UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

USING CNNs TO CLASSIFY AND GRASP CLOTH GARMENTS

A degree thesis

Submitted to the Faculty of

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Enric Corona

In partial fulfillment

of the requirements for the degree in

Bachelor's degree in electronic Systems Engineering

Advisors:

Antoni Gabàs

Guillem Alenyà

Ramon Bragós

Barcelona, June 2016

Abstract

Identification and manipulation of deformable objects are currently considered as one of the most challenging tasks in the field of robotics. Their unpredictable shape and pose makes it very difficult to identify them and retrieve their most relevant parts.

The aim of this project is divided in two tasks. First, to recognize a garment between four previously modeled types. And second, to search for suitable grasping points in order to bring the cloth from its initial random position to a known configuration. Both tasks are solved using Convolutional Neural Networks (CNNs) trained with both real and synthetically generated clothing depth images.

We developed a method to detect, after the garment is recognized, two garment-based predefined grasping points. A CNN is used to predict their visibility and position, choosing between rotating or grasping the garment. Once grasped the first, the second point is predicted similarly with a more specialized CNN.

Resum (Catalan)

La manipulació i identificació d'objectes deformables actualment es considera un dels problemes més ambiciosos en l'àmbit de la robòtica. A causa de la seva forma i posició imprevisibles, és molt difícil reconèixer-los i identificar les seves parts més rellevants.

L'objectiu d'aquest projecte es divideix en dues parts. Primer, reconèixer una peça de roba entre quatre models prèviament definits. I segon, buscar punts adients per agafar la roba, per tal de portar-la des d'una posició aleatòria a una configuració coneguda. Ambdues tasques es solucionen mitjançant Xarxes Neuronals Convolucionals (CNNs) entrenades amb imatges de profunditat reals i sintèticament generades.

Hem desenvolupat un procés per detectar, després d'identificar la peça de roba, dos punts prèviament definits per agafar cada peça de roba. Una CNN prediu la visibilitat i posició dels dos punts, per saber si girar la roba o agafar-la. Un cop agafat el primer, el segon punt es prediu de forma semblant amb una CNN més especialitzada.

Resumen (Spanish)

La manipulación e identificación de objetos deformables actualmente se considera uno de los problemas más ambiciosos en el ámbito de la robótica. Debido a su forma y posición imprevisibles, reconocerlos e identificar sus partes más relevantes es muy difícil.

El objetivo de este proyecto se divide en dos partes. Primero, reconocer una prenda de ropa entre cuatro modelos previamente definidos. Y segundo, buscar puntos adecuados para coger la ropa, para traerla desde una posición aleatoria a una configuración conocida. Ambas tareas se solucionan mediante Redes Neuronales Convolucionales (CNNs) entrenadas con imágenes de profundidad reales y sintéticamente generadas.

Hemos desarrollado un proceso para detectar, después de identificar la prenda de ropa, dos puntos previamente definidos para coger cada prenda de ropa. Una CNN predice la visibilidad y posición de los dos puntos, para saber si girar la ropa o cogerla. Una vez cogido el primer punto, el segundo punto se predice de forma parecida con una CNN más especializada.

Acknowledgements

There are many people to whom I would like to thank their support and help during the time I have been working in this project.

First, I wish to thank my tutors. Guillem, to admit me in the laboratory almost without knowing me and always suggest new approaches and ideas I had not considered. To Toni, for teaching and guiding me through all this project. To Ramon, that accepted me as his student and helped me with the bureaucracy of the thesis.

I would like to thank my colleagues in the laboratory and office 19, specially to Sergi and Gerard, who have helped me in many technical problems.

Finally, to the people who have helped me get where I am, my parents, family and friends. Thank you for your constant support and help.

Revision history and approval record

Table 1: Document history.

Revision	Date	Purpose
0	04/06/2016	Document creation
1	25/06/2016	Document revision
2	27/06/2016	Document approval

Table 2: Document distribution list.

Name	E-mail
Enric Corona	ecorona@iri.upc.edu
Antoni Gabàs	toni2332@gmail.com
Guillem Alenyà	galenya@iri.upc.edu

Table 3: Revision and approval history.

Written by:		Reviewed and approved by:	
Date:	05/06/2016	Date:	22/06/2016
Name:	Enric Corona	Name:	Guillem Alenyà
Position:	Project author	Position:	Project supervisor

Contents

List of Figures

List of Tables

1 Introduction

Robots are under increasing expectation to continue gaining autonomy up to participate in our daily lives on topics as diverse as elderly care, housework or maintenance. Their ability to perform repetitive tasks very precisely is essential to any automated process, which combined to their effectiveness makes them the optimal option for industry. The use of robots has been extended to places inherently unsafe for humans, such as space exploration or military tasks. Other types of robots can be found in the medicine field thanks to their precision and reliability.

Nevertheless, the diversity of topics they can be found working on, contrasts with their still low self-sufficiency. Robots can be accurately programmed to achieve a list of tasks, but they become ambiguous when handling new situations. Dealing with deformable objects is part of this unpredictability and involves a significant challenge for robotics, for the necessity to employ different artificial intelligence algorithms. Bringing a garment from an unknown configuration to the known initial position of the following task, such as folding it, presents as a very demanding task that not only requires complex perceptions but also the proper algorithms to interpret the information captured and abstract the relevant part.

Robotics is a highly multidisciplinary area including mechanics, electronics, computer science and physics. In order to solve the mechanical procedure of manipulating garments with two robotic arms, the project described in this document falls in the category of artificial intelligence and computer vision. We assumed the robotic arm has already grasped the garment from a pile of cloth and it is being held in the field of view of a depth camera. The pile of cloth would contain different pieces of cloth, so the garment has to be identified and grasped from a known configuration in order to perform the following task (folding the cloth, dressing a person, etc.). To bring it to this known position, intrinsic for every piece of cloth, the robot should recognize the different parts of the cloth and go to grasp the garment from the familiar point. Figure 1.1 summarizes the whole process with a pair of jeans being grasped by two Barrett WAM arms [3].

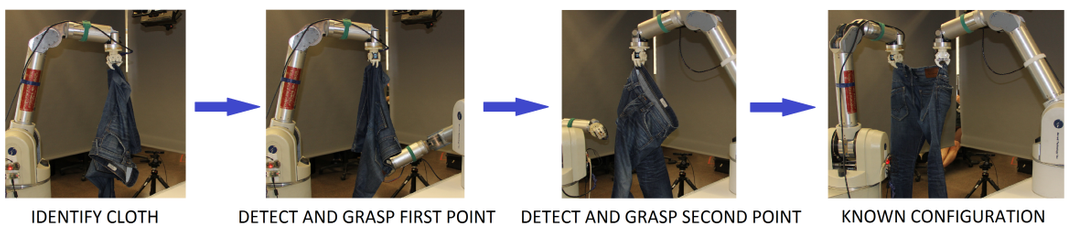


Figure 1.1: Process from grasping garment until we can perform the following task, from the known configuration.

1.1 Objectives, methods and procedures

The objective of this project is to use Convolutional Neural Networks (CNNs) on depth images to identify garments and predict how to bring them to known configurations. CNNs are being increasingly used for visual recognition tasks and their promising results made us consider using them to identify four types of garments: jeans, jumpers, t-shirts and towels. The four categories seem to adapt similar forms and have comparable sizes. Therefore, their appearance in depth images does not vary much.

This poses a challenging identification problem for which we will need many images. Important identification competitions provide up to one million images to train reliable classifiers CNNs. Needless to say, getting enough images to train an algorithm that identifies garments presents as a very time-consuming task. We propose the use of images taken from real garments combined with a synthetic source of data, simulating the clothing being grasped from points on their entire surface.

Differently, the use of convolutional neural networks is not as extended for regression tasks: providing a value in a linear scale that represents typically a physical property. In this case, we expect the algorithm to provide the position in the space of the reference grasping point for the previously identified clothing.

From our garments to be classified, the towel is a special case where the reference grasping points would be at the vertexes, which are easily identified in images: When a towel is grasped and held, the lowest point seen in the images is the vertex as showed by Maitin Shepard et al. [4]. Thereby, our method would only be applied to the other three categories, from which we will use the jeans to initially approach the problem and tune the CNN parameters.

Although the color of the cloth provides an important amount of information, we do not want the algorithm to learn to differentiate between garments based on their color, as garments can be found on almost any tint. Consequently, the recognition algorithm will be trained only with depth images. To capture them we are going to use a Xtion depth camera [5], which will be located at a fixed distance from the garment holding point, to be decided depending on a balance between wrinkle definition and amount of space observed.

1.2 Work plan, milestones and Gantt diagram

As explained in section 1.3, the work plan was modified in the critical review to add a second predictor to the regression section. The final Work Breakdown Diagram can be seen in figure 1.2. It consists of three basic modules from which the first is necessary to gather data to train convolutional networks for the second and third steps. The first and second sections also include the theoretical study to the physics engine software [1] and CNNs respectively.

Also, table 1.1 and figure 1.3 present the project milestones and the Gantt Diagram, respectively.

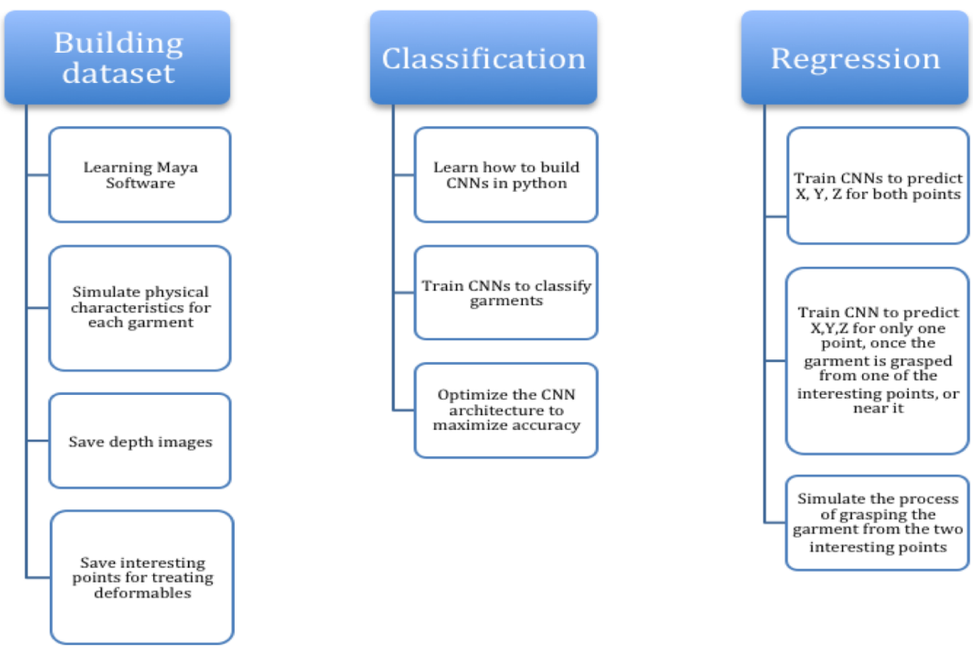


Figure 1.2: Work Breakdown Diagram.

1.3 Deviations from the initial plan and incidences

This subsection focuses on precise parts of the project and it should be better understood reading it afterwards.

As explained in section 5.1, we initially did not thought of dividing the grasping process in two parts. It was when we trained the first network that we realize that, once we had grasped the first point, we would need to predict a second. We could possibly make use of the first

Table 1.1: Milestones.

WP#	Task	Short title	Milestone/Deliverable	Date (Week)
WP1	T1	Generate synthetic data for classification	Synthetic images are similar to real ones generated	15/03/2016
WP2	T2	Accuracy testing	The final network is validated in the real test dataset	12/04/2016
WP1	T4	Generate synthetic data including the position of their grasp points	Synthetic images generated to train the regression CNN	22/04/2016
WP3	T1	Regression CNN to predict the position of two grasping points	The final network is validated in the synthetic test dataset	05/05/2016
WP3	T2	Regression CNN to predict the position of the second grasping point	The final network is validated in the synthetic test dataset	22/05/2016
WP3	T3	Simulate process of grasping point	Simulation from the garment being identified and brought to a known configuration	05/06/2016

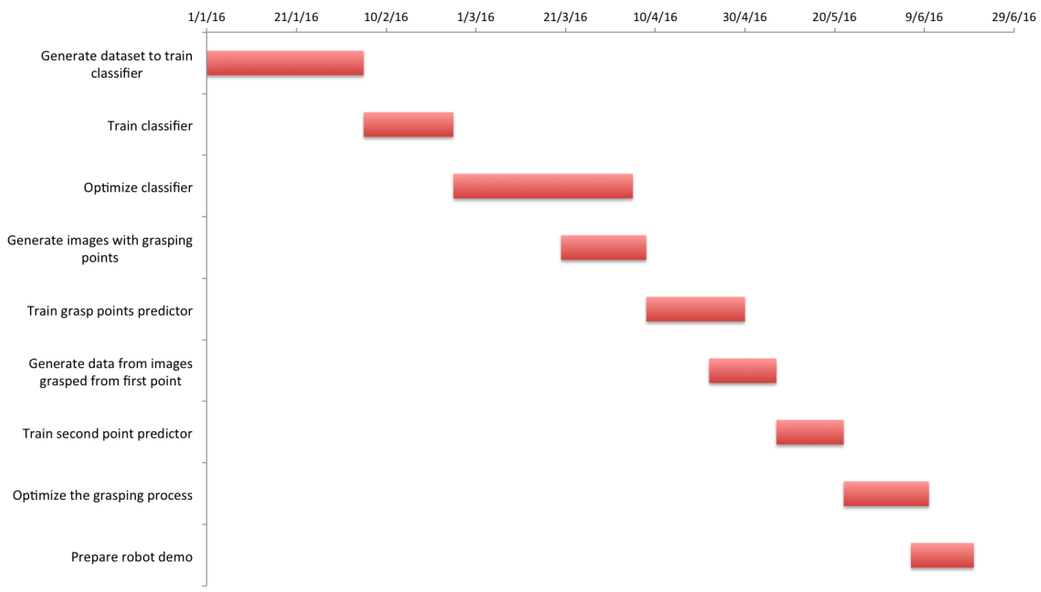


Figure 1.3: Gantt Diagram.

network also to predict the second point, but adapting to the case of the first point being already grasped (or grasping somewhere near it), a specialized network directly to face this situation proved to be considerably more efficient.

Also, predicting whether the point was visible or not and its position was not initially planned. We expected to predict their situation and, then, try to find the point in the image to decide if it was visible or we should turn the garment. When analyzing the results of the first networks, we realized of what our final approach should be.

2 State of the art

This chapter reviews the most recent research related to the topics treated in this project, including garment identification, bringing pieces of cloth to known configurations and Convolutional Neural Networks.

2.1 Garment identification

A significant amount of research has been focused in detecting and identifying deformable objects. The process of isolating a piece of cloth from a pile was identified as the one of the first tasks to be solved for laundry manipulation in a pioneer work by Kaneko and Kakikura [6, 7]. Their initial work identified three categories: shirt, pants, and towels. Previously to identification, works such as [8] by Colome et al. study the grasping process from a pile of clothing, implementing a method to know when we have grasped only one garment.

Latest works commonly consist of an image or pointcloud database taken by a kinect sensor or a pair of stereo cameras. The main difference resides in the identification method. Willimon et al. [9, 10] used interactive perception to recognize and classify different small garment types such as socks and short pants, but their model relies on a color-based image segmentation that may fail when seeing fully textured clothes. Kita et al. [11, 12] demonstrate the ability to recognize poses of different garments by matching them to a precomputed database. Li et al. [13] use a SIFT descriptor which only needs an input image and, therefore, they achieve a high-speed recognition of clothing: sweater 85%, jeans 70%, and shorts 90%.

Another common approach is based on comparing volumetric features, such as Li et al. [14] proposing to reconstruct a 3D model by using diverse images of the garment. They extract volumetric features and match them to an offline database. However, their method requires having the garment completely rotated, which slows the recognition process. Similarly, KinectFusion on different deformable objects [15], directly compare the 3D model to the pre-recorded database, which also requires powerful computational resources.

One approach using convolutional networks is presented by Lao et al. [16], that use them on color images downloaded from the Internet and achieve an accuracy of 74.5% in identifying sixteen classes of clothing. More similarly to us, Mariolis et al. [17] used CNNs on depth images to obtain an accuracy rate of 89.38 % in identification of shirts, pants and towels. In the first part of our work we presented an approach to identify clothing using only real images on convolutional networks that takes advantage of a robotic arm to rotate the cloth to have more views, which was accepted in Conference on Articulated Motion and Deformable Objects 2016 [18].

2.2 Bringing clothes to known configurations

Osawa et al. [19], first proposed two robotic arms to unfold a garment. Their method consisted of color-based segmentation with a clean background and, therefore, only works for garments with unique color. Maitin-Shepard et al. [4] propose to detect the corners of a piece of cloth using geometric cues, consisting of a sequence of grasps that end having the garment always in the same position. Their approach works for towels, whose known configuration is being grasped from two vertices. They present that the lowest point of a towel, after some re-grasps, is always one of the vertexes. This allows to fold previously unseen towels without the necessity to compute the database. Assuming this as a valid hypothesis, our project does not centre in predicting where the grasp points are for this garment. Ramisa et al. [20] present a system that uses a very efficient shape descriptor named FINDDD combining depth and color to find good grasp candidates on a cloth lying in the table. Therefore, they cannot disambiguate with more than one view nor choose known grasping points.

Other approaches to manipulate garments involve some kind of artificial intelligence algorithm to decide where the point is. Doumanoglou et al. [21] proposes the first method to unfold regular-size clothing with a similar method to ours. They use random forests on the garments' depth images, trained with a set of images manually taken and labelled, which may be a very

time-consuming task method to implement in practice. Also, it may lead to worst results when seeing new fabrics, although achieves impressive rate of success within their database.

Similarly to us, Li et al. [22] makes use of a physics engine to create a database to train an algorithm that recognizes the garments' pose, matches it to a pose achieved with the synthetic 3D model of the same garment, and relates the grasping point of the synthetic image to the real one. After some re-grasps, the process stops when the pose is matched to the known state.

2.3 CNNs

Convolutional neural networks have a long history in computer vision tasks, having examples of successful back-propagation training for digit recognition [23] in a few decades ago.

But its due to the increasing computational power of GPUs, that the number of applications based on CNNs is incrementing, leading to recent successes of deep networks for image classification, that have achieved competition-winning numbers on contests involving datasets of more than one million images, particularly the network proposed by Krizhevsky et al. [24] and GoogLeNet [25] in ImageNet competition [26].

In recent years, the evolution of new techniques to train not only CNNs but also normal artificial networks, such as dropout [27] and batch normalization [28] have boosted Convolutional Neural Networks in a very diverse number of applications that may be related to image processing.

3 Convolutional networks theoretical background

Artificial neural networks are a set of mathematical models inspired in humans' central nervous systems. Each neuron aggregates its inputs and passes them to a nonlinear activation that is fed to the next step. Neural network structures are usually formed by an ensemble of layers (Figure 3.1) that reduce the incoming dimensionality at the same time as they process nonlinearly the input values, which makes them able to approximate more complex functions. Nonlinearities are usually either rectified linear units (ReLUs) [29] or sigmoids [30]. ReLUs derivatives are easier to compute than sigmoids', resulting in networks having them train faster.

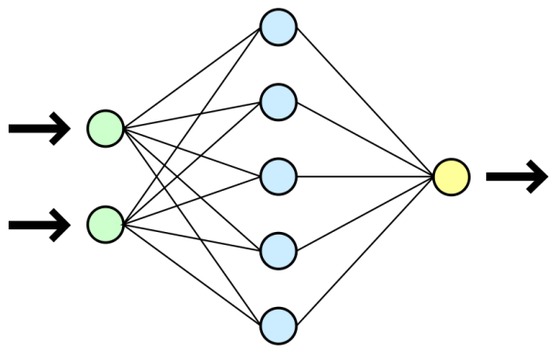


Figure 3.1: Artificial network structure.

The weights on each layer are tuned based on a training process, where all the available data is shown to the network to generate a certain loss function.

Backpropagation [31] consists of minimizing this loss function step by step by changing the parameters of every network, where the size of each step is determined by the learning rate. If it is bigger the function will be learnt more rapidly, although having it too large can make the process diverge.

When training, there are different methods that improve the results depending on the application. The ones used in this project were Adam [32] and stochastic gradient descent with Nesterov momentum [33]. The first adapts by itself the learning rate depending on the last range of losses, making it very fast to train. The second trains normally but conserving a history of weight modifications that helps to avoid local minimums.

Convolutional Neural Networks (CNNs) are a specific type of neural networks that are giving very favorable results in computer vision tasks. The input image is processed by convolutional layers that help the following layers to have a better representation of what structures appeared on the initial image. The layers used on this project are:

- Convolutional: The layers in charge of processing the image divided in portions and give a better understanding of the input image. They are attached to nonlinear activations applied usually after the convolution, adding complexity to the network.
- Pooling: After convolutional layers, it is usual to find a pooling layer that reduces the dimensionality of the image resizing the image. This can be done finding the maximum value or the average of the numbers to be pooled. Pooling layers help to produce invariance on the position of the image, but difficult the following layers to understand what portion of the image values come from.
- Normalization [28]: Layers that normalize their inputs have been proved to increase the training speed of the algorithm, although gives worst results when the network is given different types of images from the training ones.
- Fully connected: These are formed by sets of simple neurons and process the data provided by the convolutional and pooling layers. Usually, one or two fully connected layer combine all the features found previously and pass them to the last layer, which is either a softmax or a linear layer.

- Softmax [34]: The softmax layer is used to classify between different possibilities. It returns the probability for the subject to pertain in each class.
- Linear: Linear units are simple neurons without the nonlinear activation, where the amount of neurons depends on the number of outputs we want to have. Instead of classifying, they are used to predict a value related to the input data, i.e. the number of faces in an image.

Training CNNs requires a big amount of data, which usually consists of images, involving also an important amount of RAM memory and disk space. Not only that, a Graphics Processing Unit (GPU) makes the process much more efficient and helps to reduce dramatically the training time.

Ground truth is an important term for this project, meaning the correct results labelled to the corresponding image. It refers to the correct values of the dataset and are being used to train the CNNs and calculate the results of the networks.

4 Cloth identification

This section explains the process of training a garment recognizer, which includes taking photos of real images, the generation of synthetic images, developing different Convolutional Neural Networks and fine-tunning them to optimize the network and get the maximum accuracy. Figure 4.1 shows the position of this part in the whole sequence.

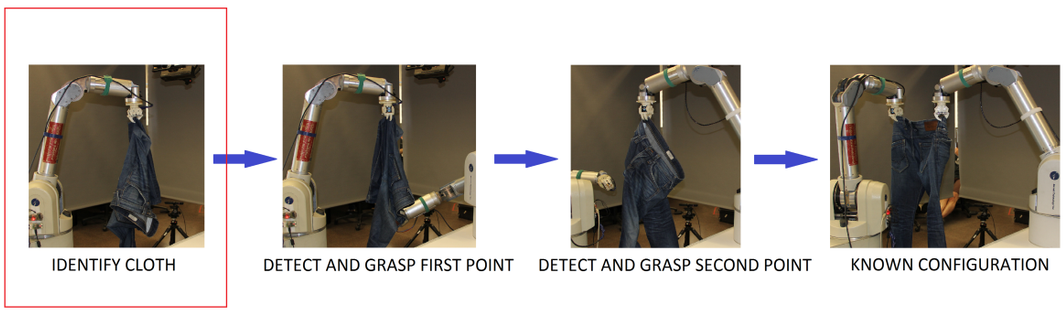
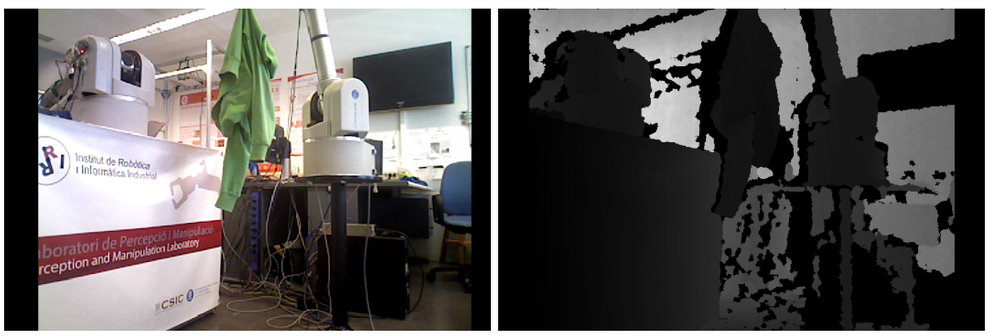


Figure 4.1: Goal of the current section in the project's framework.

4.1 Approach

After trying different distances from the camera to the garment, we set the position at 1.5 meters. Although placing the camera further involves the image to be noisier and miss small wrinkles, the image usually enclosed whole views of the clothing and poses became identifiable. Figure 4.2 shows the color and depth appearance of our setup to obtain depth images and later grasp the garments, in this case a jumper, using the two robotic arms [3]. As seen in the depth image, the garment is only part of the scene and can be filtered by depth, knowing the rest of the objects are further.

Figure 4.2: Our setup to capture depth images of the garment, in this case a jumper, involving a Xtion camera [5] and two robotic arms [3].



(a) Color image of our setup. **(b)** Depth image of our setup.

We obtained 2530 depth images of the garments we want to classify, of which some of them are repeated and the only difference is the camera noise. As our camera is located at a fixed distance from the robot, the images are captured only from a known distance and centered in the image in a fixed orientation. This simplifies the training but does not allow much data augmentation. The only way we could obtain more images from the original ones was to compute the X-axis symmetry, which doubled the number of images.

The resulting data was divided into train (60 %), to teach the CNN, validation (20 %) and test (20 %) sets. The validation set helped to determine what changes in the network structure and parameters help to improve the classification accuracy, while the test images are those the CNN has never tested until the final model is chosen, to verify the network performance. As there are very similar images in the set, the validation and test sets contain different garments than the training, to confirm our model learns to identify these garments.

At the end, we did not have much real images and the first networks would not lead to a correct approximation of the function but mostly retain in memory those training images, reaching an accuracy of approximately 87%.

4.2 Gathering data

The real data we dispose of is not enough to train an enough reliable network. Consequently, we decided to simulate synthetic piece of clothing in a physics engine [1], using 3D models of the garments to be identified in this project. This section will explain how we obtained the depth images using this Software.

4.2.1 Physics engine environment

The depth images were captured from 36 cameras at a distance of 1.5 meters, covering the whole visual range at that distance. The pose of the original downloaded garments is initially as if they were not deformable. To capture images on different poses, a vector of vertexes was defined on each garment and its being iterated to simulate the cloth is grasped from each of those points. This vector contains vertexes of only one side of the garment, as the images are being doubled by computing the X-axis symmetry, and the resulting images would be the same than computing the same vertexes on the other side. Figure 4.3 shows the interface in the physics simulator when capturing images.

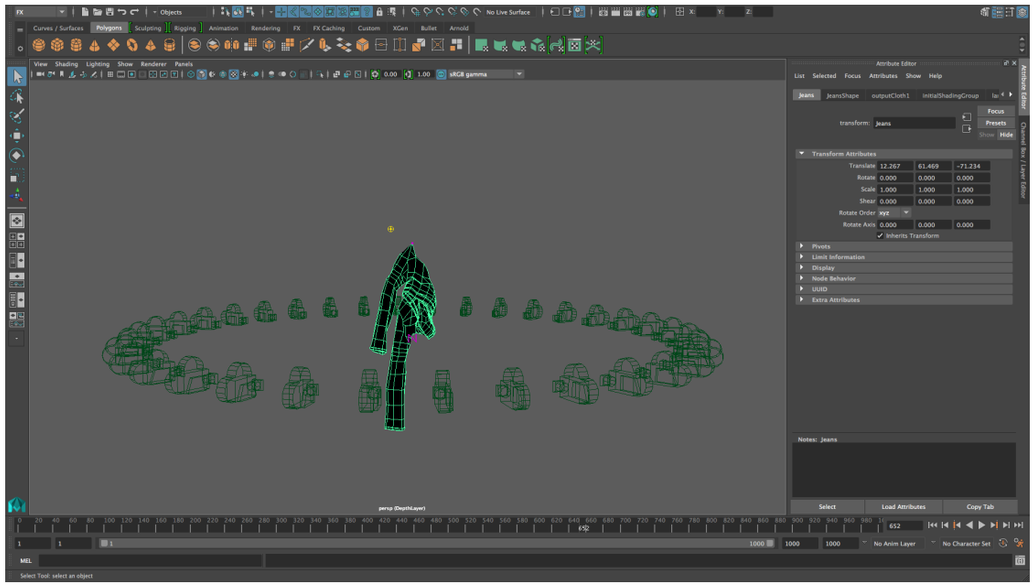


Figure 4.3: Physics engine interface when capturing depth images of jeans. Cameras are located in a fixed distance to capture different views of the same pose.

The physics engine has a programmable environment in Maya Embedded Language (MEL) [35] or python. We programmed diverse scripts in MEL, easier to communicate to the physics engine classes, that let us automatize the image capturing process. The models we used to obtain the synthetic images were open source 3D cloth in the of jeans, jumpers and T-shirts, whereas we created the towels as simple plane deformable objects of diverse sizes and widths. In order to create the cameras and initialize our working environment for the jeans cloth, we used the following MEL code. To model synthetic cameras with the characteristics of the real cameras we are using, their focal length and image size were specifically defined. Also, the pictures were saved in TIF format of 16 bits to minimize the quantification error.

```
string $cloth = "Jeans";
file -import "/Users/ecorona/Downloads/jeans.obj";
setAttr "defaultRenderGlobals.imageFormat" 4;

// Select Jeans and convert to deformable with standard characteristics:
select -r $cloth ;
createNCloth 0;
setAttr "nClothShape1.stretchResistance" 1;
```

```
setAttr "nClothShape1.compressionResistance" 0.1;
setAttr "nClothShape1.bendResistance" 0.025;

// Create cameras
for( $i=0; $i<36; ++$i){
    camera -focalLength 33.152;
    float $x = 150*cos (($i*10)*3.14159/180);
    float $z = 150*sin (($i*10)*3.14159/180);
    move -r $x 0 $z;
    float $rotation = (90 - $i*10);
    rotate -r -os -fo 0 $rotation 0;
    string $instruction = "cameraShape"+($i+1) + ".mask";
    setAttr $instruction 0;
    string $instruction = "cameraShape"+($i+1) + ".locatorScale";
    setAttr $instruction 10;
}
```

Listing 4.1: MEL code to load jeans 3D model, set our environment with the deformable properties and cameras.

As observed, MEL is syntactically similar to Perl and, although it was a new language for us, still provides two main advantages respect to python. First, the physics engine offers a built-in command echo server, which is created for MEL, and lets execute lines without having to create scripts. And second, most of inner working of the physics engine is formed by MEL commands, which provides many insights into undocumented features.

4.2.2 Capturing images

To simulate the garment being grasped, the position of a vertex is moved to the center and fixed. Then, the garment is let only affected by gravity. When the software simulates the cloth falling, it performs several computations in order to avoid physical incongruities or collisions, needing a considerable time to achieve realistic results.

Before gathering enough images to train the CNNs, we tried the synthetic images to appear very similar to reality, despite of the camera noise. By varying cloth dynamic properties such as compression, stretch and bend resistance, we modelled different types of cloth fabrics. The following MEL code needs approximately three hours to generate approximately 3.000 jeans depth images. It iterates between all the vertexes, moving each of them to the central position to be at a fixed distance from all cameras, passes trough time and takes the images. This process was repeated varying the size and physical properties of the garments until we had roughly twenty thousand images for piece of cloth.

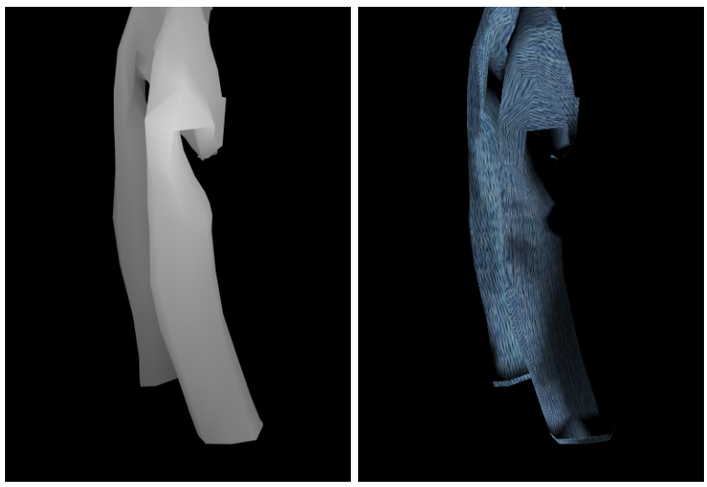
```
// Start the captures:
for( $x = 0; $x < size($vertics_Jeans); $x = $x+1){

// Move grasping point to the centre:
    currentTime 1;
    string $instruction = $cloth+".vtx[" + $vertics_Jeans[$x] + "]";
    float $pos[] = `pointPosition $instruction`;
    select -r $cloth;
    move -r (-1*$pos[0]) (60-1*$pos[1]) (-1*$pos[2]) ;
    select -r $instruction;

// Fix the point position
    createNConstraint transform 0;

// Pass trough time one by one
    for( $t=2; $t<500; ++$t){
        currentTime $t ;
    }

// Iterate through cameras to take photos:
    for( $c=1; $c<=36; ++$c){
        string $cam = "camera" + $c;
        string $final_name = "/Users/ecorona/Documents/Database/" + $cloth2 +
            "_" + $x + "_" + $c + ".tif";
        $direction = `render  $cam`;
        sysFile -move $final_name $direction;
                }
```

```
// Unfix grasping point:
    select -r dynamicConstraint1;
    doDelete;
}
```

Listing 4.2: MEL code used to iterate over the selected vertexes in Jeans and simulate them falling. Then, the cameras obtain their depth image.

Figure 4.4 shows one of the depth images obtained and compared to a color image where the jeans texture can be appreciated. There is nothing in the image apart from the garment as we want the convolutional network to focus on it. The points containing no depth were set to zero. Therefore, the real images obtained have to be set to the same configuration by filtering the depth further than the position of the jeans.

Figure 4.4: Synthetic color 4.4b and depth 4.4a images of jeans obtained with a physics engine [1].



(a) Depth image of jeans. **(b)** Color image of jeans.

4.3 Network description

We trained a series of CNNs varying the most important details of the network such as the amount of convolutional, pooling and fully connected layers, varying the size and amount of the filters, etc. This showed what features were the most important and could improve the network results.

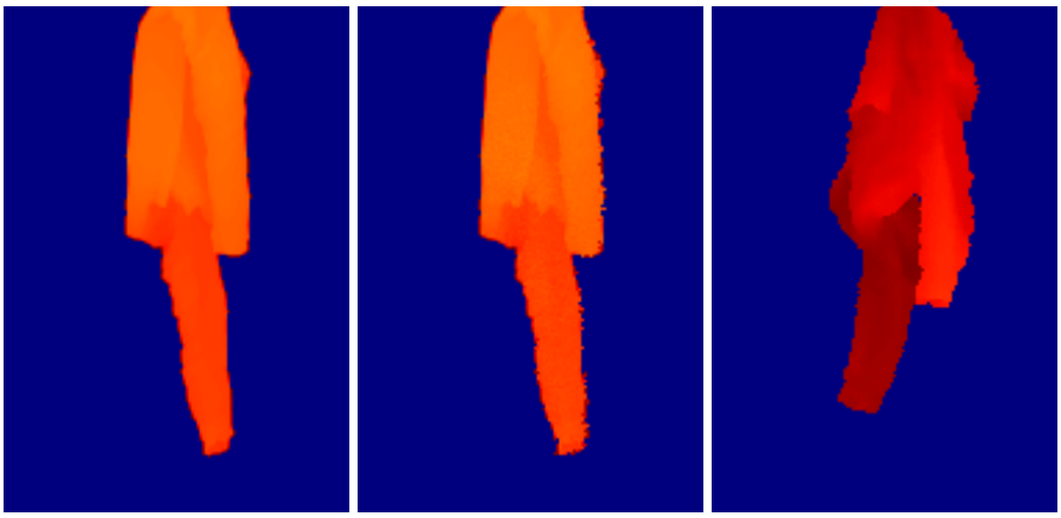
One of the first techniques we applied was to add dropout [27] to the fully connected layers, which consists of cancelling some nodes in the network with a certain probability. This helps the network not rely only on a few features, but try to extract additional properties in different ways. Although its use slows the training process, the accuracy of networks that use dropout increases considerably. Dropout is sometimes being used on the convolutional layers, but its effectiveness is still controversial [36]. Some tests were performed in this direction but the dropout in these layers did not improve the results.

Second, penalizing the parameters when they become too big is habitually used to avoid the filters from learning too complex features, usually known as regularization [37]. In addition, data normalization is a common practice to increase the training speed and accuracy of the algorithm. On the first approach, images were pixel-wise normalized by setting the mean to zero and the standard deviation values to 1. We also tried Batch Normalization [28], which consists of normalizing every set of images showed to the network instead of normalizing the entire set of images previously, showing very good performances and faster training for our case.

Both the real and synthetic images have a shape of 240 pixels height and 320 pixels width, but the garments are always in the middle. Accordingly, the sides were cropped to have a size of 240 height x 160 width which still contains all the images we have. This allows the network to train and predict faster and also reduces considerably the amount of disk space needed to save the images.

We noticed the synthetic images taken from the simulator have very precise information and the borders of the garment appear very definite. Instead, images from real cameras return different types of error. First, the uncertainty on the depth captured can be characterized as Gaussian noise correlated to the distance to each point and the pixel position. Also, considerable horizontal and vertical error appears, though it is only visible on object borders, where the depth from a pixel to its neighbors is most different. Choo et al. [2] presented a mathematical model to simulate this noise in images, which has given us our best results. After adding this noise in our synthetic training images, our model has learnt not to rely much on the borders and take depth noise into account. Figure 4.5 shows the appearance of the noise in synthetic images..

Figure 4.5: From left to right: Synthetic image with no noise, synthetic image with added kinect noise [2] and depth image taken from real jumper in a similar configuration.



The loss function used to train the network was based on categorical cross entropy, with added regularization [37].

$$Loss(p,t) = \sum(-t * log(p) - (1-t) * log(1-p)) + Regularization \tag{4.1}$$

The training process is optimized to minimize the loss function, but does not provide us enough information to evaluate the network reliability. Alternatively, we are using the accuracy to know which model provides best results.

The ReLU function [29] proved to be faster and give more accurate results than the sigmoid [30]. Table 4.1 presents the networks with best accuracy. They have common features, such as fifty per cent of dropout [27] and are regularized [37] (see section 3), but the structure is different. More details of these networks and their performance compared to the other networks trained can be seen in Appendix C. They showed what features make CNNs approach more accurately this problem. The networks trained with synthetic images giving best results were refined using the training dataset formed by real depth images, that contains 3039 images. Their results are presented in the last two columns of figure 4.1.

Table 4.1: Table of trained networks using synthetic images.

Training details			Train results		After refinement	
Training images	Added Noise	Normalization	Loss	Accuracy	Loss	Accuracy
Real	No	Pixelwise norm.	0.382	87.08 %	-	-
Synthetic	No	Batch norm.	0.381	86.48 %	0.138	95.28 %
Synthetic	No	Pixelwise norm.	0.623	82.87 %	0.201	94.26 %
Synthetic	Gaussian noise	Pixelwise norm.	0.715	72.87 %	-	-
Synthetic	**Kinect noise in x,y and depth [2]**	**Batch norm.**	**0.354**	**88.61 %**	**0.125**	**96.85 %**

The best model reaches an accuracy of almost 97 % of accuracy in real images. As seen in image 4.6, its structure is based on four convolutional layers with different size for the horizontal and the vertical axis, as the input image is not squared. After the convolutional

layers, four max-pooling layers to resize the image and increase the network invariance respect to the position of the garment features. Next to them, two fully connected layers connect to the last two softmax [34] neurons.

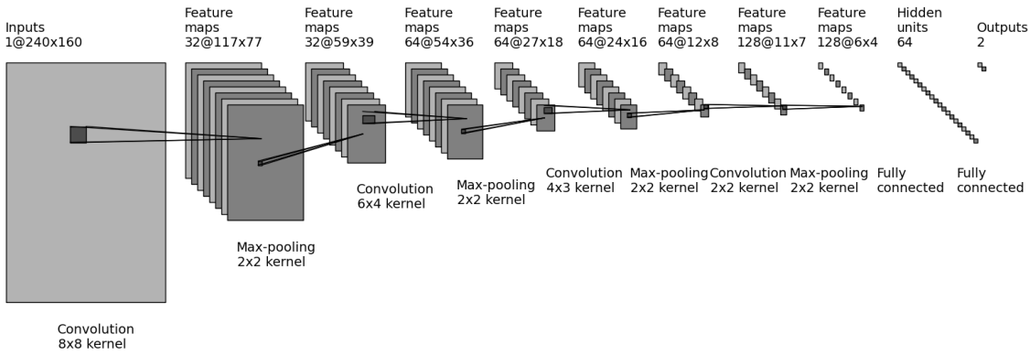


Figure 4.6: Network structure. The final model has four convolutional layers with non-squared size, four max-pooling layers and two fully connected layers.

4.4 Results

The final accuracy and loss during training are shown in figure 4.7, first using synthetic depth images and validated only by real images. When the accuracy stabilized in the first training, approximately at the 100 epochs, the network was refined using the real training images with a smaller learning rate and a higher regularization. As the training dataset is changed to a new one, the training loss has a new peak. On the refinement, the training data is much smaller and epochs pass much more rapidly. The global loss minimum is achieved in the 284 epoch and, after that, the network starts to slightly overfit the training images. The validation curve is not being reduced but the training loss is decreasing even though it has added difficulties such as dropout and regularization.

The confusion matrix at figure 4.8 shows accuracy and missclassification rates on the four garments included in the project. T-shirts appear to be very different to the other three, being correctly identified in the nearly two hundred T-shirt depth images we dispose of. Differently, the other three may have similar poses that makes the network mistake. Most significantly, the towel is the garment causing more confusion, most often confounded with jeans.

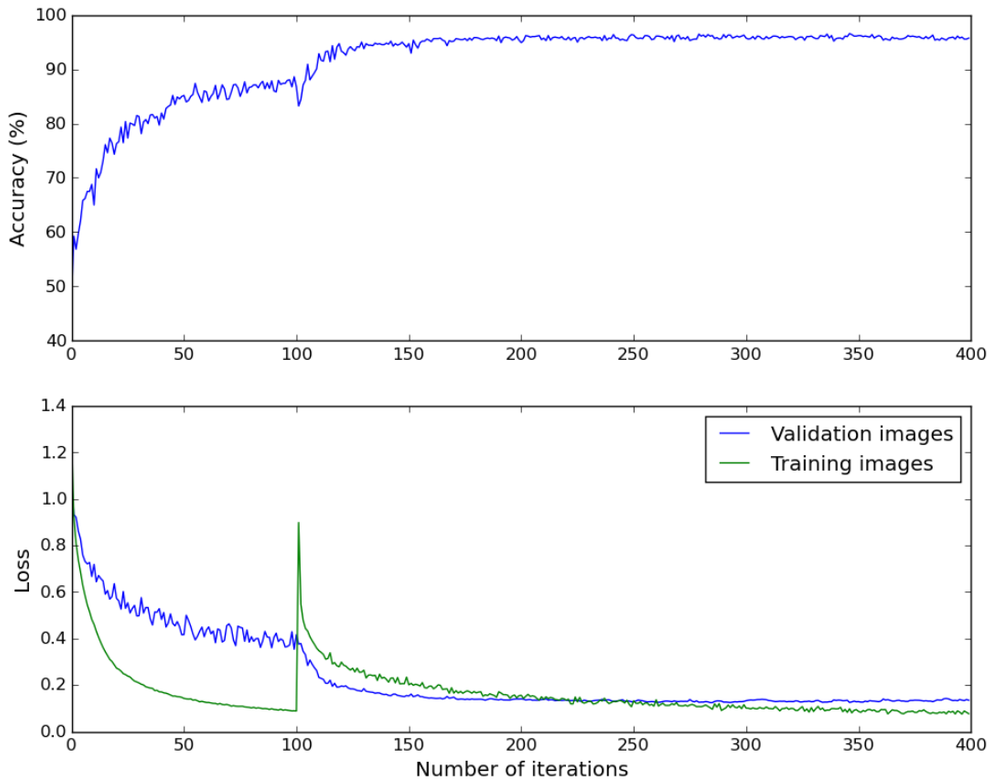
Figure 4.9 presents correct predictions and mistakes committed by the network. As seen in the second row, different garments can have very similar appearance in depth images. Among the probabilities most regularly predicted, one of the options stands out the others. Nevertheless, when the image does not contain characteristic features from the different garments, such as the sleeves, the hoods on jumpers or the necks on the T-shirts; the probabilities predicted are more balanced and imply more indecision.

To avoid being in these situations and maximize the identification confidence, the garment can be rotated and identify the garment for different views of the same pose. Then, the prediction would be the most voted output of the convolutional network. This way, we ensure the distinctive parts of the garment have been viewed.

In other occasions, the network may fail predicting a high probability to a mistaken garment due to the similarity of the poses adapted, as in the bottom row of figure 4.9. The long sleeves of the jumper are very similar to the jeans legs, which is a common source of error to the network. The confusion between the jeans and the towel could be due to the similarity of towel views with some poses of the jeans having the legs together.

Comparing to other cloth identification research projects in figure 4.2, we achieve a similar accuracy to the state of the art when training our convolutional network only with depth images from real garments. In addition, our final model being trained with synthetic images and refined with real ones, obtains a considerable improvement to other recent approaches in research. Although this topic has been studied taking into account different number of clothing, which difficulties the comparison. Furthermore, we and Mariolis et al. [17] have only

Figure 4.7: Accuracy and loss evolution during training, from epoch 0 to 100. Refinement starts at epoch 100, approximately, and generates a training loss peak, due to the change in training images.



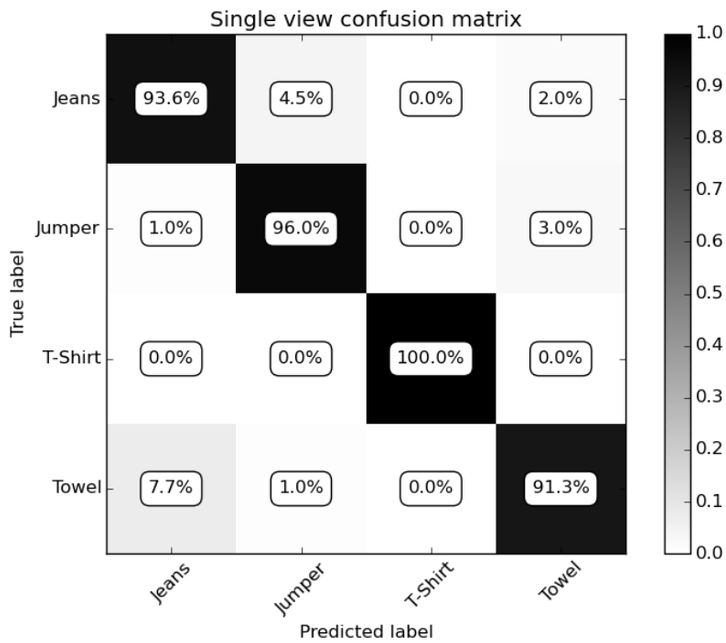
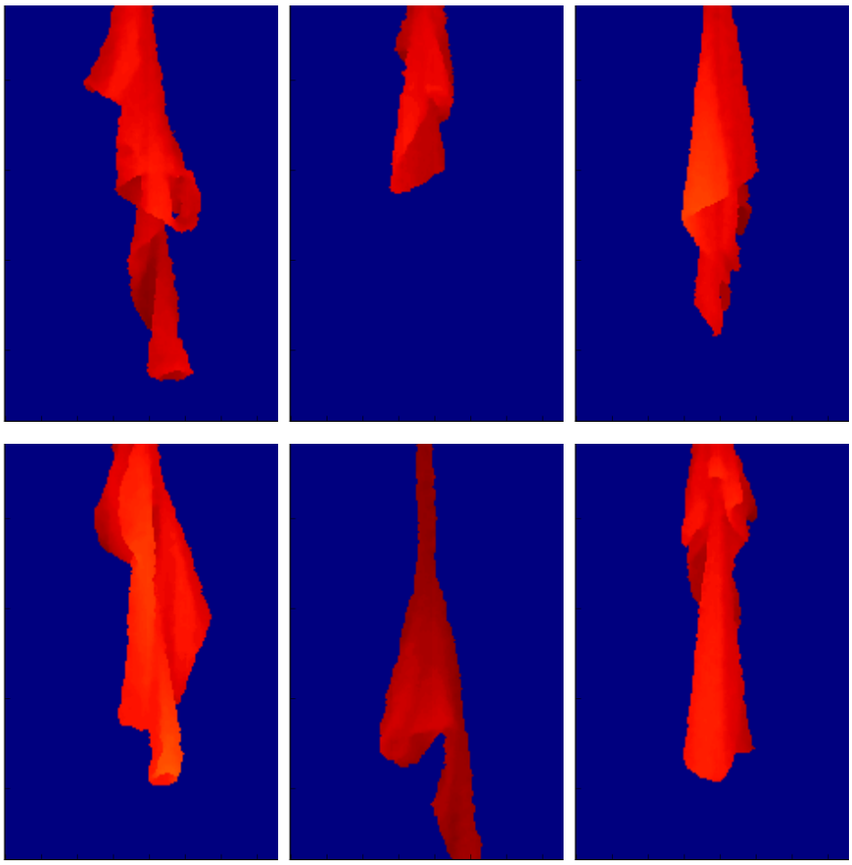


Figure 4.8: Confusion matrix. The diagonal shows correctly identification rates per each type of garment. The most relevant missclassification occurs between jeans and towels.

Figure 4.9: First row: Images of jeans, T-shirt and towel correctly identified, respectively. Second row: The same garments in misleading poses that made the classifier mistake.



considered a fixed position for the garment while other approaches such as Li et al. [13] have different views in other angles maintaining a fixed distance. Also, Lao et al. [16]) detect an ample amount of garments in color images, more oriented to fashion rather than robotics.

Table 4.2: Results of cloth identification on this project compared to state of the art.

Approach	Number of garments	Accuracy
Li et al. [13]	3	81.67 %
Lao et al. [16]	16	74.5 %
Mariolis et al. [17]	3	89.38 %
Our approach only with real images	4	88.61 %
Our approach with synthetic and real images	**4**	**96.85 %**

5 Bringing cloth to a known configuration

This section will describe how we approached the process of having an identified garment in an unknown pose until it is grasped in a familiar configuration by the robot. As figure 5.1 presents, the current chapter will try to solve the problem of locating the grasp points in the Cartesian space to provide them to two robotic arms that will grasp them in order until it is being held in a well-known pose.

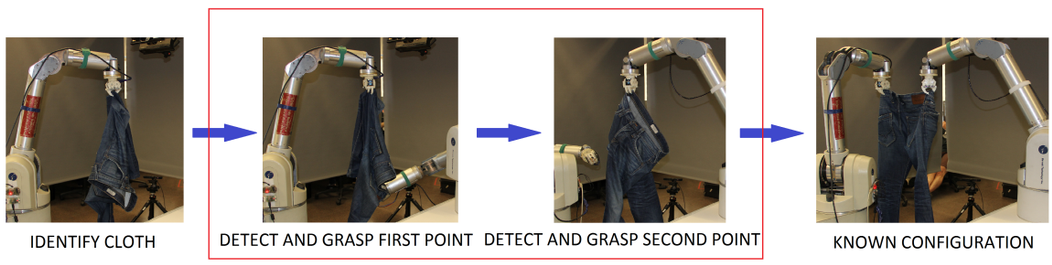


Figure 5.1: Goal of the current section in the project's framework.

5.1 Approach

When hanging garments, we naturally manipulate the garment in our hands until we grasp the cloth from these positions, having always a good sense of where they are. Differently, robots cannot handle these types of objects this way. Their grippers do not let them to slightly modify their position when handling a garment, as we can do. Instead, robots should grasp one point first and then the second, taking into account that there may not be any grasping point on the image: They may be on the other side of the garment or, in the case of jeans, they are long enough to have the grasping point below the image.

Having our image identified as presented in figure 5.1, the grasp points may be visible or we may have to rotate the garment to see them. To know when they are visible, we would need a specialized network. After this network finds a visible grasp point, a regression CNN would predict the Cartesian location of the point. Following with the grasping pipeline, having the first point grasped we would need to identify the second grasping point in a new image using the same method.

Training two networks to classify and to predict the location of the points, respectively, would involve more complexity for each of the garments. At the end, we would need four different Convolutional Networks for each cloth type. After trying different types of networks, we decided to classify and apply regression in the same network by modifying their last layer, which would increment the complexity in the training process but ease the implementation, reduce the memory used and increase the prediction speed. For each garment considered we end having two Convolutional Networks. The first would predict the visibility and location of the two known grasping points, while the second would estimate the same properties of only the non-grasped point.

5.2 Gathering data

Due to the clear difficulties of gathering enough real garment depth images linked to a grasping position so that a Convolutional Network can be trained, this part was approached similarly than in section 4. A new big amount of images from synthetic garments, referenced to the known point positions, were generated and the resulting CNNs were trained with them.

Considering the scripts presented in section 4.2, we had to add modifications in order to locate the grasping point at each pose and extract these values linked to the image, in a CSV file.

5.2.1 First point detection

Figure 5.2 shows the desired configuration for jeans. Their known grasp points are at the waist, over the sleeves. For jumpers and T-shirts, the familiar holding points were set over their shoulders.

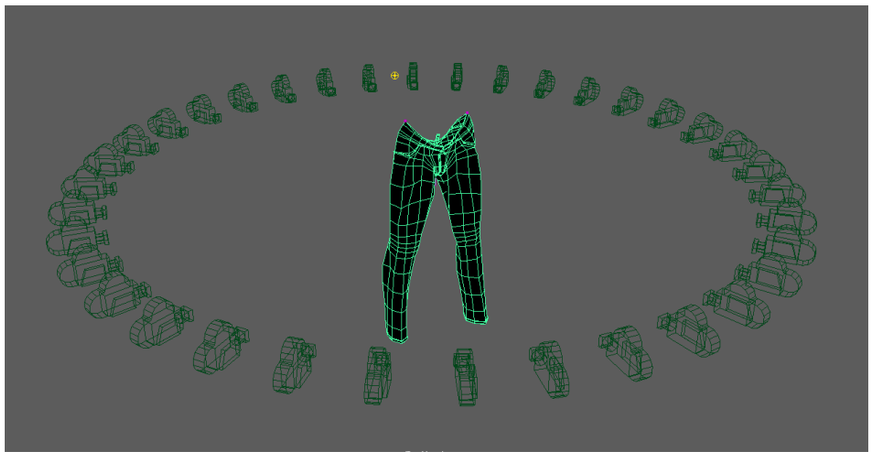


Figure 5.2: Known configuration for jeans in the simulated environment.

To train a reliable network independently of the position of the position of the objects involved, we set the Cartesian grasp point locations referenced to the camera. The origin was placed in the garment holding point, which is 60 centimeters higher than the camera and at 150 centimetres in front of it.

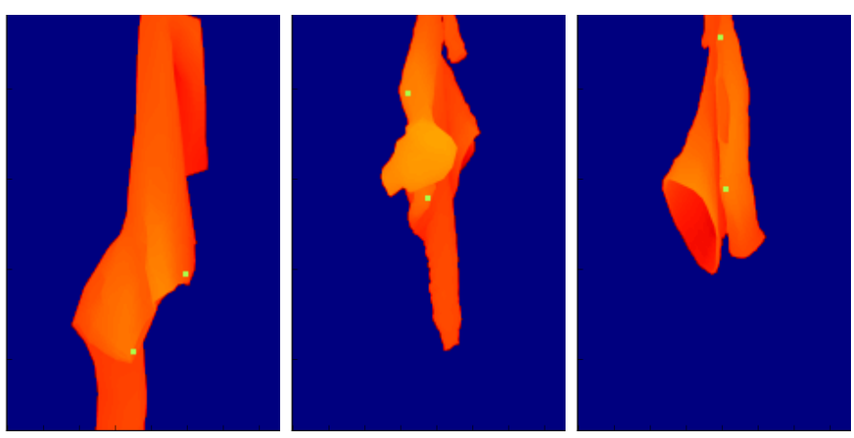
Nevertheless, the point positions are more human-comprehensible once seen in images. Knowing the field of view of the camera, 57 degrees horizontally and 42.75 vertically, and the number of vertical and horizontal pixels, 320 and 240 respectively, the grasping positions can be drawn in images. The axis orientations are defined as in the Kinect camera [38]: The X-axis matches the horizontal axis, where the positive direction is leftward. The Y-axis corresponds to the vertical axis upward and the Z-axis represents outgoing depth. The equations obtaining the pixel values are 5.1 and 5.2.

$$pixel_x = 79 - \frac{x}{(150+z) * tan(57.0/2)/160} \tag{5.1}$$

$$pixel_y = 119 + \frac{-60-y}{(150+z) * tan(42.75/2)/120} \tag{5.2}$$

Although the pixel depth does not exactly correspond to the Cartesian Z value, due to quantification error and the conversion from a decimal value of pixel to an integer, the visibility of the point can be detected when these two values are nearer than a threshold, having computed our ground truth with a limit of 1 centimeter. Otherwise, the grasp point is supposed to appear behind the cloth. Figure 5.3 shows clothes with their labelled visible grasping points.

Figure 5.3: Synthetic ground truth for localizing the first grasp point, in white circles. From left to right: Jeans, jumper and T-shirt.



5.2.2 Second point detection

Once the first point is grasped, the following CNN has to predict where the second grasping point is located. Thus, the images shown to this network have to be garments grasped from reference points or near them. Figure 5.4 shows the possible grasping points we labelled for jeans to be grasped, from which the MEL scripts iterated over them to provide approximately 60 thousand depth images more per garment. In this case, we did not compute the X-axis symmetry to double the number of images, as the selected vertexes were all around the interesting points and we would have repeated images. Apart from this zone, other points around the garment were selected, to transmit the CNN an intuition that the clothing can adapt a great variety of poses, which could help us when the previous point had not been correctly localized.

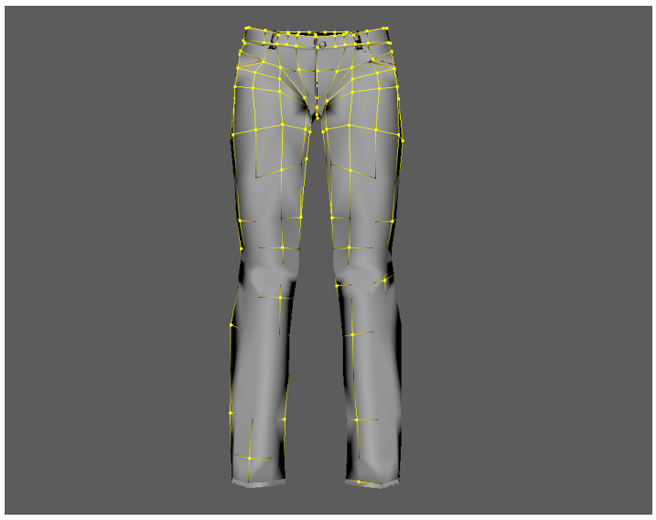
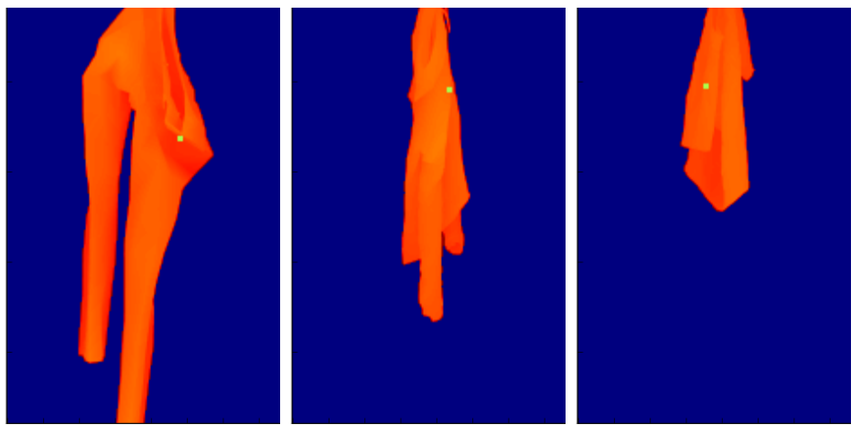


Figure 5.4: Possible grasping points, near the ground truth, selected in the simulated environment, in the case of jeans. Other points were selected further from the ground truth to capture images with a wider range of poses.

Having the images taken, the ground truth was computed to be the one of the two grasping points that was further of the origin, being it in the holding point of the vertex. We want to grasp the second grasping point once we have the first, but the images were taken from points on all the waist. Hence, even if the first network was not accurate the second would predict a point that could make the cloth end in a very similar position to our known configuration.

Figure 5.5: Synthetic ground truth for localizing the second grasp point, in white points. From left to right: Jeans, jumper and T-shirt.



5.3 Network description

Considering that the robot is going to grasp only the visible points, the regression part of the CNNs should focus on being accurate with the seen points. On the other hand, only training the network with those images where grasp points appear would decrease the number

of training images and so decrease the ability of the CNN to pursue its objective. The best results were obtained when training the network with all the available images, but penalising the error committed with the visible points.

5.3.1 First point detection

The first network has to predict the two points location and visibility. Its output layer has 10 neurons, of which 6 are linear and predict the location in the space for each point. The other four predict the probability of the points to be visible or not. When training, the ground truth has to be structured as the last layer, with the Cartesian positions to be predicted in the place of the linear neurons and the visibility in the Softmax layers.

However, being correctly arranged does not optimize the network, as there are two points to predict, and the prediction order may not always correspond to the ground truth order. If we force our network to predict the grasping locations in a certain arrangement, if random, will not understand what we are looking for. If, instead, the points have a logical order as, for example, the first point is the one on the left and the second is on the right, we are making the network learn a more complex function.

In practice the order of the points predicted does not matter, as long as each point is attached to its visibility. This would make the network train faster and improve its results. To implement this idea, we look first the order of the points predicted and match it to the ground truth for every image, following equation 5.3:

$$nearest_{point}(p1, p2, t1, t2) = argmin(sq_{err}(p1, t1) + sq_{err}(p2, t2), \\ sq_{err}(p1, t2) + sq_{err}(p2, t1)) \quad (5.3)$$

After finding the order of the predictions, the loss is computed depending on every image for classification, equation 5.5, and for regression, as in 5.4:

$$loss_{regression}(p1, p2, t1, t2) = \begin{cases} sq_{err}(p1, t1) + sq_{err}(p2, t2) & \text{when } nearest_{point} = 0 \\ sq_{err}(p1, t2) + sq_{err}(p2, t1) & \text{when } nearest_{point} = 1 \end{cases} \quad (5.4)$$

$$loss_{classification}(p1, p2, t1, t2) = \begin{cases} cl_{err}(p1, t1) + cl_{err}(p2, t2) & \text{if } nearest_{point} == 0 \\ cl_{err}(p2, t1) + cl_{err}(p1, t2) & \text{if } nearest_{point} == 1 \end{cases} \quad (5.5)$$

Both equations depend of the squared error of the point and the categorical cross entropy, shown in equations 5.6 and 5.7 respectively. The subscripts indicate the value of the ground truth and the predicted point in each axis, or their visibility, which has a binary value.

$$sq_{err}(p, t) = (p_x - t_x)^2 + (p_y - t_y)^2 + (p_z - t_z)^2 \quad (5.6)$$

$$cl_{err}(p, t) = (-t_{visibility} * log(p_{visibility}) - (1 - t_{visibility}) * log(1 - p_{visibility})) \quad (5.7)$$

To train the network to maximize both features equally, the total loss was the sum of the regression and the classification losses, for every image. But the regression loss represents the sum on squared differences in centimetres, while the classification error is in the order of units or less. Therefore, the second one was scaled by a constant factor and, as shown in equation 5.8, summed to regularization [37], to prevent overfitting.

$$Loss = \sum Loss_{regression} + k * \sum Loss_{classification} + Regularization \quad (5.8)$$

The equations shaping the final loss function have to be calculated image by image to know the correct order of the predictions. This can be vectorized in python and its implementation can be found in a maintained git [39] repository containing the code written in this project, using Theano [40] and Lasagne [41] libraries.

Incrementing the complexity of the network proved to be important to gain accuracy, since the validation results were increasingly precise as the network became deeper. Table 5.1 shows some networks trained to predict the points and their results, and how increasing their complexity also increased the accuracy. The networks shown have a dropout of 50 % [27], regularization [37] and the same first convolutional filters shape.

Table 5.1: Trained networks. As CNNs became deeper, the results improved.

Nº of convolutional layers	Average squared distance to grasping points	Accuracy
2	62.809 cm^2	61.18 %
3	28.39 cm^2	72.73 %
4	6.43 cm^2	85.88 %
ImageNet [24]	4.14 cm^2	88.97 %

Adding convolutional filters and widening the layers improved the network performance. Training a big network from scratch requires a considerable amount of time and does not give as good results. Therefore, after training networks with five layers, we started to use a pre-trained ImageNet [24] changing their last layer, with over hundred and forty million parameters that occupy 553 Megabytes. At the end, the network is slightly bigger than the original ImageNet because their input image size is smaller. This involves a few new fully connected neurons that were randomly initialized.

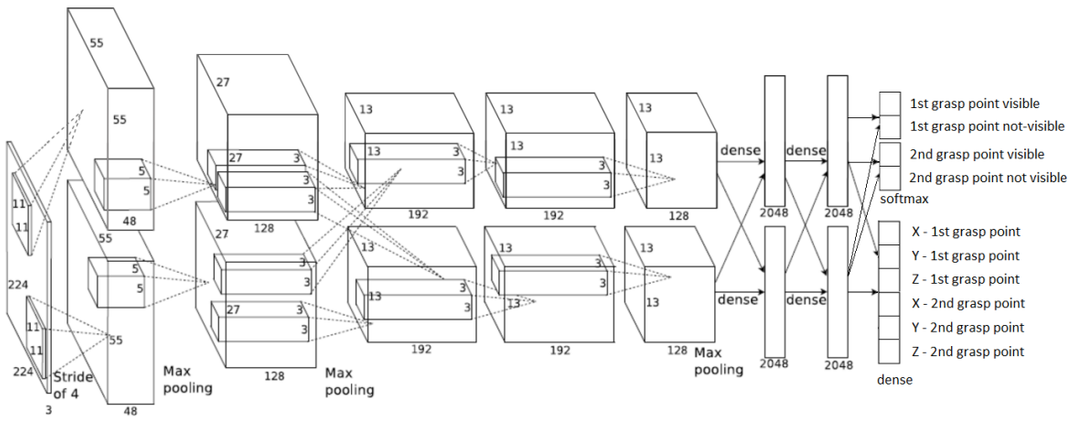


Figure 5.6: First points predicting network structure. Original image extracted from [24].

As the same network is used to predict regression and classes, both objectives share all the network weights, except for the last layer, where the fully connected layer connects to six linear neurons that predict the Cartesian position of the points and to four softmax [34] units in charge of classifying. After that, the ten neurons are concatenated and the whole network trained as one. The whole structure can be seen in figure 5.6.

5.3.2 Second point detection

In a similar approach than in section 5.3.1, the network last layer has five outputs: Three linear values and the probabilities for the point to appear and not to appear. In this case, only having one point to predict simplifies the loss functions, shown in figure 5.9, where the squared error and the categorical cross entropy refer to the equations 5.6 and 5.7 respectively.

$$Loss(p,t) = \sum sq_{err}(p,t) + k * \sum cl_{err}(p,t) + Regularization \tag{5.9}$$

To compare our method of classifying and predicting the point position in the same network to the specialized networks only predicting each feature at one time, we also trained the classification and the regression networks. The validation accuracy and the average squared distance during training, for the three networks, are shown in figure 5.7. Unexpectedly, an only classifier network gives approximately one point less than our network, predicting both properties. This may be consequence of the regression labels, being related to the visibility of the points and, therefore, contributing to the network to learn trough another manner. It is noticeable, though, that our network needs more training epochs to learn to classify the point visibility, due to the other features to be learned.

The regression results, on the other side, are comparable and both networks follow a very similar loss curve. The three networks were trained using Adam [32] optimization method which adapts the learning rate depending on the magnitude of the loss function. At the end, predicting the visibility and the position of the grasping points in only one network involves a considerable reduction of memory used when saving the parameters and running the network. Moreover, such complex networks may need a non-negligible amount of time when not running in a GPU, being more rapid to compute with only one network.

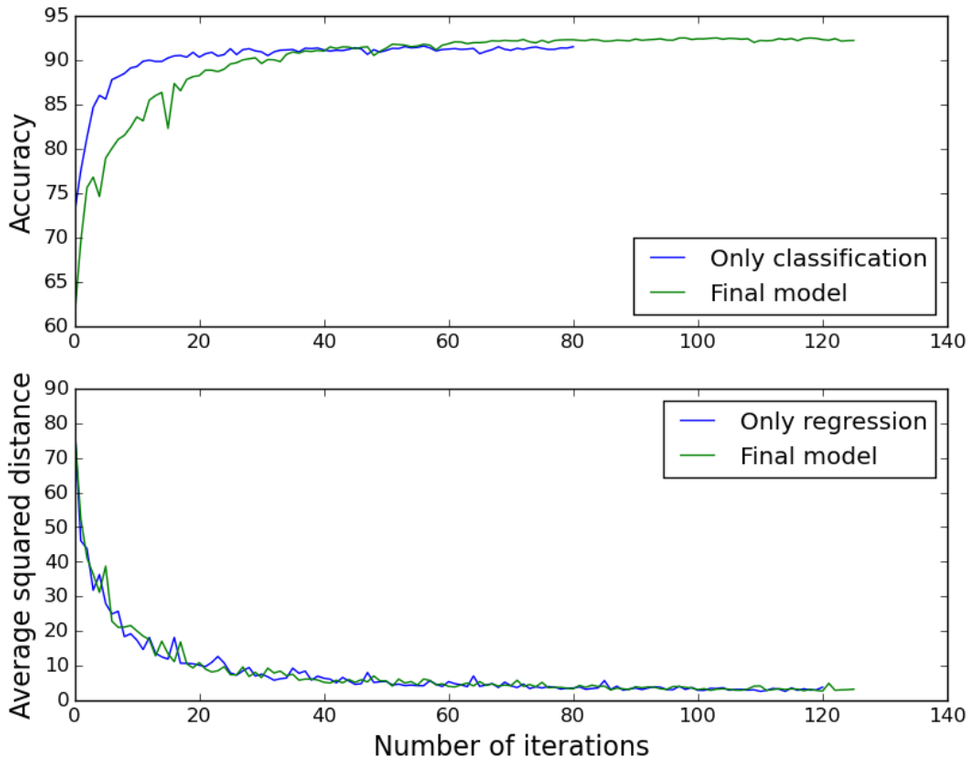


Figure 5.7: Second point predicting network accuracy (top graphic) and average squared distance to ground truth (second graphic) compared to the networks specialized only in classifying and predicting the point location, respectively.

The network structure is very similar to the one presented in figure 5.6, but the last layer predicts only three linear values and the class. The resulting structure is shown in figure 5.8.

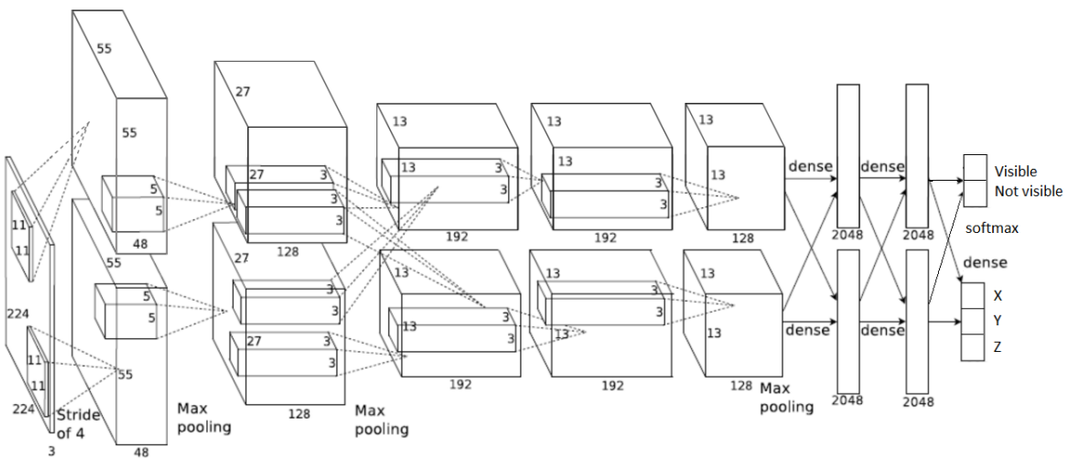


Figure 5.8: Second point predicting network structure. See figure 5.6 to compare. Original image extracted from [24].

5.4 Results

This section will describe the results of the two CNNs trained for predicting the grasping points position in synthetic and in real images.

5.4.1 Results in synthetic images

The average of squared Cartesian distance between the predicted position of the points to the ground truth, evaluated in the test set, is shown in table 5.2, with the accuracy of the visibility predicted. Logically, the first network has an error considerably bigger than the second, as has to deal with a higher number of possible poses and, therefore, have to approximate a more complex function. Apart from that, the error appears to be reasonable to grasp clothing with a good succeeding rate. The direction having more error is the Z axis, in the depth direction, that may involve problems identifying the points on the surface of the cloth. The X and Y axis are the horizontal and vertical directions respectively. The error on these two axis appear to be directly related to the shape the garments adapt. In the case of jeans, which can be grasped from the jeans legs or the waist, for example, can have their grasp points in a very wide vertical range of the image. Differently, T-shirts are rather smaller and this suggests that, comparatively, the network errors on the Y axis is also more reduced.

Table 5.2: Errors per output neuron in final network, where regression error is in cm^2.

	First point				Second point			
Garment	X	Y	Z	Visibility acc.	X	Y	Z	Visibility acc.
Jeans	0.80	1.81	1.53	88.97 %	0.50	1.30	0.78	92.46 %
Jumper	1.22	1.50	2.70	88.54 %	0.54	0.59	1.05	94.92 %
T-Shirt	2.06	1.34	3.66	83.63 %	1.59	1.86	3.39	90.32 %

Table 5.3 summarizes the best errors committed by the network in more human-understandable units: The distance from the predicted points to ground truth in centimetres. As shown, jeans and jumpers achieve comparable results but T-shirts present slightly more difficult to deal with, because of the wrinkles they form and the small characteristic features - only short sleeves - that help to find cloth features.

Table 5.3: Distance from predicted point to ground truth in centimeters.

Garment	Jeans	Jumper	T-Shirt
Average error distance on the first grasping point	1.59 cm	2.22 cm	2.76 cm
Average error distance on the second grasping point	1.52 cm	1.18 cm	2.16 cm

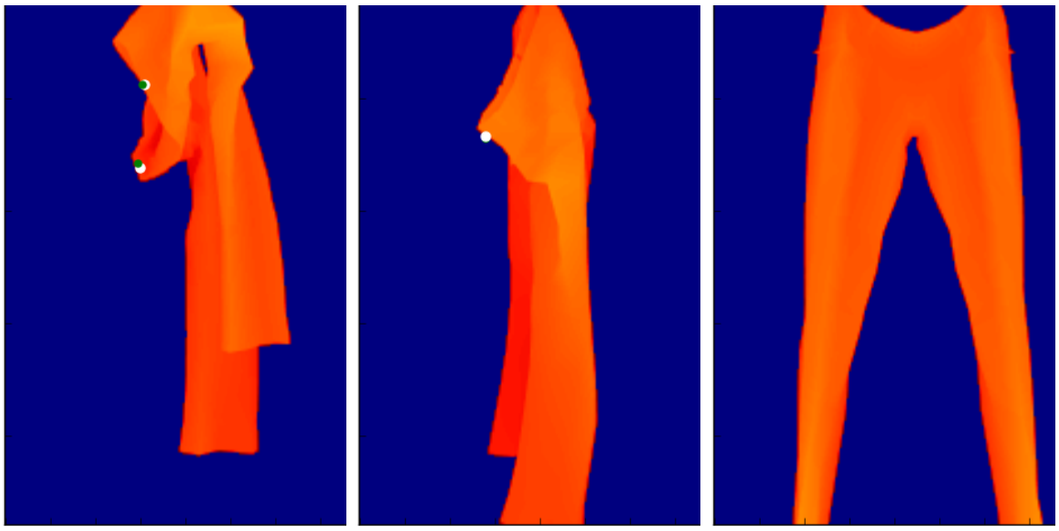
5.4.2 Synthetic images simulations

The process of bringing the cloth to known configuration was simulated using depth images obtained of synthetic garment models. Supposing the robot would be able to grasp the garment from the points found, the process is shown in the figure 5.9. We start from a random pose of the grasped garment, in this case a pair of jeans, and evaluate its depth image with the first identification CNN. Once it classifies the garment as jeans, the second CNN predicts if any of the two possible grasping point are visible. If so, the Cartesian location of the points predicted are used to validate if the values correspond to a valid point in the image. Otherwise the piece of cloth is rotated.

After having grasped the first point, the third network predicts if the second reference point is visible and, if it is, the values are used as in figures 5.3 and 5.5 to find the corresponding pixel in the image. When there is a valid point, the robot would grasp it and, oppositely, would rotate the garment. Finally, when the two points are grasped we would see the image in the last column of figure 5.9.

The current simulation eludes typical hardware and planning problems, not taken into account into this project, such as collisions between the two robotic arms or with other objects, and supposes the grasping action always succeeds.

Figure 5.9: Synthetic simulation of grasping the garment from known points. The first image shows the jeans in a random position and the predicted grasping points, in white, with the ground truth, in green. Similarly, the second image presents the jeans grasped from one of the two predicted points in the first image. It also presents the grasping point predicted in white. The third image shows the jeans being grasped from the two points predicted before.



The whole process is simulated by two different sizes of jeans, never seen before by the network, from which each of them groups 93 poses, being grasped from as many vertexes. As we have the depth images from all their views in the space, we can pretend the garment is being rotated. From all the simulations, the final configuration is summarized in table 5.4. Though to simulate the process we only have a limited position of vertexes, so the distance of the final grasping points to the ground truth in the final pose is quantified.

Table 5.4: Percentage of grasping points grasped in simulation and their distance to the ground truth. Possible grasping points had to be discretized and were labelled in the synthetic model.

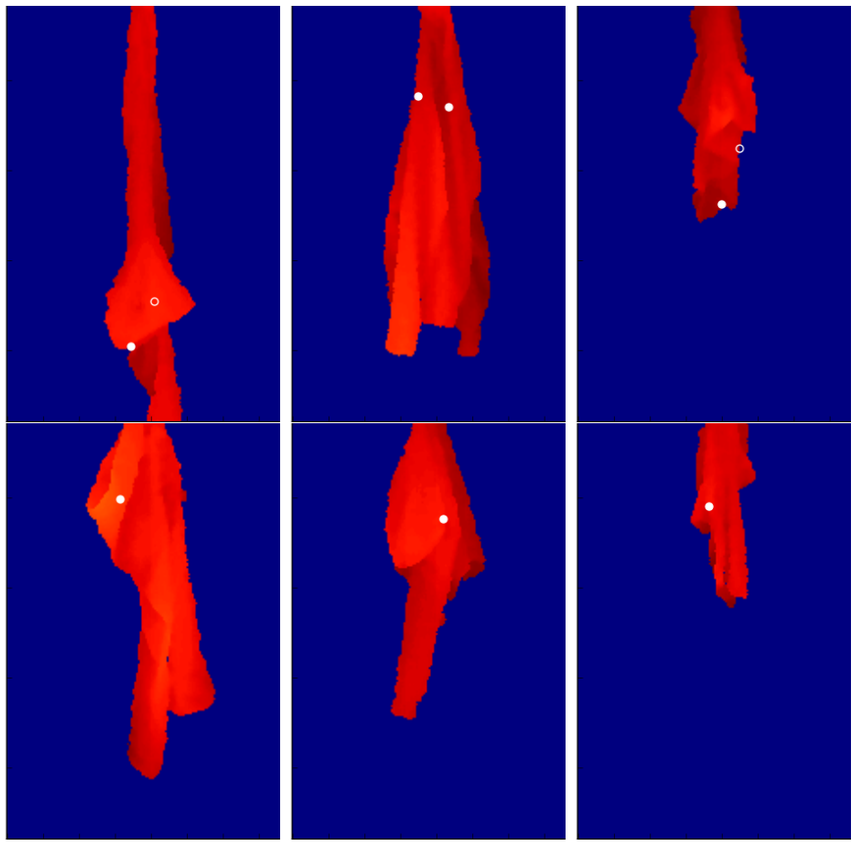
	Exact point	Between 2 - 4 cm	Between 4 and 8 cm	Between 8 and 12 cm	Between 12 and 16 cm	Between 16 and 20 cm
1st point grasped	81.18 %	4.3 %	5.38 %	5.38 %	2.15 %	1.07 %
2nd point grasped	96.77 %	2.69 %	0.64 %	0.00 %	0.00 %	0.00 %

5.4.3 Results in real images

Grasp points predictions were obviously less accurate when predicting from real garment images, never seen by the convolutional network. In these cases, training them with added kinect noise [2] helps the network to slightly increase its results, but a refinement would be necessary to improve more considerably its accuracy, as in section 4.3.

Almost two hundred depth images of real jeans were hand-labelled to refine the convolutional networks, being a highly time-consuming task that difficulties the goal of having a big set of images. Also, the labelling process contains an added error as the grasping point is sometimes not easily identifiable, apart from the inherent error committed by handmade tagging. The labelled images were used to validate the models and obtained an error of approximately 6 centimeters. Nonetheless, this small amount of images did not improve the results when refining the model, being rapidly overfitted.

Figure 5.10: Grasping point location predictions. From left to right: Grasping points prediction for jeans, jumper and T-shirt. The upper row shows the supposed position of the two first known points and the second row shows the predicted second grasp point location. A filled white circle means the grasp point is visible, and therefore the robot could approach and grasp it, while an empty circle suggests we need another view to see the point.



6 Budget

Table 6.1 shows the amortization of each hardware used in this project. The office supplies and the specialized equipment have an amortization time of three and five years respectively.

Table 6.1: Hardware resources.

Means	Amount	Price per unit	Amortization time	Price per hour	Hours used	Amortization
Server PC	1	1600	3 years	0.22 €	1210	266.2 €
Lab PC	1	1000	3 years	0.14 €	700	98 €
Barret WAM	2	100000	5 years	8.33 €	18	150 €
GPU	1	3400	5 years	0.28 €	1200	336 €
					Total:	**850.2 €**

There are no software costs as we used only open source programs. The physics simulator [1] used to simulate the synthetic garments was under a student license and the convolutional networks were trained and tested in python language.

The costs associated to human resources are described in table 6.2 and take into account different specialized jobs realized in this project.

Table 6.2: Human resources.

Position	Wage	Hours	Total
Software engineer	8 €/h	712	5695 €
Project manager	15 €/h	42	630 €
		Total:	**6325 €**

7 Conclusions and future work

In our initial problem, we have a piece of clothing unidentified in an unfamiliar position and we need to have the piece of cloth recognized and in a known position to perform another action, such as fold it. The process seems to need several recognition algorithms, for which we used CNNs because of their increasingly good performances in several computer vision tasks.

We established a pipeline of three steps to achieve the recognition and grasping of each garment. Each of the three steps contains its specialized Convolutional Neural Network, whose prediction defines how to move to the next stage.

To train the CNNs, we have used images from a synthetic source, which has proved to be a very interesting approach that helps to increase the accuracy in cases where obtaining real images may be as high time-consuming.

To grasp the garment, we decided to predict the Cartesian position of the known points only trough their images, which apparently was remarkably complex function to learn by CNNs. This would let us grasp the points in order, reaching a final known configuration dependant on the identified garment.

Garment identification

From the ideas tested in the first part of the project, identifying garments, CNNs worked with synthetic images to create comparable results to the current state of the art, that were refined with real data to outperform the current best classification accuracy.

Two lessons related to synthetic datasets were learnt in this task. First, using synthetic images confirmed to be rather useful when working with problems where gathering real datasets is a highly time-consuming task. Making synthetic garment 3D objects appear to real ones can be difficult, but some unexpected features, such as the camera noise or the deforming properties of the garment, can be modelled.

Secondly, a dataset of real images should be obtained to optimize the networks for our concise problem. When combining synthetic images with real ones, the accuracy can increase considerably as seen in section 4.3.

Related to Convolutional Neural Networks, we learned what features usually work better. In our experience, dropout [27], regularization [37], using ReLU functions [29] and deepening the network were the techniques used to increase the accuracy. Also, batch normalization [28] proved to be very useful, although has to be used carefully when predicting with images rather different to the seen in the training set. To train the networks, Adam [32] optimization method confirmed to be much faster than Stochastic Gradient Descent with Nesterov momentum [33].

Bringing cloth to a known configuration

Our results from section 5 imply this problem can also be fronted with CNNs and, using synthetic images, achieve a good success rate when bringing a garment to a familiar pose.

Also, we show how to achieve regression and classification in the same network trough modifying their last layer and expose how to approach order invariant result predictions when two or more features are to be predicted.

In practice, a dataset of real depth images would improve much more the results when grasping real clothes. Probably a refinement would decrease the distance from the predicted point to the real known reference points. Nonetheless, a refinement in regression and in such a complex function to be approximated such as this one, apparently needs a vast amount of images, differently to classifying. Also, using a newer camera that provides less noise may definitely increase the results as they may appear more similar to the synthetic images.

Future work

Approach the identification task not on a fixed distance, but more generally, could provide robots of a certain autonomy to manipulate clothes wherever they may interest to. In the physics simulator, creating cameras over the space at a different distances from the garment and different angles, as in figure 7.1, can generate enough depth images to train a convolutional network capable of identifying the garment independently on the position. However, the identification accuracy would probably decrease when getting too close, not being able to see all the pose, or too far, not to appreciate its texture.

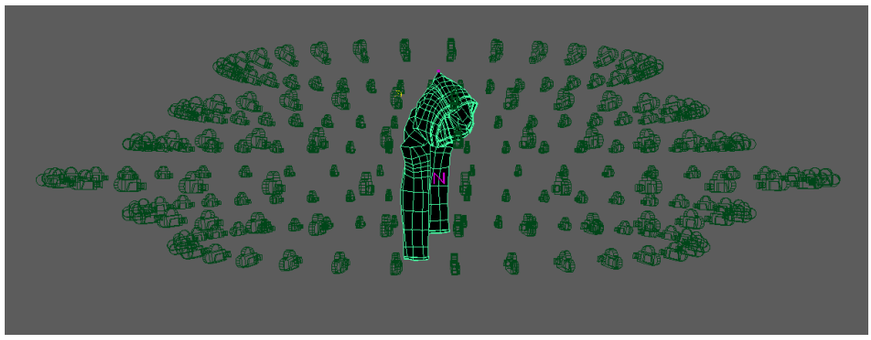


Figure 7.1: Possible position of cameras to obtain images from more positions.

Also, we could increase the difficulty of the identification problem, considering the possibility of having more than one garment grasped in the same point. When several clothing garments form a pile, more than one can be caught at the same time.

On the other hand, to improve the manipulation results in real garments a set of images linked to their grasping points should be obtained. This could be done in different ways. The points could be differently tinted to apply color segmentation in color images to find where they are, whenever they are visible. Differently, to generate a dataset where the non-visible grasping points are also positioned, they could be joined to an infrared light that would be captured by two or more calibrated infrared cameras, from which we could obtain the Cartesian position of the points. As obvious this complicates the process but, as seen when identifying, real images can rise considerably the network precision.

New approaches could be tried to predict the position of the point. The efficiency of Convolutional Neural Networks is increasing with time and other techniques giving better results in real images may appear, such as image segmentation with CNNs [42].

Similarly to identification, taking images of different distances and angles of view may give autonomy to robots when grasping garments.

References

[1] "Maya." http://www.autodesk.com/products/autodesk-maya.

[2] B. Choo, M. Landau, M. DeVore, and P. A. Beling, "Statistical Analysis-Based Error Models for the Microsoft Kinect^{TM} Depth Sensor," *Sensors*, vol. 14, no. 9, pp. 17430–17450, 2014.

[3] "WAM Arm." http://www.barrett.com/DS_WAM.pdf.

[4] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2308–2315, IEEE, 2010.

[5] "Xtion." https://www.asus.com/3D-Sensor/Xtion_PRO_LIVE/specifications/.

[6] M. Kaneko, Y. Tanaka, and T. Tsuji, "Scale-dependent grasp-a case study," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 3, pp. 2131–2136, IEEE, 1996.

[7] M. Kaneko and M. Kakikura, "Planning strategy for putting away laundry-isolating and unfolding task," in *Assembly and Task Planning, 2001, Proceedings of the IEEE International Symposium on*, pp. 429–434, IEEE, 2001.

[8] A. Colomé Figueras, D. E. Pardo Ayala, G. Alenyà Ribas, and C. Torras, "External force estimation for textile grasp detection," in *Proceedings of the 2012 IROS Workshop Beyond Robot Grasping: Modern Approaches for Learning Dynamic Manipulation*, pp. 1–1, 2012.

[9] B. Willimon, S. Birchfield, and I. Walker, "Classification of clothing using interactive perception," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1862–1868, IEEE, 2011.

[10] B. Willimon, I. Walker, and S. Birchfield, "A new approach to clothing classification using mid-level layers," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 4271–4278, IEEE, 2013.

[11] Y. Kita, F. Kanehiro, T. Ueshiba, and N. Kita, "Clothes handling based on recognition by strategic observation," in *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pp. 53–58, IEEE, 2011.

[12] M. Cusumano-Towner, A. Singh, S. Miller, J. F. O'Brien, and P. Abbeel, "Bringing clothing into desired configurations with limited perception," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3893–3900, IEEE, 2011.

[13] Y. Li, C.-F. Chen, and P. K. Allen, "Recognition of deformable object category and pose," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5558–5564, IEEE, 2014.

[14] Y. Li, Y. Yue, D. Xu, E. Grinspun, and P. K. Allen, "Folding deformable objects using predictive simulation and trajectory optimization," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 6000–6006, IEEE, 2015.

[15] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pp. 127–136, IEEE, 2011.

[16] B. Lao and K. Jagadeesh, "Convolutional neural networks for fashion classification and object detection,"

[17] I. Mariolis, G. Peleka, A. Kargakos, and S. Malassiotis, "Pose and category recognition of highly deformable objects using deep learning," in *Advanced Robotics (ICAR), 2015 International Conference on*, pp. 655–662, IEEE, 2015.

[18] "AMDO." http://amdo2016.uib.es/.

[19] F. Osawa, H. Seki, and Y. Kamiya, "Unfolding of massive laundry and classification types by dual manipulator.," *JACIII*, vol. 11, no. 5, pp. 457–463, 2007.

[20] A. Ramisa, G. Alenya, F. Moreno-Noguer, and C. Torras, "Finddd: A fast 3d descriptor to characterize textiles for robot manipulation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 824–830, IEEE, 2013.

[21] A. Doumanoglou, A. Kargakos, T.-K. Kim, and S. Malassiotis, "Autonomous active recognition and unfolding of clothes using random decision forests and probabilistic planning," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 987–993, IEEE, 2014.

[22] Y. Li, D. Xu, Y. Yue, Y. Wang, S.-F. Chang, E. Grinspun, and P. K. Allen, "Regrasping and unfolding of garments using predictive thin shell modeling," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1382–1388, IEEE, 2015.

[23] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[26] "Imagenet." `http://image-net.org/about-overview`.

[27] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[29] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[30] A. C. Marreiros, J. Daunizeau, S. J. Kiebel, and K. J. Friston, "Population dynamics: variance and the sigmoid activation function," *Neuroimage*, vol. 42, no. 1, pp. 147–157, 2008.

[31] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural Networks, 1989. IJCNN., International Joint Conference on*, pp. 593–605, IEEE, 1989.

[32] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project adam: Building an efficient and scalable deep learning training system," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pp. 571–582, 2014.

[33] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning.," *ICML (3)*, vol. 28, pp. 1139–1147, 2013.

[34] K. Duan, S. S. Keerthi, W. Chu, S. K. Shevade, and A. N. Poo, "Multi-category classification by soft-max combination of binary classifiers," in *International Workshop on Multiple Classifier Systems*, pp. 125–134, Springer, 2003.

[35] "Maya." `http://download.autodesk.com/global/docs/maya2014/en_us/index.html?url=files/GUID-F48E3B78-3E56-4869-9914-CE0FAB6E3116.htm,topicNumber=d30e144972`.

[36] H. Wu and X. Gu, "Towards dropout training for convolutional neural networks," *Neural Networks*, vol. 71, pp. 1–10, 2015.

[37] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural computation*, vol. 7, no. 2, pp. 219–269, 1995.

[38] "Kinect specifications." `https://msdn.microsoft.com/en-us/library/hh973078.aspx`.

[39] "Code used in this project." `https://gitlab.iri.upc.edu/ecorona/Maya_codes.git`.

[40] "Theano." `https://github.com/Theano/Theano`.

[41] "Lasagne." `https://github.com/Lasagne/Lasagne`.

[42] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

Appendices

A Project code

The related source code of the project is being maintained in a public Git repository in `https://gitlab.iri.upc.edu/ecorona/Maya_codes.git`.

B Abstract of AMDO article

Abstract from the accepted paper in the IX Conference on Articulated Motion and Deformable Objects [18]:

We present a system to deal with the problem of classifying garments in a pile of clothes. This system uses a robot arm to extract a garment and show it to a depth camera. The robot, then, moves the garment along the vertical axis in order to provide different views of the garment until a prediction with enough confidence is reached. For the classification task, a deep convolutional neural network has been trained to label different types of garments given a depth image. In addition to obtaining very high classification scores, compared to previous approaches to cloth classification that match the sensed data against a database, our system provides a fast and occlusion-robust solution to the problem.

C Trained classifier networks

Network parameters						Training		Refinement	
N° layers	Regularization	Nonliniarities	First conv. filter	Added noise	Dropout	Normalization	Loss	Acc (%)	Loss	Acc (%)
2	None	Sigmoid	128x11x11	None	None	Pixelwise	0.694	78.42	0.328	88.89
3	None	Sigmoid	128x8x8	None	Dropout = 0.3	Pixelwise	0.626	79.81	0.279	90.28
4	None	Rectify	128x11x11	Gaussian noise (std = 0.005 m)	None	Pixelwise	0.715	72.87	-	-
4	None	Rectify	32x8x8	None	Dropout = 0.5	Batch Norm.	0.494	80.37	0.27	90.28
4	Yes (L2)	Rectify	32x8x8	None	Dropout = 0.5	Batch Norm.	0.381	86.48	0.138	95.28
4	None	Sigmoid	64x8x8	None	Dropout = 0.3	Pixelwise	0.598	79.56	0.305	90.28
4	Yes (L2)	Sigmoid	64x8x8	None	Dropout = 0.3	Pixelwise	0.583	78.89	0.287	90.28
4	Yes (L1 + L2)	Sigmoid	64x8x8	None	Dropout = 0.3	Pixelwise	0.599	79.53	0.289	90.00
4	Yes (L2)	Sigmoid	64x8x8	None	Dropout = 0.3	Pixelwise	0.542	79.44	0.293	89.72
4	Yes (L2)	Sigmoid	64x8x8	None	Dropout = 0.3	Pixelwise	0.627	82.87	0.201	94.26
4	Yes (L2)	Sigmoid	64x7x9	None	Dropout = 0.5	Pixelwise	0.57	81.85	0.232	91.94
4	Yes (L2)	Sigmoid	64x7x9	None	None	Batch Norm.	0.502	80.80	0.290	90.46
4	Yes (L2)	Rectify	64x8x8	Kinect noise in x,y and depth [2]	Dropout = 0.5	Pixelwise	0.52	82.96	0.216	93.24
4	Yes (L2)	Rectify	64x7x9	Kinect noise in x,y and depth [2]	Dropout = 0.5	Pixelwise	0.53	85.19	0.201	94.07
4	Yes (L2)	Rectify	64x8x8	Kinect noise in x,y and depth [2]	Dropout = 0.5	Batch Norm.	0.54	84.91	0.292	92.87
4	Yes (L2)	Rectify	32x8x8	Kinect noise in x,y and depth [2]	Dropout = 0.5	Batch Norm.	0.354	88.61	0.125	96.85
4	Yes (L2)	Rectify	32x7x8	None	Dropout = 0.3	Batch Norm.	0.576	79.72	0.256	90.46

Glossary

Adam A method for stochastic optimization [32]. 22, 35

CNNs Convolutional Neural Networks. 1, 2, 5–9, 11, 12, 17, 19, 23, 27, 28, 35

CSV Comma-Separated Values. 17, 35

GPU Graphics Processing Unit. 6, 8, 22, 35

IRI Institut de Robòtica i Informàtica Industrial. 35

MEL Maya Embedded Language. 10–12, 19, 35

ReLU Rectified Linear Unit. 13, 27, 35

TIF Tagged Image File Format. 10, 35