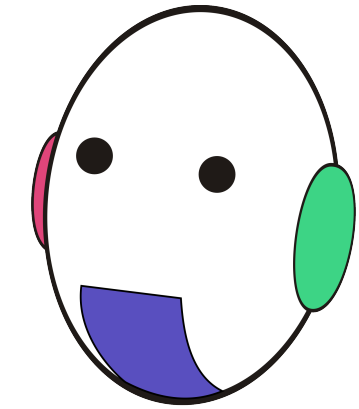

Master's degree internship thesis

Multi-modal perception for in-hand robot manipulation tasks

Auteur:
Camille TAGLIONE

Supervisor:
Carlos-Manuel MATEO-AGULLO
David FOFI

University of Burgundy Le Creusot,
MSc in Computer Vision VIBOT Laboratory

· 2020 ·

Abstract

This topic deals with the study of the pose of an object in space in order to manipulate it with a robotic hand. This topic is very large and uses a machine learning approach (deep learning) and not an approach based on object models. We will start by explaining what object pose is, and why multimodal solutions are often the most effective through the occlusion problem. We will then look at many methods that compute this pose, and we will study the commonalities between all these methods as well as the things that set them apart from each other. This topic consists of many steps: image segmentation, tactile data extraction, adaptation of data of different types for common use, use of features through different neural networks and pose estimation itself. This is one of the reasons why I really enjoyed this topic, through the in-depth study of this topic I was able to study many aspects present in my Master Vibot training in order to improve my knowledge on these aspects as well as to improve my skills for the realization of a method and tools for this topic. As mentioned previously the subject is vast with 5 months to cover it so I had to make choices and make a method that uses only color data and depth data to estimate the pose. We will see through this thesis, my understanding of the subject, the different possible approaches on this subject, the problematic that we can draw from it, my study of works on similar subjects which use a tactile detection system, a tactile detection system or both at the same time for better results, the method implemented during this internship, the constraints, the tools developed, the different experiments and manipulations performed, the results obtained and a study of these results.

Contents

List of Figures

Acknowledgments

During this internship I received a lot of help so I would like to thank all the members of the VIBOT/ImViA laboratory who helped me and who were available when I needed help and especially my supervisor Dr. Carlos-Manuel MATEO-AGULLO who knew how to guide me, help me and motivate me all along the internship period.

Chapter 1

Introduction

1.1 Motivation for the subject

The topic of this 5 months master internship is *Multi-modal perception for in-hand robot manipulation task*. My main motivation for this topic was the fact that it is extremely vast. Indeed this subject is vast for two reasons: the estimation of the pose of an object is a process that requires many steps: the segmentation of images for the management of color images and depths. The extraction of tactile signals is the management of tactile data and the fusion of data of different type, the use of features that leads to the pause by neural networks and the estimation of the pose. The other reason why this topic can be considered as vast is the numerous approaches on this topic, through the methods that use color and depth data, the one that uses tactile data only and the one that uses all three types of data. Through each method we see the databases, the common points between all these methods, but also their differences. All these steps and approaches make the topic vast, complex and exciting. This topic is vast for another reason, through my training three main poles: robotics, image and AI, and this subject has a link with these three poles, indeed the imaging through the segmentation and visualization of results, robotics for the seizure, manipulation of an object and the use of tactile data and AI through the calculation of the pose and the management of all data. It is all these reasons that have greatly motivated me on this subject.

1.2 Explanation of the subject

In this course we present the use of different sensors and their data of different types to assist the manipulation of objects with a robotic hand. To manipulate an object, it is necessary

to visualize it(and in some cases its environment). We will therefore use a visual system to estimate the 6D pose of the object to be manipulated. This topic can be categorized as a deep learning approach and not an object model based approach like most of the methods that will be studied. We will see in chapter II, that we can find a problematic around the use of tactile data. We will also see the choices that have been made to cover the subject. When we talk about 6D pose, it is a way of locating and representing an object in space from a reference frame. When we talk about 6 Dof, we mean the number of information (degree of freedom) we have. Indeed, the 6D pose is a way to give the orientation and the position of an object through two vectors: three rotation coordinates for the orientation and three translation coordinates for the position. In some cases we will study the 7 Dof pose or the 9 Dof pose which represents a 6 Dof pose to which we add the size (represented by three values) but where the rotation is represented by two or three coordinates. To grab an object with a robotic hand, you must have a physical model of the object, the *model-based* methods which uses 3D models of the object to help grab it, and the *model-free* methods, which do not have the model of the object in these cases we can use depth information to obtain a point cloud that can be used as a partial model but cannot be used as a ground-truth. This is why most 6D pose estimation methods lay one uses the RGB-D sensor (red, green, blue and depth). In addition to the three color channels, we have the depth channel that corresponds to the distance between the camera and the object, which gives us the location of the point that corresponds to the same pixel. By merging the position of points, the location of the pixels obtained a model for the object on the image. Estimating the 6D pose of an object is an important task in robotics. Indeed, one needs to know its position, orientation and model in order to manipulate (or grasp) it with the help of a robotic arm. To add precision to the estimate of the 6D pose we will add an others detection mode, tactile sensing, for which data that will come from the touch sensor put on the hand that will catch the object. We will see more details in the Problem Definition section. The more precise and accurate the estimation, the more efficiently the robotic arm can work by grasping the object often and without letting it slip.

1.3 Openings on the subject

This topic is directly related to my supervisor's PhD thesis and postdoctoral works and would be the premise of a work on non-rigid object manipulation. Non-rigid objects(tissus, clothes, ...) are deformable which implies two additional difficulties:

- the first one is that their shape is not fixed which makes it impossible to catch the object without knowing its shape. In these cases, we can develop a method of estimation of the 9D pose (we add three coordinates which will correspond to the shape of the object which

will be represented by a rectangular parallelepiped) which will not be enough to grasp the object.

- the second difficulty is that under the gripping force of the robotic hand the object will change its shape, which leads to the same problems and makes the manipulation very complicated.

1.4 Contributions

During this internship, I studied many methods that use either visual data detection or tactile data detection and the methods that use these two means of detection. I have created a 6D pose estimation method that is a fusion of the DenseFusion [14] and DualPoseNet [6] methods, which uses the architecture of the DenseFusion method [14] but replaces the global feature modules with a pose encoder that uses the spherical space of the DualPoseNet method [6]. I have also created a module to visualize the results on the images in the form of video and experiments of sensor tests and reinforcement learning on the Isaac Gym environment of NVIDIA.

1.5 Presentations from upcoming parties

In this master thesis, we will first look at the problem that has been studied by defining a problem that explains the use and results obtained with the use of tactile data. In a second time a presentation of the state of the art, of all the methods that use visual data through RGB-D images, the tactile data only and the sensor used in these cases and well extended the methods that use these two types of detection to achieve the pose estimation. While studying all these methods, I have little developed different method projects of my own.

In a third step, I will present the draft methods that have been developed to solve the problem in our way and the final method that will be chosen and why.

In the fourth part, we will see how I implemented this method and the experiments carried out around this method such as the study of reinforcement learning with the Isaac Gym environment, the visualization tool created.

Chapter 2

Problem Definition

2.1 Introduction

Today there are many methods of estimating 6D pose that give very good results. The most recent methods use machine learning approaches, concretely deep learning, that is much more efficient than rule-base methods that use 3D models of the objects under study. In many approaches to pose estimation, these methods are based on visual sensors (RGB-D camera) that usually provide data from 4 channels, three channels provide color and one provides depth. One of the problems of 6D pose estimation using visual methods is occlusion and especially when experimenting with grabbing or hand hiding the object from the camera, as some objects are not fully visible from the camera's single point of view. Other everyday objects in the databases studied for this type of exercise exhibit what is called self-occlusion, for example a mug, the handle hides part of the mug's shape. It is for all these reasons that the use of tactile data allows to compensate this defect brought by the occlusion.

Figure 2.1: Mug with self occlusion

2.2 Suggested solution

Against the problem of occlusion and to address this source of uncertainty, many methods have been developed: some propose to multiply the points of view to have an image of the object from several angles so that the points of the image correspond, or in our case add a method that uses tactile data to improve the results, the tactile data that would allow to partially replace the missing data when manipulating the grasp by the same hand that provides this tactile data. The 6D pose estimation methods that only use tactile data are very rarely used or not used at all because the tactile data are very sparse, they are only located where the sensor is in contact. This can allow us to obtain very dense local data but on small surfaces of the object. Indeed, depending on the sensors, the amount of points obtained can vary but they will always remain point clouds centered on the locations of the contacts between the sensors and the object in question. The use of both methods (simultaneously or in refinement of each other) gives better results on the databases used in this exercise and even more so on the one emphasized on the occlusion (LM-O (Linemod-Occluded)). In most of the works studied in this project, the tactile data are coupled to the depth data obtained by the RGB-D camera. Indeed, after processing, the two depth data are combined. Indeed, after processing, both the depth data and the tactile data are point clouds that represent the surface of the object studied.

2.3 Restriction

This internship subject being extremely vast, we had to make choices in the axes on which we were going to concentrate for the realization of a method. In this master thesis, we will therefore deal in more detail with the visual part of the method, that is to say that the method developed would use color data and depth data to estimate the 6D pose of objects on a scene. However, the state of the art and the study of different methods of using tactile data will be presented, but they will not be implemented in the final method and therefore the final results will not depend on the tactile data.

Chapter 3

Stat of the art

To begin this project, I studied a large number of articles during the first months of the internship, indeed the initial idea was to publish one or more articles during the period of the training course (which requires a great knowledge of the methods studying the same problematic), the topics covered by these articles revolve around three main topics: pose estimation methods that use only visual detection, methods that use only tactile detection and methods that use both detection systems simultaneously or not. As the internship progressed, I decided to concentrate on the methods that use only visual detection because time constraint. But the study of the other works allows me to see at what points the addition of tactile detection can improve the estimates and how one can use this data. Another possibility that was discussed at the beginning of the internship is the use of simulators (normally used in reinforcement learning works) to generate rich datasets. Unfortunately, I let this aspect of the work as a line of research and future works due to time constraints.

3.1 Semantic Segmentation

Almost all pose estimation methods that use visual data, start with a segmentation step. Particularly, nowadays this is often addressed from a semantic perspective.

Object classification, detection or segmentation are common applications of deep learning, there are two types of segmentation techniques: Semantic segmentation is the process of classifying each pixel belonging to a particular label. Instance segmentation gives a unique label to each instance of a particular object in the image. Semantic segmentation is the method of assigning a class label to each pixel of an image. There are three types of segmentation methods: the use of a sliding window that divides the input image into many small frames of the image;

the use of fully convolutional networks; and the use of oversampling and undersampling of the image.

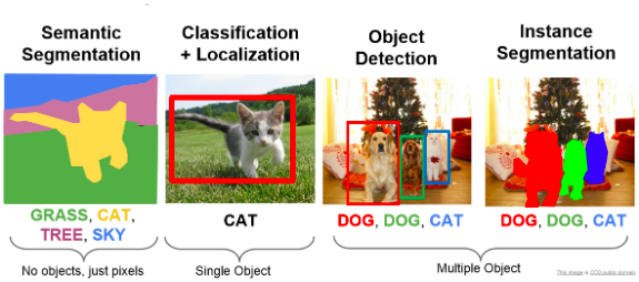

Figure 3.1: Semantic segmentation

In the literature we can find several approaches to implement semantic segmentation. The following summarize the most popular approaches. The fully conventional network consists of a stack of convolution layers to obtain a final segmentation map. The network starts with a process of downsampling and upsampling via deconvolution and then merges the outputs of the different samplings. At each layer of sampling convocation, we obtain more accurate information on the image which discards the classes too far away. Then we use the oversampling layers to obtain the class map by pixel. Another possible method is the U-Net which is also a fully convolutional network. It contains an encoder that uses the subsampling layers of the input image into a feature map, and the decoder uses the oversampling feature map. The advantage of the U-Net is the ability to shortcut directly to the decoder at each layer grouping.

The R-CNN mask is the most common segmentation method used in the papers studied for pose estimation.

R-CNN or RCNN is the acronym for Region-Based Convolutional Neural Network The fast R-CNN uses selective search to generate regions of interest, while the faster R-CNN uses a "region proposal network" (RPN). RPN is simply a neural network that proposes multiple objects in the image. Faster R-CNN uses an attention mechanism on a fast R-CNN architecture.

Mask R-CNN was developed from Faster R-CNN, a convolutional region-based neural network.

Mask R-CNN was built from Faster R-CNN. While Faster R-CNN has two outputs for each candidate object, a class label and a bounding box offset, Mask R-CNN is the addition of a third branch that outputs the mask of the object. The additional mask output is separate from the class and box outputs, requiring the extraction of a much finer spatial layout of an object.

Mask R-CNN adds to Faster R-CNN an object mask prediction branch (region of interest) in parallel with the bounding box recognition branch.

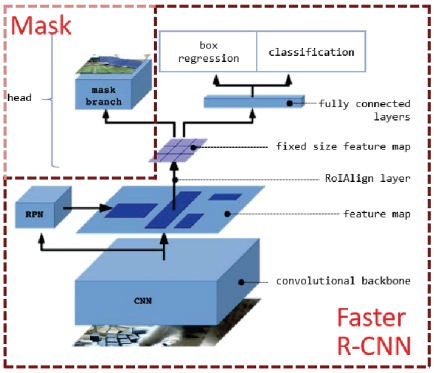

Figure 3.2: Mask R-CNN architecture

3.2 Methods using only the visual channel

The 6D pose estimation methods that inspired me the most for the creation of my final method are, of course, the methods that use only visual detection. Most of these methods use the color channel to perform segmentation and they use depth data. These methods are the following:

- Deep Object Estimation for Semantic Robotic Grasping Housefold Objects [11]
- DualPoseNet: Category-level 6D Object Pose and Size Estimation Using Dual Pose Network with Refined Learning of Pose Consistency [16]
- DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion [14]
- NeRF-Pose: A First-Reconstruct-Then-Regress Approach for Weakly-supervised 6D Object Pose Estimation [5]
- PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes [17]

And indirectly, since our architecture is pretty much the same as for the DenseFusion method [14] which itself uses modules of the following methods:

- PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation [18]
- PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation [8]

3.2.1 Deep Object Estimation for Semantic Robotic Grasping Housefold Objects [11]

The method presented in this article [11] uses synthetic data to train the network, which allows to obtain a large amount of already labeled data. The common point with our subject is the use of the YCB-Video dataset. To achieve good performance in this area, they use a combination of random and photorealistic data. This paper uses one-shot learning to learn images (using only a small number of images to categorize an object).

First, the network will estimate the belief map of all objects in the image scene. To obtain the pose and detect the objects, they extract the peaks from the belief map, then associate the detected peaks with the centroids of the objects, and finally compare the detected vector field at the peaks with the direction of the peaks to the centroids. Finally, they use the PnP algorithm to obtain the pose. The purpose of this algorithm is to estimate the position of the camera with respect to a 3D point in the scene and their 2D projections in the image.

Figure 3.3: Type of data

3.2.2 DualPoseNet: Category-level 6D Object Pose and Size Estimation Using Dual Pose Network with Refined Learning of Pose Consistency [16]

The DualPoseNet method [16] presented in this article is one of the two methods that inspired me the most for this project (with Dense Fusion [14]), so its study will be a bit more detailed than the others. Indeed, the encoder used in this method will be part of the method proposed for this project. This method is used to estimate (only with visual detection) both the 7D pose (rotations with only yaw angle, 3 translations and 3D size). It uses two pose decoders (one implicit and one explicit) connected in parallel to the output of a pose encoder. Both

decoders predict the pose but not with the same method which allows to supervise the learning of the encoder. The encoder is built using spherical convolution and a spherical fusion module, the study of the encoder will be a bit light in this part because it will be soon more detailed in the method chapter.

The different steps of the method are the following: define the canonical pose of an object in a categorical way (and align the instances), perform the deep learning of the rotations is part of the 2D image (easier for the translations), estimate the pose of an object without the presence of models using only the visual detection.

The two decoders running in parallel use these enhanced features to reconstruct a partial canonical pose point cloud to estimate the pose. They refine the pose consistency between the two decoders using the self-adaptive term. The method starts with a segmentation (MaskR-CNN). The segmented results are adjusted in touch and color are used to train the encoder which learns the characteristic representations of the pose, then they use the explicit decoder to estimate the pose and they use in parallel the implicit decoder to create the canonical points from the observed points. In the training phase, the method improves pose consistency between the two decoders, which improves pose estimation.

The explicit decoder consists of 3 sub-arrays of multilayer perceptrons that are used to regress the pose of the region of interest. To decode the implicit pose, they establish an affine transformation of the point cloud P into a canonicalized point cloud Q. They can then use this canonical prediction to obtain the three pose components (R, t, s) using Umeyama's algorithm to solve the alignment problem.

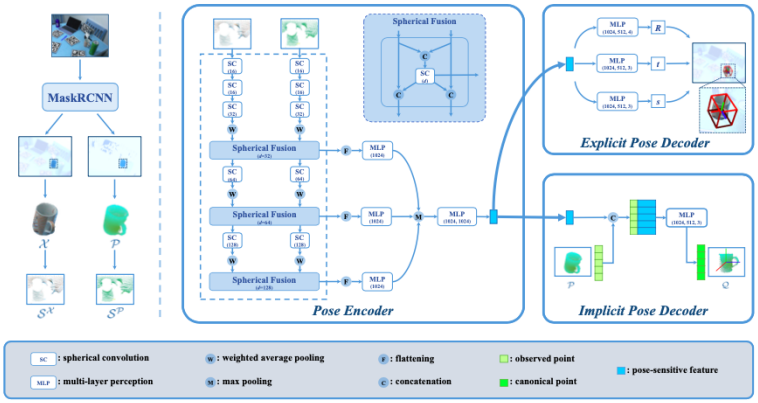

Figure 3.4: DualPoseNet framework

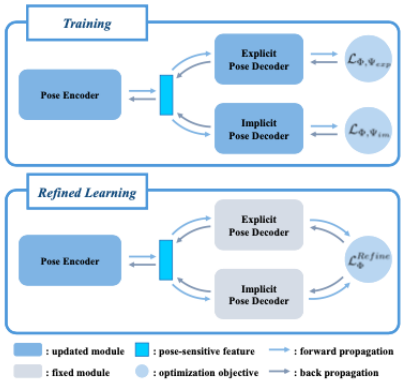

Figure 3.5: DualPoseNet methods for training and for refined learning

3.2.3 DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion [14]

The article presented here is the one that with the DualPoseNet article inspired me the most, indeed the final architecture of the method to realize my project is greatly inspired by that of this article, we will not see the details of the architecture in this part, they will be detailed in the method part. In this method the data of different types are processed separately to be merged to obtain the dense data integration per pixel which is used to estimate the pose.

In this paper, the presented method uses an end-to-end deep network. The objective here is to integrate and merge color data and point clouds. This method allows the reuse of local geometric features for the model that manages the occlusion.

One of the problems with this method is that all methods that use these two types of channels in a complementary way and that these data are heterogeneous. The DenseFusion method presents a structure to manage the data of different types separately, it also uses a dense fusion network per pixel that allows to merge the two types of data. Finally, the framework also contains a differentiable iterative refinement module (trainable along with the rest of the architecture) to estimate the pose.

3.2.4 NeRF-Pose: A First-Reconstruct-Then-Regress Approach for Weakly-supervised 6D Object Pose Estimation [5]

In this paper [5], the method presented uses a weakly supervised pipeline that uses reconstruction. The goal is to reconstruct in 2D from multiple views, then refine the result with a pose regression network (PRN) to connect pixels with points in the reconstructed model. A

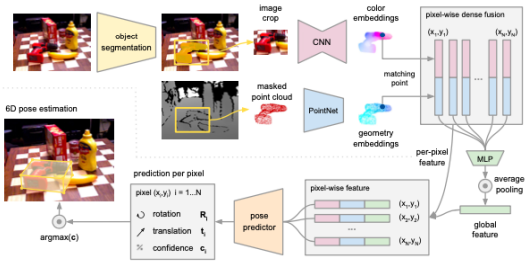

Figure 3.6: DenseFusion Network Architecture

PnP+RANSAC algorithm is used to obtain an accurate result.

The method proposes to recover the 3D representation of objects from training images with weak labels: 2D segmentation masks, Relative camera positions (obtained by methods such as: Structure from Motion (SfM), Simultaneous Localization and Mapping (SLAM), visual inertial odometry, a marker array).

With this representation, they supervise the pixel regression, the correspondences between the training images and the representation. It starts by using real images with 2D segmentation masks and camera poses, obtaining an implicit representation of the 3D model, It is used to create object correspondence maps, which are used to train a pose regression network, once regressed, the correspondences pass into a PnP+RANSAC algorithm to iteratively estimate the pose.

The pose estimation is done in three steps; the first step is the trained 2D detector (CNN separated inspired by DPoD and CDPN) on images overlapping the studied object, they use ResNet as encoder and the decoder consists of four layers of oversampling . The second step, they then use a pose regression network is trained (also separate CNN) on the images to predict the coordinates of the object under study and to predict the segmentation mask. The third step is optimization, and this is the part done in this paper (PnP+RANSAC used in NeRF) .

3.2.5 PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes [17]

PoseCNN is one of the reference methods of pose estimation. This method starts by predicting the translation of an object using the coordinates of its center in the image thanks to the distance between it and the camera and it predicts the rotation of the object by regression to quaternions. One problem concerns symmetrical objects that may appear to have the same orientation from different viewpoints.

To begin with the network classifies the pixels of the image to assign them an object class

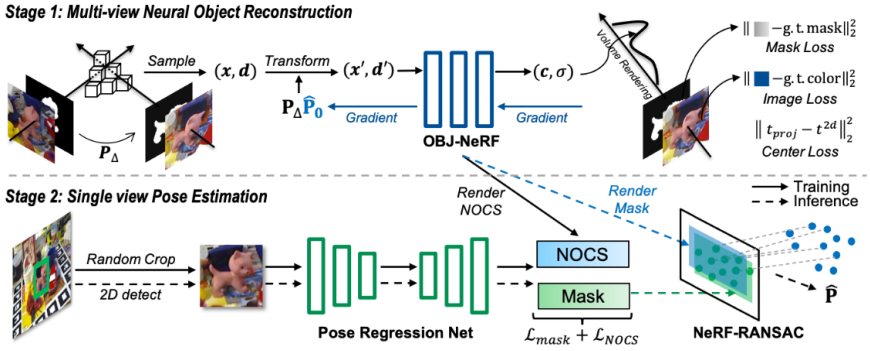

Figure 3.7: Nerf-Pose pipeline

(semantic labeling) and then uses as input two feature maps (obtained by feature extraction) which will be summed. The sum of the maps is given the image size, and finally through a conventional layer, a labeling score is generated (a channel of the image size for each class in output).

To estimate the translation of an object in the real world (3D), they need to predict the position of the center of the object using a regression network(inspired by an implicit shape model (ISM)) that regresses to the direction of the center for each pixel of the image. Two characteristic layers are added to the network (smoothed L1 loss, Hough's vote) that allow to obtain for each class the vote score that will lead to the probability that the location is the center of the object of this class (they select the center according to the one that has the maximum score.)

To get the rotation of an object in the real world, they use bounding boxes (obtained by Hough's voting layer), they apply clustering layers and fully connected layers that give us the rotation per quaternion as output for each class. To train the regression network, they use a Pose Loss function that measures the MSD between the estimated rotation points and those in the model.

3.2.6 PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation [18]

This article presents the method *PointFusion* which detects objects from images and 3D point clouds. Like our method, they use separate color and depth data through a CNN and PointNet architecture. And the results are then merged and used to predict the 3D boxes and their confidence.

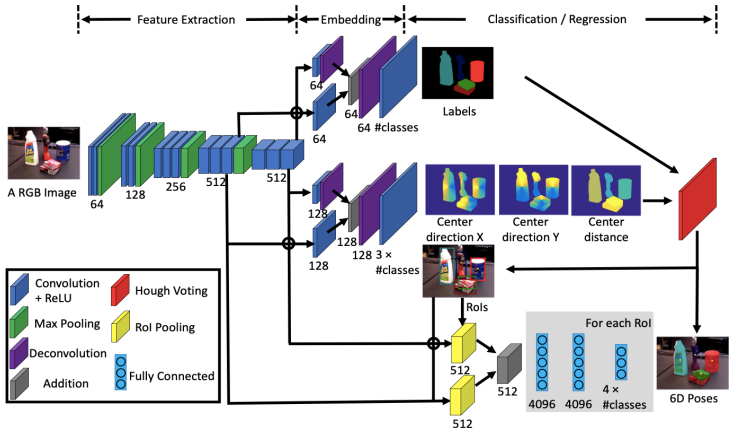

Fig. 2. Architecture of PoseCNN for 6D object pose estimation.

Figure 3.8: Nerf-Pose pipeline

The architecture presented here has been designed to work regardless of the number or range of 3D sensors used. The network created can easily merge heterogeneous color and depth data from their raw form (no voxelization). To manage the point clouds, they use the PointNet architecture [8].

The bounding box regression network uses a CNN that obtains the appearance and geometry features from the color data, the PointNet [8] subnetwork for the depth data and the fusion module.

Their merging method uses a dense 3D box prediction structure with input points, estimates corner locations, and uses the data to create the bounding box.

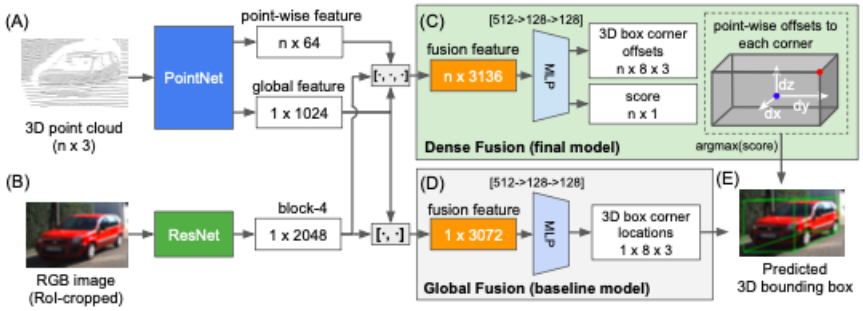

Figure 3.9: PointFusion Framework

3.2.7 PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation [8]

This article presents a method for using raw point clouds (without voxelisation). The presented network is used for object classification and segmentation. Touch data in voxel form

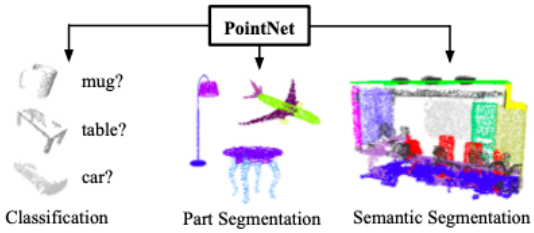

Figure 3.10: PointNet Task

is very large and with artifacts, so they are looking here for another way to extract and use depth data. The point cloud is the simplest pure form of representing 3D data. The PointNet performs a symmetric calculation because it takes into consideration that the point cloud is invariant to internal permutations. The PointNet allows to obtain class labels for the input point clouds (in a semantic way each point is assigned a class because the points are treated separately.). The network uses Cartesian coordinates with the symmetric max pooling function to learn the optimization criteria that select the points of interest. Then the fully connected layers use these points and criteria to create a label map for classification or segmentation. As the points are processed independently, a spatial transformation subnet (rigid and affine) canonicalizes the input data to refine the result.

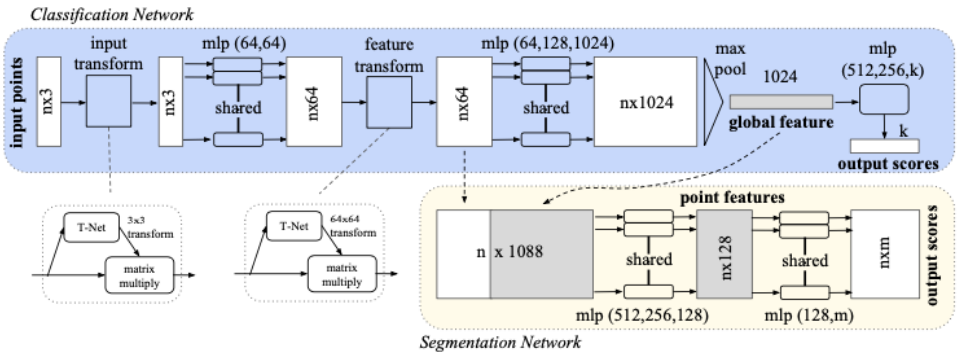

Figure 3.11: PointNet Framework

3.3 Tactile Sensing

As mentioned before, the methods using only Tactile detection are non-existent or extremely rare, that's why in this part they will see the articles that I had to study which concern the sensors that they had chosen and how to use the data that they transmit us. The reason why pose estimation methods by tactile detection only are few is the fact that tactile data are not representative of the global shape of the studied object. Indeed, one obtains an accurate representation of the shape, texture and friction coefficient, in a local way which is not sufficient. For this reason, tactile data are considered as sparse.

Within the framework of this project, I studied different articles that were close to our subject in terms of tactile detection only. Here is the list:

- Use of tactile feedback to control exploratory movements to characterize object compliance [10]
- Multi-Fingered In-Hand Manipulation With Various Object Properties Using Graph Convolutional Networks and Distributed Tactile Sensors [4]
- Interpreting and Predicting Tactile Signals for the SynTouch BioTac [7]

Through these articles I could discover how the BioTac sensors we had chosen worked, what kind of raw signals it returns and how to exploit the data provided by the sensor.

3.3.1 Use of tactile feedback to control exploratory movements to characterize object compliance [10]

In this article we will learn how to use the tactile detection of BioTac sensors.

BioTac sensors are tactile sensors that look and feel like fingertips. The force exerted on the fingertip, on the soft part, is measured by the electrical resistance of the conductive fluid between the main electrodes inside the sensor and the elastic skin. The BioTac® is also a sensor capable of detecting contact forces and normal/tangential forces.

This sensor consists of a rigid core on which are fixed the electrodes surrounded by an elastic skin filled with a conducting fluid. When an object is in contact with the sensor, the force will deform the skin, which will displace the fluid and modify the impedance of the electrodes. It can directly relate the change in impedance at the electrodes to the normal vector of the force of the electrodes.

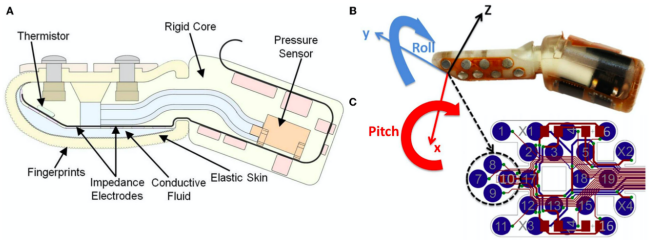

Figure 3.12: BioTac sensor

3.3.2 Multi-Fingered In-Hand Manipulation With Various Object Properties Using Graph Convolutional Networks and Distributed Tactile Sensors [4]

This article [4] presents a method for stable object manipulation and grasping. They use sensors on the fingers of the robotic hand (Allegro), the data from these triaxial sensors (1152 measurements) must be preprocessed because the classical neural networks cannot use them because they do not have a rectangular shape. This is why the method is based on a graphical convolutional network (GCN) which allows to obtain the geodesic characteristics of the tactile data thanks to the configuration of the sensors. Concerning the training, they use a data glove to reproduce human dexterity. The U Skin sensors (which use the Hall effect so that the deformation is a function of the force applied to the sensor) are present on the whole surface of the hand. The GCN is used to link the sensors close to each other according to the position of the fingers.

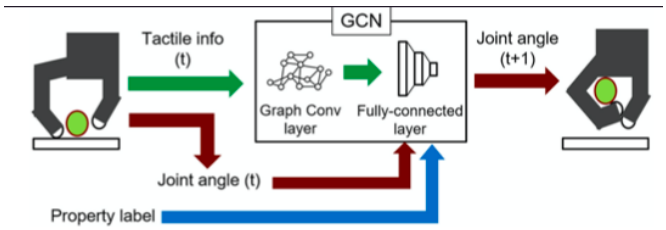

Figure 3.13: Use of the GCN

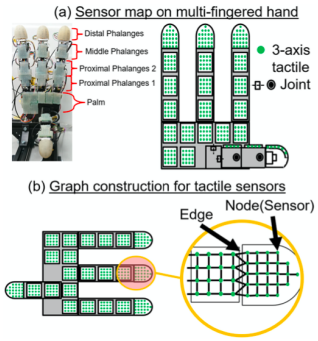

Figure 3.14: Allegro Hand with all the sensor

3.3.3 Interpreting and Predicting Tactile Signals for the SynTouch BioTac [7]

In the article [7] they show us how to use tactile detection to perform a manipulation under visual occlusion. They studied the 3D contact position and the force vector from the raw tactile signals.

The objective of this paper is to create a database (53 109 data points in the end) with 3D contact points, force vectors and electrode values.

They use the electrode signals to map the contact. They use an FE model that segments a complex geometric shape into simple geometric subregions (tetrahedral or hexahedral for 3D shapes). This model also allows to transform the partial differential equation into a group of algebraic equations. They use the regression with MLP and 3D CNN on the values of the electrodes to obtain the contact position and the force vector, in the case of PointNet++, but from the values and coordinates of the electrodes to obtain the same final value.

3.4 Methods using both visual channel and tactile channel

In this section, I will present articles that use both tactile and visual detection to estimate the pose of an object. These methods are more complex because the data received are not all of the same type but they are also the most efficient against occlusion and the most promising to assist robotic manipulation. Concerning the methods that use both types of detection, I have studied in depth these five articles:

- Adaptive Control for Pivoting with Visual and Tactil Feedback [13]
- Active Visuo-haptic Object Shape Completion [9]

- CAPTRA: CAtegory-level Pose Tracking for Rigid and Articulated Objects from Point Clouds [16]
- Multi-Modal Geometric Learning for Grasping and Manipulation [12]
- VisuoTactile 6D Pose Estimation of an In-Hand Object Using Vision and Tactile Sensor Data [2]

3.4.1 Adaptive Control for Pivoting with Visual and Tactil Feedback [13]

In this paper [13], they present a method of rotating a grasped object into a desired orientation by rotating it between the fingers of a gripper. The method controls the gripping force so that the object follows the desired trajectory. The rotation to the desired orientation will be based on the application of the gravitational torque (gravity) created by the weight of the object and applied at its center (extrinsic dexterity). They will use the force applied by the clamp on the object as a braking force. The method presented here, manages the friction between the object and the gripper by using the closed loop with an adaptive controller that would use the friction measurements or by using a touch sensor to control the holding force.

To achieve the rotation, we modify the pressure applied by the fingers on the object so that the gravity by the mass of the object applied in its center makes it rotate. By decreasing this pressure, the torsional friction on the pivot is modified.

To do this, they use the initial orientation (angular position obtained by vision) as an input with the desired angular position of the adaptive controller. This controller calculates the normal force of reference that will be used to control the trajectory of the object, this force is used through a PI controller to refine the movements of the finger.

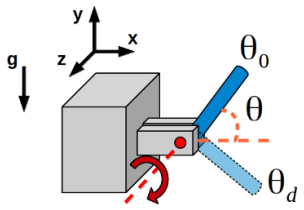

Figure 3.15: Modeling of the pivot task

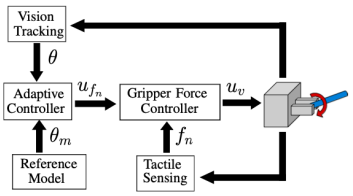

Figure 3.16: Rotation Control framework

3.4.2 Active Visuo-haptic Object Shape Completion [9]

This paper [9] presents a method (Act-VH) for object reconstruction using visual detection with tactile (haptic) refinement to overcome the occlusion problem. It uses point clouds; it calculates the uncertainty on the modeling with the implicit geometric regularization. To avoid taking a lot of time and a large number of trials like the heuristic touch approach, it uses a more efficient closed loop system. The method uses a deep implicit surface network to create many possible reconstructions that will be iteratively and randomly refined until the point cloud and the reconstruction match. The differences will form the uncertainty voxel grid, when the uncertainty is maximum, tactile exploration is applied.

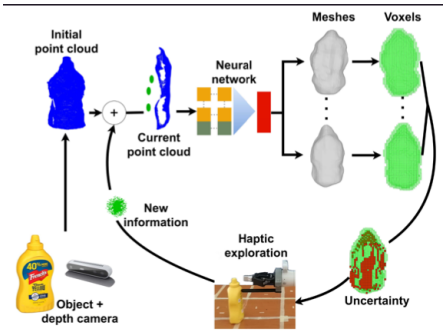

Figure 3.17: Act-VH framework

Other methods use probabilistic completion for reconstruction, they use a 3D CNN with Monte Carlo dropout variational inference technique for sampling.

The CNN is trained from visual data and haptic data retrieval in random exploration. The choice of architecture is an implicit geometric regularization(it is a multilayer perceptron (MLP)) that is trained by the distance function of the signal. To avoid sampling adjacent shapes, a multi-form IGR is used by changing the latent vector for each shape.

3.4.3 CAPTRA: CAtegory-level Pose Tracking for Rigid and Articulated Objects from Point Clouds [16]

The following article is the one that inspired me the most at the beginning of this project [16], it is an article that estimates the 9D pose (Three translations, three rotations, three sizes). The method presented in this article is used to estimate the pose of articulated objects (shape with a possibility of modification). The method uses the point clouds of the depth channel of the RGB-D image and the predicted pose on the last image to update this pose.

The architecture includes three modules such as a pose canonization module to normalize the point clouds, the RotationNet module that regresses rotations from one image to another and the CoordinateNet module that analytically computes the size and translation from the segmentation and coordinate estimation. When talking about pose estimation for articulated objects, the procedure here is to track through a series of images the individual pose of all rigid parts of the articulated objects.

Other methods use coordinates with RANSAC (to sort the values), or they regress the pose directly which favors the estimation speed against a higher error rate. In this paper there is a will to merge these types of methods to obtain a differentiable method from end to end in order to have a fast and accurate estimation frame after frame. To do this, they use canonicalization which consists of using previous poses (in the form of a point cloud) to refine the current pose (also a point cloud) which is similar from one image to another. This is where the RotationNet module (inspired by PointNet++) comes into play to handle small rotations. The CoordinateNet module is used to estimate the normalized dense coordinates because they allow us to obtain the size and translation by comparing with the previous image.

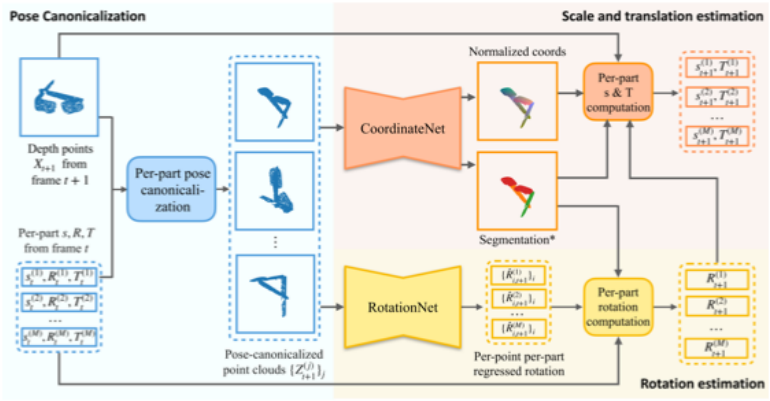

Figure 3.18: CAPTRA framework

For this part, we start with the canonicalization of the pose, to estimate the pose from the point clouds, two alternatives have been proposed: use a neural network to regress the pose directly or estimate the normalized coordinates; they use the Umeyama algorithm coupled with RANSAC to obtain the 6D pose and use the distance of the axes for the size. They use the canonicalized point cloud as input to the RotationNet which regresses the rotation and replaces the old one with the new one. So they design a CoordinateNet to segment the canonicalized point cloud map, which when used, along with the output of the RotationNet, by the Umeyama algorithm provides the scale of the object and its translation. The CoordinateNet module can

be separated into two parts: one that manages the segmentation (using the IoU pert function) and the other that predicts the normalized coordinates, the prediction being done according to the classes

3.4.4 Multi-Modal Geometric Learning for Grasping and Manipulation [12]

In this paper [12], they present a method that creates 3D models for a robotic manipulation task.

Here, they obtained a point cloud that represents the visible part, they are voxelized to obtain a first draft of geometry and the hand is moved to the contact of the object.

The CNN is used to create a mesh of the target object using all this input data. The output after the final Sigmoid layer corresponds to a probability. If the output is close to 0, the voxel is unoccupied and if it is close to 1, the voxel is occupied. The optimizer chosen for this CNN is Adam to update the learning rate.

The tactile database consists of 500,0 00 triplets of voxel grids: depth (marks occupancy if visible by vision), tactile (marks occupancy if contact is present), and ground truth.

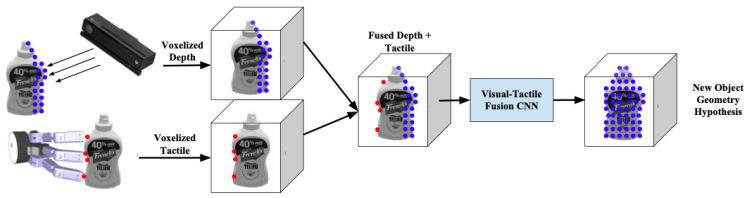

Figure 3.19: Framework of the Multi-Modal Geometric method

3.4.5 VisuoTactile 6D Pose Estimation of an In-Hand Object Using Vision and Tactile Sensor Data [2]

In this paper [2], the authors addressed the following question: How to use data from two sensors that do not have the same detection type?

The main use of visual data is segmentation, but they can also be used to capture the textures of objects, depth data can also have a second use, under different lighting conditions, it can provide several geometric accuracies.

Concerning the fusion of sensor data (dense pixel fusion), it uses the method seen in the article DenseFusion [14] as our method.

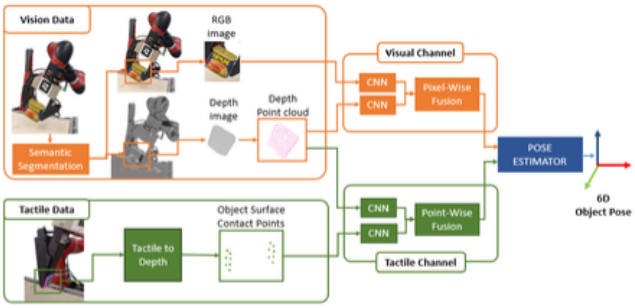

Figure 3.20: Global Framework of the VisuoTactile method

For training, here it uses synthetic data (photorealistic and tactile) obtained with NVIDIA's Deep Learning Data Synthesizer (NDDS).

The network presented in this article is one of those that inspired my first framework that used tactile data.

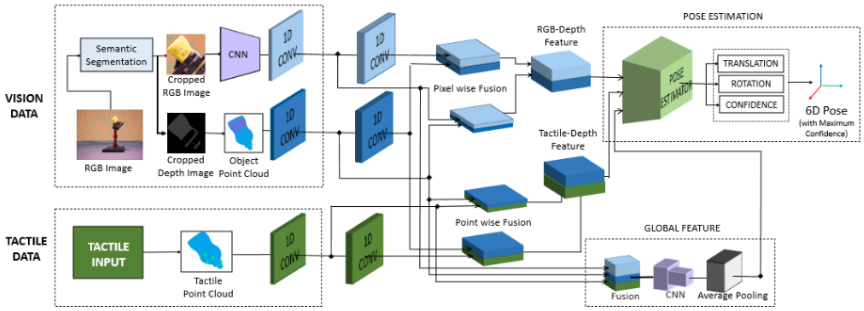

Figure 3.21: Detailed Framework of the VisuoTactile method

The visual channel starts with a semantic segmentation tool on the color images (assigning each pixel a class). The color and depth images are then adjusted according to the segmentation result and the depth images are converted into a point cloud of the camera reference frame. Then, the pixels are assigned their corresponding 3D location. The processed color and depth images are then incorporated and integrated by two successive CNNs before being merged in the dense per-pixel fusion module. The overall feature is obtained by merging the depth, color, and tactile data that have been integrated into a CNN with an average pooling as output.

In the pose estimation network, the translation vector, rotation vector and confidence level are predicted for each input. Through the network, a large number of estimates are made, and

the one with the highest confidence level is retained.

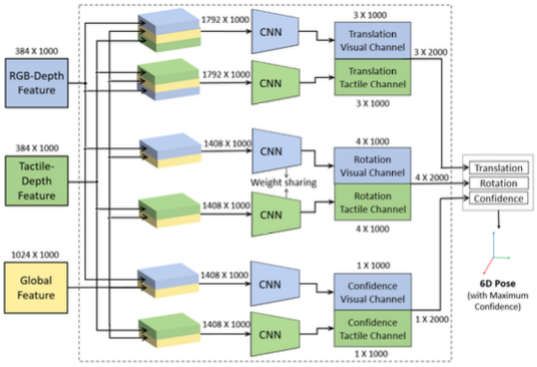

Figure 3.22: Pose estimation Framework of the VisuoTactile method

3.5 Autres

At the beginning of this internship we explored the track of supervised reinforcement learning with visual detection to obtain a high grasping rate. To do so, I studied this type of machine learning about which I didn't know much through the playlist of videos on the subject on Youtube made by professor Steve Brunton from the University of Washington. Reinforcement learning consists in imitating animal learning, with a reward system for desired behaviors (results) and negative reward for negative behaviors. We make an agent evolve in an environment and for each choice of action made we look at the proximity of the result to a desired state and we reward the agent in a positive or negative way depending on whether it is closer to the desired result than to its initial state. Performing this on a series of actions and looking for the series of actions that corresponds to the best global reward and optimize the network to obtain an optimal solution. To explain this method we often use the example of a mouse in a maze, or a chess game. There are many kinds of reinforcement learning, they are all presented in the video playlist: methods that use a model or not, methods that use a gradient or not and methods that use deep learning.

In our case the type of reinforcement learning we would have used is the *Q-learning* It is a method that does not use a model or gradients The letter 'Q' designates the quality function measured for an action executed in a given state of the system. It is an off-policy algorithm. The Q-learning allows to learn a strategy, to define which action has the best reward. It works with a quality function noted Q which allows to obtain the reward on the long term which is

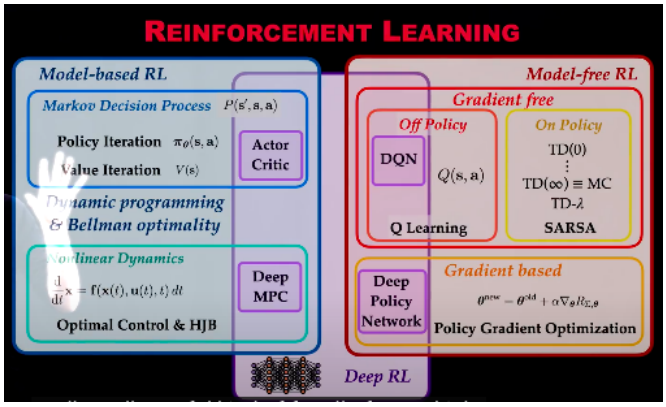

Figure 3.23: Type of reinforcement learning

obtained by the choice of an action in the current state by following an optimal policy. The advantage of Q-learning is that despite the lack of initial knowledge about the environment, it is possible to achieve learning.

Many applications of reinforcement learning can be found in robotics, as mentioned above in the methods of grasping or manipulating objects by robotic arms is an often explored track. I have studied the article *Isaac Gym: GPU-based high-performance physics simulation for robot learning* which presents a simulation tool developed by Nvidia that uses the GPU to accelerate the simulation of robotic learning tasks and this simulation environment can be used for reinforcement learning . One of the examples that will be presented by this simulation tool is the manipulation of a colored cube by a robotic hand (Open Ai Shadow Hand) to a defined final position. In this environment, we multiply the simulations which allow to greatly accelerate the learning time GPUs are used for parallel simulations, in fact we duplicate the simulation environment a lot of times and apply variations on each copy and then run all simulations in parallel. In the Isaac Gym environment, three simulations based on robotic hands are available (Shadow Dexterous Hand, TriFinger robot, Shadow Hand to Allegro Hand). The simulation that uses the Shadow Hand robotic hand is the cube can be applied to our subject, indeed the final goal of this course would be to achieve maximum manipulation with the hand. This cube manipulation simulation (inspired by the Open AI one), which uses a feed forward network for the observation with an observation that has as input: joints, speed, force, sensor, previous actions, object position, ...

Thanks to this environment I was also able to simulate the contact between the BioTac sensor and variable surfaces, in order to study the results obtained as much in the electrical signal provided by the electrodes as in the point cloud transmitted by the sensor.

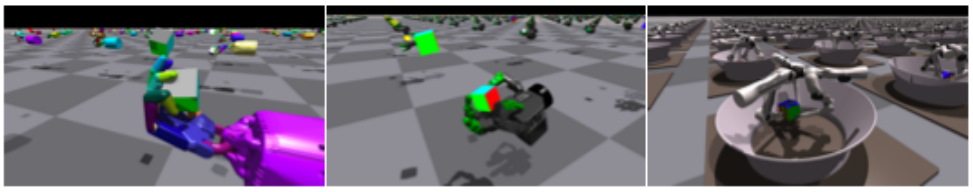

Figure 3.24: The three type of hand

3.6 Datasets

For this project we used two types of databases to train and test the studied and realized networks. The visual databases containing the studied images for pose estimation (YCB-Video, LineMod, NOCS). The tactile databases, where we mainly studied the format of the data provided and studied and finally the different sensors that we could have used to obtain tactile data (Bio Tac, ...).

3.6.1 Visual Datasets

For this project we worked mainly on the YCB-Video database, because of its size it allows to obtain more representative results than Linemod. When downloading this database I encountered a problem, the computer I was working on didn't have enough space for me to download it (this database is 150 GB in .zip format) and I couldn't download it directly from my session on the asterion server (even with the Linux command "Wget") so it took me a little while to get it.

YCB-V Dataset

The YCB-Video dataset created Y. Xiang, T. Schmidt, V. Narayanan, and d. Fox for the PoseCNN method, contains 21 YCB objects of various shapes and texture levels under different occlusion and lighting conditions. The dataset consists of 92 RGB-D videos, these videos are annotated with 6 Ds poses and segmentation masks. Similar to the preparation for PoseCNN and Dense Fusion, we separate the data into 80 videos for training and 2,949 keyframes were selected from the remaining 12 videos for testing. We use the 80 000 synthetic images.

LineMod

Another very well known database in this kind of application is Linemod which has 1000 images with about 15 objects present on each of them. One of the problems of using Linmod is its size, indeed this database is smaller than standard training databases. There is another

low database on Linemod called Linemod-Occluded which allows to study more precisely the effect of the occlusion

NOCS

The name of this database comes from the article for which it was created [15]. This database is the database used for the DualPoseNet algorithm, consisting of two datasets CAMERA25 and REAL275. CAMERA25 is a mixed reality dataset with 6 object categories; this dataset consists of 300,000 images of 1,085 object instances, with 25,000 images of 184 instances for evaluation. REAL275 is a real-world dataset obtained with more difficult conditions, clutter, occlusions and different lighting conditions. This dataset contains over 7000 images with 13 different scenes including 2750 images from six scenes for testing. Both datasets have the same object categories and they can be used in combination.

3.6.2 Tactil Datasets

As mentioned in the early parts of this dissertation, for this project I focused on pose estimation based on visual detection only but studied how to obtain and use tactile data.

YCB Binvox

At the beginning of my internship, I was provided with tactile data corresponding to a manipulation with a robotic hand equipped with sensors on objects in the YCB database. These data were in ".binvox", it is a format which allows to store in a file with a header and binary data a compressed and voxelized image.

BioTac sensing

For the choice of the tactile sensor, we would have used the BioTac sensor presented in the articles "Sim-To-Real for robotic tactile sensing via physics-based simulation and learned latent projections" [9] and "Interpreting and predicting tactile signals for the syn-touch biotac" [10]. We could have used the DIGIT sensor or a depth camera system for each hand area.

Chapter 4

Methodologie

4.1 First approach

After having read all the above articles and others I started to think about a method to realize a framework to use all the data for pose estimation. At that time I saw a little big, indeed there was only three months left of the internship and the method I had elaborated was very ambitious to estimate the 9D pose from visual detection, tactile detection, a global features and the canonicalization pose.

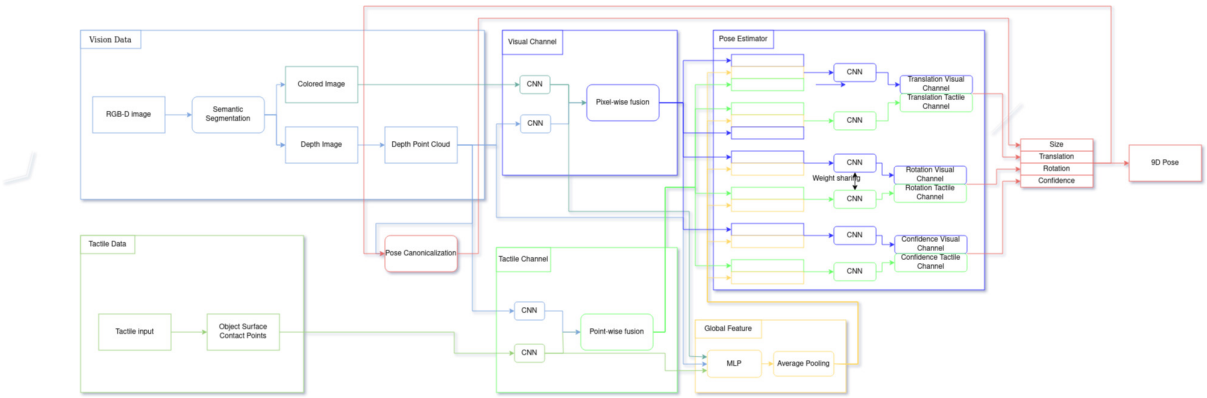

Figure 4.1: Frame work of the first idea of methods I create

In this method, with an architecture based on that of the method "Visuo Tactile 6D Poses Estimation of an In-hand Object Using Vision and Tactile Sensor Data" [2]. The data from the

visual detection pass through a segmentation tool (MaskRCNN) then from the depth image we obtain point clouds then these point clouds will be merged through a pixel wise fusion module. The touch data in the form of a point cloud and the depth point cloud are used in a point wise fusion module. The output of the two fusion modules and the depth point cloud will be used to create a global feature through a Multilayer perceptron network and then averaging. Then we will use the global feature, the result of the pixel wise fusion and the point wise fusion in a module of pose estimation. As a result, we obtain the translation, rotation and confidence and then we will use these values to obtain with the depth point clouds through a module of pose canonicalization, the size of the object.

4.2 My choice

So I decided with the help of my supervisor, to start on a more reasonable method, indeed I decided to create a method based on the architecture of the DenseFusion [14] method but I replaced the global feature module by an encoder which uses the segmented images of color and depth to calculate a pose sensitive feature. The encoder is the one of the DualPoseNet method [6], it is based on spherical convolution.

4.2.1 Dense fusion architecture

As with the Dense Fusion method, the depth data and the color data will be used separately to be able to extract the features that will lead to the break for each pixel of the images. We will also use the iterative method to improve our pose As mentioned in the first chapters and as most of the methods studied in the previous chapters*Stat of the art*, the method developed for this internship is a neural network method based on end-to-end deep learning and not on 3D models. Our method will merge depth data in the form of point clouds and color data in the output of the segmentation module to obtain features. Our method will also consist in a pose refinement from the point cloud depth data to obtain a better pose. Our method uses directly the color and depth data in their basic form, it looks like PointFusion [22].Here we will use a local texture fusion. We will of course use the coordinates of the camera in order to calculate the position in the real world of the pixels that correspond to each object. The first challenge for this method is to be able to use the depth data and the data from the color channels in a complementary way despite the fact that these data do not come from the same space. This is why as for the DenseFusion [14] method we will use a parallel architecture until the two streams of data are the same form, namely a group of features, at which point they can be merged (concatenate). Then the structure will also allow iterative refinement of the model to refine the estimation of the 6D pose.

Semantic segmentation

A greater explanation of segmentation is present in Chapter III because this module is common to almost all pose estimation methods. The first part of our method is semantic segmentation. Semantic segmentation is a tool for object detection in the image that uses segmentation to assign to each pixel of an image a label with respect to the object of which this pixel represents a part (example if on an image we can see a mug, the semantic segmentation will assign to each pixel that composes the mug in the image the label *Mug*).

Features extraction

To solve the challenge of merging the depth and color information, we must first pass them into a feature. Then for each pixel of the image, we add the depth value provided by the depth channel.

Color channel In the second part, we will then adjust the image around each object detected in the image by segmentation to estimate the pose of the objects by taking a closer look. Then this image segmented and adjusted will be used through a conventional neural network (CNN) more precisely a U-Net that assigns the pixel to a grouping of color features. The U-net also called fully conventional network that is composed of two streams that are an assembly of many layers of convolution and pooling, the first is the encoder that creates a map of features reduced from the input image and the second is the decoder that is symmetrical to the encoder allows to obtain a map of color feature in our case, of greater size. The color feature is obtained in the form of an image map that for each pixel vector (HWC) gives us the aspect of the image in position.

Depth channel For the other side of the parallel network structure, on the depth data management side, our method is also derived from PointNet [12], which is as can be seen in Chapter III a CNN that does segmentation/classification from point cloud, in our project we use their geometric feature extraction modules. Our method uses the depth data in the form of a point cloud to obtain the geometric features. Before using the PointNet module we convert the pixels of the depth channel of the adjusted and segmented image into a point cloud thanks to the intrinsic parameter of the camera that allows us to locate the surface represented by the pixel in the real world. The PointNet module is used to encode the information in the neighborhood of each point studied in the cloud and with a geometric integration, it generates a geo-dimensional feature (dense).

Pixel-wise-Dense Fusion

Then we will perform the fusion (concatenation) of the color features and the geometric features for each pixel to obtain a group containing all the features for each pixel. To realize the concatenation/merger of the features, we link to each pixel of the image segment and adjust its features of dense color and its geometric features by projection with the intrinsic parameters, then for each pixel we obtain a vector that will complete the dense features of the pixels to give a group of features will lead to the pose.

6D Object Pose Estimation

To train the pose estimation module [R—t], like in the DenseFusion method [14] we try to minimize the estimation loss which can be defined as the distance between the model points and the estimated pose points.

$$L_i^P = \frac{1}{M} \sum_j \|(Rx_j + t) - (\hat{R}_i x_j + \hat{t}_i)\|$$

With x_i is the randomly chosen point in the model and M the number of points in the model. This loss function is used for asymmetric objects (only one canonical frame). For symmetric objects, the loss function is defined as follows:

$$L_i^P = \frac{1}{M} \sum_j min_{0<k<M} \|(Rx_j + t) - (\hat{R}_i x_k + \hat{t}_i)\|$$

To optimize generally, they minimize the average loss, but if confidence is taken into account for estimation, a weighted loss formula is obtained.

$$L = \frac{1}{N} \sum_i (L_i^P c_i - w\, log(c_i))$$

With N the number of dense pixel features and w is a balancing hyperparameter. So we obtain the translation vector t, the rotation R and the confidence of the prediction which will allow us to define which estimate is the best.

Iterative Refinement

To iteratively refine our network (the DenseFusion part [14] and the encoder) pose prediction, the creators of the method used a deep learning network that uses dense fusion integration, the fused features of pixel, color and depth. This method will allow to correct the error on the pose estimation in an iterative way from the previous pose thanks to a recurrence system and from the point cloud estimated in the output of PointNet [12]. Indeed the estimated pose is used as

a prediction of the canonical frame, so they use the modified point cloud to encode the pose that will be reused. This gives us for K iterations, a pose equal to

$$\hat{p} = [R_K|t_K].[R_{K-1}|t_{K-1}]....[R_0|t_0]$$

But they activate the estimation by refinement only when the initial network is stable (convergence) for noise reasons.

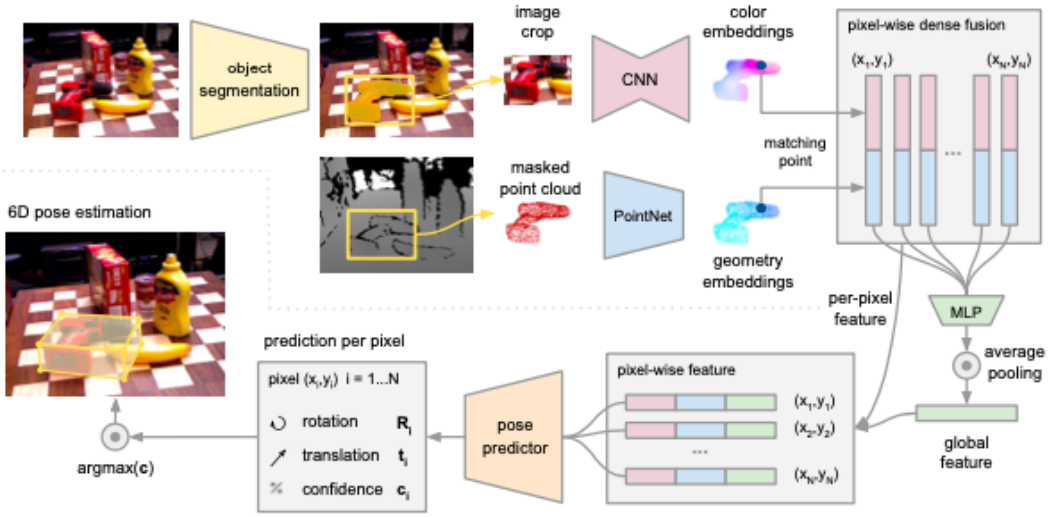

Figure 4.2: DenseFusion framework [14]

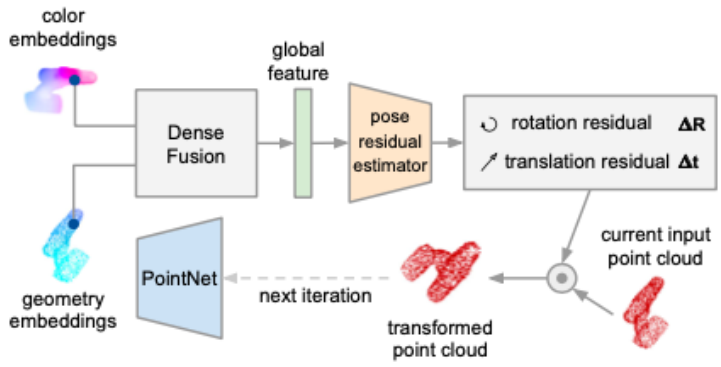

Figure 4.3: DenseFusion refinement framework [14]

4.2.2 Encoder

The encoder used here is from the DualPoseNet method [6] , it is built using spherical convolution and a spherical fusion module.

The input to the encoder will use the segmented and adjusted images. The encoder uses spherical convolutions to learn deep surface and shape features by rotational equivariance. The encoder also uses a spherical fusion module that is used to improve the shape feature integration of the input segmented image.

In the encoder, the inputs are color and depth images that have been segmented and adjusted, they are used in two identical parallel structures that contain spherical convolution modules and the results are merged with spherical fusion modules.

Spherical convolution Convolution in general is very efficient to capture useful features, if we represent the data on a sphere we can do 3D data analysis, the difficulty is to fit the convolution with planar data because the reference frame is here non-normed and the points are spaced in a non-uniform way. We can try to model the points of the spherical domain as a uniformly spaced Cartesian grid and perform a 3D convolution on it (imprecise method). To begin we represent all the points under spherical coordinate (r, θ, ϕ) with r = 1, we define f as the surface function of the unit sphere $f(\theta, \phi) = r$ we then obtain a spherical heat map which represents our points. In the spherical convolution, as in the plane convolution the kernel moves on the surface, here of the sphere, and takes the inner product between the kernel and the surface function, the spherical kernel will capture the patterns that are on the surface of the sphere as for a plane convolution. Equivariant results (according to translation and rotation) will be obtained regardless of the orientation of the object in the image due to the sharing of weights between the kernel and the plane and the presence of the unit sphere. The spherical convolution is realized in the spherical harmonics domain to allow to transform the convolution in a simple manipulation. The spherical harmonics are a complete and orthogonal set of basis functions on the surface of the unit sphere.

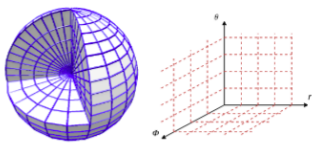

Figure 4.4: Spherical domain

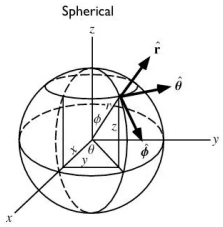

Figure 4.5: Spherical coordinate

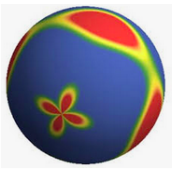

Figure 4.6: Spherical heat map

Spherical fusion The spherical fusion uses as input the color depth images modified by the previous spherical convolutions, it has as output the spherical signals of size definable by the internal spherical convolution function which will become the pose sensitive features after they are passed in the multilayer perceptron subnetwork (fully connected). The fusion module works on the basis of the spherical convolution, we concatenate the two studied streams and the concatenation passes in the spherical convolution modules, the outputs are used by the MLP where they are concatenated with their respective initial streams before being reinjected in the encoder.

Encoder architecture Indeed each input image goes through two spherical convolution modules with a transformed image of size 16 in output. Then these features are used as input for a spherical convolution with a transformed image of size 32 in output. We then apply a weighted average pooling function before merging the two streams in a spherical fusion module with a transformed image of size 32 in output. Then on the output of each stream at the output of the spherical fusion we apply a spherical convolution with 64 output features and another weighted average pooling function. And we use a second spherical fusion with a transformed image of size 64 in output this time. And a last time at the output of the merge we use a spherical convolution with a transformed image of size 128 in output,his time and another weighted average pooling function. Then we finish with another 128 output merge. The outputs of the three merges are first flattened (but in vector form) by a *Flatten* function and then processed through a fully connected, multilayer perceptron (MLP) subarray with 1024 features outputs. At the output of the three MLP has fusion features with a maximum pooling layer, and the fusion passes into another MLP with a feature map of size 1024 x 1024 at the output, feature sensitive to the pose.

Encoder operation More precisely, here, the encoder transforms the input image into a discrete sampling of spherical signals, it calculates the geometric center of the image, then it refocuses the point cloud and divides it into regions (drawing regular rays from the center). For each region, they find the largest distance between the center and the point of the cloud. So he can create the spherical signal from this shape:

$$S^X(w,h) = x_{h,w}^{max}$$

$$S^P(w,h) = \|P_{h,w}^{max} - c\|$$

with S^X the spherical signal of the color signal, S^P the spherical signal of the point cloud signal, w and h the indexes of the studied region, $x_{h,w}^{max}$ designate the RGB values corresponding to $p_{h,w}^{max}$ which corresponds to the maximum distance of a point with the center.

The spherical fusion module is based on as mentioned above two flows (X, P) which pass through the spherical convolution layers and are then pooled by weighted average We define S_l^X and S_l^P as the spherical feature map for each layer l . This map can be computed with the spherical convolution function (layer) and then the maps of both streams are merged (concatenated). This module can be added after each spherical convolution layer passage, in this case it is used 3 times as shown on the framework. Once out of the fusion module the S-features will be used through a Flatten layer that flattens the data (tensor) to the desired shape (vector) and a sub-network of multi-layer Perceptrons that pool through a discretization process based on sampling. Once the pooling is done we can use the data as output for the encoder.

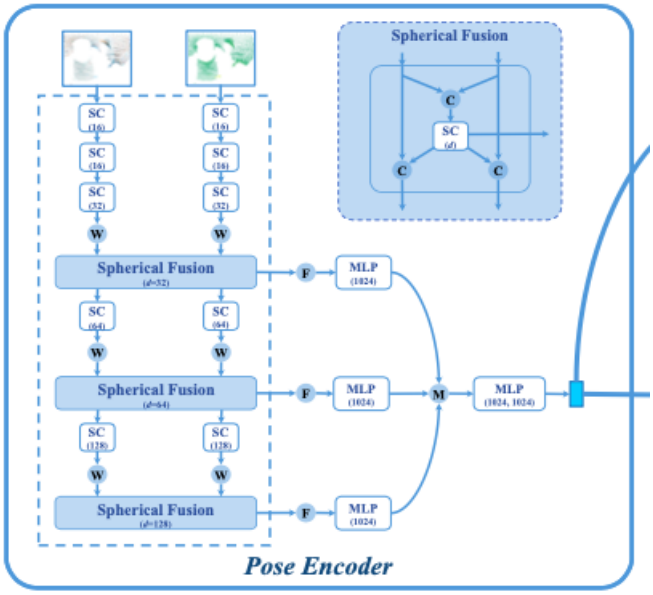

Figure 4.7: Encoder framework

Our method

For the encoder, as mentioned in DualPoseNet [6], pose-sensitive features are used to estimate the pose. This encoder uses spherical convolutions to learn the texture and shape features. In the encoder, depth and color segmented inputs are used in parallel in spherical convolution modules and then merged with spherical fusion modules. The encoder transforms the input images into a discrete sampling of spherical signals.

The spherical fusion module is based on two input channels that are passed through spherical

convolution modules and then a sub-network of multilayer perceptrons is applied that group the data and a weighted average pooling (as for the global feature).

The color images will pass through a CNN U-net to obtain the color features, the depth images pass through the PointNet modules to obtain the depth features. All features are merged through a wise dense fusion pixel module to obtain an image where each pixel is associated with a feature. This image will be merged with the encoder output in a pixel wise feature module. Then the merged features are used in a pose estimation module.

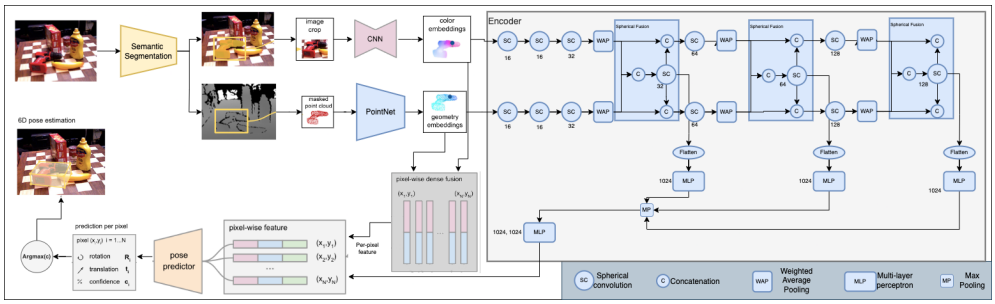

Figure 4.8: Framework of my method

To calculate the pose we use the same method as in Dense Fusion defining a loss function which is defined as the difference between the estimated pose (rotation and translation) and the true pose, we will also obtain the confidence for each pose. The thought that led me to replace the global feature module of the DenseFusion method [14] by the encoder of the DualPoseNet method [6] , is that at the output of this module we have a feature that is derived from the average pooling of features (that are already used) that is processed by a multilayer perceptron subnetwork. So replacing this module with another way to obtain a pause sensitive feature will bring a new estimate which will give us a better final pose. We also notice that, as in the global feature modules, ends with a multilayer perceptron subnetwork and a average pooling by adjusting the parameters we will obtain an output of the same form as for the global feature. As mentioned above, we will use the results of the segmentation as input, and as output we will have a pause sensitive feature, which can be concatenated in the wise fusion module by pixel of the features.

4.3 Visualization tools

For most of the methods presented by the articles, a GitHub link that leads to the code. The results obtained by all the methods I tested were in vector form as explained in the introduction. So I created a module that allows to visualize the pose in an image the pose of the objects,

in the form of a 2D bounding box to show the location of the object in the image thanks to the result of the segmentation and a three axis reference that shows the pose of the object (its translation and rotation).

Chapter 5

Experimentations and Results

5.1 Isaac Gym experimentation

At the beginning of the internship I realized the installation, the manipulation and the test of simulation on the Isaac Gym environment of NVIDIA because I thought of leaving on an approach based on the learning by reinforcement this experimentation was realized on a powerful computer(DELL precision 7740) at the university of Condorcet. The rest of the experiments will be done on a session on the Asterion server of the ImVia lab(Consists of 3 GPUs (NVIDIA Titan Xp, Quadro P400 and NVIDIA RTX A5000) and 126G of RAM, 48 CPU). I also tested on the Isaac Gym environment, the BioTacs sensor simulation to see how the sensor reacted to contact.

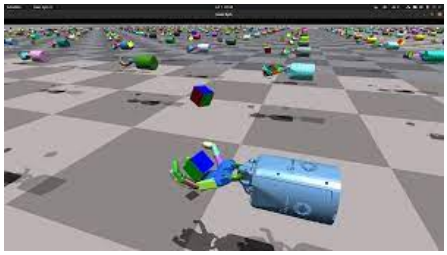

Figure 5.1: Isaac Gym Shadow Hand simulation

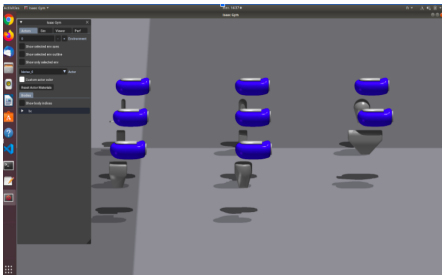

Figure 5.2: Isaac Gym BioTac sensor simulation

5.2 Pose estimation method

To merge the DenseFusion network [14] and the DualPoseNet encoder [6] , we create the code in Visual studio code. The main challenge here is that we can't just replace the global feature module with the encoder by changing the name and importing the file because the whole code of the DenseFusion method [14] is done with the Pytorch programming framework while the encoder that comes from the DualPoseNet [6] code is programmed under the TensorFlow framework. The compatibility problem is explained by the different functions, we cannot train a TensorFLow module with a Pytorch network because when the module is created with TensorFlow the class is of type *object* whereas for a model coming from Pytorch the type of the class is generally *nn.Module* so it does not have the same training and testing functions, nor the same parameters. Moreover the functions of the encoder are not issued from *torch.nn* so they are not usable in a Pytorch model The last problem with this implementation is that the spherical convolution function is also coded with TensorFlow, so it is also unusable in these circumstances. The spherical convolution function of DualPoseNet [6] and the encoder itself are derived from the work of Learning SO(3) Equivariant Representations with Spherical CNNs [3] on spherical convolution networks which seeks to achieve a 3D classification task by ignoring the rotation in the image thanks to the principle of rotation equivariance.

We therefore did not have a spherical convolution function under the Pytorch framework, so we used the functions created for [1] which uses the same version of Pytorch as we do (0.4) and presents a method for the construction of spherical CNN networks. for the study of situations like omnidirectional vision (drones). I then built with this spherical convolution function and the libraries available in Pytorch, an equivalent encoder by following both the diagram of the encoder as describe before in the *Methodology* chapter and its code in TensorFlow.

The problem with pose estimation methods that use touch data or depth data is that 3D data is heavy to use. With the means I used, the server, it takes 3h 30 for a training epoch, that's why I can't provide any result yet with the new network, I hope to be able to launch the testing phase in the next few days.

5.3 Visualisation tool

The visualization tool I created allows to use the results of the segmentation to draw bounding boxes around the objects on each frame and the tool also uses the results of the pose estimation, the rotation goes from a quaternion form to a standard rotation matrix(3*3). We then use the translation vector and the rotation matrix to draw for each object present in the frame a three axis reference that represents their position and orientation. The visualization tool also puts the frames together to create a video.

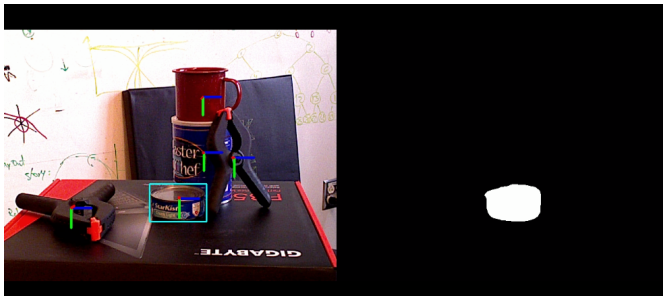

Figure 5.3: Example of result

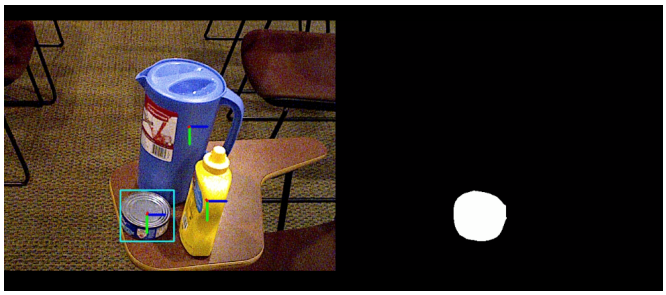

Figure 5.4: Example of result

As mentioned before, the result of the segmentation can be seen as a bounding box and for the pose of each object as a triaxial marker for each frame of the video.

Chapter 6

Conclusion

6.1 Technical conclusion

On the technical side, after having studied the articles that presented many methods in order to write one myself during the internship and after having tested the codes of several methods, I developed an ambitious method of estimating the 9D pose from tactile, depth and color data using the methods seen previously. As the internship period progressed, I had to think of a more reasonable method, so I chose to merge the DenseFusion [14] and DualPoseNet [6] methods to create a new one, unfortunately it is a method that does not use tactile data but can produce better results than the two above methods. I also created a tool to visualize the pose on each frame of a video by representing the 2D bounding boxes and the axial marker that represents the pose of each object in the image.

6.2 Personal conclusion

On a personal level, this training course was very formative and enriching but sometimes a source of frustration. Indeed, spending months to discover many methods which try to answer a problem and to find the way to obtain a result with a maximum of precision can import the conditions (occlusion) of study, was a true means to satisfy my curiosity on the subject. Concerning the frustration I can only blame myself, I think I should have allocated less time to the study part of the existing work and find an idea quickly enough which would have allowed me to go further in the subject because the fact of working with 3D information lengthens the time of training and testing which can be prejudicial. Adding this time with the difficulties to download the database, the period of testing the different possibilities in terms of code and

the creation of the visualization tool, I had badly estimated the time needed to develop the method. Thanks to this internship I was able to greatly improve my understanding of deep learning models, their architecture, the way to set up, the choice of layers, ... I was also able to realize a research work on a specific subject, which allowed me to use note taking and smart file management tools (Obsidian) and which allowed me to learn how to find the best articles around a subject. I was also able to experience what a doctoral thesis could be like in a short period of time which allowed me to confirm my choice of professional project, indeed given that the only frustration is the lack of time I think that given the period of the doctoral thesis, I could avoid this frustration.

Bibliography

[1] Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *CoRR*, abs/1801.10130, 2018.

[2] Snehal Dikhale, Karankumar Patel, Daksh Dhingra, Itoshi Naramura, Akinobu Hayashi, Soshi Iba, and Nawid Jamali. Visuotactile 6d pose estimation of an in-hand object using vision and tactile sensor data. *IEEE Robotics and Automation Letters*, 7(2):2148–2155, 2022.

[3] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so(3) equivariant representations with spherical cnns. *CoRR*, 2017.

[4] Satoshi Funabashi, Tomoki Isobe, Fei Hongyi, Atsumu Hiramoto, Alexander Schmitz, Shigeki Sugano, and Tetsuya Ogata. Multi-fingered in-hand manipulation with various object properties using graph convolutional networks and distributed tactile sensors. *IEEE Robotics and Automation Letters*, 7(2):2102–2109, apr 2022.

[5] Fu Li, Hao Yu, Ivan Shugurov, Benjamin Busam, Shaowu Yang, and Slobodan Ilic. Nerf-pose: A first-reconstruct-then-regress approach for weakly-supervised 6d object pose estimation, 2022.

[6] Jiehong Lin, Zewei Wei, Zhihao Li, Songcen Xu, Kui Jia, and Yuanqing Li. Dualposenet: Category-level 6d object pose and size estimation using dual pose network with refined learning of pose consistency. *CoRR*, abs/2103.06526, 2021.

[7] Yashraj S. Narang, Balakumar Sundaralingam, Karl Van Wyk, Arsalan Mousavian, and Dieter Fox. Interpreting and predicting tactile signals for the syntouch biotac. *CoRR*, abs/2101.05452, 2021.

[8] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[9] Lukas Rustler, Jens Lundell, Jan Kristof Behrens, Ville Kyrki, and Matej Hoffmann. Active visuo-haptic object shape completion. *IEEE Robotics and Automation Letters*, 7(2):5254–5261, apr 2022.

[10] Zhe Su, Jeremy Fishel, Tomonori Yamamoto, and Gerald Loeb. Use of tactile feedback to control exploratory movements to characterize object compliance. *Frontiers in neurorobotics*, 6:7, 07 2012.

[11] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *CoRR*, abs/1809.10790, 2018.

[12] Jacob Varley, David Watkins-Valls, and Peter K. Allen. Multi-modal geometric learning for grasping and manipulation. *CoRR*, abs/1803.07671, 2018.

[13] Francisco E. Viña B., Yiannis Karayiannidis, Christian Smith, and Danica Kragic. Adaptive control for pivoting with visual and tactile feedback. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 399–406, 2016.

[14] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. *CoRR*, abs/1901.04780, 2019.

[15] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. *CoRR*, abs/1901.02970, 2019.

[16] Yijia Weng, He Wang, Qiang Zhou, Yuzhe Qin, Yueqi Duan, Qingnan Fan, Baoquan Chen, Hao Su, and Leonidas J. Guibas. CAPTRA: category-level pose tracking for rigid and articulated objects from point clouds. *CoRR*, abs/2104.03437, 2021.

[17] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *CoRR*, abs/1711.00199, 2017.

[18] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. *CoRR*, abs/1711.10871, 2017.