Efficient learning methods for robot grasping oriented pose estimation

September ２０２１

Gabas Nova Antonio

Efficient learning methods for robot grasping oriented pose estimation

Graduate School of Systems and Information Engineering

University of Tsukuba

September ２０２１

Gabas Nova Antonio

Abstract

Perception and manipulation are two of the most fundamental processes that a robot needs to perform. In order to execute a task that includes moving, assembling or disassembling an object, the robot first needs to efficiently acquire some knowledge about the object itself and the environment surrounding it. This is the case for industrial robots working in a production line as well as service robots in a household. Service robots can assist humans by doing tasks that are considered repetitive, tedious, or dirty. This often includes chores in a household, but also any place like a small store that does not have a controlled environment like in the industrial case. In the past years several solutions have been implemented for industrial robots. However, there is still a lot of work to do in order to have service robots assisting at households. These environments require much higher degrees of adaptability to changing scenarios and the ability to manipulate many different objects with different properties. Object perception is the previous step to object manipulation and a step that will have an immense influence on the success of the process. By gathering enough and accurate information of the object, we can greatly improve the chances of success.

The main goal of this thesis is to obtain information about an object 3D pose i.e. its position and orientation. More specifically, we want to obtain this pose information in a way that is robust, but also efficiently applicable to small-scale, non-controlled scenarios, and easily implementable for new objects.

By robustly obtaining the object pose, we can define grasping points anywhere in the object and also control not only the position where

we place it, but also the orientation in which it is placed. This is important in scenarios like a grocery shop, where the robot may need to place an item on a shelve with the label facing the client.

There are three important challenges in training a robot to detect the 3D pose of an object: Gathering enough data to successfully train it, attaining robustness to work in changing scenarios (different surfaces, light conditions, etc.), and work with a wide range of different objects that may have different properties.

Acknowledgements

First, I would like to express my gratitude to my supervisors Prof. Yoshida and Prof. Sagawa for their constant guidance and encouragement, and Dr. Yoshiyasu for his support and discussions that helped to shape this project.

I would also like to thank my sub-supervisors and members of the committee Prof. Aiyama and Prof. Kanehiro, for their valuable feedback during my presentations; and Prof. Kita for her assistance and support, specially during my first two years of research in Japan.

I am very grateful for the funding received from the Japanese Government MEXT scholarship and the opportunity to do my PhD in the University of Tsukuba in collaboration with AIST.

I would also like to thank Ryodo, Kota, Rohan, Kevin, Arnaud, Luca, Naoko, my colleagues in CNRS-AIST Joint Robotics Lab, and Takahiro Ito for being a great tutor and friend. I am also extremely thankful to my colleagues in the Institut de Robòtica in Barcelona, specially Guillem Alenà, Sergi Foix, Carme Torras and Sergi Hernandez, from whom I have learned so much and continue to be inspired from.

I am most grateful to my parents, for their endless love and support, for always believing in me, and for being such amazing parents. To my sister, my brother and my niece, whom I have missed enormously during this time.

The journey towards a PhD is a tortuous one, and I am fortunate to have my life partner Lara Solà by my side, whose love and support has been essential to make this thesis possible.

Last but not least, I want to thank my friends in Tsukuba: Martin, Ralph, Vishal, Christina, Stina, and Petra, who have filled the last five years with humongous amounts of fun and wonderful memories; the Spanish Tsukubians: Guli, Dani, Santi, Jacobo, Antonio, Marc, Jose, Ayaka, Ayu and Masako; and the friends who have always been there even when we were thousands of kilometers apart: Sam, Andrés, Jordi, Sebas, Christian, Constan, Cónsul, Roger, Alberto and Pablo.

Contents

List of Figures

LIST OF FIGURES

List of Tables

Glossary

CNN Convolutional Neural Network; Class of deep neural networks, most commonly applied to analyzing visual imagery. Their architecture consists in shared-weight convolution kernels that scan the hidden layers.

DL Deep Learning; Family of machine learning methods based on artificial neural networks.

DoF Degrees of Freedom. Number of axes in which movement or rotation is possible.

DRL Deep Reinforcement Learning; subfield of machine learning that combines reinforcement learning (RL) and deep learning

MDP Markov Decission Process. A mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker

NBV Next Best View. Name of the methods that estimate more informative camera viewpoints.

OPE Object Pose Estimation. Method to estimate an object's pose (i.e. position and orientation)

RGB-D Red Green Blue - Depth; The four channels in a depth image sensor.

SAC Soft Actor Critic. off-policy actor-critic deep reinforcement learning algorithm based on the maximum entropy reinforcement learning framework

1

Introduction

1.1 Perception for Manipulation

Object 3D pose estimation is a very important step in robotic manipulation as well as other Computer Vision areas like Augmented Reality. It is defined as the problem of finding the position and orientation of a given object.

For object manipulation, knowing the 3D pose of the object allows several advantages over other methods that try to, for example, detect grasping positions in the object without knowing the actual pose. These advantages, include being able to grasp the object at a desired point, have information about occluded parts (if the 3D model of the object is available), and placing the object on a specific point with an specific orientation (see Fig.1.1).

This area has received lots of attention following the Amazon Picking Challenge (1) which led to several contributions that tackle the pick-and-place problem using data-driven methods.

These methods are often effective for industrial applications in which the object is always the same and it is found in a controlled environment. Furthermore, the data-gathering methods, while very successful in finding the pose, introduce a series of challenges (explained in section 1.2) that make them difficult to use in more common scenarios.

Our goal is to design efficient methods that can more easily be trained and applied to new objects, with minimum effort by the user, and requiring only

Figure 1.1: Robot manipulating a bottle - By knowing the pose of the bottle, the robot can grasp it at a desired position and orientation and place it on a shelve with the label facing the front.

simple and easily available hardware. Thus bringing these advances closer to more common manipulation actions (Fig.1.2).

First, it is important to distinguish between two kinds of objects: rigid and deformable.

For the first kind of object, we propose a data-driven method that requires minimum hardware and works only with RGB images as inputs. The system consists in a Neural Network with different submodules to deal with the estimation of the position and orientation of the object. We also propose a system for automatic data labelling to generate a dataset of images and poses and train the neural network with minimal effort.

For deformable objects, we take squared cloth items as an study case. In this case, since the object can take an infinite number of configurations, there is not a clear relationship between object and pose like the case of rigid objects. To deal with this in a way that is still efficient, we propose to focus on the shape and type of the edges of the garment. We distinguish between two types of edges and present a categorization of cloth folding patterns. This allows to locate two consecutive corners of the garment and spreading it. Bringing it to a position

Figure 1.2: Examples of common object manipulation actions - Examples of common manipulation actions in everyday scenarios.

ready to manipulate. We also propose an automatic dataset generation method for the case of deformable objects. In this case we automatically generate inputs consisting of depth images of the cloth item and label images with the segmented target edges.

Finally, to better improve the results of the pose estimation, we present a method for obtaining views of the object that are more informative. For this, we mount the camera in the robot arm and define the problem of Next Best View estimation as a Markovian problem. Our Reinforcement Learning agent takes as observations the current camera views and the robot joints and performs actions to move the camera with the objective of maximizing a reward function that consists, among other factors, in the pose regression error of the main Neural Network. This method can be extrapolated to any pose regressor improving the results by providing views of the object that are more informative and contain less occlusions.

The combination of these methods provides a pipeline that is easy to implement, requires little effort to train, is robust to changing scenarios, and does not require expensive or difficult to obtain hardware. These are conditions of efficiency that make it easier to implement pose estimation methods in common everyday scenarios.

1.2 Challenges

The main goal of this thesis is to design methods for 3D pose estimation of rigid and deformable objects, that can interact with the environment to provide more information and robustness. Particularly, we want to do it in an efficient way that can be easily adopted by any user in a common scenario like a household or a small shop. This creates a series of restrictions which in turn pose a series of challenges on top of the main challenges of estimating the pose of rigid and deformable items.

1.2.1 Automatic dataset generation.

Data-driven methods have demonstrated a lot of success in recent years. However, compared to classic methods, they require a much bigger amount of data. Capturing pairs of inputs and labels is a very tedious task that often involves a person having to go through all the images marking the labels. Many of the current methods use one of the commonly available datasets, this helps to compare the performance of different methods but it means that the problem of generating the data is not dealt with. For industrial applications it is often the case where the object to manipulate is always the same and the dataset needs to be generated only once, but in more common everyday applications, it tends to be the opposite way and the robot needs to learn to manipulate new objects. This would mean learning almost from scratch and having to generate a dataset every time. We propose methods to automatically generate datasets for rigid objects as well as cloth items. This means that almost no user intervention is needed and it does not require any labelling from the user.

1.2.2 Efficiency in hardware and input simplicity.

If we want to apply these methods in non-industrial environments and simplify the requirements, we need to use easily available hardware for capturing the inputs. Many methods use systems like motion capture devices, or multiple expensive cameras. Our intention is to use the minimum necessary information: RGB images for rigid objects, and RGB-D images for deformable objects. This means

that our input information will not be as rich as other methods. To overcome this challenge, we propose two novel neural networks that take only RGB images (RGB-D in the case of cloth items) as inputs and return the object pose as an output. Another common approach used by many methods is to use the 3D model of the object during the training. This helps us to get a more detailed loss function to train the neural network. However, if as described above, we need to often train with new objects, generating a sufficiently detailed 3D model for each new item is a task as difficult and time consuming as manually labelling a dataset. Hence, we choose not to use any 3D model during training, making it even more challenging to compete with state of the art methods.

1.2.3 Interaction with the environment.

An important step towards robust object pose estimation is to obtain informative views of the object. It is often the case where just one image of the object is not enough, it may be because the object of interest is occluded or because the current view is ambiguous. For example, if we try to obtain the pose of a mug but we cannot see the handle, we cannot determine the pose with enough confidence. Therefore, interaction with the environment becomes necessary. Moving around, moving the camera, or moving the object are effective methods to obtain better views of the object. The challenge here, becomes how to do this interaction in an efficient and clever way.

In the case of the cloth item, we are interested in locating the corners, and these corners may be self-occluded by the garment. This is why we propose an active perception method to choose the best action to perform in order to reveal the location of the corners and grasp them.

We also propose a reinforcement learning based method that learns to move the camera to positions with less occlusions of the object and less ambiguity. We employ this method to improve the results with rigid objects but it can potentially be applied to any kind of object.

1.3 List of Publications

1.3.1 Journal Publications

Antonio Gabas, Yasuyo Kita, Eiichi Yoshida, *Dual edge classifier for robust cloth unfolding*, ROBOMECH Journal, 2021, vol.8,1,pages 1–12, SpringerOpen

Enric Corona, Guillem Alenyà, Antonio Gabas, Carme Torras, *Active garment recognition and target grasping point detection using deep learning*, Pattern Recognition, vol. 74, pages 629–641, 2018, Pergamon

1.3.2 Peer Reviewed Conferences

Antonio Gabas, Enric Corona, Guillem Alenyà, Carme Torras, *Robot-aided cloth classification using depth information and CNNs*, International Conference on Articulated Motion and Deformable Objects, pages 16–23, 2016,Springer.

Antonio Gabas, Yasuyo Kita, *Physical edge detection in clothing items for robotic manipulation*, 18th International Conference on Advanced Robotics (ICAR), 2017, pages 524-529, IEEE

Antonio Gabas, Yusuke Yoshiyasu, Rohan Pratap Singh, Ryusuke Sagawa, Eiichi Yoshida, *APE: A More Practical Approach To 6-Dof Pose Estimation*, IEEE International Conference on Image Processing (ICIP), pages 3164–3168, 2020, IEEE

Kalenga-Bimpambu Tshilombo, Yusuke Yoshiyasu, Antonio Gabas, Kota Suzui, *Automatic dataset generation for object pose estimation*, SIGGRAPH Asia 2018 Posters, pages 1–2, 2018

Kota Suzui, Yusuke Yoshiyasu, Antonio Gabas, Fumio Kanehiro, Eiichi Yoshida, *Toward 6 DOF Object Pose Estimation with Minimum Dataset*, IEEE/SICE International Symposium on System Integration (SII), pages 462–467, 2019, IEEE

Rohan P Singh, Iori Kumagai, Antonio Gabas, Mehdi Benallegue, Yusuke Yoshiyasu, Fumio Kanehiro, *Instance-specific 6-DoF Object Pose Estimation from Minimal Annotations*, IEEE/SICE International Symposium on System Integration (SII), pages 109–114, 2020, IEEE

1.4 Thesis Overview

This thesis is organised as follows:

Chapter 2 provides an overview of the related work in rigid and deformable object pose estimation as well as other related topics like data gathering and best view estimation.

In chapter 3, we discuss the solutions we take to make our learning methods more efficient in terms of usability. We present the methods for automatic dataset generation for rigid and cloth items.

Chapter 4 describes the pipelines for object pose estimation providing implementation details of the neural networks and experimental results to validate the proposed methods.

In chapter 5 we present a reinforcement learning based method that learns to find better views of the object to provide to the pose estimator. We describe the framework and provide experimental results.

Finally, on chapter 6 we provide some conclusions about the results and in chapter 7 we explain the limitations of our method and possible future avenues of research.

2

Related work

2.1 Rigid Object Pose recognition

2.1.1 6-DoF pose detection

The most basic approach to 6-DoF pose detection is to directly regress the pose of the object in the image. In (2) they regress the projected 2D coordinates of the 3D bounding box and use the PnP algorithm to obtain the 3D point correspondences. Mahendran et al. (3) directly regresses the pose of the objects from a pre-existing dataset. The BB8 method (4) obtains the pose of the objects in the dataset using only RGB data but trains a second Neural Network to refine the results. Both methods claim to use only RGB data, however, both of them require a 3D model for the data generation and for the loss function.

Another type of approach, which is more similar to classic methods, consists in detecting a series of keypoints in the image and matching them to a model (5, 6). In these cases, to obtain the labels it is often necessary to do manual labeling where the human needs to go through many or all images locating where the keypoints are.

Recent methods (7, 8) show high accuracy using richer information like dense feature maps but in turn require several types of labels that are difficult to obtain like semantic segmentation, Normalized Object Coordinate Space maps for each object, etc. As stated in (8), a major challenge in 6-DoF pose still remains in the unavailability of ground truth data.

2.1.2 Data gathering

Following the aforementioned advances in 6-DoF pose estimation, the main bottleneck has become how to gather the necessary data for training. Many works assume the existence of a labeled dataset. (2, 3, 4, 9). The requirements of the labels are different for each solution and vary in how difficult it would be for a final user to obtain them.

Several works have tried to facilitate different aspects of this process. Suzui et al. (10) proposes a semi-automatic method using AR markers to obtain a small dataset that can train a pose detection network for a particular scene. In (11) a system to automatically annotate data is presented, although it reduces the amount of work needed it requires a Motion Capture device which also limits the scene. In the work of (12) the authors propose a robot-based solution for self-supervised learning. This approach uses a robot arm with an attached camera to take different views of the object and also bring the object to different poses. Same as the previous approach, this limits the backgrounds captured during training limiting the scenario to the place where it was trained and being weak to light changes.

Both of the last two works make it easier to annotate data but in turn require expensive hardware and limit the variability in the backgrounds. In our work we seek to obtain a method that can make data gathering easier in terms of effort and hardware. The most similar work to ours is the one presented in (13), where the authors use a method of copy and paste with blending techniques to obtain more variability in the background when training for 2D object detection. We extend this to 6-DoF pose and provide a Neural Network that can work with only RGB images as an input.

With the idea of being able to exhaustively sample more object poses and reduce human work, recent methods resort to training partially or completely with synthetic data (14, 15, 16). Although these methods claim to free the human from tedious tasks of labeling and object re-positioning, they assume that the user has a 3D model of the object, which can be a problem for applications where it is needed to introduce new objects very often. This 3D model needs to be very

accurate or otherwise the synthesized images will look differently from the real object, which will affect the training of pose detector.

2.2 Deformable Object Pose recognition

2.2.1 Semantic Edge Detection

The use of learning techniques to detect edge information allows to perform edge segmentation with respect to more subjective criteria than classical methods such as Canny (17). In (18), they use random forests to learn a mid-level representation based on object contours called sketch tokens. Similarly, in (19), they use boosted decision trees to extract depth maps.

Semantic edge detection goes one step further by turning this binary classification into a multiclass problem. CASEnet (20) proposes a network that classifies each pixel in the edge to one or more semantic labels. They demonstrate the results using Semantic Boundaries Dataset and Cityscapes datasets. The work in (21) improves the results of CASEnet by doing full deep supervision.

This paper expands on previous work (22), which was, to our knowledge, the first attempt to teach machines semantic edge segmentation for the perception of deformable objects. In (22), edge detection is successful in finding the corner to be grasped to unfold a towel. In cases where the corner is hidden, however, the unfolding of the garment cannot be completed. In this work, we present a detection and manipulation technique that allows us to identify and grasp a corner that is hidden behind curled up cloth, and then bring the garment to a complete unfolded state.

2.2.2 Cloth Manipulation

Feature detection is the approach most commonly used to locate a point to be grasped for unfolding. In (23) they detect the hem and propose grasping points that are later manually selected. If the garment lies on a surface and only presents some wrinkles as in (24), topology analysis can be used to generate a strategy for flattening. Yuba et al. (25) uses a “pinch and slide” action that involves locating

a corner, grasping it, and then pinching the edge close to it before finally sliding toward the next corner.

With the advent of deep learning, several studies have tried to solve the cloth manipulation problem. Triantafyllou et al. (26) uses horizontal edges and junctions found in the depth images as grasping points. This approach considers all of the depth edges without distinguishing whether they really belong to a physical edge or are produced by folds or noise. This can lead to selecting incorrect grasping points.

Doumanoglou et al. (27) uses random decision forests to learn to find specific points of garments (e.g., the shoulders in a t-shirt or corners in a cloth). To solve a problem where the points are not visible, the authors use a probabilistic action planner to acquire new views of the object by rotating it. However, soft garments, tend to wrinkle in a way that can hide big parts of the object, including these specific points (see Fig.2.1). In those cases, such points cannot be found, even by rotating the garment 360 degrees.

Figure 2.1: Some methods that try to locate specific points like corners; however, cloth tends to curl over, which can hide these points. The green lines are the painted physical edges. Physical edges are detected in the RGB image using color segmentation and are used to generate labels. During training, we only use the depth image, and the neural network never receives this color information.

Similarly to Doumanoglou's method, Corona et al. (28, 29) detect specific points for each garment using deep convolutional neural networks to find the grasping points on a garment after a neural network identifies the garment type.

In the work by Hu et al. (30), the authors hold the unknown garment to form one shape from a small set of limited shapes and match it with ones in a database prepared in advance. For bringing the item to such a limited shape, they first grasp the garment by the lowest hanging point and then by the farthest point from the vertical axis through the holding position, considering that the farthest point should be a characteristic point such as a shoulder. This second grasping strategy may not be applicable to all kinds of garments especially in the case of soft garments.

2.3 Best View estimation

When faced with the problem of pose estimation, one common difficulty is finding views that are not occluded or ambiguous. A classic approach is to use multiple views in order to try to increase the chances of obtaining a good view (31, 32, 33, 34).

However, even if we acquire several views of the object, that does not guarantee that we will obtain unambiguous and nonoccluded views.

As stated in the challenges remarked in the previous chapter, in order to increase the robustness it is necessary to interact with the environment. We propose searching for the Next Best View (35) of the object in an intelligent way.

The idea of searching for the NBV has been applied to many problems like 3D reconstruction (36, 37), navigation (38), 3D workspace exploration (39, 40), visual failure detection (41), object recognition (42), and extracting support relations (43).

Hashim et al. (44) also used techniques to find the next best view in order to find unambiguous views for pose recognition by moving the camera around the vertical axis of the object. In our case, we extend the problem to cover rotations in all 3 axes and finding the next best viewpoint defined by a 3D position and a 3D orientation.

3

Efficiency in Learning Methods

3.1 Bridging the Gap

The differences between industrial applications and other small-scale applications mentioned in Section 1 are not the only differences that make object pose detection and manipulation difficult to apply in more common scenarios. Specially since the advent of Deep Learning, the necessity to provide richer inputs to the neural networks, has lead to the use of more complicated sensors. This reduces the efficiency of such methods. Whether it is expensive pieces of equipment (e.g. High Definition cameras, distance sensors, etc.) or sensors that are difficult to introduce in non-controlled environments (e.g. motion capture devices, etc.), such hardware is difficult to obtain or introduce and this limits the possibility of using many state of the art methods for simple applications that are meant to be used in scenarios like households or small stores.

Another efficiency challenge derived from the use of Deep Learning methods and, in particular, the most prevalent subcategory: supervised learning methods, is the need for pairs of inputs and labels. Obtaining labelled inputs is often one of the most difficult and time consuming parts of a system based on deep learning. First, the inputs should be captured with a sufficiently large amount of data, often thousands of instances. These instances also require enough diversity to sample as best as possible the space of inputs. Then, depending of the system necessities it is needed to label the data. These labels can take a wide range of forms: discrete categorical labels, continuous numerical values, or annotated

images. For the creation of many of the available datasets, large amounts of human labour is been used to annotate the labels. This process of annotation, is not only very time-consuming but also highly tedious.

In the following sections we provide solutions and design choices that can be followed in order to make the learning framework more affordable to use in common scenarios.

3.2 Hardware

In learning methods for object pose estimation, the input to the neural network is an observation of the object. In general, the more information we can acquire about the scene, the richer the input will be, and this will facilitate obtaining a correct pose. However, this often comes at the cost of expensive hardware like cameras (sometimes several of them), depth sensors, motion capture devices, etc. To mitigate this, we propose methods that only use RGB sensors or, in the case of deformable items RGB-D sensors. We describe these methods in Chapter 4 and present novel neural networks that take only RGB(D) images.

3.3 Automatic Label Generation

Generating a dataset for training a pose estimator requires obtaining labels for each input sample in the dataset. This process, as shown in the beginning of the chapter, is very time consuming and requires highly repetitive tasks from the person doing the labelling.

In the following sections, we propose methods to generate pairs of input an labels in an automatic way. First, we focus on rigid objects, for which we generate pairs of images and pose labels. In the case of deformable objecs, as we explain later in Section 4, we focus on detecting the physical edges. Hence, we provide pairs of RGB-D images and image labels with the segmented edges.

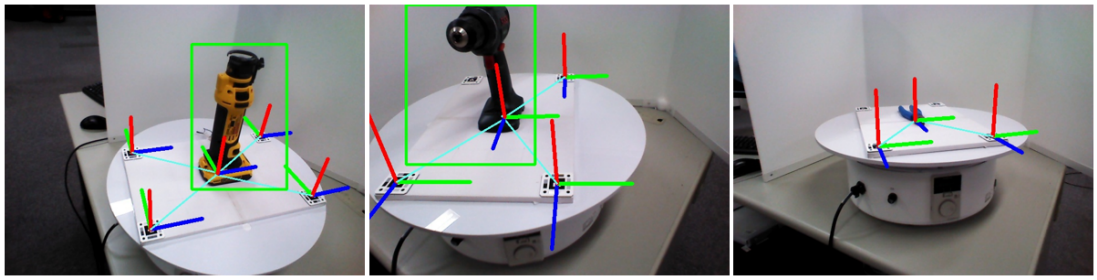

Figure 3.1: Data capturing system - Objects placed in the markers board for data capture.

3.3.1 Rigid Objects

In order to obtain pairs of object images and pose labels, we place the object in a board with AR markers (see Fig. 3.1), and, if available, we place the board on a turntable. The transformation from each marker to the centre of the board is fixed and known. When placing the object in the board, the contact point between the board centre and the object will define the origin of the object coordinate system.

Next, we take a video around the object. This part is even easier if a turntable is available since the user does not need to capture views around the object, only at different distances and from a different height. Typically we only require 2-3 min. of video.

The object centre position on each frame is obtained as follows: First we read each visible marker $M = \{M_1..M_N\}$ and we obtain its position with respect to the camera $\mathbf{T}_M^{Cam}$. Then add it to the fixed transformation from the marker to the board centre $\mathbf{T}_C^M$. We do this for each marker and then average the result:

$$\mathbf{T}_{\text{Object}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{T}_{M_i}^{Cam}\mathbf{T}_C^{M_i} \tag{3.1}$$

From each processed frame, the system will save the pose of the object and the 3D bounding box, which can be easily obtained from the dimensions of the tool (WxLxH). The 2D bounding box is obtained by projecting the 3D points into the 2D image and keeping the top-left point and bottom right point.

Figure 3.2: Examples of images in the automatically generated dataset - As the background color and intensity randomly changes, they have somewhat similar effects as changing ambient illumination conditions.

With this acquisition system we are able to obtain a large number of labelled images without much effort. However, the background is kept the same in all images and this will make it very difficult to generalize to other images with different backgrounds. In order to increase the diversity of backgrounds, we use Poisson image editing (45), a technique to cut a patch of an image and blend it into another image. To do so, we extract a binary mask around the tool with the previously obtained bounding boxes. The content of the bounding box is blended into a random background. Example of Poisson image editing is shown in Fig.3.2. We found that after blending the image into a new background the colour of the object is also slightly modified. This makes the result more robust to changes in light conditions.

3.3.2 Deformable Objects

When dealing with deformable objects, since they can take an infinite number of shapes, we can not establish the same kind of object to pose correspondence. As it will be explained in Chapter 4, we propose to detect the physical edges of the item. Our goal to generate the dataset is to obtain pairs of images and labels consisting of pixel-level segmentation of the physical edges. Manually labelling

the physical edges in thousands of images is not feasible owing to time constraints. To overcome this, we again use a semi-automatic dataset generation method. We paint the physical edges of a cloth item (as seen in Fig.3.3) and then with an RGB camera, we detect and label the edges by doing colour segmentation. It should be noted that we use the RGB images only to generate the labels; this colour information is never seen by the neural network used for pose detection, as it only uses depth information. Using this method, we are able to obtain hundreds of labelled images with minimal human intervention. The garment is hung from the robot end effector and rotated while the images are captured. After a full rotation of the garment, the shape of the garment is modified and another round of images is captured.

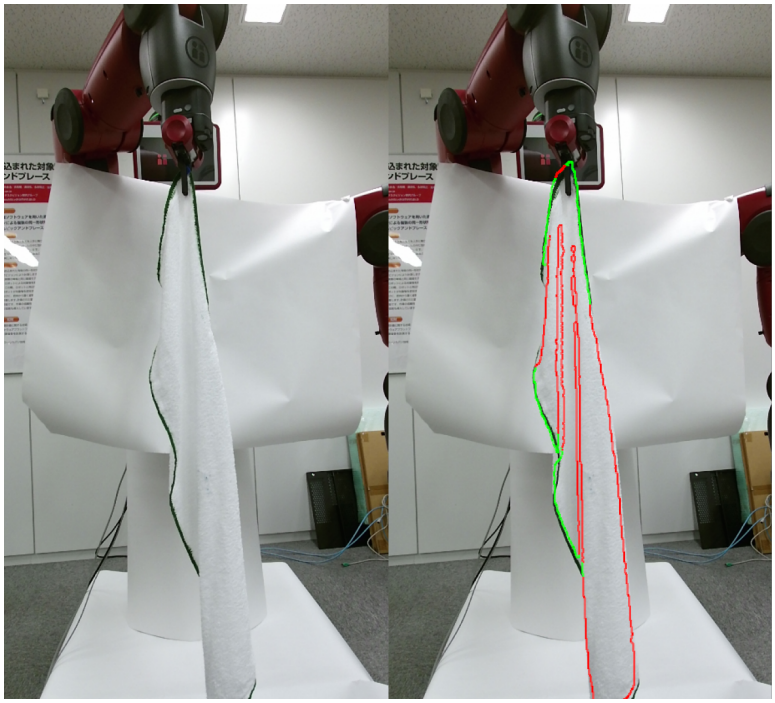

Figure 3.3: Example of automatic label generation for cloth - We paint the physical edges and use colour segmentation to generate the labels automatically. The classifier only uses depth information and does not have access to this colour information used to generate the labels.

4

Pose Detection and Manipulation

4.1 Rigid Items

In the following section we describe the details of our method for pose estimation of rigid objects and conduct experiments to validate it.

4.1.1 Framework Architecture

Since for the sake of practicality we use only RGB images, a big challenge is to design a network structure that can improve the trade-off between information in the input and accuracy in the output. For that purpose, we design a novel architecture (shown in Figure 4.1) divided in 4 sub-networks:

Object detection network
Methods like (2, 4) divide the image into a grid to generate a binary mask and search for the object centroid. We take a more direct approach and use Faster-RCNN (46) with ResNet-50 as a backbone to get a bounding box around the object. This network is trained independently with the 2D bounding boxes obtained from the markers.

Position and depth regressor
For estimating the value of the position, directly regressing the 3D coordinates

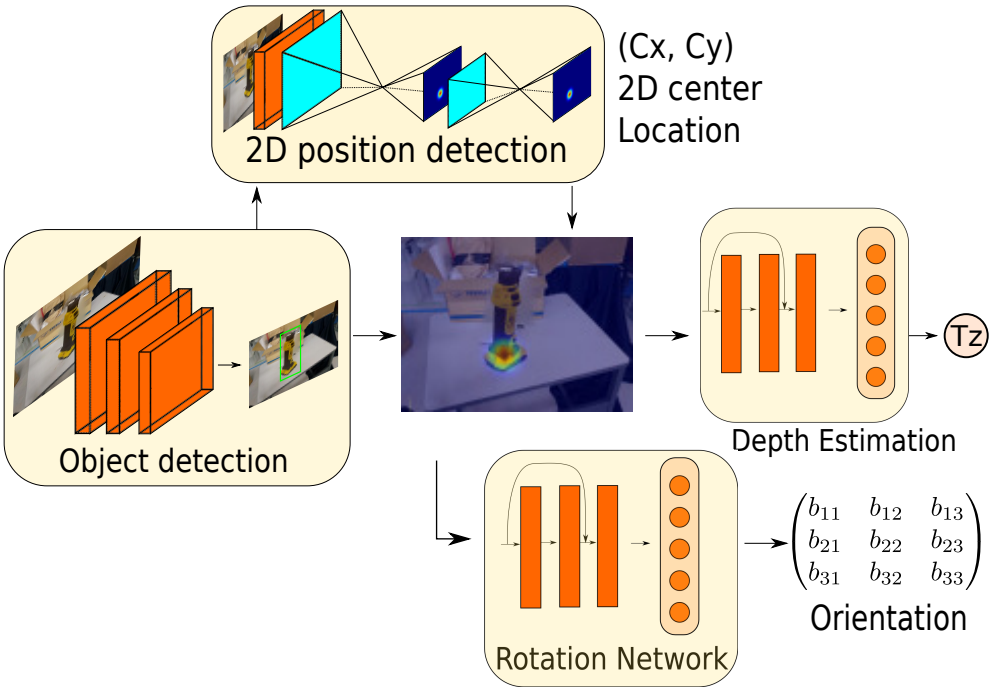

Figure 4.1: Pose detection network -Structure of the Network consisting on 4 submodules.

(T_x, T_y, T_z) would require the neural network to learn about the size of the object as well as the camera intrinsic parameters.

As a result, the trained network would only work on the camera used for training. Instead, we locate the image coordinates of the object origin (C_x, C_y) and regress the distance to the point in the Z-axis only (T_z). Then, same as (47), we obtain the 3D coordinates following equation 4.1, where (f_x, f_y) are the focal length in each direction and (p_x, p_y) are the coordinates of the principal point.

$$T_x = (C_x - p_x)\frac{T_z}{f_x}, T_y = (C_y - p_y)\frac{T_z}{f_y} \tag{4.1}$$

The position network consists of a stacked hourglass network (48) with two stacks of four residual modules. Unlike other methods we output a heatmap with maximum value on the origin of the object coordinates in image space, (C_x, C_y) this results in more accuracy than directly regressing the position. We concatenate this heatmap with the input image forming a 4-channel image which is run through a modified ResNet-152 with the mentioned 4 channel input layer and a linear layer as an output with a single value, the estimated distance (T_z) in the Z-axis of the object coordinate origin.

Orientation regressor

Taking as an input the RGB image masked around the bounding box, we regress a rotation matrix, which is a unique representation of a rotation. The structure is similar to the depth estimation network but the output consists on the 9 values of the 3x3 matrix. Directly using the output values from the linear layer does not guarantee that the resulting matrix is a rotation matrix. For that, it needs to meet two conditions: Must be orthogonal: $\mathbf{M}\mathbf{M}^T = \mathbf{M}^T\mathbf{M} = \mathbb{I}$ and must not be a reflection: $\det(\mathbf{M}) = 1$

In order to enforce this conditions, we add an orthogonalisation layer at the output. First we perform singular value decomposition (SVD) on the output matrix $\mathbf{M} = \mathbf{U}, \mathbf{S}, \mathbf{V}$. We obtain a new rotation matrix by $\mathbf{R} = \mathbf{U}\mathbf{V}^T$. Note that $\mathbf{U}\mathbf{V}^T$ is enough to satisfy the first condition. However, in the cases where the second condition is not satisfied, we need to replace the singular vector $\mathbf{u}_3$ with $-\mathbf{u}_3$. All these operations are differentiable and the resulting error can be

back-propagated.

Training loss

All 3 networks are trained together and the total loss ($\mathcal{L}_{\text{total}}$) is computed as: $\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{2D} + \beta\mathcal{L}_{\text{depth}} + \gamma\mathcal{L}_{\text{rot}}$ where α, β and γ control the relative strengths of the losses. The 2D position loss $\mathcal{L}_{\text{2D}}$ and the depth loss $\mathcal{L}_{\text{depth}}$ are computed as the Minimum Squared Error loss between the output T_z and the ground truth target value. The rotation loss is defined as the geodesic distance between the output orientation $\mathbf{R}_{\text{out}}$ and the target orientation from the ground truth $\mathbf{R}_{\text{target}}$.

4.1.2 Experimental Results

4.1.2.1 Evaluation of 6-DoF pose detection network

To evaluate our 6-DoF pose network, we compare to other methods by using the LINEMOD dataset (49) which is an standard benchmark for 6-DoF pose recognition. Different works report results in a variety of metrics according to their application. For Augmented Reality applications it is often measured the overlap between the object in the image and the projection of the model using the obtained pose. In our case, we are interested in robotic manipulation and need a metric that takes into account depth as well. We thus adopt the 5cm 5 degree metric (4, 9) which considers that an object is correctly classified if the error is below 5 cm and 5 degrees. For details about the implementation settings refer to the supplementary material. Table 4.1 shows the results compared to other methods that report on the 5cm/5deg metric. Our results are slightly superior to the ones reported in (4) even though they use a refinement process that uses a 3D model.

4.1.2.2 Evaluation of data gathering method

To validate our data gathering technique, we generated a dataset with 3 novel objects following the proposed method. We split this dataset and use different backgrounds for the test set.

Table 4.2 shows the average position error and orientation error for the three tools. As can be seen from Table 4.2, the position errors are below 5 cm for all

	5cm/5deg Metric		
	(9)	(4)	Ours
Ape	34.4	**80.2**	71.2
Can	48.4	**76.8**	68.9
Cat	34.6	**79.9**	65.2
Driller	54.5	69.6	**72.1**
Duck	22.0	53.2	**80.1**
Glue	23.6	54	**66.6**
Holepuncher	47.3	**73.1**	72.9
Average	37.8	69.5	**70.91**

Table 4.1: The 5cm/5deg. metric considers an instance of an object correctly classified if the error is below 5 cm and 5 degrees

Object	Position Error (cm)	Orientation Error (deg.)
Yellow Tool	3.37	6.3
Blue Pliers	3.31	3.29
Drill	1.54	0.91

Table 4.2: Average error on the validation dataset.

the tools. In the case of the yellow tool, due to its cylindrical shape, the rotation error is a bit higher than 5 degrees since its more difficult to regress the rotation in the vertical axis. However, the characteristic "L" shape of the drill makes it very easy to regress.

4.1.2.3 Evaluation in live setup and application to robot grasping

Although the results on the captured test set and LINEMOD dataset show satisfactory results, an important challenge to demonstrate its usefulness is to be able to use the system on data captured live. We placed the camera in front of a robot in an scenario with a table, a target object and several other objects. We obtain the pose of the target object and then grasp it from the desired direction. Figure 4.2 shows a sequence of grasping. Other sequences can be found in the

video accompanying this article.

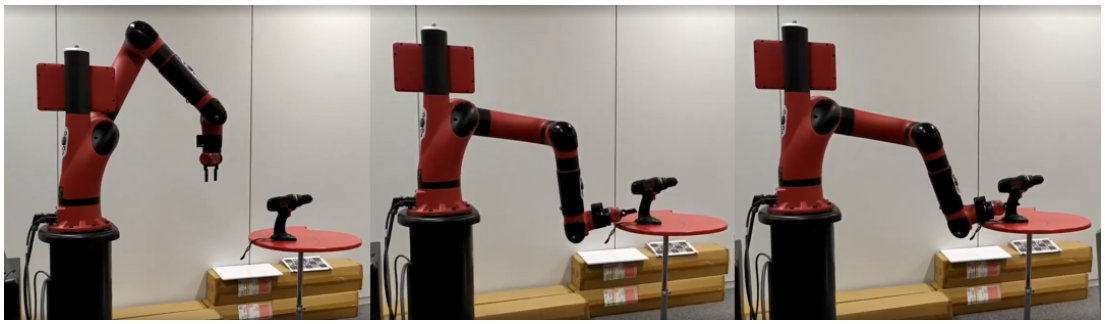

Figure 4.2: Sequence of tool grasping.

4.1.3 Conclusions

We presented a novel neural network that can produce results similar to those of the state of the art methods even though it only uses RGB images as inputs. Having only RGB images as inputs allows us to use a method of automatic dataset generation that facilitates training with new objects in a quick and effortless way.

In a first experiment, we validated that our results are similar to the state of the art in a common dataset for object pose detection.

A second experiment shows that if we use our method for dataset generation, we can obtain large amounts of data for training that result in a better pose regression score.

Finally, experiments with the robot show that we can use our system for robot grasping and even in the cases with greater error in the pose estimation the robot managed to successfully grasp the object.

4.2 Deformable Items

Many of the objects we manipulate in our daily lives are not rigid objects. Deformable objects behave in a very different way and require different perception and manipulation methods. There are many types of deformable objects, we take as a study case rectangular cloth shapes. Cloth manipulation is an operation that is central to many household tasks, including laundry, assisted dressing, and bed

making. This skill, which is simple for most humans, is actually very difficult for robots to perform. The difficulty of cloth manipulation lies in the deformability, nonlinearity, and low predictability of the behaviour of the materials. Because of their deformable nature, compared with rigid objects, cloth objects are also inherently difficult for robots to recognise. This is why it is often necessary to completely unfold cloth items prior to starting a task. An unfolded garment is easier to recognise and manipulate because a robot can then approximate the shape to a model or locate interest points like corners.

A common method of cloth unfolding is to lay the garment flat on a surface and unfold it, as in a pick-and-place problem (23, 50, 51, 52). In (51), similar to our method, the authors present an analysis of the types of corners in order to find strategies for unfolding. By contrast, our approach does not require a table or any flat surface, and involves simply grasping one point of the garment, lifting it into the air, and letting it hang from that point by the effect of gravity.

Typical methods used to open rectangular cloth garments while hanging require locating predefined points and grasping them (27, 53). However, because there are often hidden folds, we analyse the depth of the garment's edges instead of searching for specific points, which allows us to extract information for forming a manipulation strategy. We distinguish between two types of edges: those that belong to the hem of the garment, which we call physical edges, and the remaining nonphysical edges, often formed by folds. Figure 4.3 shows an example of this edge classification. In the image on the right, the physical edges are marked in green and the nonphysical edges are marked in red. Locating physical edges is very useful for find grasping points and to better understand the shape of the garment. Opening the garment requires locating two corners formed by physical edges, which we call physical corners. These two corners should be consecutive i.e., connected by the same physical edge. Once located, grasping each corner with one hand leads to unfold the garment.

The configuration of edge types in the whole garment reveals some patterns. However, the high dimensionality of clothing items makes it very difficult to find global features that could identify an edge as physical or not. On the other hand, local features around edges tend to show slight differences between physical and nonphysical edges.

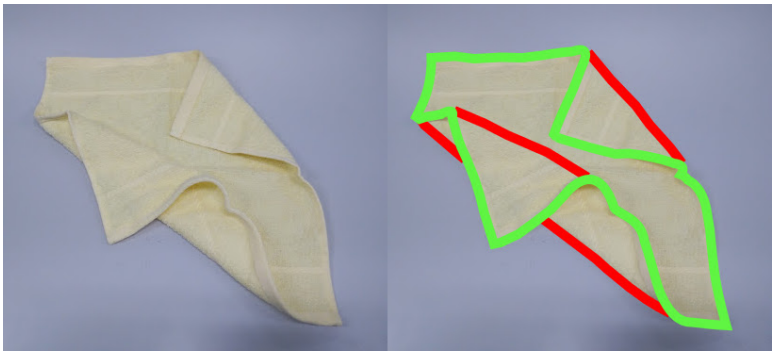

Figure 4.3: Example of physical edges - Green: physical edges. Red: non-physical edges.

Therefore, to classify the edges, we propose a system that combines the results from two classifiers: a local one that selects a small patch around a pixel as an input and a global one whose input is the whole image. Finally, we present a categorisation of the types of corners found in the image of the garment and use this categorisation in an algorithm to actively choose the best robot action for opening the garment.

4.2.1 Cloth Shape Observations

The main problem with working with deformable objects is that the number of configurations they can take is infinite. In order to limit the possible configurations of the garment, we leverage a simple observation to grasp the garment by one of its corners. If the garment is grasped by any random point, then the lowest point of the garment from a frontal view corresponds to one of the corners (see Fig. 4.4). The same observation was used in (27, 29).

Regrasping by that lowest points ensures that the garment is grasped by one of its corners. We thereafter assumed this to be the initial position for all of the experiments. After grasping one corner, we gained insight by looking at how humans manipulate cloth before unfolding it. We found that the first action is often to look for any other contiguous corner and grab it. If the corner is not visible, humans tend to grasp one of the edges and slide the hand towards the corner.

Figure 4.4: Initial position for cloth grasping - When grasping the garment from a random point, the lowest point from a frontal view corresponds to one of its corners.

4.2.2 Analysis and Categorisation of Cloth Folding Patterns

We present a categorisation of the possible configurations of a cloth item. Next, we use the result to reveal and grasp the hidden corner. To understand how the garment is folded, expanding on the work in (22), we focus on distinguishing between physical and nonphysical edges, as mentioned earlier in the *Introduction* section.

Based on the edge types, the type of corner made by the edges can be classified. In the method proposed in this paper, we focus on the lateral (leftmost and rightmost) corners of a cloth held in the air and identify its type, as shown in Fig.4.5.

With one physical corner being held, the bottom point always corresponds to the opposite corner. For the other two corners, there are three possible states: visible, curled forward, and curled backward. To evaluate the state of the corners of a garment, we observe the leftmost and rightmost corners of the perimeter as shown in Fig. 4.6. If two physical edges are coming out of that corner, it is a real corner (e.g., the right corner in Fig. 4.6a). If one or more edges are nonphysical,

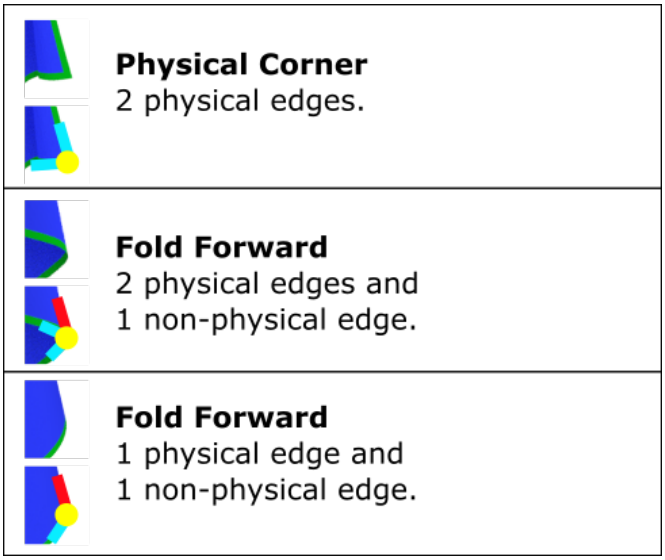

Figure 4.5: Corner categorisation - The three possible configurations of the leftmost and rightmost corners of the cloth depth image.

then it is a pseudo corner. In the case of two edges coming out of the corner, the real corner is folded backward (e.g., left corner in Fig. 4.6a–c). If three edges are coming out of the corner, the real corner is folded forward (e.g., right corner in Fig. 4.6b–e). In the case of a corner folding forward, the actual physical corners are either visible (e.g., right corner in Fig. 4.6b) or hidden (e.g., right corner in Figures 4.6c–e). In cases where it is hidden, further manipulation is needed to reveal it before grasping. Figure 4.7 shows the process that needs to be followed to identify the pattern in the leftmost and rightmost corners.

From this observation, we can see that it is possible to obtain crucial information about how the garment is folded simply by identifying the types of edges leading to the corners in these two points.

4.2.3 Pipeline

Figure 4.8 shows the whole pipeline of the system. First, the robot takes the cloth to the initial position and then, from the depth image, the edges are extracted. Next, the leftmost and rightmost points are located and their folding pattern is classified according to the type and number of edges at each point. Finally, the

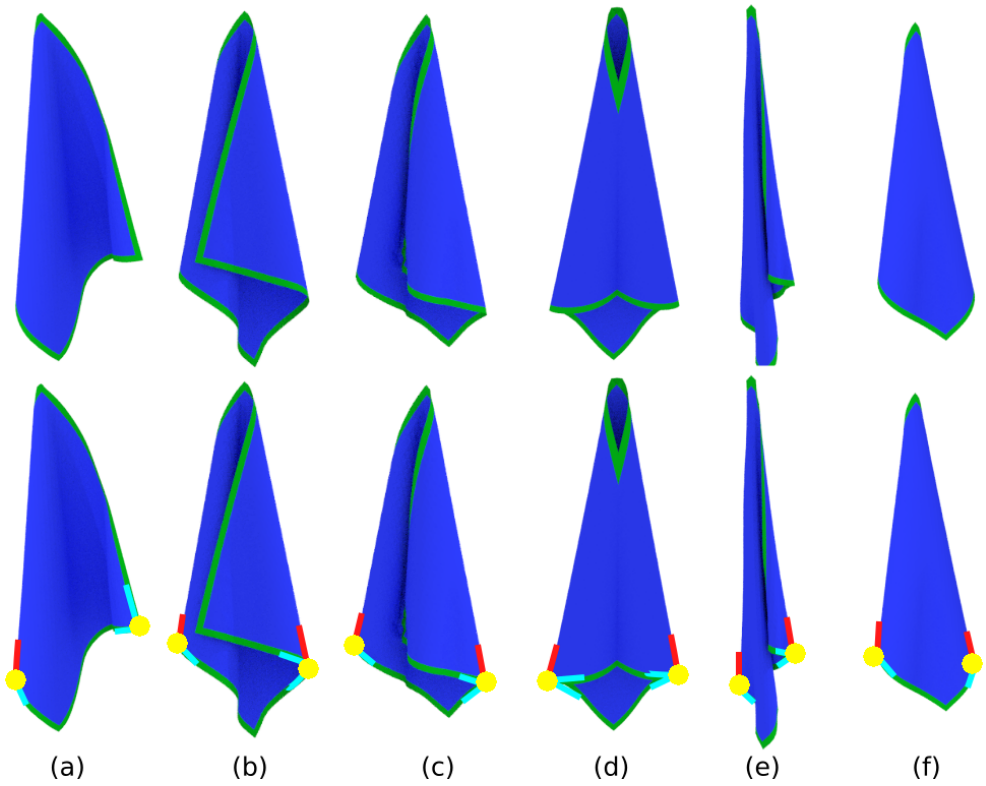

Figure 4.6: Synthetically generated examples of folding patterns - a) has a backward fold on the left and a physical corner on the right; b) has a backward fold on the left and a forward fold with a visible corner; c) is the same as b) but the corner is not visible; d) has forward folds with non-visible corners on both sides; e) has a backward fold on the left and forward fold on the right with a hidden corner; and f) has backward folds on both sides.

robot executes an action according to the observation.

In the next sections, we explain the details of each stage.

4.2.4 Framework

The vision system takes a depth image of a garment as input data and classifies its edges as physical or nonphysical. It consists of two detectors: a local one and a global one. The local one only considers small patches in the image, around the point that it classifies. This is useful for the generalisation of other garments, but it lacks the ability to consider the global structure in the current item. For

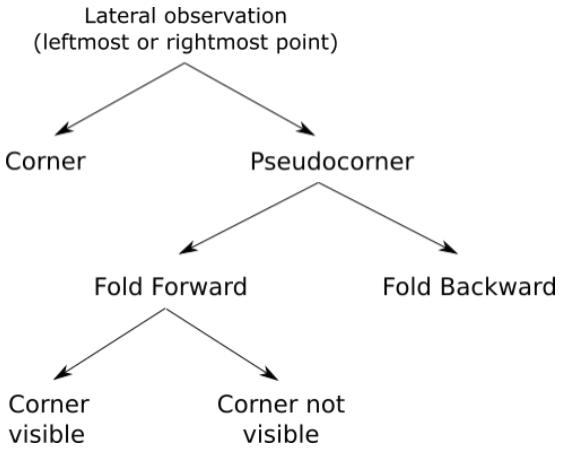

Figure 4.7: Corner classification process

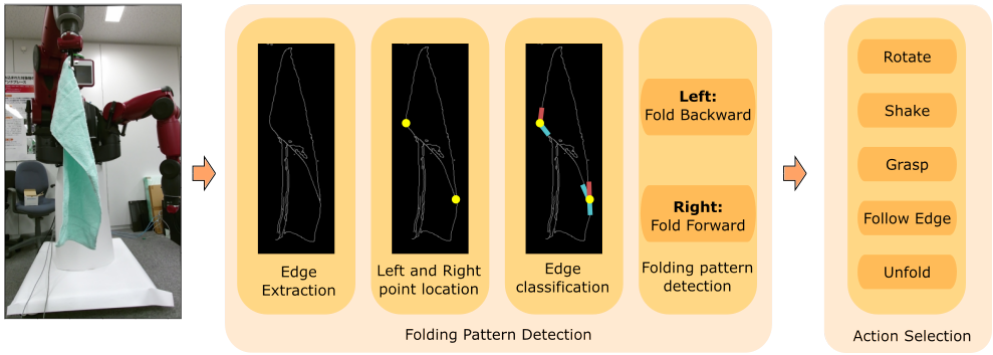

Figure 4.8: Sequence of processes for cloth unfolding - First, the edges are extracted from a depth image. Then leftmost and rightmost points are located and classified. According to the type of folding pattern, an action is selected and executed by the robot.

this purpose, we introduce a global detector that takes into account the whole image as it classifies the pixels.

Training a neural network requires large quantities of labelled data. Manually labelling the physical edges in thousands of images is not feasible owing to time constraints. To overcome this, we use a semi-automatic dataset generation method. We paint the physical edges of a cloth item (as seen in Fig.2.1) and then with an RGB camera, we detect and automatically label these edges. It should be noted that we use the RGB images only to generate the labels; this colour

information is never seen by the neural network, as it only uses depth information. Using this method, we are able to obtain hundreds of labelled images with minimal human intervention. The garment is hung from the robot end effector and rotated while the images are captured. After a full rotation of the garment, the shape of the garment is modified and another round of images is captured.

4.2.4.1 Image Acquisition and Preprocessing

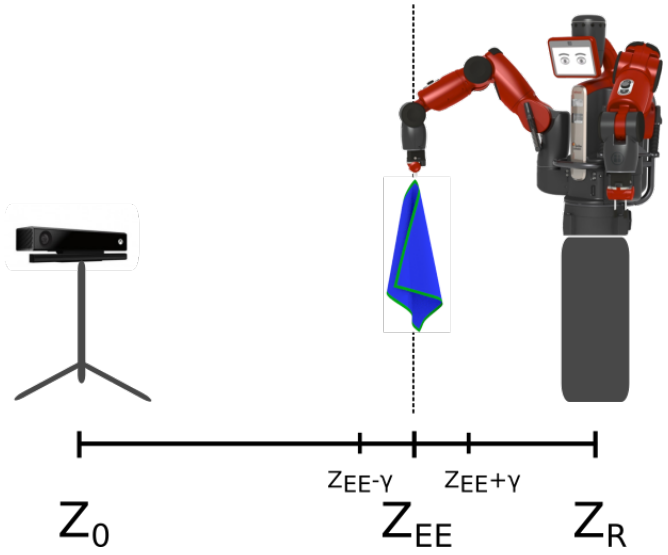

Figure 4.9: Cloth unfolding experimental setup - The robot holds the cloth by one of the corners placing it between the camera and the robot itself

We use a Kinect One sensor placed as shown in Fig. 4.9. The sensor provides an RGB image matrix $I(p)$ and a depth image matrix $D(p)$. Both cameras are calibrated so that each pixel $p = (x, y)$ in the images corresponds to the same location in the real scenario. The camera is also calibrated with the robot so that its position relative to the robot is known.

To remove the pixels that do not correspond to the cloth, we filter by depth, keeping only the pixels that are at a distance $Z_{EE} \pm \gamma$ near the end effector (as shown in Fig. 4.9). Next, we extract the edges from the filtered image using the Canny algorithm (17). We denote V_d as the set of pixels in the resulting binary image.

The RGB image is only used during training to generate label images $\{\hat{Y}(p)_0 ... \hat{Y}(p)_N\}$. When we train using a cloth with painted edges, we segment each image by colour to extract a binary image label in which $\hat{Y}(p) = 1$ if the pixel p corresponds to a physical edge. Otherwise, it is zero.

4.2.4.2 Local Detector

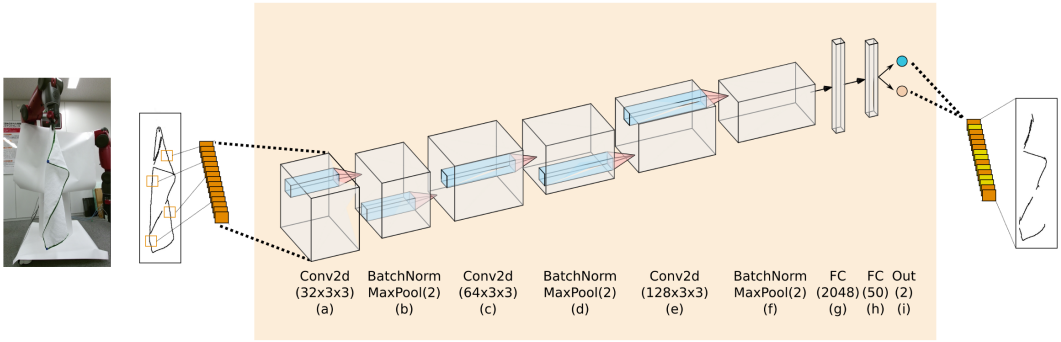

Figure 4.10: Local edge detector network - The Local Detector consists of a convolutional layer (a) of size (32 x 3 x 3) followed by a batch normalization layer, a max pool of size 2 and ReLU activation. The same structure is repeated through (c–d) and (e–f) with convolutions of sizes (64 x 3 x 3) and (128 x 3 x 3) respectively. Then the output of (f) is rearranged as a vector of size 2,048 in (g) and followed by two fully connected layers of sizes 500 and 2.

As a local detector, we use the same structure as we did in our previous work (22). Figure 4.10 shows the way the inputs and outputs to the network are arranged. For each pixel in V_d, a patch $h(p)$ of size 50 x 50 is extracted around that point from $D(p)$. The patch size was determined empirically by visually analysing the images. It corresponds to a size that is big enough to contain some context surrounding the point and small enough to avoid capturing other nearby edges that could affect the classification. Batches of patches are fed into the neural network. After the input layer, we set a convolutional layer (Fig. 4.10a) with 32 convolutional kernels of size 3 x 3 and stride 1. The next layer (Fig. 4.10b) is a batch normalization layer followed by a max pool layer of size 2 and rectifying linear unit (ReLU). This structure is repeated in the subsequent layers (see Fig. 4.10c–d), with a 64-kernel convolution of the same size. The last set

of convolution layers (Fig. 4.10e–f) consists of 128 kernels of the same size as the previous ones. The output of (f) is linearly rearranged, forming a vector of length 2,048 (g), which is then passed to a fully connected layer of 500 neurons (h). Finally, the output layer (i) has two neurons that activate, indicating the probability of the pixel belonging to a physical or nonphysical edge.

For each batch of N samples $X = \{h(p_0), ..., h(p_N)\}$ (with p from the set V_d), the neural network returns $\{y(p_0), ...y(p_N)\}$ with $y(p)$ being the probability of pixel p belonging to a physical edge. We then evaluate the binary cross entropy loss:

$$BCE_{loss} = \frac{1}{N}\sum_{p=0}^{N} -\hat{Y}(p)\log y(p) - (1-\hat{Y}(p)\log(1-y(p))) \tag{4.2}$$

4.2.4.3 Global Detector

Since the local detector classifies pixels individually without taking into account the full cloth, it is susceptible of presenting discontinuities in an edge. To compensate this effect, we use a global detector that takes into account the whole image and classifies every pixel in the image by using a fully convolutional neural network.

Figure 4.11 shows the structure of the network. The orange boxes represent the feature maps at each convolution layer. The yellow boxes are the feature maps at each deconvolution layer merged with the features from early stages of the neural network (represented by the gray arrows). Each box follows the ResNet architecture (54) and is followed by a batch normalization layer and ReLU activation.

In this case, we formulate the problem as a multi-label problem. Each of the N-label images $\bar{Y}_N^{(k)}$ contains K binary images, one for each of the K categories. We use $K = 3$ with one category corresponding to the physical edges ($\hat{Y}$), one for the nonphysical edges ($V_d - \hat{Y}$), and the rest of the pixels corresponding to the background.

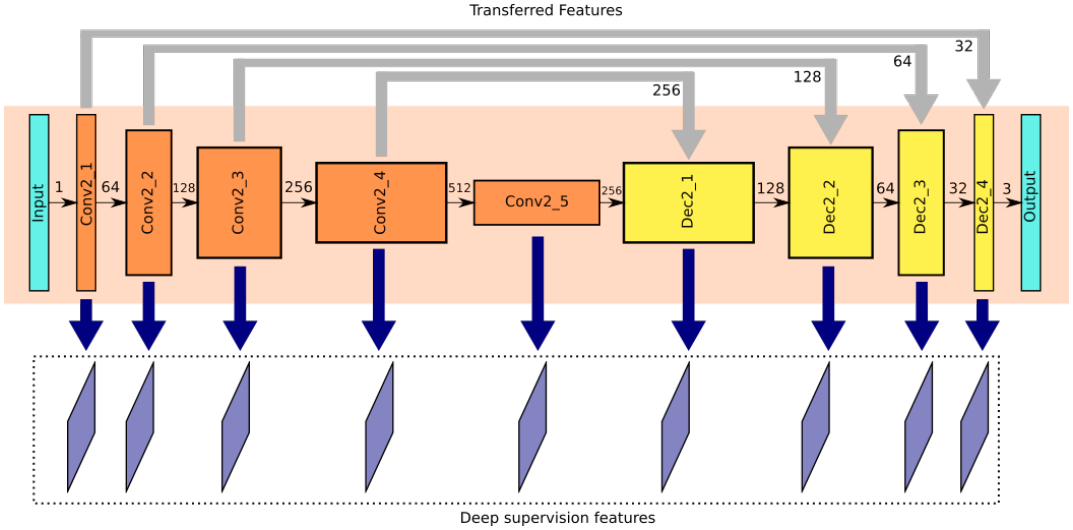

Figure 4.11: Global detector network - The global detector is a fully convolutional neural network with deep supervision. The orange boxes represent the feature maps at each convolution layer. The yellow boxes are the feature maps at each deconvolution layer merged with the transferred features from early stages (grey arrows). The blue arrows represent the feature extraction for deep supervision at each layer.

The multi-label loss (ML_{loss})is defined as

$$\begin{aligned} ML_{loss} = \sum_{k=0}^{3} \sum_{p} & -\epsilon \bar{Y}(p)^{(k)} \log Y(p)^{(k)} \\ & - (1-\epsilon)(1-\bar{Y(p)}) \log (1 - Y(p)^{(k)}) \end{aligned} \tag{4.3}$$

To compensate for the skewness in the dataset, we use ϵ and $(1-\epsilon)$, which represent the percentage of non-edge and edge pixels respectively.

Similar to other works (20, 21) we perform supervision at each stage. Supervision layers (represented by blue lines in Fig.4.11) extract feature layers at each stage. We denote the weights as $W = \{w^0, ..., w^n\}$ for each of the $n = 9$ layers. The supervised loss is evaluated as the sum of the multi-label loss of each of the individual layers:

$$L_{supervision}(W) = \sum_{i=0}^{n} ML_{loss}(Y_{w_i}) \tag{4.4}$$

The final loss $\mathcal{L}$ consists of the loss at the output layer and the supervision loss:

$$\mathcal{L} = ML_{loss}(Y_{out}) + \lambda L_{supervision} \tag{4.5}$$

where λ is a parameter between 0 and 1 that defines the weight of the supervision in the final loss.

4.2.4.4 Output

For each pixel, we have two classification results, one coming from the local detector and the other from the global one. We can ponder the outputs to give more importance to generalisation or global structure by tuning β.

$$Y = \beta Y_{local} + (1 - \beta)Y_{global} \tag{4.6}$$

4.2.5 Action Planning

We introduce five actions the robot can take to accomplish the goal of unfolding the garment: Grasp, Rotate, Shake, Follow-Edge, and Unfold. The action Unfold, is the last action (as shown in Fig. 4.12g) and after that the garment should be in an unfolded state. Otherwise, the process starts again from the beginning. The Rotate action performs a rotation of the garment around the vertical axis by rotating the end effector of the robot arm that holds the cloth. The Grasp action is performed with the free hand by grasping a point on the garment, usually a corner. In the Shake action, the arm that is holding the garment allows it to spread vertically by the effect of gravity. Finally, Follow-Edge, moves the right hand's end effector along one of the physical edges.

The algorithm starts from the initial position i.e., the robot holding one of the garment's corners (Fig. 4.12a). We assume this position can be reached following the observation in the *Cloth Edge Classification* section. In other words, the robot first grabs the garment by any point and then, with the other arm, grasps the lowest point, which corresponds to a corner.

Next, the farthest horizontal point is examined (Fig. 4.12b). A hanging garment will typically take the shape of a rough triangle, with its hypotenuse

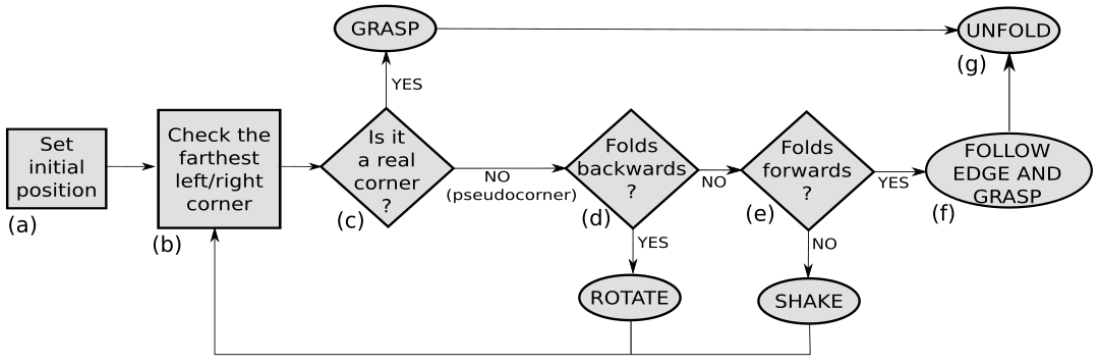

Figure 4.12: Unfolding algorithm.

along the vertical axis. We showed in the *Analysis and categorisation of cloth folding patterns* subsection that this outer corner is crucial to understanding how the garment is shaped.

We then analyse the edges that are connected to the farthest corner. If there are two physical edges (Fig. 4.12c), the corner in question is a real corner and we can proceed to grasp and then unfold it by extending it.

If it is a pseudo corner, we look more closely at the edge types and determine the type of folding, as shown in the *Analysis and categorisation of cloth folding patterns* subsection. If the edge folds backward (Fig. 4.12d), the corner is probably behind the garment and the appropriate action is to rotate the garment to reveal the corner.

If it folds forward (Fig. 4.12e), we will move the end effector of the free arm along the trajectory defined by the physical edge to reveal the corner, grasp it, and then unfold the garment.

If the detected edges do not correspond to any of the defined categories, we will perform an action to shake the garment to loosen any folds and extend it by the effect of gravity. Then, we start the process again.

4.2.6 Experiments

4.2.6.1 Experimental Setup

In all of the experiments, we use a Baxter robot with a Kinect One camera facing each oher, as seen in Fig. 4.9. The neural networks are implemented using the

open software Pytorch (55). The GPU is an NVIDIA GTX1080 with 8 GB of memory, and the CUDA edition is 10.0. In all of the experiments, unless stated otherwise, $\lambda = 1$ in Eq.4.5 and $\beta = 0.6$ in Eq. 4.6. Training was done with a garment with painted edges (see Figs. 2.1 and 4.4), from which we extract more than 1,600 images. This amounted to more than 3.2 million patches.

Each experiment begins with the robot holding a cloth with its right arm as an initial state, then taking actions to unfold it with the left arm (Fig. 4.14). The camera is calibrated and its position with respect to the robot is known. We conducted three types of experiments. First we analysed the robot's performance in edge classification and grasping for 20 attempts using the same garment (see Figs. 2.1 and 4.4). Then we validated the results of our method by having the robot unfold several previously unseen garments. Finally, to demonstrate the effectiveness of the global detector, we show an ablation study comparing the local + global detector with the local only detector from previous work (29).

4.2.6.2 Training

The training progress is shown in Fig.4.13. The top row shows the loss and accuracy during training and validation. We train for 15 epochs, stopping before any signs of overfitting. To demonstrate that the amount of data gathered is enough, we trained the system with increasing amounts of samples in the dataset. The left graph in the bottom row of fig. 4.13 shows that the loss quickly decreases as we increase the size of the dataset. After around 750 samples, the change in the loss relative to the dataset size decreases more slowly, indicating less significance of adding more data. The graph on the bottom right shows the accuracy, which inversely grows at a similar rate.

4.2.6.3 Classification and Grasping

In order to show the effectiveness of the method and determine the stage at which possible errors might occur, we performed 20 attempts to unfold a single garment (seen in Figs. 2.1 and 4.4). We studied these attempts to ascertain whether the edge classification had been produced correctly and the grasping and unfolding were successful. The results are summarized in Table 4.3. Figure 4.14 shows an

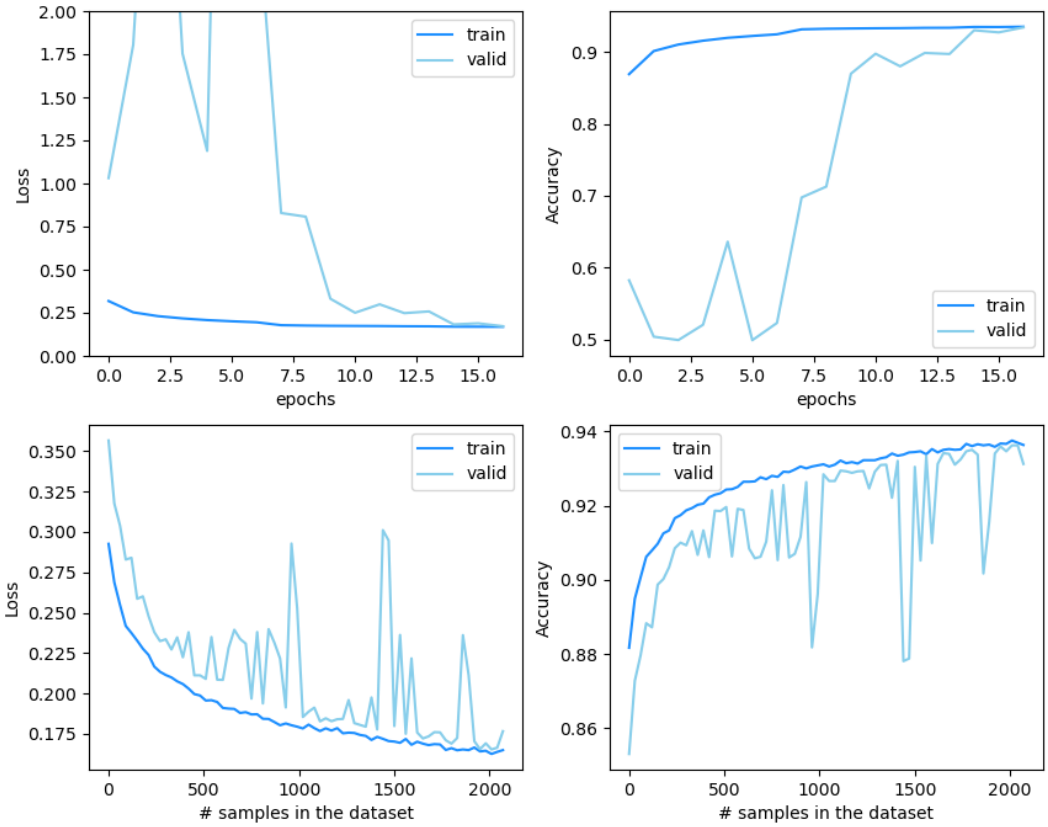

Figure 4.13: Edge detection training details - Top left: training and validation loss. Top right: training and validation accuracy. Bottom left: loss for different dataset sizes. Bottom right: accuracy for different dataset sizes.

example of unfolding when the inner corner is hidden. The robot followed the trajectory of the physical edge to reveal the corner, grasp it, and successfully unfold the cloth.

Table 4.3 shows the four possible outcome cases depending on the success or failure in corner classification and unfolding. A circle in the corner classification column indicates that the corners were correctly classified in all the steps. The first row indicates that 75% of the times the unfolding was successful with correct corner classification in every step. The second row indicates that in the 10% of the cases in which the Edge classification was not successful, the Grasping was. This result is produced in cases where, somewhere in the process, there are errors in the classification, but after some action (like Rotate) the next step led to

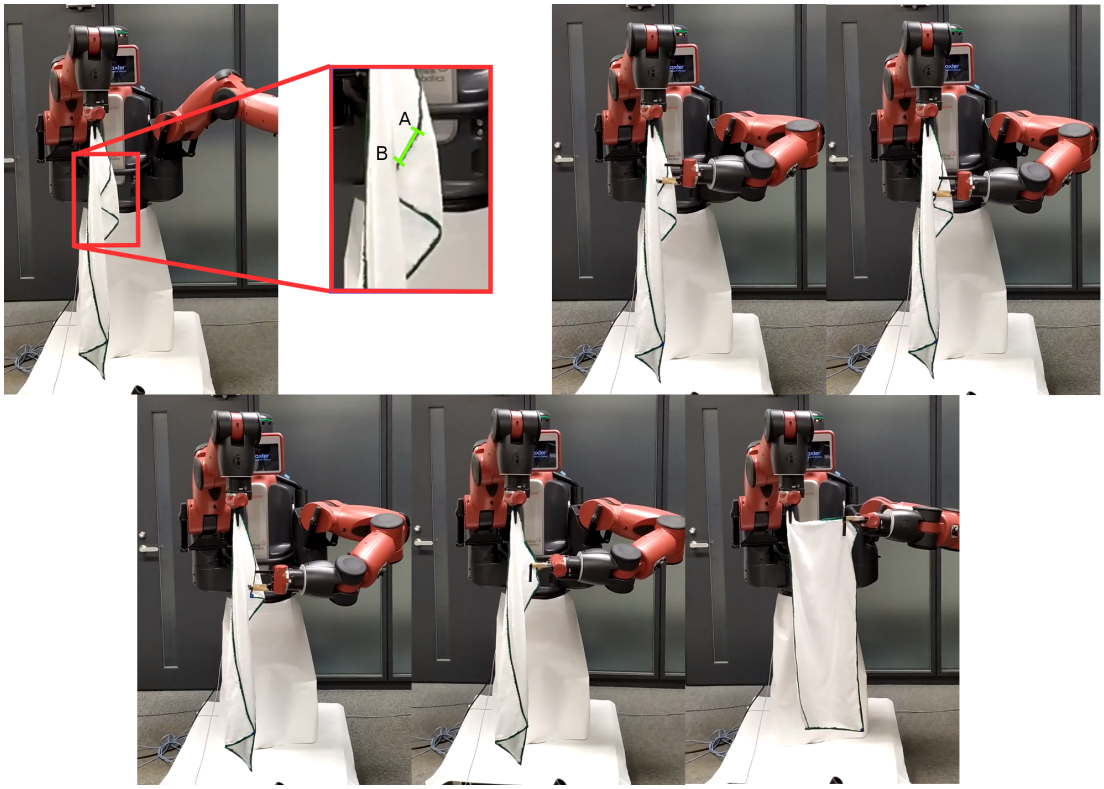

Figure 4.14: Cloth unfolding sequence - Sequence of the robot unfolding a hidden corner folded forwards. By moving the robot's gripper through a trajectory defined by the physical edge (the points from A to B) we can reveal the hidden corner.

a correct classification and grasping. In this case, the corner classification was correct in 85.72% of the steps

Note that we do not differentiate between success in grasping and success in unfolding because a successful grasp led to successful unfolding in every attempt.

4.2.6.4 Generalisation

We tested the results of the system through experiments using four cloths of different sizes and textures that were not seen during training (shown in Fig. 4.15). The robot attempts to grasp each garment 20 times and the success rate of edge classification and unfolding are shown in Table 4.4. For each attempt, we consider that the corner classification is successful if it was correct in all the steps,

Corner Classif.	Grasping Unfold	Ratio
O	O	75%
X(1)	O	10%
O	X	15%
X	X	0%

(1) Corner classification was correct in 85.72% of the steps

Table 4.3: Outcomes of 20 unfolding attempts. An "X" in a cell means the operation in that column was not successful, whereas a circle indicates success. The ratio indicates the percentage of attempts with that particular corner classification and unfolding outcome.

and the unfolding is considered successful if the physical edges form a square. The success rate represents the percentage of success in the 20 attempts. The cloth A (seen in Fig. 4.15) reaches 100% in corner-type classification. This garment is, in fact, the most similar to the one used during training. Cloths B and D are smaller and have different folding patterns. Cloth B is the most different in terms of colour texture and it has the lowest classification accuracy. That said, the cause of the lower corner classification success ratio is not the colour texture, but the tendency of this cloth to curl up and hide its edges more often than others. Images with a physical edge present are correctly classified most of the time regardless of the cloth's texture. Error cases tend to appear in cases with hidden edges. These are more difficult to classify, and an increased tendency of a cloth to curl is the main factor affecting the classification success ratios.

4.2.6.5 Ablation Study

To show the benefits of using both global and local classifiers, we compare the percentage of correctly classified pixels in the edges when taking into account both local and global classifiers with the ablated version using only the local classifier as in (22). For this experiment, not only the classification of the edges in the corner is considered, but the classification of all the edges in the image.

Cloths	Corner Classif.	Unfolding
Cloth A	100%	85%
Cloth B	85%	80%
Cloth C	95%	95%
Cloth D	95%	85%

Table 4.4: Success ratios for different unseen cloths during training. The middle column shows the success ratio in edge classification, and the right column the success ratio in grasping and unfolding.

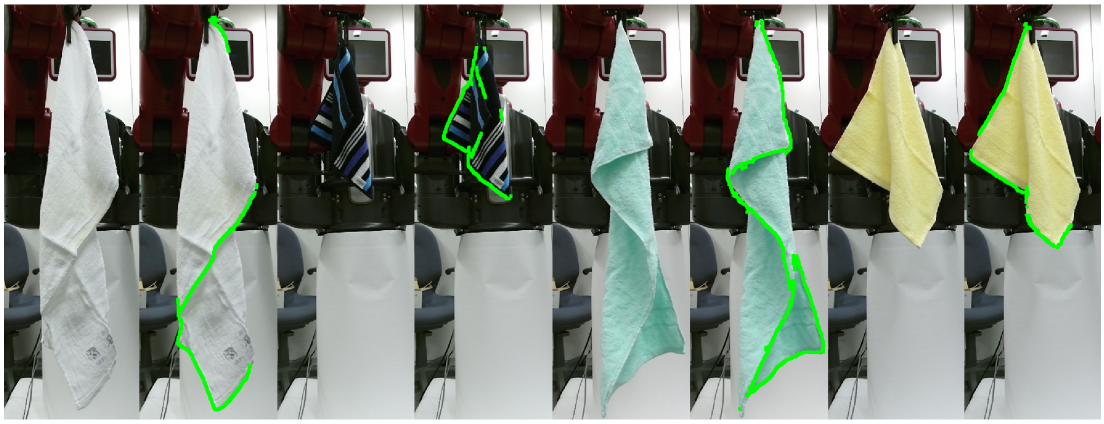

Figure 4.15: Edge classification results - Examples of edge detection in unseen garments (Cloths A to D respectively).

The success ratio represents the ratio of pixels in the image's edges that are correctly classified to the total number of pixels in the image's edges. Table 4.5 shows the results for each cloth. Again, cloths A and C, which are the biggest in shape (the length is similar and they are rectangular) and more similar to the one used during training in terms of cloth texture, are the ones with the best accuracy. The global detector does not add a big increase in the accuracy, since the results of the local detector are already high. Cloths B and D are shorter and more squared; benefit from knowing the whole structure of the garment and significantly improve their results when taking into account the global classifier.

Cloths	Local	Global+Local
Cloth A	81%	88%
Cloth B	78%	86%
Cloth C	83%	89%
Cloth D	74%	89%

Table 4.5: Success ratios for different unseen cloths during training. The middle column shows the percentage of correctly classified pixels using only the local classifier and the right column shows the result after the refinement with the global classifier.

4.2.6.6 Failure Cases

Figure 4.16 shows the most common example of failure in edge detection. Because we are exclusively using the depth image to detect the edges, having the edges very close to another layer of cloth can lead to failure in their detection.

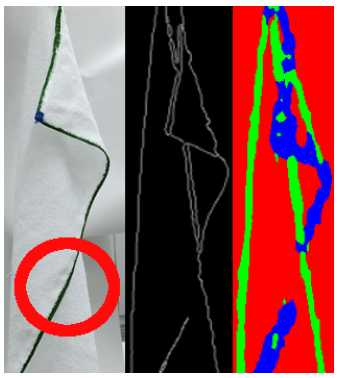

Figure 4.16: Label generation failure case - When the edges are very close to another layer of cloth, this can lead to errors in the edge detection.

This kind of error, however, only tends to happen around the center of the cloth. The proposed method for analysing the leftmost and rightmost corners generally avoids this kind of error because there is usually a background behind those points and not another layer of the cloth. There are two main possibilities to increase the accuracy in pixel detection. First, by improving the inputs, that is improving the resolution of the sensor or adding more channels (colour RGB).

The other possibility is to focus in the design and optimization of a new model of neural network.

The most common cause of failure in manipulation and, also the main general cause of failure, is the incapability of finding a solution in the inverse kinematics for the grasping point. This is more common when trying to reveal a hidden corner. To solve this we relaxed the tolerance of the goal position and orientation, and tried to find other configurations within a short distance and angle from the desired configuration. We also use an L-shaped gripper to grasp the edge at an angle, making it easier to find a solution for the inverse kinematics. In order to further improve the results, a more task-specific robot could be designed to better satisfy the task. However, we chose to use a two-arm robot with 7 degrees of freedom in each arm which is a common design.

4.2.7 Conclusions

We presented a method that overcomes the difficulties in understanding the shape of deformable objects by analysing their edges and providing a method to detect edges that are good for grasping.

The same method is used in a proposed categorisation of folding patterns. This categorisation allows to find and grasp two consecutive corners. Which is necessary when unfolding garments, an operation necessary as a previous step to several operations like garment recognition, folding or ironing. The method presented allows to locate the two consecutive corners even when the garment is curled up and the corners are hidden.

We showed experiments to validate the edge detection system as well as the garment unfolding pipeline. The experiments also showed the generalisation skills to other cloths.

5

Finding a better view

5.1 Problem description

In the previous chapter, we saw how in the case of deformable objects often the interest point we are looking for is hidden and to reveal it we need to interact with the environment. Rigid objects are not exempt of this problem and when we provide a single view of the object we might find that the view is ambiguous or the object is occluded (see Fig.5.1).

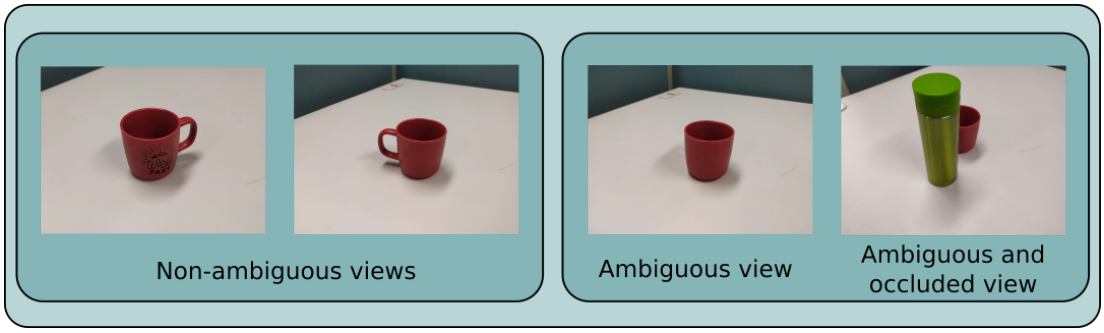

Figure 5.1: Ambiguous and non-ambiguous views of an item -

Humans, naturally, when faced with an ambiguous view of an object, move to a new viewpoint, move the object to view a different part of it, or move the occluding object away if there is any.

In this chapter, we tackle the problem of finding a good viewpoint for pose recognition by moving a camera mounted on a robot arm. First, we need a trained

pose estimator. The system works independently of the type of pose estimator and any estimator can be used, including the one presented in section 4.1, but also any simpler or more complicated method. The goal of the new system is to move the camera to a viewpoint that provides images to the pose estimator that are unambiguous and easy to obtain the pose of the object.

We define the problem as a Markovian process (Fig.5.2).

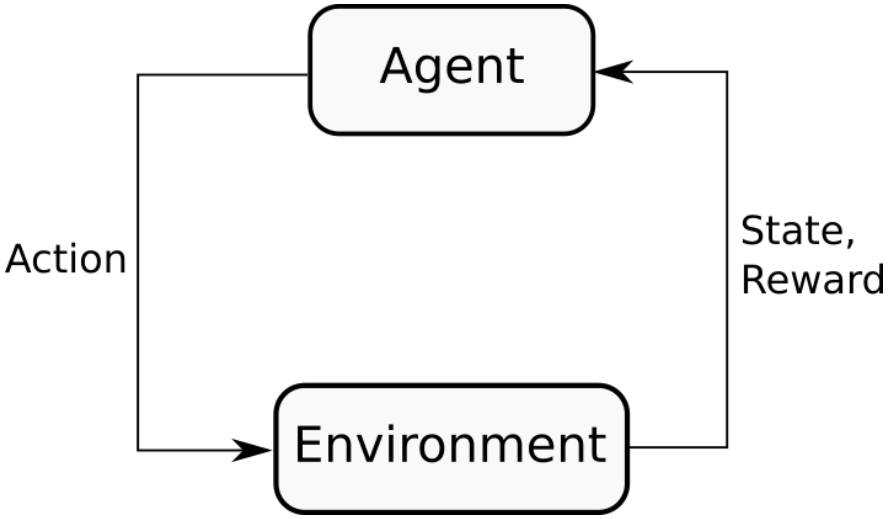

Figure 5.2: Structure of the Markov Decision Process - On each timestep the agent takes an action upon the environment and observes the new state and reward.

- Our **environment** (Env) is the real or simulated world.
- The **state** S_t is an observation of the environment $O(Env)$ at time t made with a 3D camera together with the current joint positions Jp_t and velocities Jv_t (Fig.5.3).
- The agent can act on the environment by taking an **action** (a) which consists in moving the end effector towards certain direction ΔEE_{dir} and orientation ΔEE_{rot}.
- This action comes from a **policy** $\pi(a|s)$ that chooses the best action according to the current state and reward.

- **The reward** (r) consists in 3 factors:

$$r = k_{RE} \cdot r_{RE} + k_{IG} \cdot r_{IG} - k_D \cdot r_D \tag{5.1}$$

Pose regression error r_{RE}: The error of the trained pose estimator.

Information Gain r_{IG}: The percentage of points of the object that are visible compared to the previous view.

Distance to other objects r_D: Negative reward for getting too close or touching objects.

Factors k_{RE}, k_{IG} and k_D allow to regulate the weight of each component towards the total reward.

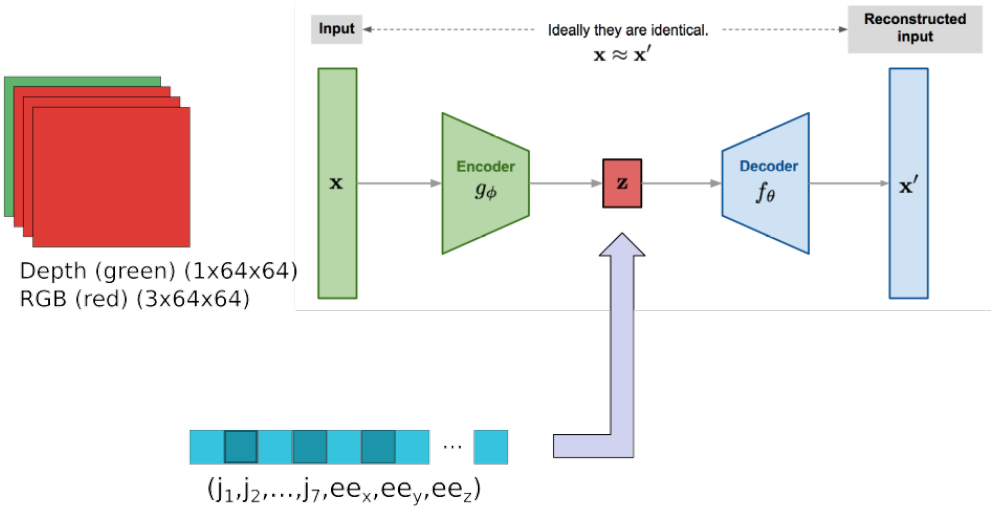

Figure 5.3: State observations - An autoencoder is used to obtain a vector representation of the image which is then combined with the robot joint information.

One **episode** is a sequence of observation-action iterations. An episode will end after a number of iterations has passed. The **accumulated reward** is the discounted sum of all the rewards in the episode (Eq.5.2), where γ is the discount factor that accounts for how important are future rewards to the current state.

$$R = \sum_{t=0}^{T-1} \gamma^t \cdot r(s_t, a_t) \tag{5.2}$$

If our actions come from an stochastic policy π parametrised by θ, for a trajectory $\tau = (s_0, a_0, s_1, a_1, ...)$ the expected return is:

$$J(\pi) = \int_{\tau} P(\tau|R(\tau)) = E_{\tau\sim\pi}[R(\tau)] \tag{5.3}$$

Then, to maximise the reward, the goal is to find the parameters θ of the optimal policy π^*, expressed as:

$$\pi^* = \underset{\pi}{\text{argmax}} J(\pi) \tag{5.4}$$

5.2 Experiments

5.2.1 Locating unambiguous views

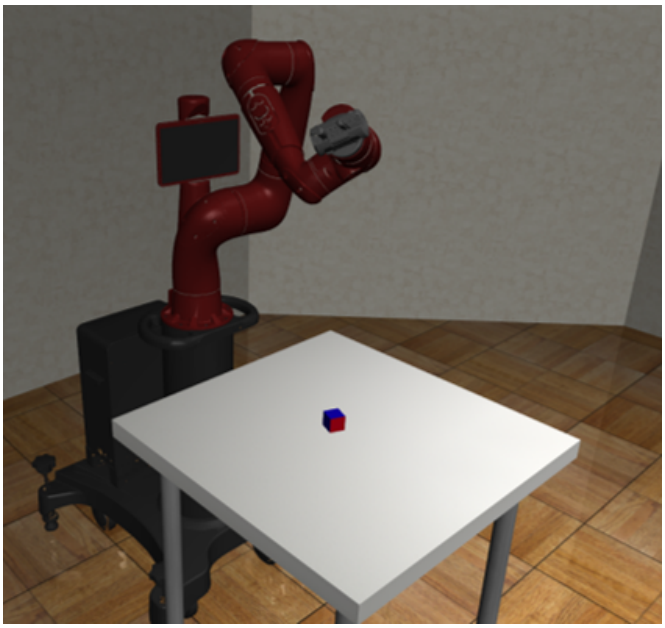

Figure 5.4: Experimental scenario for locating unambiguous views - To obtain an unambiguous view of the object, the robot needs to learn to bring the camera to a viewpoint where the red side of the cube is visible.

First, we want to validate that our method is capable of locating characteristic unambiguous views of an object. For that purpose, we create a scenario where

the robot has to find the pose of a cube. This cube has all sides of equal size and all of them are coloured blue except for one red side. This means that in order to get an unambiguous view of the cube, the agent needs to find a viewpoint where the red side is visible (Fig.5.4).

In this scenario, we focus on validating that the robot is capable of learning to find the most informative views of the object and we do not add any occluding objects yet. At the start of each episode, the cube spawns at a random position on the table and with a random rotation in the vertical axis. The cube never spawns with the red face on top (the solution would be too easy) or with the red face facing the table (the solution would be impossible to find without lifting the cube).

The camera on the end effector starts facing down, the episode horizon is set to 300 interactions and we train using the Soft Actor Critic algorithm (SAC) (56).

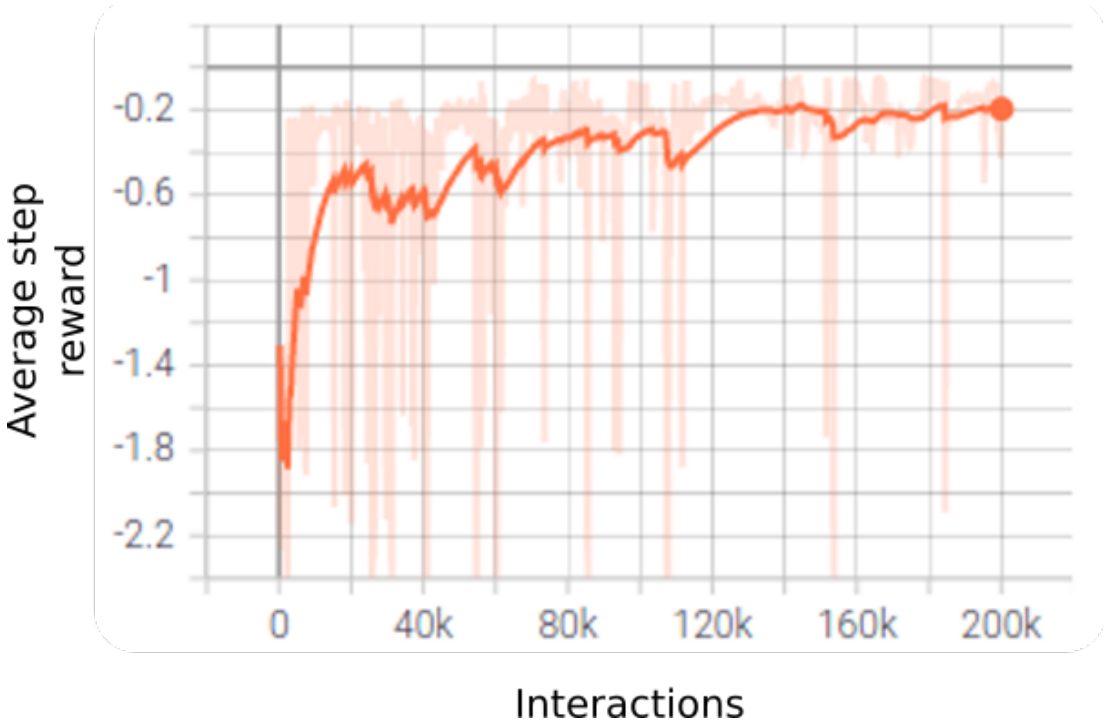

Figure 5.5: Training of the NBV agent to locate unambiguous views - The training ran for 200k episodes and the final average reward was around -0.2.

Details of the training progress can bee found on Figure 5.5. After 200k episodes, the average episode reward (on each interaction) is -0.2. Since the reward is defined negative, the maximum possible reward is 0.

Figure 5.6 shows sequences of 3 episodes of the trained system. It can be observed that the learned behaviour is to move the camera up until the object is visible in the image and then, descend towards the object locating the red face and getting close to it.

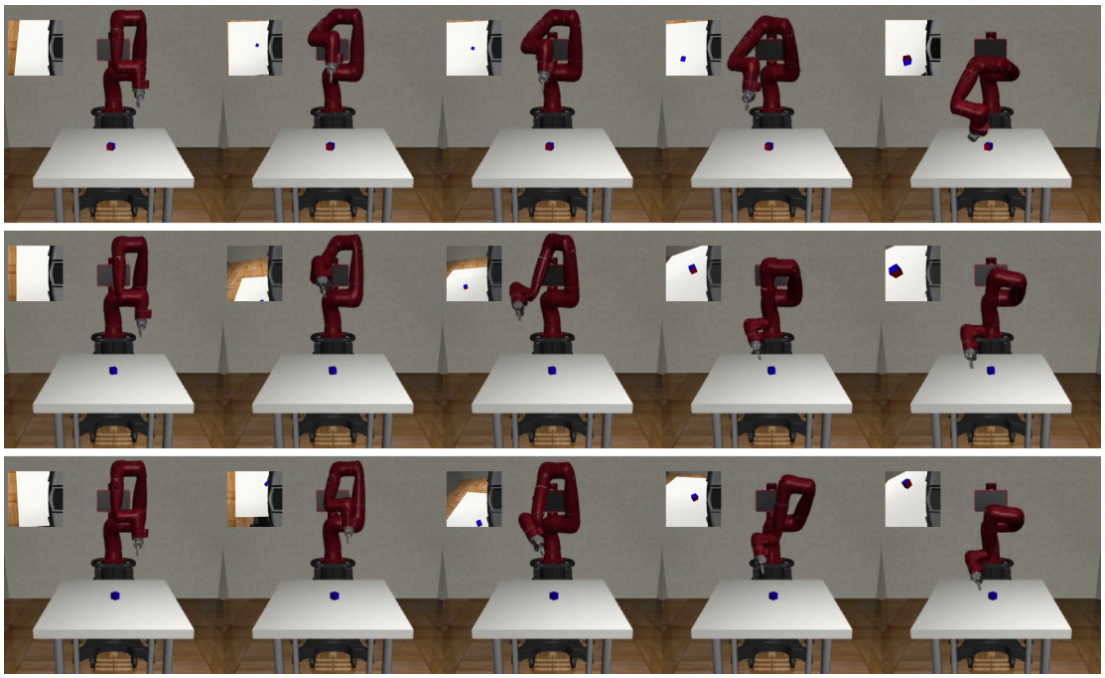

Figure 5.6: Sequences of 3 attempts at finding the best view of the cube without occlusions - The learned behaviour of the agent is to move up searching for the cube on the table and then descend facing the red side of the cube in a close view.

To show that our system improves the results of the single pose estimator even when this pose estimator is not very precise, we compare the results of a simple pose estimator model at different stages of its training process with the results of that semi-trained model in the NBV method.

The pose regressor was trained for 25 epochs, after each epoch the model is saved and used in the NBV scenario. We compare the results of the average error

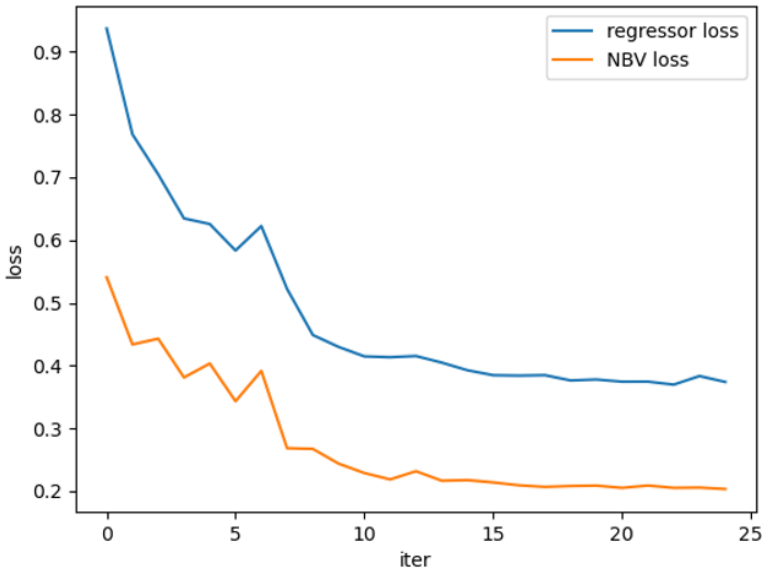

Figure 5.7: Results of NBV vs Single view - Comparison of the results of a pose regressor model at different states of training with the same semi-trained model in the NBV system.

in obtaining the pose from 100 images from the validation dataset evaluated by the pose regressor model (regressor loss), with 100 images obtained after the full rollout of an episode of the NBV scenario (NBV loss). The results show that the learned policy for finding better views improves the results each time, even in the early stages of training (Fig. 5.7).

5.2.2 Occlusions scenario

For the next experiment, we introduce occlusions in the scenario (Fig. 5.8) in order to test the ability of the system to find viewpoints in which the object in the image is free of occlusions or in which the occlusions are minimal.

To the previous scenario, we add a number occluding cubes. These cubes at spawned randomly at the beginning of each episode and their positions and sizes are sampled randomly from a uniform distribution.

Figure 5.8: Scenario with occlusions. - As occluding objects, we add two cubes of random sizes around the target object.

One occluding object.

First we test spawning one single occluding object at a random position between the target object and the robot. The training progress is shown in Figure 5.9, the policy network starts with pre-trained weights from the previous experiments and ran for 300k episodes.

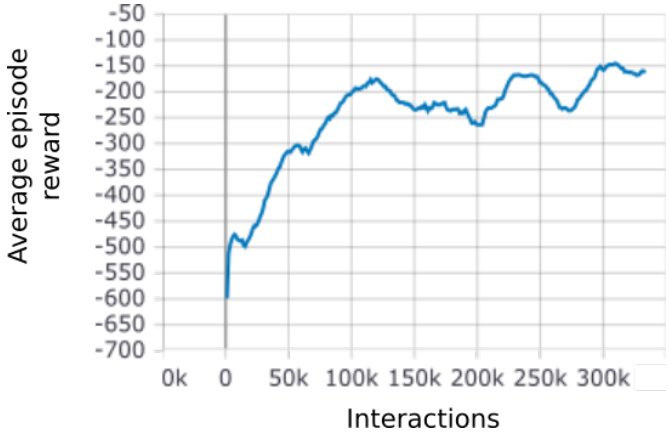

Figure 5.9: Training process with one occluding object. - Training is done starting with pre-trained weights from the previous experiment.

Figure 5.10 shows some qualitative results. Since in this case the occluding object is always between the target object and the robot, the agent learns a policy in which it often reaches a viewpoint from the direction opposite to the robot. However, as shown in the two cases in the bottom, sometimes it fails to find the

red face.

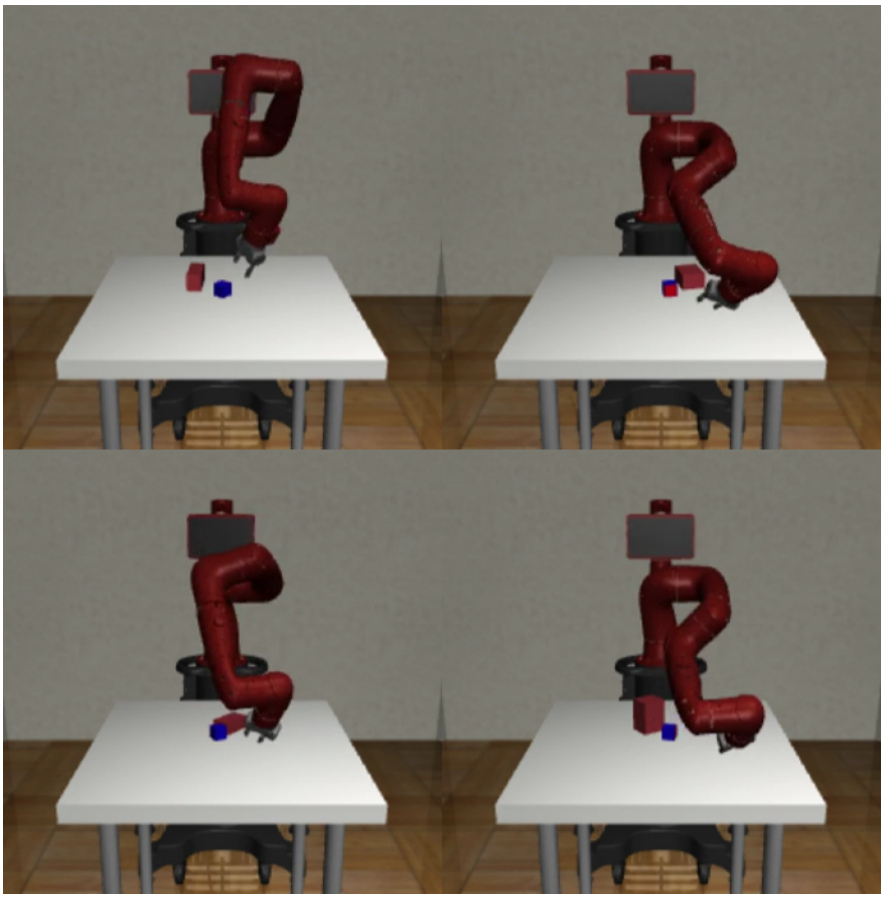

Figure 5.10: Scenario with one occluding object. - The agent learns to get a view from behind the occluding object. In the cases in the bottom the agent manages to avoid the occlusions but fails to locate the red face of the target cube.

Two occluding objects.

Next we add one more occluding object to the scenario. This second object can spawn randomly anywhere around the target object. This makes impossible the strategy learnt in the previous experiment of always looking at the object from the opposite view. Figure 5.11 shows an example of a sequence of finding the best view.

For this experiment, we removed the fingers in the end effector since there is no need to use them at this point. We found this helped avoid unnecessary collisions with the occluding objects and find a good policy earlier.

Finally, to validate our learned policy, we compare our method with two baseline policies:

- **Random policy** This policy detects the centre of mass of the objects in the table and selects a random viewpoint that faces that direction from a random position.

- **Top-down policy** This policy detects the centre of mass of the objects in the table and selects a viewpoint looking directly down from the top of the centre of mass.

We compare the two baseline policies with our policy by running them and taking as a final estimation the average of the last 5 frames in the sequence and calculating the geodesic loss of the pose.

Policy	Loss
Random	1.1
Top-down	0.55
Ours	0.19

Table 5.1: Evaluation of different policies.

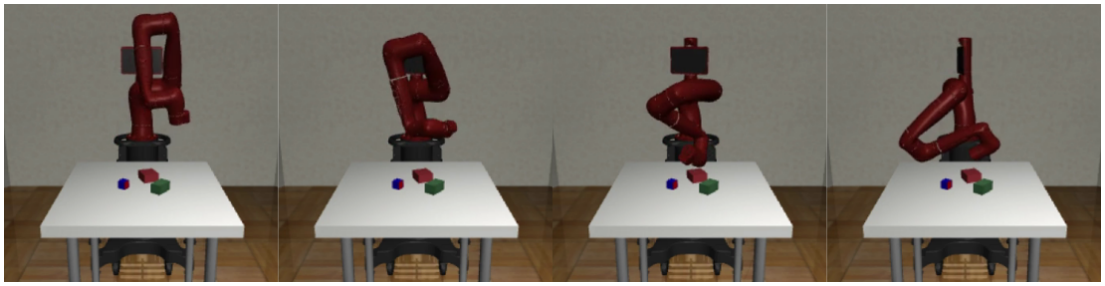

Figure 5.11: Sequences of an attempts at finding the best view of the cube with two occluding objects - The robot learns to reach the object and the twist to avoid the occluding objects.

5.2.3 Conclusions

As explained in section 1.2, interaction with the environment is a key element to increase the robustness and success rate of a pose estimation method. In fact, in the previous chapter about cloth manipulation, we showed how it was necessary to interact with the item by rotating it or revealing a hidden corner. In this chapter we presented a method for interaction with the environment that provides similar characteristics to the case of pose estimation of rigid items. Despite that our experiments were done with rigid items, the same method can potentially be extended to any kind of object.

We experimented with our next best view estimator in a virtual environment. First, showing that our method is capable of locating unambiguous views of an object by locating a view of a cube that contains the only distinctive face of it. This toy problem is akin to the case of locating a view of a mug that includes the handle, as seen in the example in the beginning of this chapter.

Then we showed that our method is also capable of finding viewpoints in which the target object is not occluded or the occlusion is minimal.

The results show our method can learn to feed images to any pose estimator to exploit the viewpoints that return better results.

The presented system can be useful in occluded scenarios where a single view of the object is not enough to obtain a reliable pose of the object.

6

Conclusions

We started this thesis with the goal of bridging the cap between the latest results in learning methods for pose recognition and their use cases in common scenarios. Recently, we have seen a surge in methods that use data-driven approaches for solving problems of pose recognition and robot manipulation. However, these results have had little impact in scenarios like households or small stores.

6.1 Dataset generation

One of the big challenges we identified for applying data-driven learning methods is the difficulty in creating a dataset for an object. Obtaining enough pairs of labelled inputs with enough variability in the pose and covering different backgrounds and different light conditions is a very complicated problem that is often left out in most methods that just use any available dataset. Moreover, many methods require other kind of inputs like a 3D model or semantically labelled images, which are very difficult to obtain. We proposed automatic dataset generation systems for both rigid and deformable objects and we showed that, even though as a consequence of generating our labels automatically we have simpler inputs than other methods, by training with larger amounts of data, that is obtained effortlessly in our case, we can compensate for it and obtain results similar to the state of the art. In the experimental section, we showed how the number of generated data is enough for training the neural network and also make use of a

robot to demonstrate how our method can train a system with our automatically generated dataset for doing manipulation of rigid and deformable objects.

6.2 Pose estimation for manipulation

Another challenge we stated at the beginning of this thesis is the necessity to adapt to different kinds of objects. Rigid and deformable objects behave in very different ways and it is not feasible to cover both of them with the same method. We first showed how our proposed method for rigid objects can obtain state of the art results even though it only uses RGB images as inputs. In our experiments with a robot arm, by knowing the 3D pose of the object, the robot was able to grasp the object at a desired point from a desired orientation. Then, we showed our method for deformable objects in which we use rectangular pieces of cloth as a study case. In our method we overcome the difficulties faced by methods that search for specific points by analysing the garment edges instead. This method is also simpler than methods that fit a deformable 3D model to the observation and also does not require having a model of the item.

6.3 Robustness

Finally, even though we made some things simpler for the sake of usability, we still need to guarantee the robustness of our method. For that purpose, we identified interaction with the environment as a means to obtain better information and recover from possible errors.

In our experiments with cloth items we interacted with the item by moving it (rotating it or opening it). This allowed to recover from errors in corner detection after getting more views when rotating the garment or having access to corners that otherwise would not be accessible when the garment is curled up hiding them.

We also proposed a method for obtaining better viewpoints of an object. These are views with less ambiguity and taken from viewpoints that avoid having occluding objects in the image. This method for next best view estimation is not limited to our pipeline but can also work with any trained pose regressor and it

will learn to exploit it by providing the images that the pose regressor is best at estimating.

7

Limitations & Future Work

In the next sections we present the limitations of the methods we have shown for rigid and deformable objects and future avenues of research to solve them.

7.1 Rigid Object Pose Estimation

Our method allows to create a dataset of pairs of images and labels effortlessly. We consider that this is the most critical line of work since the task of creating a dataset is always associated with tedious tasks of labelling. However, there is still a necessity in working towards the time it takes to have a trained posse regressor. The main factor is reducing the training time. Our method allows to gather large amounts of data, but larger datasets can also mean larger training times.

One possible solution is to come up with efficien pre-training methods that allow to start the training with some prior knowledge.

7.2 Deformable Object Pose Estimation

There are many types of objects that fall under the category of deformable objects. We chose cloth items as a working example because they are representative of many items used at household chores. The idea of semantically analysing the edges of the deformable item can be useful to all kind of deformable items, in (22) we analysed the results of directly applying the method to other types of

garments. Figure 7.1 shows an example of a t-shirt. In it the network is supposed to detect as physical edges of the holes in the collar, sleeves, and bottom. The image shows the true positives and negatives in green and yellow respectively and the false positives and negatives in red and blue respectively. The results show that it is able to find most physical edges but mistakenly classifies the folding line in the shoulder as a physical edge. Directly training with more types garments, learning to find their semantical edges, and devising strategies for unfolding them is a promising strategy for this complicated task.

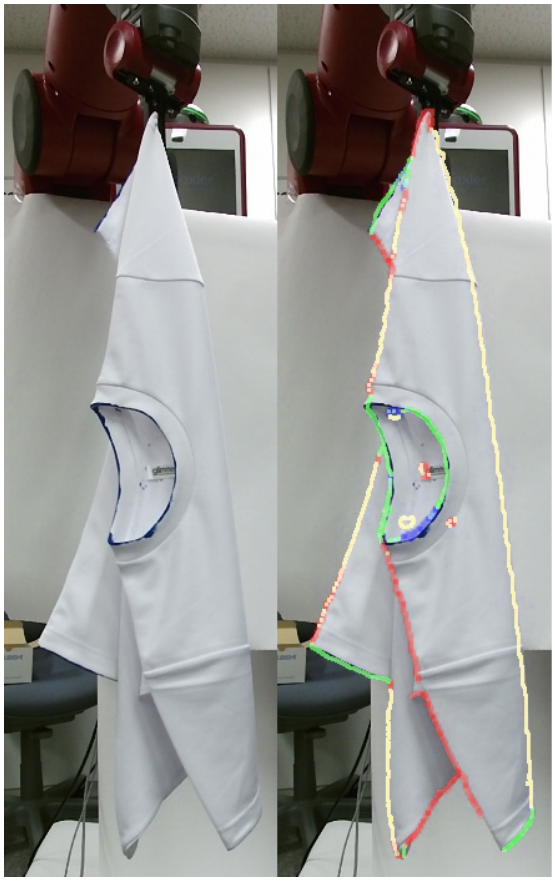

Figure 7.1: Physical edge detection on a t-shirt - Result of directly applying the method of physical detection on items other than squared cloths. In green True Positives. Yellow: True Negatives. Red: False Positives. Blue: False Negatives.

There is also the category of articulated items, which can be considered a mix of both rigid and deformable objects. In our experiments we used a pair

of pliers in our dataset. However, we only considered their closed state. In the case of articulated items, finding semantical edges like those that constitute each articulated part, can be an interesting topic for further research.

References

[1] Nikolaus Correll, Kostas E. Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M. Romano, and Peter R. Wurman. **Analysis and Observations From the First Amazon Picking Challenge**. *IEEE Transactions on Automation Science and Engineering*, **15**(1):172–188, 2018. 1

[2] B Tekin, S.N. Sinha, and P. Fua. **Real-Time Seamless Single Shot 6D Object Pose Prediction**. *CoRR*, **abs/1711.08848**, 2017. 9, 10, 21

[3] S. Mahendran, H. Ali, and R. Vidal. **3D Pose Regression Using Convolutional Neural Networks**. In *ICCV*, pages 2174–2182, 2017. 9, 10

[4] M. Rad and V. Lepetit. **BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth**. In *International Conference on Computer Vision*, pages 3848–3856, 2017. 9, 10, 21, 24, 25

[5] S. Peng, Y. Liu, Q. Huang, H. Bao, and X Zhou. **PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation**. *CoRR*, 2018. 9

[6] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. **6-DoF Object Pose from Semantic Keypoints**. *CoRR*, 2017. 9

[7] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese. **DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion**. *CoRR*, 2019. 9

[8] 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*, jan 2019. 9

[9] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, and carsten Rother. **Uncertainty-Driven 6D Pose Estimation of Objects and Scenes From a Single RGB Image**, 2016. 10, 24, 25

[10] K. Suzui, Y. Yoshiyasu, A. Gabas, F. Kanehiro, and E. Yoshida. **Toward 6 DOF Object Pose Estimation with Minimum Dataset**. In *International Symposium on System Integration*, pages 462–467, 2019. 10

[11] J. M. Wong, V. Kee, T. Le, S. Wagner, Gian-Luca M., A. Schneider, L. Hamilton, R. Chipalkatty, M. Hebert, D. M. S. Johnson, J. Wu, B. Zhou, and A. Torralba. **SegICP: Integrated Deep Semantic Segmentation and Pose Estimation**. *CoRR*, 2017. 10

[12] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox. **Self-supervised 6D Object Pose Estimation for Robot Manipulation**. *CoRR*, 2019. 10

[13] D. Dwibedi, I. Misra, and M. Hebert. **Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection**. *CoRR*, 2017. 10

[14] Li. Y., G. Wang, X. Ji, Y. Xiang, and D. Fox. **DeepIM: Deep Iterative Matching for 6D Pose Estimation**. *CoRR*, 2018. 10

[15] J. Tremblay, T To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield. **Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects**. *CoRR*, 2018. 10

[16] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. **SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again**. In *International Conference on Computer Vision*, pages 1530–1538, 2017. 10

[17] John Canny. **A Computational Approach to Edge Detection**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **8**(6):679–698, nov 1986. 11, 33

[18] Joseph J Lim, C Lawrence Zitnick, and Piotr Dollar. **Sketch Tokens: A Learned Mid-Level Representation for Contour and Object Detection**. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3158–3165, jun 2013. 11

[19] P. Dollar, Zhuowen Tu, and Serge Belongie. **Supervised Learning of Edges and Object Boundaries**. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, **2**, pages 1964–1971, 2006. 11

[20] Zhiding Yu, Chen Feng, Ming-Yu Liu, and Srikumar Ramalingam. **CASENet: Deep Category-Aware Semantic Edge Detection**. *CoRR*, may 2017. 11, 36

[21] Yun Liu, Ming-Ming Cheng, Deng-Ping Fan, Le Zhang, Jia-Wang Bian, and Dacheng Tao. **Semantic Edge Detection with Diverse Deep Supervision**. *CoRR*, apr 2018. 11, 36

[22] Antonio Gabas and Yasuyo Kita. **Physical edge detection in clothing items for robotic manipulation**. In *2017 18th International Conference on Advanced Robotics (ICAR)*, pages 524–529, 2017. 11, 29, 34, 42, 63

[23] Kimitoshi Yamazaki. **Grasping Point Selection on an item of Crumpled Clothing Based on Relational Shape Description**. *IEEE International Conference on Intelligent Robots and Systems*, pages 3123–3128, 2014. 11, 27

[24] Li Sun, Gerardo Aragon-Camarasa, Simon Rogers, and J. Paul Siebert. **Accurate garment surface analysis using an active stereo robot head with application to dual-arm flattening**. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 185–192, 2015. 11

[25] Hiroyuki Yuba and Kimitoshi Yamazaki. **Unfolding an Item of Rectangular Clothing Using a Single Arm and an Assistant Instrument**. *IEEE/SICE International Symposium on System Integration, SII 2014*, pages 571–576, 2014. 11

[26] Dimitra Triantafyllou, Ioannis Mariolis, Andreas Kargakos, Sotiris Malassiotis, and Nikos Aspragathos. **A Geometric approach to Robotic Unfolding of Garments**. *Robotics and Autonomous Systems*, **75**:233–243, jan 2016. 12

[27] Andreas Doumanoglou, Andreas Kargakos, Tae-Kyun Kim, and Sotiris Malassiotis. **Autonomous Active Recognition and Unfolding of Clothes Using Random Decision Forests and Probabilistic Planning**. *Proc. IEEE International Conference on Robotics and Automation (ICRA14)*, pages 987–993, 2014. 12, 27, 28

[28] Enric Corona, Guillem Alenyà, Antonio Gabas, and Carme Torras. **Active Garment Recognition and Target Grasping Point Detection Using Deep Learning**. *Pattern Recognition*, **74**:629 – 641, 2018. 12

[29] Antonio Gabas, Enric Corona, Guillem Alenyà, and Carme Torras. **Robot-Aided Cloth Classification Using Depth Information and CNNs**. In *Articulated Motion and Deformable Objects*, pages 16–23, 2016. 12, 28, 39

[30] Jingyu Hu and Yasuyo Kita. **Classification of the Category of Clothing Item After Bringing It into Limited Shapes**. *IEEE-RAS International Conference on Humanoid Robots*, pages 588–594, 2015 December. 13

[31] Özgür Erkent, Dadhichi Shukla, and Justus Piater. **Integration of probabilistic pose estimates from multiple views**. In *European Conference on Computer Vision*, pages 154–170. Springer, 2016. 13

[32] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker, Alberto Rodriguez, and Jianxiong Xiao. **Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge**. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1386–1383. IEEE, 2017. 13

[33] Alvaro Collet and Siddhartha S Srinivasa. **Efficient multi-view object recognition and full pose estimation**. In *2010 IEEE International Conference on Robotics and Automation*, pages 2050–2055. IEEE, 2010. 13

[34] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. **Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints**. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5010–5019, 2018. 13

[35] C Connolly. **The determination of next best views**. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, **2**, pages 432–435, March 1985. 13

[36] Daryl Peralta, Joel Casimiro, Aldrin Michael Nilles, Justine Aletta Aguilar, Rowel Atienza, and Rhandley Cajote. **Next-Best View Policy for 3D Reconstruction**. In *Computer Vision – ECCV 2020 Workshops*, pages 558–573. Springer International Publishing, 2020. 13

[37] Wen Zhou and Jinyuan Jia. **Training deep convolutional neural networks to acquire the best view of a 3D shape**. *Multimed. Tools Appl.*, **79**(1):581–601, January 2020. 13

[38] Kumar Ashutosh, Saurabh Kumar, and Subhasis Chaudhuri. **3D-NVS: A 3D Supervision Approach for Next View Selection**. *CoRR*, December 2020. 13

[39] Riccardo Monica and Jacopo Aleotti. **A Probabilistic Next Best View Planner for Depth Cameras Based on Deep Learning**. *IEEE Robotics and Automation Letters*, **6**(2):3529–3536, April 2021. 13

[40] Sergi Foix, Guillem Alenyà, and Carme Torras. **Task-driven active sensing framework applied to leaf probing**. *Comput. Electron. Agric.*, **147**:166–175, April 2018. 13

[41] Akanksha Saran, Branka Lakic, Srinjoy Majumdar, Juergen Hess, and Scott Niekum. **Viewpoint selection for visual failure detection**. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5437–5444, September 2017. 13

[42] Andrea Roberti, Marco Carletti, Francesco Setti, Umberto Castellani, Paolo Fiorini, and Marco Cristani. **Recognition self-awareness for active object recognition on depth images**. In *BMVC*, page 15, 2018. 13

[43] Markus Grotz, David Sippel, and Tamim Asfour. **Active Vision for Extraction of Physically Plausible Support Relations**. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 439–445, October 2019. 13

[44] Nik Mohd Zarifie Hashim, Yasutomo Kawanishi, Daisuke Deguchi, Ichiro Ide, Ayako Amma, Norimasa Kobori, and Hiroshi Murase. **Best Next-Viewpoint Recommendation by Selecting Minimum Pose Ambiguity for Category-Level Object Pose Estimation**. *Journal of the Japan Society for Precision Engineering*, **87**(5):440–446, 2021. 13

[45] P. Pérez, M. Gangnet, and Blake. **Poisson image editing**. In *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, **22(3)**, page 313, 2003. 18

[46] S. Ren, H. He, Girshick R. B., and Sun J. **Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks**. *CoRR*, **abs/1506.01497**, 2015. 21

[47] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. **PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes**. *CoRR*, 2017. 23

[48] Alejandro Newell, Kaiyu Yang, and Jia Deng. **Stacked hourglass networks for human pose estimation**. In *European conference on computer vision*, pages 483–499. Springer, 2016. 23

[49] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. **Gradient Response Maps for Real-Time Detection of Textureless Objects**. *Transactions on Pattern Analysis and Machine Intelligence*, **34**(5):876–888, 2012. 24

[50] Li Sun, Gerardo Aragon-Camarasa, Simon Rogers, and J Paul Siebert. **Robot Vision Architecture for Autonomous Clothes Manipulation**. *CoRR*, **14**(8):1–15, 2015. 27

[51] Dimitra Triantafyllou and Nikos A. Aspragathos. **A Vision System for the Unfolding of Highly Non-Rigid Objects on a Table by One Manipulator**. *Lecture Notes in Computer Science*, pages 509–519, 2011. 27

[52] Pin Chu Yang, Kazuma Sasaki, Kanata Suzuki, Kei Kase, Shigeki Sugano, and Tetsuya Ogata. **Repeatable Folding Task by Humanoid Robot Worker Using Deep Learning**. *IEEE Robotics and Automation Letters*, **2**(2):397–403, 2017. 27

[53] Yinxiao Li, Danfei Xu, Yonghao Yue, Yan Wang, Shih-Fu Chang, Eitan Grinspun, Peter K. Allen, and Abstract—Deformable. **Regrasping and Unfolding of Garments Using Predictive Thin Shell Modeling**. In *International Conference on Robotics and Automation*, 2015. 27

[54] Sasha Targ, Diogo Almeida, and Kevin Lyman. **Resnet in Resnet: Generalizing Residual Architectures**. *CoRR*, **abs/1603.08029**, 2016. 35

[55] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. **PyTorch: An Imperative Style, High-Performance Deep Learning Library**, 2019. 39

[56] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. **Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor**. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018. 51