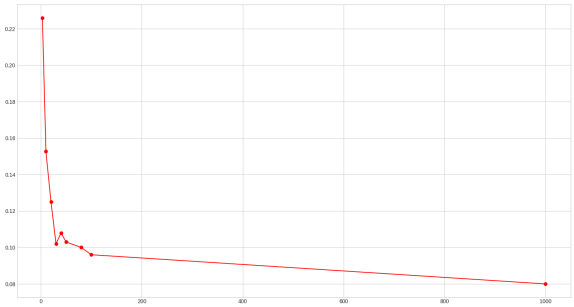


Figure 5.17. Plot of the loss of the VAE changing latent space dimension. For high dimension (1000) we used an architecture without fully connected layers (FCL), so without variational layer, because they are computationally too expensive; This value has been plotted only to show how the reconstruction changes.

In a second moment we tried to improve VAE's ability to select specific features for the grasp. For this reason we proposed the VAE-ic model. As explained in section 4.4.3 it has an intern layer which discriminates the original grasps from the modified ones. The final loss function is:

$$L_{VAE-ic} = \alpha L_{Rec} + \beta L_{Cla} \tag{5.3}$$

In Tab. 5.1 it is possible to see a comparison between VAE-ic and VAE ($\alpha = 1,\ \beta = 0$): we can observe how the models with bigger α reconstruct better the images and, in general, classify better the grasp images.

Anyway it is interesting to notice that the VAE-ic accuracy is less independent from the reconstruction, in fact it can reach good results also with higher reconstruction loss and it has a minor tendency to over-fit. Since the simple VAE model works better we continue to use it for later experiments.

Table 5.1. Comparison VAE and VAE-ic.

α, $(\beta = 1 - \alpha)$	Reconstruction Loss	Accuracy Classification
0.0	0.533	0.712
0.2	0.260	0.7602
0.4	0.262	0.796
0.6	0.247	0.785
0.8	0.237	0.798
1.0	0.230	0.864

In the following section we will use the reduced grasp representation and the GP model to transfer human knowledge with minimum amount of training. The proposed experiments have been studied to have increasing difficulty, in this way we can explore the limits of our implementation.

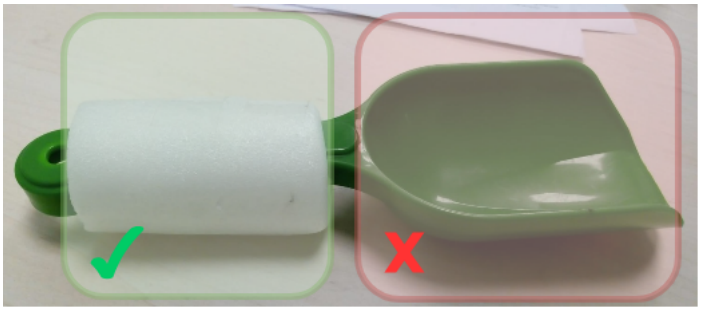


Figure 5.18. Grasp classification is done over this particular object: the grasp is considered good if it is in the white part with the correct orientation otherwise it is bad.

5.5 Gaussian Process (GP) for classification

In this first experiment we want to test the ability to teach a simple classification problem: given an object we define a good and a bad way to grasp it and we want the GP to solve the classification problem given a minimum training set.

We used as object for the experiment the scoop in Fig. 5.18.

Dex-Net, using the mask obtained with Mask-RCNN, returns a set of grasps on that object. A human operator evaluates each grasp, his criterion is based on the position and the orientation of the grasp: each grasp is considered good if it is in the white part with the correct orientation, otherwise it is bad.

After that, the prediction pipeline is so defined: the set is divided in train and validation set and each image x, of both datasets, is passed to the VAE which reduces the dimension from $|x|$ to $|z|$; the train set is used to evaluate the parameters of the GP, while the validation set is used to evaluate the results.

In this first experiment the accuracy reaced with a latent space of dimension 20 and a dataset of dimension 60 is 95%.

In Fig. 5.19 the confusion matrix for the validation set is showed, we can clearly observe how the two classes are correctly classified.

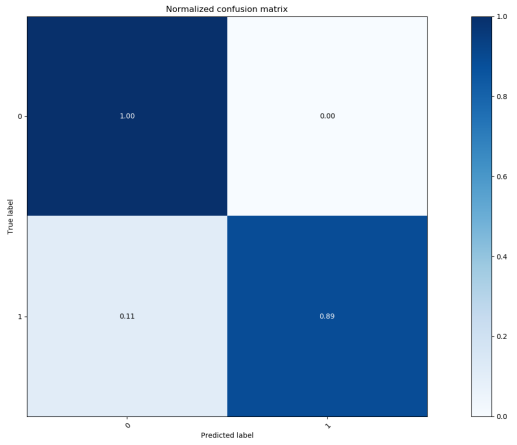


Figure 5.19. Classification confusion matrix.

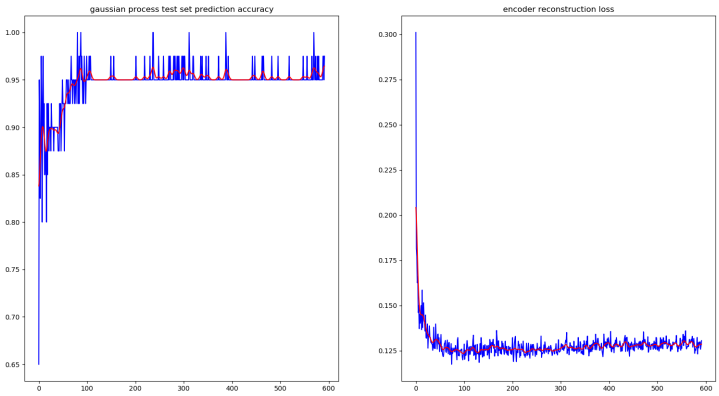


Figure 5.20. On the left the accuracy of the classification, on the right the the loss of the reconstruction of the VEA. Latent dimension $|z| = 20$.

In Fig. 5.20 we can observe how, while we train the VEA, the loss of the VAE is reduced (right plot); at the same time, the accuracy of the classifier increase (left plot). This result demonstrates that the results of the GP are highly correlated with the quality of the latent space representation.

In Fig. 5.21 it is possible to visualize how the accuracy increases while we augment the dimension of the training set: after 30 examples the accuracy is stationary around 90%. This plot demonstrates that our method can be used to teach human grasp preference with minimum size training set.

In Fig. 5.22 is plotted how the accuracy of the classifier changes with respect to the latent space dimension. We can observe that the maximum accuracy is reached for latent space dimension 30. This is due to the fact that the classifier needs a good grasp representation (so big enough) but, on the other hand, the GP works better for low dimensional vectors (so not too big).

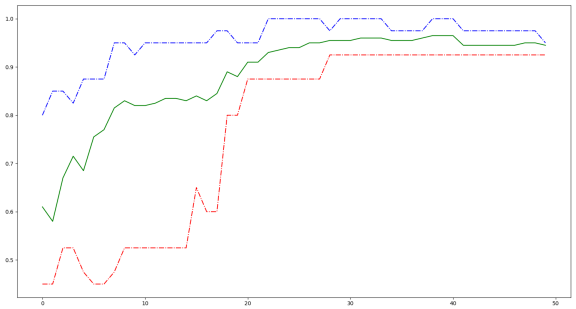


Figure 5.21. Plot of the accuracy with respect to training set size. In green the mean value, in red the minimum value, in blue the maximum value. We can observe that with a dataset with size bigger than 30 the accuracy is stationary over 90%.

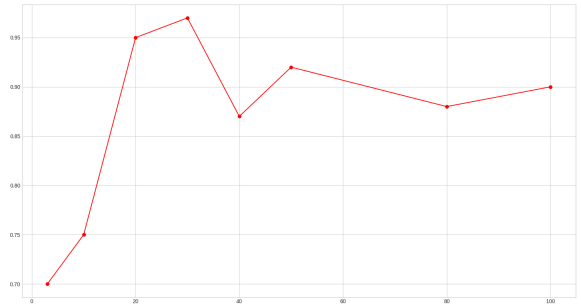


Figure 5.22. Plot of the classification accuracy changing latent space dimension.

From the solutions obtained we can infer that our model works for classification. In the next section we will present a more difficult task: predict evaluation function and correction function.

5.6 GP for regression

In the following experiments we want to teach to the planner more complex information:

1. the planner has to predict a target function assigned by the human operator, this function $Q_H(g)$ represents the evaluation of the grasp given by the human operator H.

2. the planner has to predict three target functions assigned by the human operator H, these functions $T_H(g) = (\Delta x(x), \Delta y(x), \Delta\theta(x))$ represent the transformation from the proposed grasp to a correct grasp. This information can be also interpreted as an error function over the proposed grasp.

As in the classification task we use as object to manipulate the scoop in Fig. 5.18: the grasps are considered good if they are in the white part with the correct orientation, otherwise they are considered bad.

In both cases a human operator H assigns the target functions $Q_H(g)$ and $T_H(g)$ to a dataset of one hundred images. The dataset is then divided in training and validation set. Each image x of both datasets is passed to the VAE which reduces the dimension from $|x|$ to $|z|$; the train set is used to calculate the parameters of the GP while the loss on the validation set is used to evaluate the results.

5.6.1 Human grasp evaluation $Q_H(g)$

After the human operator has defined the evaluation function $Q_H(g)$, the training set is used to evaluate the GP's parameters in such a way to predict that function.

With the same dataset as before, namely with a training set of size 60, the difference between true value and predicted value stations around the 20% of the true value.

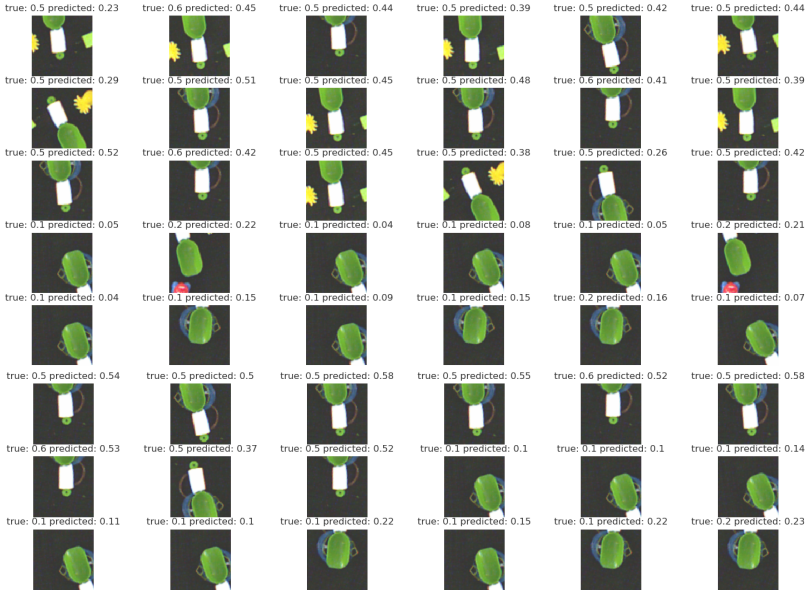


Figure 5.23. Examples of regression on grasp evaluation.

In Fig. 5.23 we can observe some results: over each image is presented the true value and the predicted one. We can see how, in most cases, high evaluations are associated to high prediction and, vice versa, low evaluations are associated to low predictions.

In Fig. 5.24 we can observe how, while we increase the size of the training set, the loss on the prediction decreases. It is interesting to notice that, with a dataset of 40 or more elements, the loss is always lower then 30% also in the worst cases (blue plot).

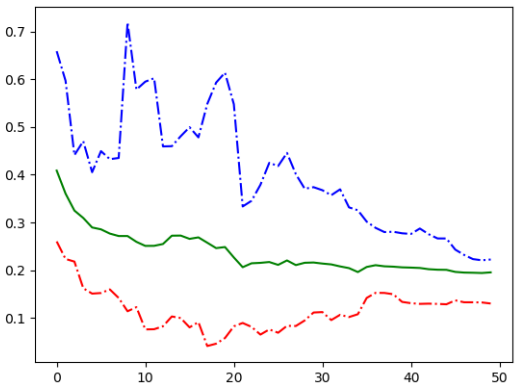


Figure 5.24. Plot of the regression loss while we increase the size of the training set. The target function is the evaluation given by the human operator; The latent space dimension is 20; The loss function is the difference between the true value and the predicted one divided by the true value.

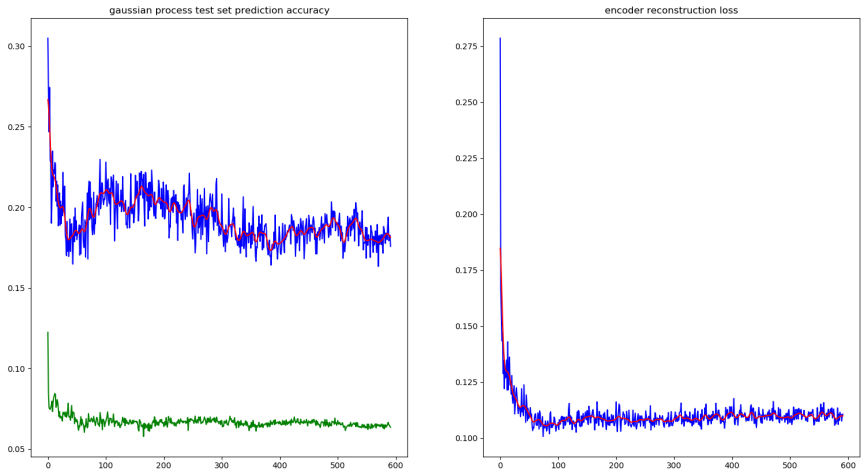


Figure 5.25. Plot of the regression loss while training the VAE. On the left the regression loss, on the right the reconstruction loss. In green is plotted the loss on the training set, in blue is plotted the loss on the validation set.

In Fig. 5.25 we can see how, while we train the VAE, the loss for the GP decreases, both for the train and for the validation set. From this graph we can infer that the GP prediction ability for the designed model is highly correlated with the quality of the latent space representation.

In Fig. 5.26 it is plotted the loss function with respect to the latent space dimension. From this graph we can infer that the minimum loss is reached with latent space dimension between 20 and 40, and, in particular from this graph, it results evident how the GP regressor works better for low dimensional vectors.

From the results obtained we understood that our model correctly imitates an human evaluation function, also with a minimum size training set. In next section we will try to increase the level of difficulty of our experiment: the target function to predict will be the correction over the proposed grasp.

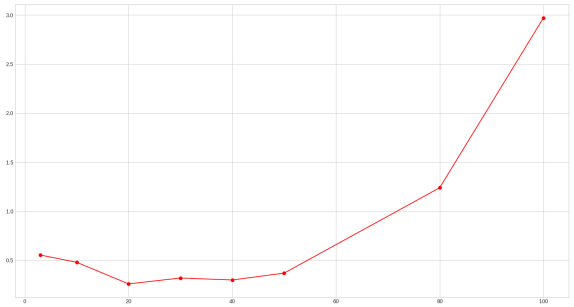


Figure 5.26. Loss of the GP trained for regression on grasp evaluation function with respect to the latent space dimension of the VAE.

5.6.2 Human grasp Transformation $T_H(g)$

In this experiment the GP tries to predict the human correction $T_H(g)$. This correction is a transformation composed by a translation on (x, y) and a rotation of an angle θ.

The human operator associates to each grasp the transformation. Then all the values are normalized dividing by the maximum possible transformation (50 pixels for the translation, π for the rotation). Grasps already in a good position present zero transformation. The dataset used for this experiment is always the same dataset composed by 60 grasp's images in the training set.

The results we obtain suggest: the predictor has problem when no corrections are present; when the correction is present the predictor, in most cases, has the correct direction but underestimate the correction; in many cases the translation is correct while the rotation is wrong.

An example of these results is available in Fig. 5.27.

We have also analyzed the individual corrections (x, y translations and θ rotation): we noticed that for the x and y corrections the loss seems related to VAE training but it has high values. In Fig. 5.28 it is possible to see how, while we train the VAE, both the losses for the X and the Y corrections decrease. On the other side we can notice that the losses have high values (in fact the GP predictor underestimate the corrections). These considerations suggest that these results could be improved with a more accurate latent space and a bigger dataset for training the GP.

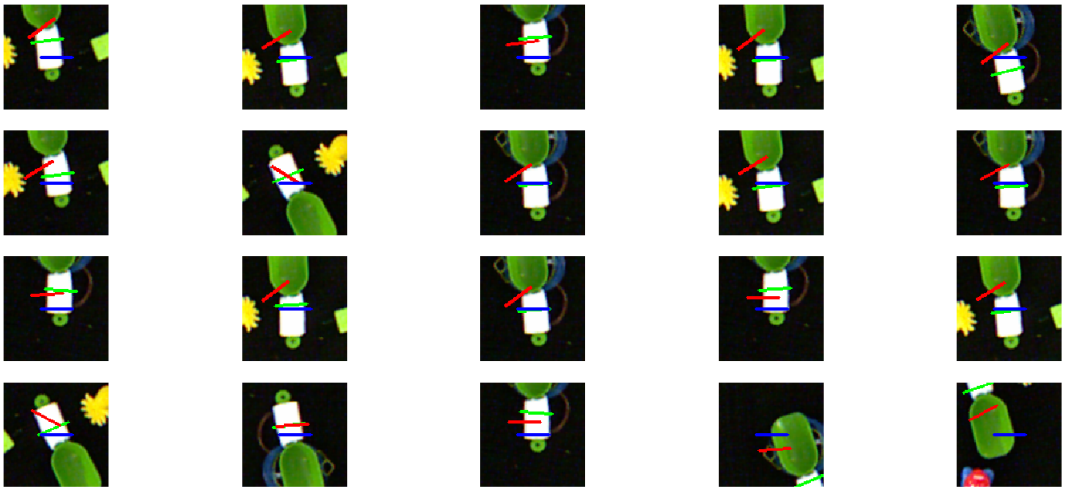


Figure 5.27. Examples of regression over the correction function: in blue the Dex-Net proposed grasp, in green the human correction, in red the predicted correction.

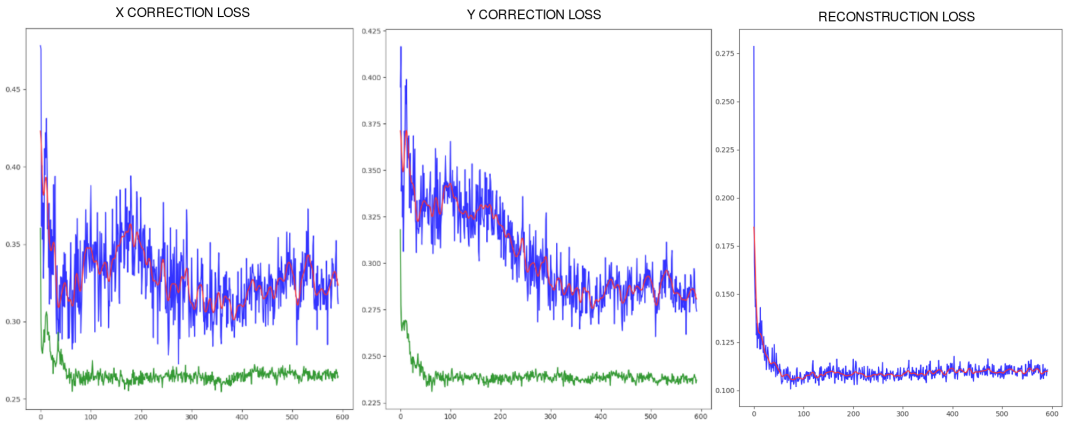


Figure 5.28. Plot of the regression loss while training the VAE. On the left the regression loss for the x correction, in the middle the regression loss for the y correction, on the right the reconstruction loss. In green is plotted the loss on the training set, in blue is plotted the loss on the validation set.

For the θ's correction instead, we noticed a lower correlation with the reconstruction (Fig. 5.29), but also, a lower loss. This result is good but suggests that maybe it will be more difficult to improve the θ correction.

Similar results come with different latent space dimensions, as we can see in Fig. 5.30. From this graph it is possible to notice how the GP regressor works better in a low dimensional space.

An aspect that is relevant is the fact that we do not need a perfect prediction from the GP regressor because the space of the grasps proposed by Dex-Net is discrete; this means that if the correction is partial we can always choose the closest grasp among those proposed by Dex-Net. In future works we will try to improve the results for this third experiment training more the regressor and considering this aspect.

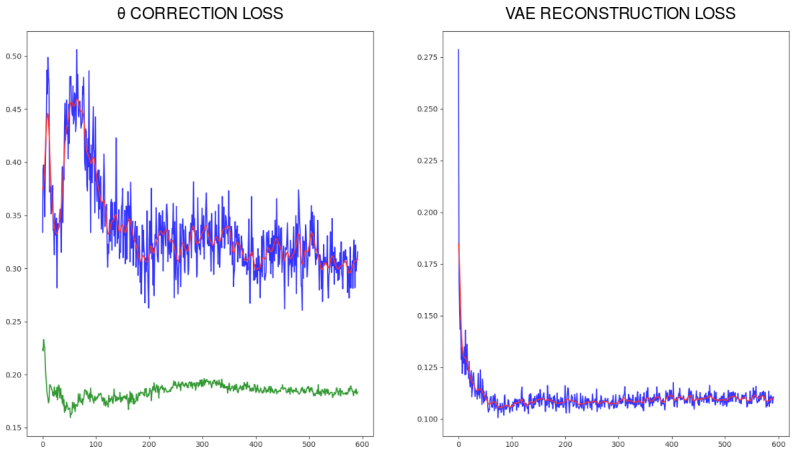


Figure 5.29. Plot of the regression loss while training the VAE. On the left the regression loss for the θ correction, on the right the reconstruction loss. In green is plotted the loss on the training set, in blue is plotted the loss on the validation set.

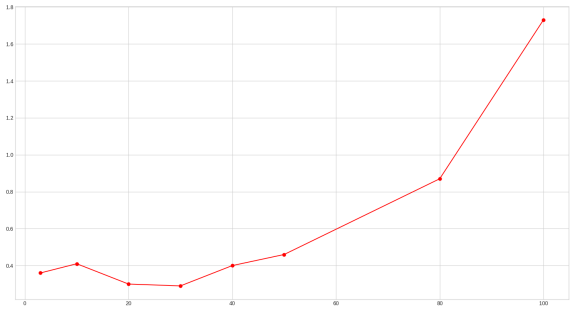


Figure 5.30. Loss for regression on correction changing the latent space dimension.

5.7 Task executions

One time the planner selected the best grasp following the human preferences the robot executes the trajectory for picking and placing the desired object.

This is the standard trajectory:

1. Robot starts from the neutral position. This position has been chosen in such a way to not disturb the camera's images acquisition (Fig. 5.31).
2. Robot collocates over the grasp with gripper facing down (Fig. 5.32).
3. Opening the gripper.
4. Robot executes linear Cartesian's path perpendicular to the table and downwards.
5. Closing gripper (Fig. 5.33).
6. Robot executes linear Cartesian's path perpendicular to the table and upwards (Fig. 5.34).
7. Object is released in a second location.

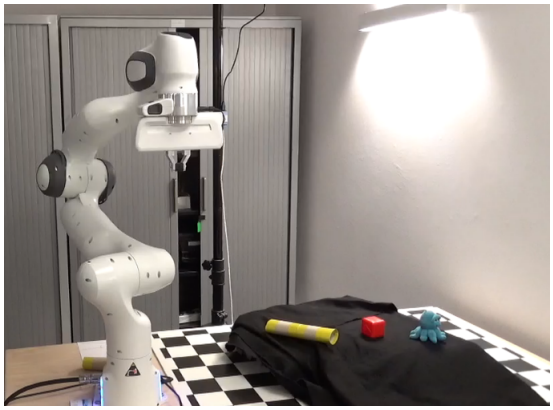


Figure 5.31. Robot in neutral position.

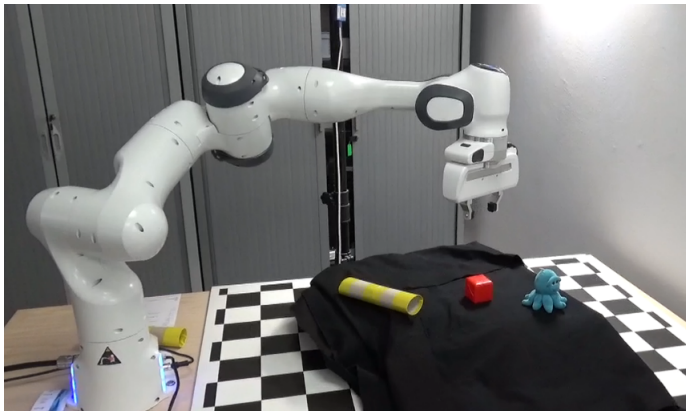


Figure 5.32. The end effector is over the object facing down.

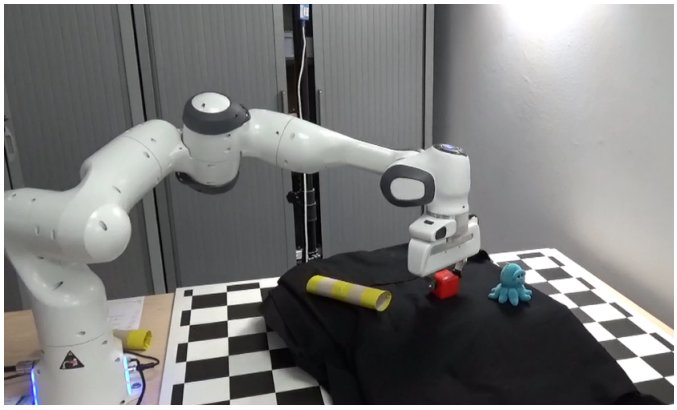


Figure 5.33. Robot executes the grasp.

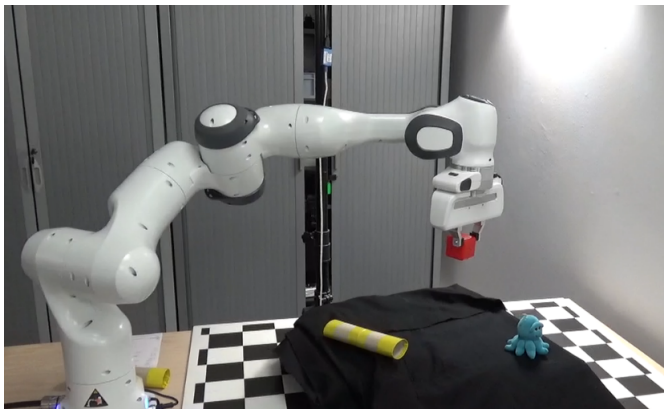


Figure 5.34. The object is raised and transported.

Chapter 6

Conclusions

Autonomous grasp planning is an active research area. Most recent works use two main approaches to solve this kind of problem: analytical approach requires a deep knowledge of the environment and of the object to be manipulated; data driven approach requires less engineering work but needs a huge amount of manipulation data.

Dex-Net uses the advantages of both the formulations, for this reason it is considered one of the state-of-the-art methods in grasp planning.

In our work we try to further improve Dex-Net's ability inserting human suggestion in the grasp decision process. The main contribute of our work is an algorithm which can transfer human knowledge with a minimum amount of data. The structure of our planner is composed by a Mask-RCNN node for object selection and a Gaussian Process regressor for grasp evaluation. Since Gaussian Process works better with low dimensional data we used a Variational Autoencoder (VAE) to reduce the size of the grasp representation.

I also present an alternative architecture to the classical VAE, namely the Variational Autoencoder with Intrinsic Classification (VAE-ic). The latent space obtained with this network has the ability to have good classification results also with bigger recontruction loss.

Our architecture for grasp evaluation has been tested in three experiments of increasing difficulty: grasp classification, grasp evaluation and grasp correction.

In grasp classification the predictor reaches an accuracy over 90% with a training set of size 30 or bigger. In grasp evaluation the predictor is able to associate high evaluation to good grasp, low evaluation to bad grasps with a minimum size dataset. In grasp correction the regressor has some limits but presents also promising results.

In future works we will further investigate these architecture trying to improve these results.

Bibliography

[1] Mahler, Jeffrey and Pokorny, Florian T and Hou, Brian and Roderick, Melrose and Laskey, Michael and Aubry, Mathieu and Kohlhoff, Kai and Kröger, Torsten and Kuffner, James and Goldberg, Ken., "*Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards*", IEEE International Conference on Robotics and Automation (ICRA), 2016

[2] Mahler, Jeffrey and Liang, Jacky and Niyaz, Sherdil and Laskey, Michael and Doan, Richard and Liu, Xinyu and Ojea, Juan Aparicio and Goldberg, Ken., "*Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics*", Robotics: Science and Systems (RSS), 2017

[3] Mahler, Jeffrey and Matl, Matthew and Liu, Xinyu and Li, Albert and Gealy, David and Goldberg, Ken., "*Dex-Net 3.0: Computing Robust Robot Suction Grasp Targets in Point Clouds using a New Analytic Model and Deep Learning*", arXiv:1709.06670, 2017

[4] Mahler, Jeffrey and Matl, Matthew and Satish, Vishal and Danielczuk, Michael and DeRose, Bill and McKinley, Stephen and Goldberg, Ken., "*Learning ambidextrous robot grasping policies*", AAAS Science Robotics, 2019

[5] Bohg, Jeannette and Morales, Antonio and Asfour, Tamim and Kragic, Danica, "*Data-driven grasp synthesis — a survey*", IEEE Transactions on Robotics, 30.2,(2013), pp.289-309

[6] Carlo Ferrari, John F. Canny., "*Planning Optimal Grasps*", IEEE International Conference on Robotics and Automation, 1992

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick., "*Mask R-CNN*", arXiv:1703.06870, 2017

[8] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, P. Dollár, "*Microsoft COCO: Common Objects in Context*", arXiv preprint arXiv:1405.0312 (2015)

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, "*ImageNet Large Scale Visual Recognition Challenge*", International Journal of Computer Vision, 2015

[10] Mousavian, Arsalan and Eppner, Clemens and Fox, Dieter., "*6-DOF GraspNet: Variational Grasp Generation for Object Manipulation*", arXiv preprint arXiv:1905.10520, 2019

[11] Bohg, Jeannette and Morales, Antonio and Asfour, Tamim and Kragic, Danica., "*Data-driven grasp synthesis — a survey*", IEEE Transactions on Robotics 30.2.289–309, 2013

[12] Rasmussen, Carl Edward and Williams, Christopher K. I., "*Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*", 026218253X The MIT Press, 2005

[13] Mahler, Jeffrey and Matl, Matthew and Satish, Vishal and Danielczuk, Michael and DeRose, Bill and McKinley, Stephen and Goldberg, Ken., "*Learning ambidextrous robot grasping policies*", Science Robotics 4.26.eaau4984 AAAS, 2019

[14] Saut, Ivaldi, Sahbani, Bidau, "*) Grasping objects localised from uncertain point cloud data.* ", Robotics and Autonomous Systems, 62(12): 1742-1754 (2014).

[15] Jacopo Aleotti, Stefano Caselli, "*Part-based robot grasp planning from human demonstration*", IEEE International Conference on Robotics and Automation (2011)

[16] Yun Lin, Yu Sun, " *Robot grasp planning based on demonstrated grasp strategies* ", International journal of robotics research, Volume: 34 issue: 1, page(s): 26-42 (2014)

[17] Kappler, Daniel and Bohg, Jeannette and Schaal, Stefan., "*Leveraging big data for grasp planning*", 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015

[18] Rubert, Carlos and Kappler, Daniel and Morales, Antonio and Schaal, Stefan and Bohg, Jeannette., "*On the relevance of grasp metrics for predicting grasp success*", 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017

[19] Roa, Máximo A. and Suárez, Raúl, "*Grasp quality measures: review and performance*", Autonomous Robots, (2015), 38 pp.65–88

[20] Michael A. Rosen, Eduardo Salas, Davin Pavlas, Randy Jensen, Dan Fu, Donald Lampton, "*Demonstration-Based Training: A Review of Instructional Features*", The Journal of the Human Factors and Ergonomics Society 52(5):596-609 (2010)

[21] Kalashnikov, Dmitry and Irpan, Alex and Pastor, Peter and Ibarz, Julian and Herzog, Alexander and Jang, Eric and Quillen, Deirdre and Holly, Ethan and Kalakrishnan, Mrinal and Vanhoucke, Vincent and others., "*Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation*", arXiv preprint arXiv:1806.10293, 2018

[22] Liangliang Ren, Jiwen Lu, Jianjiang Feng, Jie Zhou., "*Uniform and Variational Deep Learning for RGB-D Object Recognition and Person Re-Identification*", IEEE Transactions on Image Processing, Oct. 2019

[23] D. Wout, J. Scholten, C. Celemin, J. Kober, "*Learning Gaussian Policies from Corrective Human Feedback*", Uncertainty in Artificial Intelligence (UAI) Conference (2019)

[24] C. L. Isbell, C. Shelton, M. Kearns, S. Singh, and P. Stone, "*A social reinforcement learning agent*", inProc. of the 5th Intl. Conf. on Autonomous Agents, pp. 377–384, 2001.

[25] H. S. Chang, "*Reinforcement learning with supervision by combining multiple learnings and expert advices*," inProc. of the American Control Conference, 2006.

[26] P. M. Pilarski, M. R. Dawson, T. Degris, F. Fahimi, J. P. Carey, and R. S. Sutton, "*Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning*", inProc. of the IEEE ICORR, pp. 1–7, 2011

[27] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L. Isbell, and Andrea Thomaz, "*Policy Shaping: Integrating Human Feedbackwith Reinforcement Learning*", Advances in Neural Information Processing Systems 26, pp. 2625–2633 (2013)

[28] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, M. Riedmiller, "*Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards*", CoRR, abs/1707.08817, (2017)

[29] A. Singh, L. Yang, K. Hartikainen, C. Finn, S. Levine, "*End-to-End Robotic Reinforcement Learning without Reward Engineering*", arXiv preprint arXiv:1904.07854 (2019)

[30] F. Ficuciello, A. Migliozzi, G. Laudante, P. Falco and B. Siciliano., "*Vision-based grasp learning of an anthropomorphic hand-arm system in a synergy-based control framework*", Science Robotics, Jan. 2019

[31] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine., "*Time contrastive Networks: self supervised learning from multiview observation*", arXiv:1704.06888, 2018

[32] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, Konstantinos Bousmalis., "*Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks*", arXiv:1812.07252, 2019

[33] Sergey Levine, Peter Pastor, Alex Krizhevsky, Deirdre Quillen., " *Learning Hand Eye coordination for robotic grasping with deep learning and large-scale data collection*", arXiv:1603.02199, 2016

[34] David Wang, David Tseng, Pusong Li, Yiding Jiang, Menglong Guo, Michael Danielczuk, Jeffrey Mahler, Jeffrey Ichnowski, Ken Goldberg., "*Adversarial Grasp Objects*", International Conference on Automation Science and Engineering (CASE), 2019

[35] Khaled Elgeneidy, Peter Lightbody, Simon Pearson, Gerhard Neumann., "*Characterising 3D printed Soft Fin Ray Fingers with Layer Jamming for Delicate Grasping*", IEEE International conference on soft robotics (RoboSoft), 2019

[36] D. P. Losey, M. K. O'Malley, "*Including Uncertainty whenLearning from Human Corrections*", 2nd Conference on Robot Learning (CoRL 2018)

[37] H. HeH. Daumé III, J. Eisner, "*Imitation learning by coaching*", Advances in neural information processing systems 4:3149-3157 (2012)