Imperial College London
Department of Electrical & Electronic Engineering

Active Recognition and Pose Estimation of Rigid and Deformable Objects in 3D Space

Andreas Doumanoglou

Supervised by Dr. Tae-Kyun Kim

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Electrical & Electronic Engineering and
the Diploma of Imperial College London, February 2017
The work presented in this thesis is entirely my own, except where explicitly
stated and appropriately referenced in the text.

Abstract

Object recognition and pose estimation is a fundamental problem in computer vision and of utmost importance in robotic applications. Object recognition refers to the problem of recognizing certain object instances, or categorizing objects into specific classes. Pose estimation deals with estimating the exact position of the object in 3D space, usually expressed in Euler angles[1]. There are generally two types of objects that require special care when designing solutions to the aforementioned problems: rigid and deformable. Dealing with deformable objects has been a much harder problem, and usually solutions that apply to rigid objects, fail when used for deformable objects due to the inherent assumptions made during the design.

In this thesis we deal with object categorization, instance recognition and pose estimation of both rigid and deformable objects. In particular, we are interested in a special type of deformable objects, clothes. We tackle the problem of autonomously recognizing and unfolding articles of clothing using a dual manipulator[2]. This problem consists of grasping an article from a random point, recognizing it and then bringing it into an unfolded state by a dual arm robot. We propose a data-driven method for clothes recognition from depth images using Random Decision Forests. We also propose a method for unfolding an article of clothing after estimating and grasping two key-points, using Hough Forests. Both methods are implemented into a POMDP framework allowing the robot to interact optimally with the garments, taking into account uncertainty in the recognition and point estimation process. This *active* recognition and unfolding makes our system very robust to noisy observations. Our methods were tested on regular-sized clothes using a dual-arm manipulator. Our systems perform better in both accuracy and speed compared to state-of-the-art approaches.

In order to take advantage of the robotic manipulator and increase the accuracy of our system, we developed a novel approach to address generic active vision problems, called *Active Random Forests*[3]. While state of the art focuses on best viewing parameters selection based on single view classifiers, we propose a multi-view classifier where the decision mechanism of optimally changing viewing parameters is inherent to the classification process. This has many advantages: a) the classifier exploits the entire set of captured images and does not simply aggregate probabilistically per view hypotheses; b) actions are based on learnt disambiguating features from all views and are optimally selected using the powerful voting scheme of Random Forests and c) the classifier can take into account the costs of actions. The proposed framework was applied to the same task of autonomously unfolding clothes by a robot, addressing the problem of best viewpoint selection in classification, grasp point and pose estimation of garments. We show great performance improvement compared to state of the art methods and our previous POMDP formulation.

Moving from deformable to rigid objects while keeping our interest to domestic robotic applications, we focus on object instance recognition and 3D pose estimation of household objects. We are particularly interested in realistic scenes that are very crowded and objects can be perceived under severe occlusions. Single shot-based 6D pose estimators with manually designed features are still unable to tackle such difficult scenarios for a variety of objects, motivating the research towards unsupervised feature learning and next-best-view estimation. We present a complete framework[4] for both single shot-based 6D object pose estimation and next-best-view prediction based on Hough Forests, the state of the art object pose estimator that performs classification and regression jointly. Rather than using manually designed features we propose an unsupervised feature learnt from depth-invariant patches using a Sparse Autoencoder. Furthermore, taking advantage of the clustering performed in the leaf nodes of Hough Forests, we learn to estimate the reduction of uncertainty in other views, formulating the problem of selecting the next-best-view. To further improve 6D object pose estimation, we propose an improved joint registration and hypotheses verification module as a final refinement step to reject false detections. We provide two additional challenging datasets inspired from realistic scenarios to extensively evaluate the state of the art and our framework. One is related to domestic environments and the other depicts a bin-picking scenario mostly found in industrial settings. We show that our framework significantly outperforms state of the art both on public and on our datasets.

Unsupervised feature learning, although efficient, might produce sub-optimal features for our particular tast. Therefore in our last work, we leverage the power of

Convolutional Neural Networks to tackled the problem of estimating the pose of rigid objects by an end-to-end deep regression network. To improve the moderate performance of the standard regression objective function, we introduce the Siamese Regression Network. For a given image pair, we enforce a similarity measure between the representation of the sample images in the feature and pose space respectively, that is shown to boost regression performance. Furthermore, we argue that our pose-guided feature learning using our Siamese Regression Network generates more discriminative features that outperform the state of the art. Last, our feature learning formulation provides the ability of learning features that can perform under severe occlusions, demonstrating high performance on our novel hand-object dataset.

Concluding, this work is a research on the area of object detection and pose estimation in 3D space, on a variety of object types. Furthermore we investigate how accuracy can be further improved by applying active vision techniques to optimally move the camera view to minimize the detection error.

Acknowledgements

First, I would like to thank my supervisor Dr. Tae-Kyun Kim for his great support and all the valuable conversations we had during our collaboration. He showed a great interest for our work and spent a lot of hours formulating our ideas that was beyond my expectations. I would also like to thank Sotiris Malassiotis for his great support and leadership during the project CloPeMa. Last, thanks to all my colleges, friends and family without them I couldn't be able to overcome all the difficulties of the PhD.

Dedication

To Maria Petrou

‘Any sufficiently advanced technology is indistinguishable from magic.’

Arthur C. Clarke

Contents

List of Tables

List of Figures

List of Abbreviations

POMDP	Partially Observable Markov Decision Process
6D	6 Dimensions, also used in place of 6DoF
6DoF	6 Degrees of Freedom
RGB(-D)	Red Green Blue (colored image) (-Depth image)
CNN	Convolutional Neural Network
SARSOP	Successive Approximations of the Reachable Space under Optimal Policies
ARF	Active Random Forest
CAD	Computer-aided design

Chapter 1

Introduction

1.1 Object Detection in Robotics

The term object detection refers to various problems in vision, all of which are related to automatically detecting and understanding objects in scenes captured using various types of sensors. The most widely used sensor is the RGB camera, which can capture and represent a scene by 3 basic colors (channels): red, green and blue. In recent years there was a significant improvement of such systems, which sometimes could be compared with human level accuracy[7]. In the field of robotics however, the ultimate goal for a robot is to interact with the objects, such as grasp them, manipulate, or avoid them. Such tasks require, apart from the object type, richer information about the objects such as its accurate position in space. Dealing with 3D space had been a challenging problem for the years with expensive laser scanners being the only option for 3D sensing. Recently, 3D data can be much easier acquired after the appearance of low cost commodity depth cameras such as Microsoft Kinect[8]. Such depth cameras operating in real-time gave robotics a significant boost in accomplishing perception tasks. The robots can now have one more and very important channel in the images that acquire, that represents the distance of each pixel from the camera. Although such 3D information is very important, methods developed for RGB cameras only are not directly applicable to 3D data. Therefore, new methods need to be developed in order to take full advantage of the 3D information available. They also need to be robust to 3D camera artifacts since data from a depth camera are sometimes inaccurate or even missing. This is a compromise of such cameras between low cost and real-time performance versus data accuracy and availability. Nevertheless, real-time availability of 3D information gives robots a great ability to interact with objects and their environment.

On the other hand, the ability of robots to move creates another possibility for object detection methods. Being able to look an object from a different perspective can be used to improve object detection systems. That is, when there is uncertainty about what is being perceived from the current point of view, the robot may decide to look from another side to disambiguate its outcomes. Such behavior is very common to humans who explore objects from various viewpoints when they do not understand what an object is. However, although humans naturally decide how to further investigate an object in order to identify it, for robots this is an open problem: how to decide which viewpoint a robot should look an object from in order to maximize the probability to identify it correctly. This research topic is called *Active Vision* and in this thesis we have developed various methods in order to solve the so called *next best view prediction* problem.

1.2 Challenges in Rigid and Deformable Object Detection

The robotic problems this thesis tackles are related to objects that a robot needs to manipulate, which means their size is comparable to the robot's size. Such objects fall into two main categories: rigid and deformable. Regarding deformable, we are interested in a specific type of objects: clothes. We want to solve the laundry problem by using autonomous robots and our focus is on the part of folding clothes, after grasping them from a table. Clothes are one of the most deformable types of objects that also have an unusual property: they can appear in any color or texture with almost no limitations. Therefore, one should solely rely on the shape information of a garment. That being said, the problem becomes even harder since the shape of garment lies in an almost infinite configuration space. However, although garments can be greatly deformed, they can maintain some structure when being properly manipulated by a robot. All these make clothes recognition and pose estimation a very difficult problem.

Regarding rigid objects, we investigate the problem of localizing and estimating the 3D pose of objects found in domestic and industrial environments. Such problem is also challenging and arises in many robotic applications that require object manipulation. Low-cost availability of depth data facilitates pose estimation significantly, but still one has to cope with many challenges such as viewpoint variability, scene clutter and occlusions caused by nearby objects. These challenges are so common in real-life problems that in order to describe them shortly we introduced the term "object detection and pose estimation *in the crowd*". Rigid objects are categorized according to their appearance into two general types: textured and texture-less. This categorization is mainly technical because techniques in literature are mostly able to deal with only one type of objects, and not both by using the same method. That is because in textured objects, it is easy to detect several key-points and match them with key-points of objects in the training database. However, texture-less objects do not have this property and therefore key-point matching techniques are not applicable to this type of objects. On the other hand, methods that tackle texture-less object usually rely on holistic-shape template techniques or shape-aware key-points, which when applied to textured object do not take advantage of the texture. In the following sections, we describe our object detection method which is one of the earliest works to tackle both types of objects in the same framework.

1.3 Thesis Work Overview and Contributions

In this thesis we present contributions and novel methods regarding both deformable and rigid object detection, which are enhanced by novel active vision techniques. Below we describe our work and contributions for each of the sub-problems that we tackled. The first two subsections (1.3.1,1.3.2) refer to the autonomous recognition, pose estimation and unfolding of clothes using a dual-arm manipulator (1.3.1), and how our methods can be further improved using our novel active vision framework called *Active Random Forests* (1.3.2). The next two subsections (1.3.3,1.3.4) describe our rigid object detection and 3D pose estimation methods in cluttered environments and under severe occlusions. Our first approach (1.3.3) uses patches and unsupervised feature learning trained using Hough Forests. We also leverage the clustering property of the forest to build a novel next-best-view prediction technique. Our second method (1.3.4) uses a Siamese Regression Network to propose a complete end-to-end regression network for object pose estimation which also produces efficient feature embeddings. Below we present an overview of our work and contributions in more detail.

1.3.1 Active Recognition and Unfolding of Clothes

The focus of this work is to solve the task of folding clothes using a robot, and particularly the first part of the procedure, which is the unfolding of an article of clothing. Starting from a crumbled initial configuration, we want to recognize the article and then bring it into an unfolded state so that it is ready for folding. One of the key challenges in clothes perception and manipulation is handling the variabilities in geometry and appearance. These variabilities are due to the large number of different configurations of a garment, self-occlusions and the wide range of cloth textures and colors.

Research on clothes perception and manipulation started in the middle 90s [9], pre-

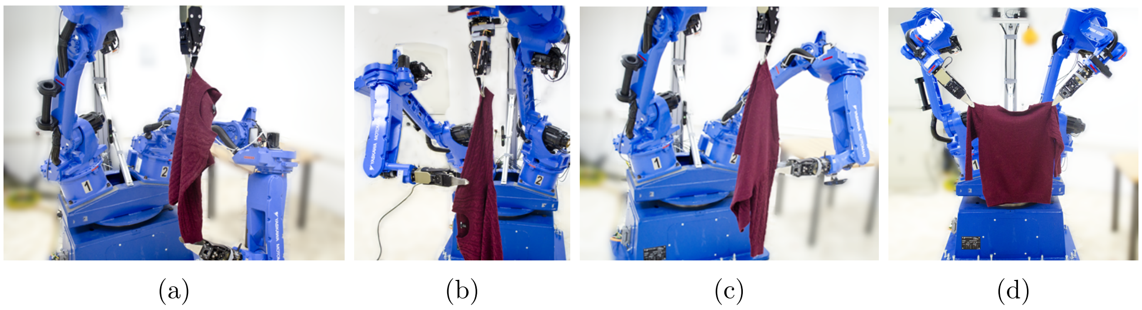


Figure 1.1: Robot autonomously unfolding a shirt. a) Grasping lowest point. b) grasping 1st grasp point. c) grasping 2nd grasp point. d) final unfolded configuration

senting some first clothes recognition techniques with the help of a dual manipulator. Later, research has been made in garment modelling and feature extraction [10] [11], while only recently scientists were able to completely fold an article of clothing starting from a crumpled initial configuration [12] [13] [10]. The main limitations in the state-of-the-art are the slow performance and the difficulty to generalize to a variety of shapes and materials. This stems mainly from the model-driven approaches used and associated simplifying assumptions made.

To address these limitations we propose a data-driven approach for clothes recognition and unfolding. We first recognize the type of the article from raw depth data using Random Forests. Based on the recognition result, a pair of key-points are identified such that the article will naturally unfold when held by these two points. Point estimation is based on Hough Forests, a random forest framework with Hough voting. An active manipulation (perception-action) approach based on POMDPs is also proposed that accounts for uncertainty in the vision tasks and thus leads to superior performance. In summary, our main contributions are:

- A fast method for unfolding an unknown item of clothing by means of gravity (previous approaches have to go through a flattening phase using a table):A robot can detect and grasp two key-points on the garment so that it become flattened while hanging in the air.
- Fast data-driven machine learning algorithms for robust scale-invariant classification of the garment type and key-points estimation from noisy depth data.

These are based on Random Forests and Hough forests respectively.

- A probabilistic perception-action framework based on POMDPs for optimal action policy of the robot. The robot is able to rotate the garment, moving each time to the next viewpoint until it reaches a certain confidence level, in order to better estimate the garment type and the grasp points. This way it accounts for uncertainty in each individual viewpoint.

Compared to the state of the art, our system requires less movements and therefore can operate faster. Furthermore, to our knowledge, this is the first work that autonomously unfolds regular-sized clothes. While most researchers work on small or baby clothes for easier manipulation, regular-sized clothes allow higher degree of deformation and pose more challenges to the recognition and unfolding task.

A demonstration of the unfolding steps using our dual arm robot is shown in Fig. 1.1.

1.3.2 Active Random Forests

As we observed with clothes recognition, single-view methods are often unable to distinguish objects which depict similar appearance when observed from certain viewpoints. An autonomous system can overcome this limitation by actively collecting relevant information about the object, that is, changing viewpoints, zooming to a particular area or even interacting with the object itself. This procedure is called *active vision* and the key problem is how to optimally plan the next actions of the system (usually a robot) in order to disambiguate any conflicting evidence about the object of interest.

The majority of state of the art techniques in active vision [14, 15, 16] including our POMDP formulation discussed in the previous subsection (1.3.1) share the following idea: one single-view classifier is trained to recognize the type and pose of target objects, whereas a subsequent step uses the inference probabilities to plan the next actions so that conflicting hypotheses are disambiguated. Although intuitive, this approach makes the combination of features from multiple views difficult whereas hypotheses from different views can only be exploited a posteriori (i.e. Bayesian formulations). In addition, their performance heavily relies on the performance of the single-view classifier. However, designing a classifier that can generalize across views is particularly challenging especially when illumination variations or deformations are considered. Another problem in active vision which hasn't been addressed by many state of the art techniques [15, 16], is defining the cost associated with each action.

To cope with the above challenges, we propose *Active Random Forests* [3] which can be considered as an "*active classifier*". The framework is based on standard Random Forests [17] having also the ability to control viewing parameters during on-line classification and regression. The key difference is that the classifier itself decides which actions are required in order to collect information which will disambiguate current hypotheses. As we will demonstrate, this combination of classification and viewpoint selection outperforms solutions which employ these two components in isolation [14, 15, 16]. Furthermore, inference is made using the entire set of captured images, taking advantage of the various feature associations between different viewpoints. The on-line inference and action planning become extremely fast by the use of Random Forests, making the framework very suitable for real-time applications such as robotics. In summary, the main contributions of our framework are:

- **A multi-view active classifier** which combines features from multiple views and is able to make decisions about further actions in order to accomplish classification and regression tasks.
- **A decision selection method** during classification and regression using the powerful voting scheme inherent to Random Forests.

- A method for taking into account the possible **costs of actions**.

Letting the classifier decide the next disambiguating actions introduces much discriminative power to the framework.

We demonstrate our proposed framework in our the challenging problem of recognizing and unfolding clothes autonomously using a bimanual robot (introduced in Sec. 1.3.1), focusing on the problem of best viewpoint selection for classification, grasp point and pose estimation of garments.

1.3.3 Rigid Object Detection, 3D Pose Estimation and Next-Best-View Prediction

Moving from deformable to rigid objects, detection and pose estimation of such objects is a challenging problem arising in many practical applications, like robotic manipulation, tracking and augmented reality. Low-cost availability of depth data facilitates pose estimation significantly, but still one has to cope with many challenges such as viewpoint variability, clutter and occlusions. When objects have sufficient texture, techniques based on key-point matching [18, 19] demonstrate good results, yet when there is a lot of clutter in the scene they depict many false positive matches which degrades their performance. Also, holistic template-based techniques provide superior performance when dealing with texture-less objects [20], but suffer in cases of occlusions and changes in lighting conditions, while the performance also degrades when objects have not significant geometric details. In order to cope with the above issues, a few approaches use patches [6] or simpler pixel based features [5] along with a Random Forest classifier. Although promising, these techniques rely on manually designed features which are difficult to make discriminative for the large range of everyday objects.

Last, even when the above difficulties are partly solved, multiple objects present in the scene, occlusions and distructors can make the detection very challenging from a

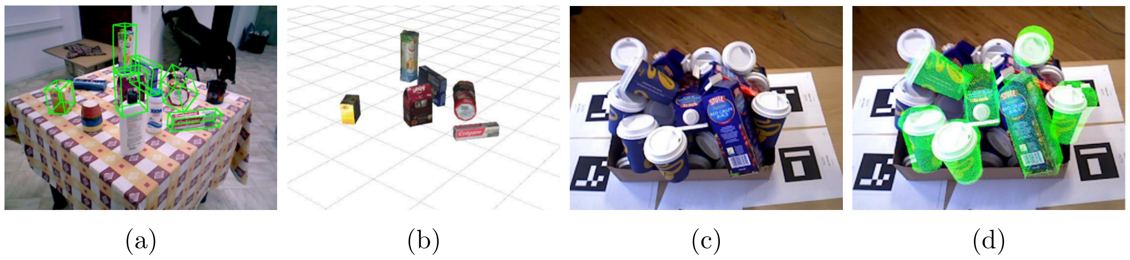


Figure 1.2: Overview of our datasets and example results. a) Example scene of our dataset, b) our framework result on (a), c) Example scene of the bin picking scenario, d) results of our framework on (c).

single viewpoint, resulting in many ambiguous hypotheses. When the setup permits, moving the camera to another viewpoint can be proved very beneficial for accuracy increase. However the problem is how to select the next best viewpoint, which is crucial for fast scene understanding.

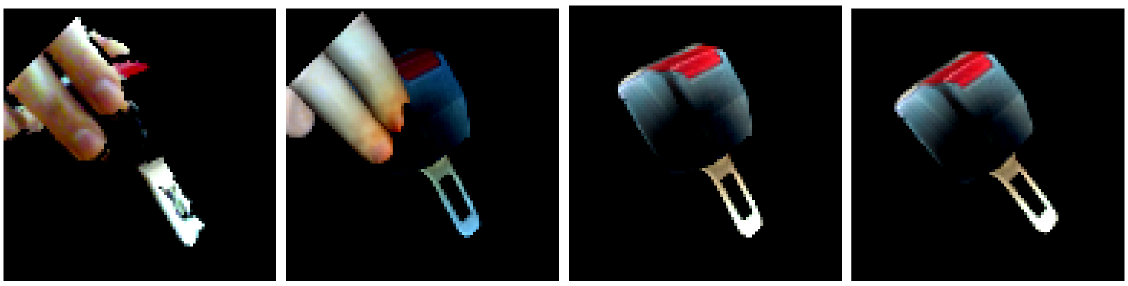
The above observations motivated us to introduce a complete framework for both single shot-based 6D object pose estimation and next-best-view prediction based on Hough Forests, a variant of Random Forest that performs classification and regression jointly [6]. We adopted a patch-based approach but contrary to [20, 6, 5] we learn features in an unsupervised way using deep Sparse Autoencoders. The learnt features are fed to a Hough Forest [21] to determine object classes and poses using 6D Hough voting. To estimate the next-best-view, we exploit the capability of Hough Forests to calculate the hypotheses entropy, i.e. uncertainty, at leaf nodes. Using this property we can predict the next-best-viewpoint based on current view hypotheses through an object-pose-to-leaf mapping. We are also taking into account the various occlusions that may appear from the other views during the next-best-view estimation. Last, for further false positives reduction, we introduce an improved joint optimization step inspired by [22]. To the best of our knowledge, there is no other framework jointly tackling feature learning, classification, regression and clustering (for next-best-view) in a patch-based inference strategy, and doing all by an end-to-end deep network is not trivial.

In order to evaluate our framework, we do an extensive evaluation for single shot detection of various state of the art features and detection methods, showing that the proposed approach demonstrates a significant improvement compared to the state of the art techniques, on many challenging publicly available datasets. We also evaluate our next-best-view selection to various baselines and show its improved performance, especially in cases of occlusions. To demonstrate more explicitly the advantages of our framework, we provide an additional dataset consisting of two realistic, everyday scenarios, shown in Fig. 1.2. Our dataset also reveals the weaknesses of the state of the art techniques to generalize to realistic scenes. In summary, our main contributions are:

- A complete framework for 6 DoF object detection that comprises of a) an architecture based on Sparse Autoencoders for unsupervised feature learning, b) a 6D Hough voting scheme for pose estimation and c) a novel active vision technique based on Hough Forests for estimating the next-best-view.
- Extensive evaluation of features and detection methods on several public datasets.
- A new dataset of RGB-D images reflecting two usage scenarios, one representing domestic environments and the other a bin-picking scenario found in industrial settings. We provide 3D models of the objects and, to the best of our knowledge, the first fully annotated bin-picking dataset.

1.3.4 Siamese Regression Networks

The learned unsupervised features using sparse auto-encoders described in the previous subsection are very efficient in handling different types of objects, but are sub-optimal for the task of object pose estimation, since they are learned without using an objective. Leveraging the discriminative power of convolution neural networks, we are interested in learning end-to-end regression networks, which are able to learn efficient mid-level feature embeddings. The lack of research on Deep Networks for angle regression prove that directly regressing object poses in angle space



(a) (b) (c) (d)

Figure 1.3: Hand-object dataset overview. a) real RGBD image of hand-object. b) synthetic groundtruth from tracker. c) object pose groundtruth, d) our Siamese Regression Network result on (a).

is not trivial, with the objective function appearing to have many local minima. Albeit the recent advances in classification tasks using CNNs, a framework that is able to perform direct regression in angles, while jointly learning discriminative features has yet to be built.

Recent works [23, 24] demonstrated successful results by using siamese networks, which can improve the network learning capabilities by exploiting additional information about the relationship between the training samples. Inspired by this, we present the Siamese Regression Network that enforces a relationship between feature and pose space by applying a novel loss function, which can boost the performance of a regression network layer. Thus, this network is able to perform single-shot end-to-end regression for object pose estimation, without requiring pairs of inputs at testing time. Apart from that, we experimentally evaluate the effect of some other factors that play an important role in successful regression such as feature normalization and batch formation. Our Siamese Regression Network, on the other hand, was shown to learn more discriminative features optimized for our particular problem, as compared to a state of the art feature learning technique using CNNs [25]. Finally, we are interested in handling severe occlusions in the object pose estimation task, a particularly interesting problem constantly arising in real-life applications. However, estimating accurate pose when a significant portion of the object is missing is a very challenging task which drastically degrade the performance of previous arts [25, 26, 5, 27, 28, 29]. We show how our loss function can be easily modified to handle cases of severe occlusions. To evaluate our regressor on such cases, we built our own challenging dataset which demonstrates an object being manipulated by a human hand (Fig. 1.3). Results show that our method can very well handle severe occlusion, reaching accuracy levels of non-occluded objects.

In summary our work offers the following contributions:

- We present Siamese Regression Network which, to the best of our knowledge, is the first CNN-based framework for regressing object poses in angle space.
- We boost the performance of our system by introducing a novel loss function for feature-guided pose regression.
- In turn, we show that pose-guided feature learning results in more discriminative features than the ones of [25] and are optimized for the particular task of 3D object pose estimation.
- We show how our loss function can be adapted to deal with severe occlusions and evaluate our system on a new challenging dataset containing an object captured under severe occlusions. Furthermore, experimental evaluation on a benchmark dataset [26] provides evidence of our system outperforming the state of the art.

1.4 Publications

This PhD work has produced the following publications:

- **Autonomous Active Recognition and Unfolding of Clothes using Random Decision Forests and Probabilistic Planning**
  Andreas Doumanoglou, Andreas Kargakos, Tae-Kyun Kim, Sotiris Malassiotis
  Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA), Hong Kong, China, 2014 *(Best service robotics paper award, sponsored by KUKA)*

- **Active Random Forests: An application to Autonomous Unfolding of Clothes**

  Andreas Doumanoglou, Tae-Kyun Kim, Xiaowei Zhao, Sotiris Malassiotis

  Proc. of European Conference on Computer Vision (ECCV), Zurich, Switzerland, 2014.

- **Folding Clothes Autonomously: A Complete Pipeline**

  Andreas Doumanoglou, Jan Stria, Ioannis Mariolis, Andreas Kargakos, Vladimír Petrík, Libor Wagner, Tae-Kyun Kim, Václav Hlaváč, Sotiris Malassiotis

  IEEE Transactions on Robotics, accepted to appear in 2016.

- **Recovering 6D Object Pose and Predicting Next-Best-View in the Crowd**

  Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, Tae-Kyun Kim

  Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Las Vegas, USA, 2016.

- **Siamese Regression Networks with Efficient mid-level Feature Extraction for 3D Object Pose Estimation**

  Andreas Doumanoglou, Vassileios Balntas, Rigas Kouskouridas, Tae-Kyun Kim

  arXiv:1607.02257

The first three correspond to the work on clothes, with the third one describing the complete pipeline of autonomously folding garments. Our unfolding procedure has been combined with other collaborator's work resulting in an end-to-end autonomous folding procedure. The last two publications correspond to rigid object detection and pose estimation.

1.5 Thesis structure

Thesis starts with an overview of the most significant related work in Chapter 2, including an overview of the Random Forest and Hough Forest framework that would be helpful for the reader to understand our methods in depth. After that, Chapter 2.4 presents the datasets that we built during the thesis in order to train and test our methods. Following, the next chapters correspond to the work related to clothes recognition, pose estimation and active manipulation of garments for the specific task of autonomously unfolding clothes and covers our contributions described in Sec. 1.3.1 and Sec. 1.3.2. Chapter 3 describes our first work on clothes recognition and grasp point detection using Random Forests and Hough Forests. Chapter 4 presents our novel framework *Active Random Forests* applied to active manipulation of clothes in order to enhance type, pose and grasp point detection performance. Finally, in the Appendix A we provide an overview of our collaborative work for project CloPeMa about the full pipeline for autonomously folding a garment starting from a random initial state.

After that, we present our work on rigid object detection, 3D pose estimation and next-best-view prediction mentioned in Sec. 1.3.3 and Sec. 1.3.4. Chapter 5 presents our patch-based unsupervised feature learning technique based on Hough Forests along with our novel next-best-view prediction framework built upon our trained Hough Forest. Finally, Chapter 6 introduces our Siamese Regression Network and its efficient feature embeddings for 3D object pose estimation. These works are included in our last two publications mentioned above. Last, for clarity, each chapter contains the related work and all experiments that are related to the methods described.

Chapter 2

Related Work, Backgrounds & Datasets

2.1 Representative & Most Relevant Literature Work

Regarding clothes unfolding, the state of the art is the work of Cusumano-Towner et al. [12]. They use a sequence of moves to bring a garment in a predefined state, they estimate the state of the garment using a simulator in an HMM framework. The unfolding is completed on a table, where they plan a sequence of grasps in order to bring it to the desired unfolded configuration. Fig. 2.1(a) and 2.1(b) shows the contour estimation of the method and a shorts that were unfolding using the PR2 robot. The results were promising when this method had been applied to small-sized or baby clothes. Another work from the same group (Maitin-Shepard et al. [13]) was the first to accomplish the complete task of folding towels autonomously. The method was based on corner detection and grasping with proper manipulations, however it required about 20 minutes in order to fold one towel. Fig. 2.1(c) and 2.1(d) show the initial and the final configuration of the towels after folding.

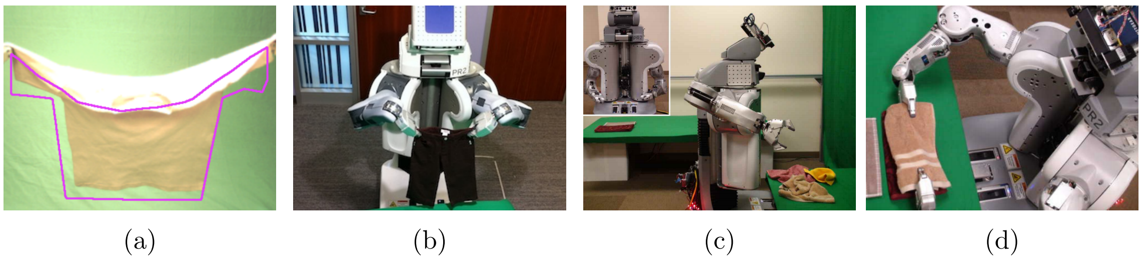


Figure 2.1: a-b) Contour estimation and Shorts unfolded as presented in [12]. c-d) Pile of towels and a folded towel as presented in [13]

About object detection and 3D pose estimation there are few recent, well known and representative techniques. One of them is LINEMOD [20], a template matching technique that introduces an object representation for fast detection that is prune to clutter. LINEMOD also provided a dataset that has been widely used by researchers to test their method against clutter. Tejani *et al.* [6] used the LINEMOD representation to train a Hough Forest which proved to be more efficient. Hough Forests have been also used by Brachmann *et al.* [5] for estimating 3D object coordinate and class labeling, making use of the simple pixel test as split function in the tree nodes. All these methods however, are not able to tackle occlusions efficiently as we will see later, compared to our methods. Apart from these standard techniques, few tried to use Deep Learning to solve the 6DoF problem. Wu *et al.* [30] presented a deep belief network that used 3D convolutions to train a neural network. Wohlhart *et al.* [25] tried to learn discriminative feature descriptors by training a CNN over pairs and triplets of training samples and used Nearest Neighbor method to determine the object label and pose from templates. On the other hand, our work on Deep Networks tackles the problem of regressing the object pose directly, which is more efficient for real time applications.

2.2 Random Forests & Hough Forests

Many algorithms developed during this thesis used Random Forests [31] as their major classifier. Although they have been introduced in 2001 [31], they have recently proved to be a powerful and versatile tool for many computer vision tasks. Partic-

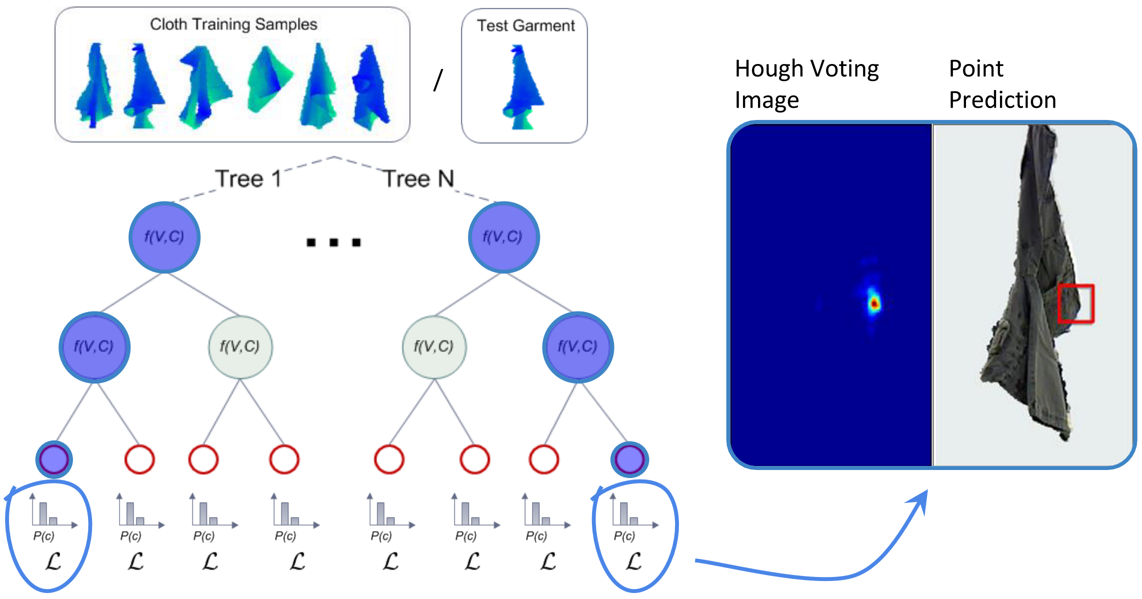


Figure 2.2: Hough Forest example demonstration on clothes training. On left there is the ensemble of trees that store the class distribution along with the voting vectors. On the right there is the Hough voting space and the corresponding estimation, that is the position of the estimated grasp point.

ularly, two works have successfully used them and inspired a research direction for others as well. The first is the work of Shotton et al. [32] who used random forests for human pose estimation from single depth images. The other work is published by Gall et al. [33] who successfully used Hough Forests for pedestrian detection, a Random Forest framework with the hough voting property. In this paper we demonstrate that the above machine learning techniques can produce robust results with challenging data such as various highly deformable clothes. Below we will describe the theory behind both frameworks so that the reader can gain a wide understanding of them, which is essential in order to continue reading the rest of the thesis.

Random Forest is a collection of Random Decision Trees of which the outcome is averaged to produce a better result. This idea is based on the observation that many week classifiers (Random Decision Trees), if combined, can produce a more accurate result. Thus, a Random Decision Tree can be regarded as a week classifier which is generally trained independently from the other trees. Each decision tree is able to perform classification or regression. Assume that each tree is trained over a subset V of the training samples, and each sample is represented by C feature channels.

Initially the tree contains only the root node which receives the full set V. The root node and each subsequent node of the tree is trained in the same way: we need to find the best split function $f(V, C)$ that produce the best split over the data which maximizes an *objective function*. The data is always split into two subnodes, the left and the right. The intuition behind this split is that we want to separate our training set in such a way, so that the leaf nodes (the last nodes of each path in the tree) contain samples only from one class in case of classification, or samples close to the same mode in the space of desired outcome in case of regression. When this is achieved, then a test sample can follow the proper path in the tree and end up to a leaf with other similar samples from training so that we can easily classify it or estimate a regression result.

A split over subset V is performed using a split function f over some values of feature channels C. For a particular sample v, if $f(v, C) < t$ where t a threshold, then the sample goes to the left child node, otherwise to the right. In order to evaluate how good a split is, the most common objective is the entropy of the two children nodes. When the entropy is low, it means that the produced nodes are more *pure* and *less uncertain* (i.e. more confident) about their prediction. In case of classification the entropy is defined as:

$$H = \sum_{c \in Classes} p(c) log p(c) \tag{2.1}$$

In case of regression the entropy can be defined over a continues Gaussian distribution estimated from the samples. For more details you can refer to Sec. 4.3. The best split function is selected over a random set of different split functions, as the one that minimized the entropy of the children nodes. One of the main contributions of [32] is that very simple split functions, such as pixel value subtraction of 2 different pixel locations in an image, can perform significantly well when there is sufficient training data.

A node does not split further when there are not sufficient training data left, or a maximum tree depth has reached. In this case the node becomes a leaf node.

When there is no node that can be further split, the training terminates. The leaf node can store information about the training samples that arrived in it. For classification, they can simply store the class distribution $P(c)$ of the classes and for regression, they usually store one or two the modes of the regression space. Such information can be aggregated from all trees in order to estimate the final result. As was described earlier, during testing, a test sample can traverse down the tree, evaluating each split function of nodes in its path, and end up to a leaf node which is used to estimate the final result. Fig. 2.2 demonstrates the idea, using clothes as training and testing data.

It should be mentioned that randomness plays a major role in the performance of the Random Forests. In order for the week classifiers to perform well when averaged, they need to be as uncorrelated as possible. Such reduction in correlation is achieved by randomly selecting a subset of training samples to train each tree (usually by using bootstrap), and randomly selecting split functions to evaluate in each node.

Hough Forests are a simple yet powerful extension of the standard Random Forests. Early works, for example Mikolajczyk et al. [34], used a tree as a hierarchical codebook to cluster features, while [33] used binary trees for clustering and Hough Voting for localizing the center of the objects. The idea is that the training samples stored in the leaf nodes can vote in a Hough Space for some desired variable, given that such information is provided from the training set. For example, in [33] samples voted for the position of the center of each pedestrian, while in our problem of unfolding clothes, samples voted for the location of the desired grasp points as shown in Fig. 2.2. In the figure, the information of the location of the desired point of each training sample is stored in vector $\mathcal{L}$. After voting, the Hough space usually contains some modes that can be extracted using non-maximum suppression and can be regarded as the hypotheses of the true value of the variable we want to estimate. Last, the dimensionality of the Hough space can take any value depending on the problem. In the examples that we mentioned earlier the dimensionality was 2, but in the problem of object detection and pose estimation, we used a Hough space of 6 dimensions (Chapter 5).

2.3 Defining the Object Pose in 3D Space

One of the main problems that this thesis tries to solve is the pose estimation of objects. Therefore it would be helpful for the reader to define what is object pose for both rigid and deformable objects.

In order to define a position of an object in 3D space we need to define at least 6 variables. The first 3 variables, usually referred to as x, y, z, define the object location in 3D space. However this is not enough, since the object might be arbitrarily rotated. Therefore we need another 3 variables to measure the relative rotation with regard to the three principal axes. In Euler angles, the rotation around the x axis is called *roll*, around the y axis is called *pitch* and around the z axis is called *yaw*. The pose of a rigid object is defined as the position of this object in 3D space, relative to a predefined position of the same object. In our case, this predefined position corresponds to the position of the object as appears in its CAD model that is used as ground truth. This ground truth pose can be set arbitrarily from the designer of the object or the method used to capture such model.

For deformable objects however the above definition cannot apply. Therefore, the way we define the pose of a garment applies only to our problem and it cannot be generalised in any way to other deformable objects, or even to clothes if the problem is different. Our problem is how to drive the robot gripper to grasp our desired grasp point on a garment. Ideally, there should be a 3D vector that has its origin on the grasp point, and its direction corresponds to the direction that the gripper should approach it in order to grasp it correctly. In Fig. 2.3 the location of the grasping point is the end-point of vector $\mathbf{g}$, and the grasping direction is represented by vector $\mathbf{p}$ (in fact the gripper should approach in the opposite direction of $\mathbf{p}$). If we can detect, along with the location of the grasp point the direction of this vector, we

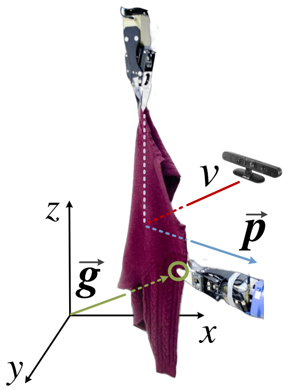


Figure 2.3: Pose estimation as defined on clothes. Vector **g** represents the location of the grasp point and vector **p** the grasping direction of the gripper.

can plan the motion of the gripper accordingly. Our final estimation should contain again 6 variables, 3 for the location of the grasp point, and 3 for the direction of the vector in 3D space. As we see, there is an implicit analogy in representation between the rigid object pose and the pose of the grasp point of a garment, thus we defined it as *garment pose*. However, as mentioned above, this definition cannot apply to other general deformable objects.

Furthermore, it should be now clear why the problem of object pose estimation is sometimes referred to as *object 6D pose estimation* or 6 degrees of freedom (6DoF) estimation. There are 6 variables (degrees of freedom), or a 6 dimensional vector that needs to be determined in order to define an object pose in 3D space.

2.4 Datasets

2.4.1 Introduction

During this thesis, in order to train and evaluate our methods it was necessary to create our own datasets. This was due to the fact that either no such dataset existed in the literature, or the existing ones didn't cover the whole range of challenges that are inherent to the problem of interest. In this chapter we present three different datasets that were created specifically for each of the problems we tackled: one dataset about clothes unfolding, one dataset about rigid object localization and pose estimation, and a hand-object interaction dataset. In the next sections we describe each dataset in detail and show example images from each one.

2.4.2 Clothes Unfolding Dataset

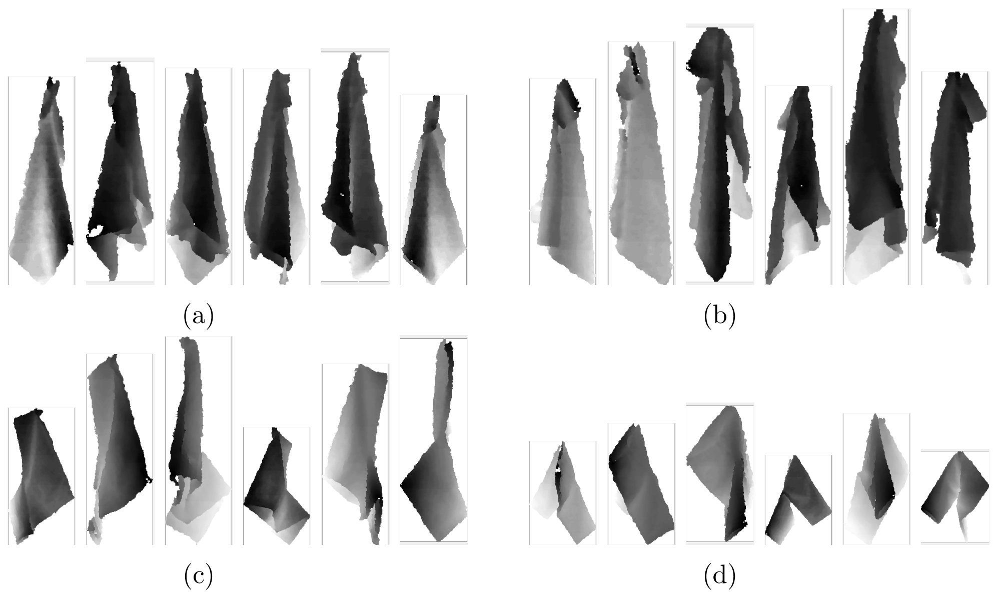


Figure 2.4: Clothes Unfolding Dataset Examples. a) Shirt hanging from 1st grasp point, b) shirt hanging from 2nd grasp point, c) shorts hanging from 1st grasp point, d) shorts hanging from 2nd grasp point. For each case we show 6 of the 40 different viewpoints.

The purpose of this dataset was to train our robot to recognize various types of garments as well as detect certain grasp points on them in order to unfold them. The main challenge of this task is to build such a dataset to generalize to various clothes and sizes, which exhibit very high inter- and intra-class variations. Our dataset comprises a training and testing set. The training dataset contains 24 regular-sized clothes of various sizes and fabric types, 6 of each category. As we will see in Chapter 3, in order to reduce the configuration space of clothes hanging freely in the air, we prefer to grasp the lowest points first, which was also adopted in [9]. Therefore, each garment was grasped 20 times from each lowest point and 40 images were captured while the garment was rotating covering the viewpoints of all 360 degrees. The final database contains 28,800 depth images (since we are not interested in color which has much more variation, we do not keep the RGB images). Image labelling was done manually using fiducial markers over key points to facilitate the task. The labeling includes the garment type, the grasp point coordinates (x, y) in the image, and the grasp direction vector, as described in Chapter 4. Testing was based on a dataset containing only novel items (not in the training dataset). The clothes used for testing were 3 per category (12 total). Example images of the training and testing set are shown in Fig. 2.4. Since the images of the dataset are cropped to the bounding box of the garment, please check Fig. 3.10 for a correspondence to real unfolding images with the robot.

2.4.3 Object Pose Estimation Dataset Description

Our proposed dataset consists of two usage scenarios. One is related to domestic environments, where everyday objects are placed on a kitchen table. The second depicts a bin-picking scenario mainly found in industrial settings where a robot should pick objects successively from a bin, which contains many stacked objects of the same or different categories. In the following subsections we describe each usage scenario.

Usage Scenario 1

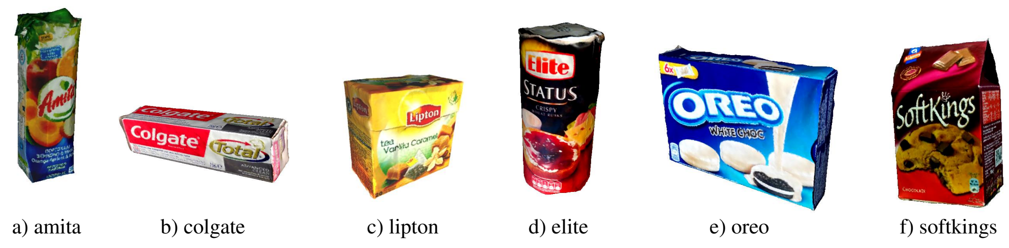


Figure 2.5: Dataset Objects. Images show renderings of the 3D models of the objects used in training.

The objects of this scenario are shown in Fig. 2.5. We collected six objects usually bought from a supermarket, and captured their 3D models using 123D Catch from Autodesk [35] (with the Android application on a smartphone). Fig. 2.5 shows real

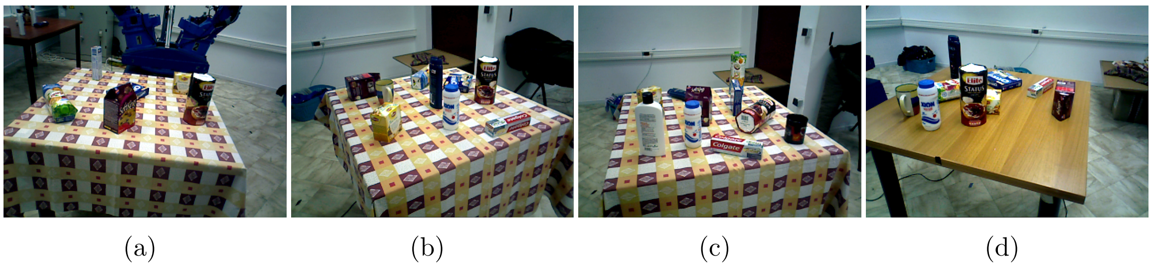


Figure 2.6: Examples of test images

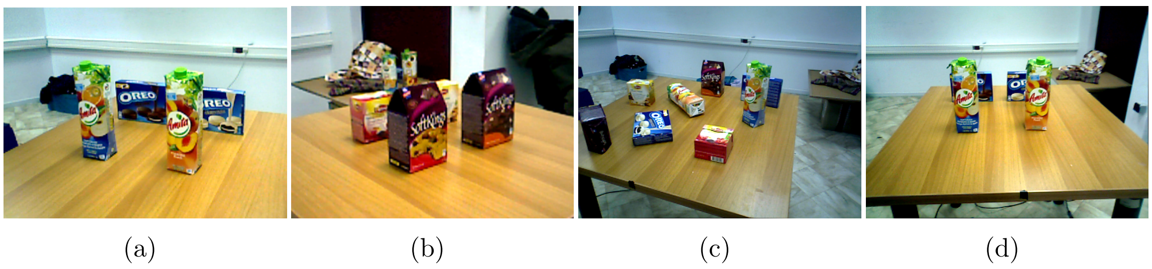


Figure 2.7: Examples of test images for evaluating active vision methods

renderings of the 3D models used for training. We can see that the quality of the models created by this Structure From Motion solution is much better compared to Kinect Fusion [36] (Fig. 2.8(d),2.8(e)) for textured, non-glossy objects. Furthermode, these models capture the complete object including the bottom part so that they can be used to detect objects lying on the table in any possible orientation.

Fig. 2.6 shows examples of the test images of our dataset. We created a variety of different scenes, with and without a table top, including sometimes objects not present in the training set. We capture RGB-D images covering 360 degrees around the table from two different heights for various object arrangements. These simple arrangements often create occlusions, which combined with the table top and the out-of-training objects make 6 DoF methods produce many false positives. In addition, we provide full annotation of the test images for the objects in the training set manually annotating the objects in the scene and estimating the camera movement (using our own tools), to avoid placing markers that makes the scenes look artificial. This scenario contains 6 different scenes with 170 test images in total.

For evaluating active vision methods, we created some additional scenes shown in Fig. 2.7. We added objects that differ with the existing ones only in some parts

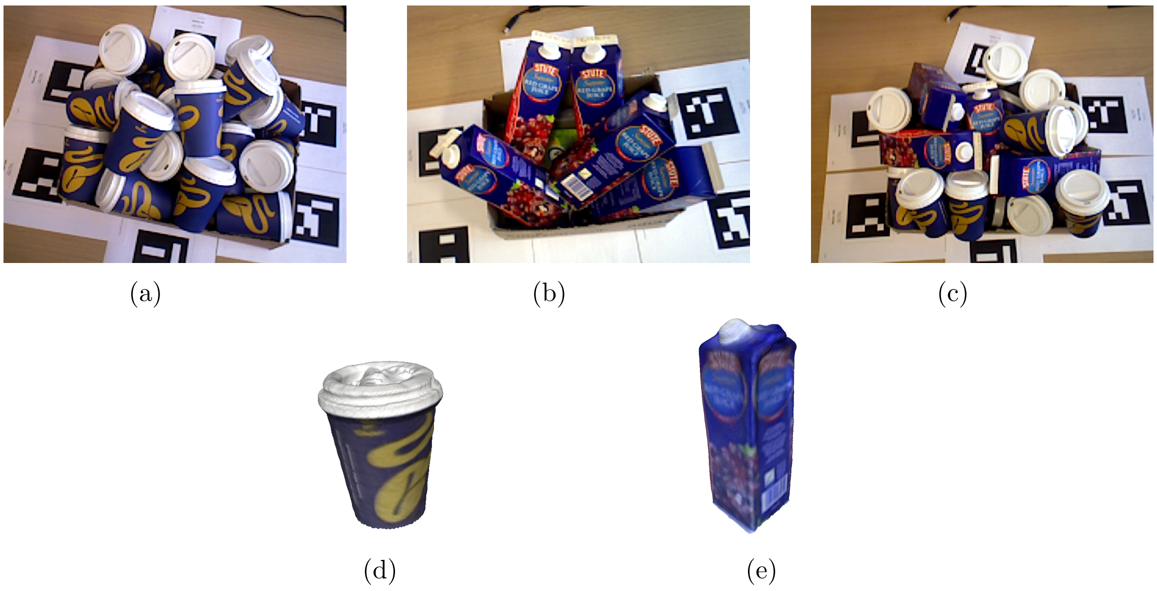


Figure 2.8: a) test image with a bin of coffee cups, b) test image with a bin of juices, c) test image with both objects, d) coffeecup 3D model, e) juice 3D model

of the object (for example, oreo with white and dark chocolate). Thus, the next-best-view of the camera should ideally focus on these distinctive parts and one can qualitatively evaluate an active vision method. Also, we have arranged the objects in such a way so that objects are occluded in viewpoints where the distinctive parts should have been observed (Fig. 2.7(d)). Such situations can be resolved by the refinement step of our active method described in Chapter 5. We created 6 additional scenes with a total of 181 test images.

Usage Scenario 2

The second scenario in our dataset, named as **bin-picking scenario**, is shown in Fig. 2.8. We used two objects, a coffee cup and a juice, and created three different scenes, two containing each object separately (Fig. 2.8(a)-2.8(b)) and one containing both objects (2.8(c)). This is a very challenging scenario with much occlusion, while stacking similar objects makes it hard to combine features extracted from different locations and estimate an object pose. To compare with the state of the art in this challenging problem, we chose the objects of [6] that showed the best performance on their dataset and also used the same 3D model given by the authors captured with

Kinect Fusion [36] (Fig. 2.8(d)-2.8(e)). We provide full annotation of the objects that are visible in each scene. The three bin-picking scenes contain 183 test images in total.

Our datasets can be found on our laboratory's website in the corresponding section.

http://www.iis.ee.ic.ac.uk/ComputerVision/

Dataset Structure

For all the objects in the dataset we provide 3D CAD models in the form of plain-text ply files. Each test scene is contained in a different folder. The images of each scene have the form *rgbXX.png* and *depthXX.png* that contain the rgb and depth image respectively. Also there are annotation txt files of the form *object_name_XX.txt* which contain the ground truth pose of each object present in the image. These txt files contain the homogeneous 4x4 rotation matrix of the object pose in right-handed system, with the z axis being negative as we move from the camera to the objects. The x and y axis are parallel to the image axes (going to the right and up respectively) and the centre of the coordinate system is at the centre of the image with $z = 0$. When an object appears more than once in an image, then the annotation file starts with a number N corresponding to how many objects of this type appear in the image. Then N rotation matrices follow, 4 lines per matrix.

Chapter 3

Autonomous Recognition and Unfolding of Clothes

3.1 Overview

Our work is about solving the problem of doing the laundry autonomously and in this part we are working on folding clothes. Our interest is focused particularly in the first part of the procedure, which is the unfolding of an article of clothing. Starting from a crumbled initial configuration, we want to recognize the article and then bring it into an unfolded state so that it is ready for folding. Fig. 3.1 shows an example of our unfolding procedure. One of the key challenges in clothes perception and manipulation is handling the variabilities in geometry and appearance. These variabilities are due to the large number of different configurations of a garment, self-occlusions and the wide range of cloth textures and colors.

As we will see in the next section (Sec. 3.3) researchers mainly resorted to model-driven approaches in order to recognize and manipulate garments [10, 11], while only recently scientists were able to completely fold an article of clothing starting from a crumpled initial configuration [12, 13, 10]. Such approaches, however, exhibit slow performance and difficulty in generalizing on a variety of shapes and materials.

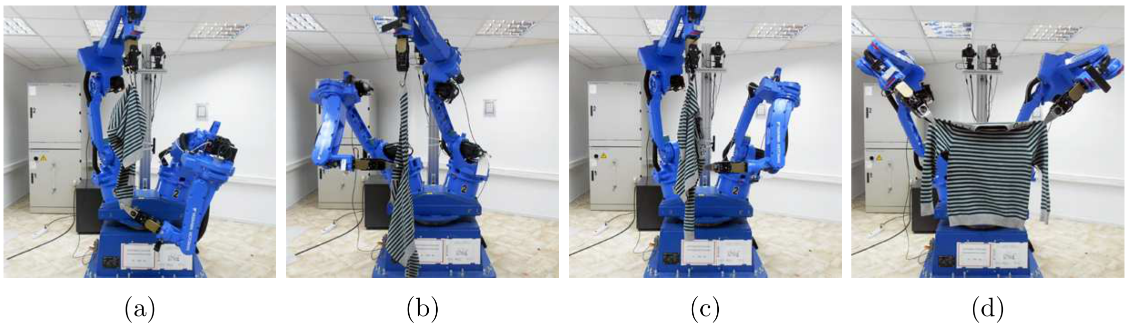


Figure 3.1: Robot autonomously unfolding a shirt. a) Grasping lowest point. b) grasping 1st grasp point. c) grasping 2nd grasp point. d) final unfolded configuration

On the other hand, we propose a data-driven approach for clothes recognition and unfolding. We are able to recognize the type of a garment only from raw depth data, using Random Forests and very simple features. We then detect a pair of key-points such that the garment will naturally unfold when held by these two points (Fig. 3.1). Point estimation is based on Hough Forests (for more details, refer to Sec. 2.2). On top of that, we propose an active manipulation (perception-action) approach based on POMDP that accounts for uncertainty of the forests output, further increasing the accuracy. Our system favorably compares to the state of the art: it requires almost half the movements, operates faster (about 40% relative reduction in execution time) and, to our knowledge, it is the first to autonomously unfold regular-sized clothes.

In the following sections we first present the related work, and then we analyze the garment recognition method, the grasp points detection and the active planning. Last, we experimentally evaluate our methods using a dual-arm robot.

3.2 Assumption and Constraints

Our solution on garment unfolding was indented to be applied and evaluated on a real robot. Before we started solving the problem, the robot for the evaluation and the cameras were already provided, enforcing us to make some assumptions about how the problem can be solved according to the capabilities of our robot. We found out that the working space of the robot, i.e. the maximum area in which the robot is able to manipulate objects (clothes in particular) is larger if we manipulate the garments in the air than on a table. Therefore our solution was designed in this direction. It turned out that this had also the advantage of unfolding garments faster since they can be easier flattened while hanging in the air. On the other hand, due to the same constraint we could not work with very long garments or sheets. Apart from that, there were some other factors that constraint the choice of the garments we dealt with. We chose only the main types of garments (shorts, shirts, trousers, T-shirts, towels) whose shape is somewhat well defined, as opposed to various other clothes such as skirts of different shapes, jackets that can be opened or closed or garments of very thin materials such as sink. The first reason was that we preferred to use cheap depth cameras (Asus Xtion) that can very quickly provide 3D data at the expense of low quality and resolution. This means that we could not work easily with thin materials or with garments having many wrinkles and high degree of deformation since we would not have the expected resolution in depth. Apart from that, for such garments it is also very hard for the gripper we had on our robot to grasp points correctly, for example grasping a point on the front piece of cloth without grasping the back one, in case they are touching each other. Last, in order to generalise well among various types of garments, one needs to collect a large dataset (hundreds or even thousands of garments) covering as many different shapes as possible, something which was out of the scope of our research. Last, a final assumption that we made in order our unfolding procedure to work is that a garment has a clear lowest point when hang arbitrarily in the air. However this assumption almost always hold even for garments that are not in our dataset due to the fact that they are designed for humans and should contain certain corner points.

3.3 Related work

The first attempts in clothes manipulation have been made by Hamajima et al. [37] who tried to detect and grasp hemlines aiming to unfold clothes and Kaneko et al. [38] who used basic 2D shape analysis to recognize different types of clothing. Osawa et al. [9] dealt with the recognition task and were the first to introduce iterative grasping of the lowest point of a garment, converging to a finite set of possible configurations, an idea adapted to our work. The classification was based on template matching of the final state of the garment after re-grasping the lowest point several times. While they also unfolded the garment using the same procedure, it can be mostly considered as flattening rather than unfolding.

Later, researchers focused on clothes features and characteristics. Triantafyllou et al. [39] used 2D images to identify different types of corners inside a piece of fabric lying on a table, aiming to unfold it by a robotic arm. Willimon et al. [40] proposed an unfolding method based on clothes features for estimating and moving certain grasping points to gradually flatten a piece of clothing placed on a table. This method required a large number of movements to fully unfold a towel while performance on other types of clothes was not mentioned. The same authors [41] proposed a recognition method using features from two binary silhouettes taken from two vertical viewpoints and re-grasping the garment 10 times to improve accuracy. However the overall procedure became very time consuming. In the final work of the same authors [42], a multi-layer classifier was developed with features extracted from depth images. Results were not very accurate except for the case of considering only shirts, socks and dresses.

Closer to our approach is the work of Kita et al. [43] [44]. It is based on fitting a deformable model to the current pose of a shirt hanging from a random point. Using the model, they are subsequently able to estimate the next grasping point (e.g. the shoulders of a shirt) in order to unfold it. The authors show promising results, that are however limited to a single shirt.

Recently, Maitin-Shepard et al. [13] was the first to complete the whole task of folding a towel using the PR2 robot. The algorithm was based on a corner detector for appropriately grasping the towel, however the amount of time required for completing the task makes the approach intractable in a practical scenario. Bersch et al. [45] also used the PR2 robot to autonomously fold a T-Shirt with fiducial markers, again performing in a prohibited running time. The state of the art in unfolding articles of clothing is the work of Cusumano-Towner et al. [12]. Adopting the technique of sequentially grasping the lowest point they can estimate the resulting state of the garment using a cloth simulator in an HMM framework. After putting the garment on a table, they plan a sequence of grasps in order to bring it to the desired unfolded configuration. The method was applied on small or baby clothes and results were promising. However, applying this method to regular-sized clothes requires large working space (table) and long manipulators attached on a moving base for better results. In contrary, our approach does not require a table for unfolding and no movement of the base of the manipulators is necessary. Also, we have further reduced the number of moves a robot should make in order to unfold an article of clothing.

3.4 Clothes Recognition

The recognition is based on Random Forests, first introduced by Breiman et al. [46] as a method of classification and regression. They achieve state of the art performance compared to other classifiers like SVM [32] [33] [47] while they provide very fast inference appropriate for real-time applications.

To define our problem, we first assume that an article of clothing has already been isolated from a pile of clothes and the robot grasps it from a random point. The space of possible configurations of a garment hanging randomly is very large, therefore we reduce it by grasping the lowest point [9]. Fig. 3.2 shows the possible lowest points

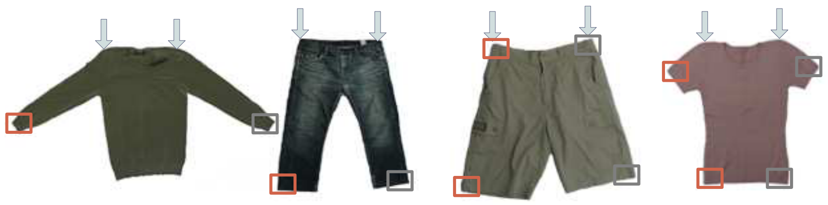


Figure 3.2: Possible lowest points. Gray squares are the symmetric points of the red ones. Arrows show the desired grasping points for unfolding.

of four types of clothes considered in this paper: *shirts, trousers, shorts* and *T-shirts*. There is only one possible lowest point for shirts and trousers and two possible lowest points for shorts and T-shirts without counting the symmetric ones. Our classifier is able to distinguish between both garment types and hanging points. Thus, there are six classes defined as $\{Shirt, Trousers, Shorts_1, Shorts_2, Tshirt_1, Tshirt_2\}$ where subscripts $1, 2$ indicate the different lowest points.

A set of trees is trained over a database of depth images captured from clothes of all the six classes. Each training sample is a pair $(\mathbf{I}, c)$ where $\mathbf{I}$ is a vector containing the depth image and c is the class of the garment labelled manually. Each tree is trained over a randomly chosen subset of the initial training set. At each node, a set of tests are randomly generated with each test containing the following parameters:

- $C_i, \quad i \in \{1, 2\}$: the channel used.
- $\mathbf{V}$: a set of random vectors indicating random positions in the image.
- $f(\mathbf{V}, C_i) > t$: a binary test over the set $\mathbf{V}$ and channel C_i using threshold t

We have used two different channels. Channel C_1 corresponds to the *depth values* as captured from the sensor filtered by a bilateral filter and channel C_2 is the *mean curvature* H calculated from the depth data filtered by an average filter. The mean curvature at a point on a surface is defined as:

$$C_2 = \frac{EN + GL - 2FM}{2(EG - F^2)} \tag{3.1}$$

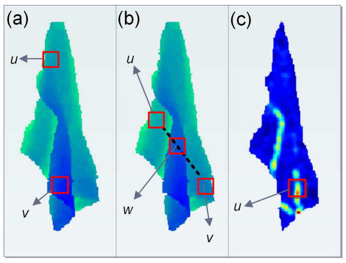


Figure 3.3: Binary tests: a) 2 pixel test in depth channel, b) 3 pixel test in depth channel, c) 1 pixel test in curvature channel.

where E, F, G and L, M, N are the First and Second Order Fundamental Coefficients respectively evaluated on the point [48]. Vectors in $\mathbf{V}$ are normalized to the width and height of the bounding box of the garment for scale invariance so that $v_x, v_y \in [0, 1], \mathbf{v} \in \mathbf{V}$.

Three different types of binary tests were used:

- Two pixel test in the depth channel: $\mathbf{V} = \{\mathbf{u}, \mathbf{v}\}$, $f(\mathbf{V}, C_1) = d_u - d_v$, where d_x is the depth value at position $\mathbf{x}$.
- Three pixel test in the depth channel: $\mathbf{V} = \{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$, with $\mathbf{w}$ being a random point on the line between $\mathbf{u}$ and $\mathbf{v}$, $f(\mathbf{V}, C_1) = (d_u - d_w) - (d_w - d_v)$.
- One pixel test in the curvature channel: $\mathbf{V} = \{\mathbf{u}\}$, $f(\mathbf{V}, C_2) = |c_u|$ where $|c_u|$ is the absolute value of the curvature at position $\mathbf{u}$.

Tests are illustrated in Fig. 3.3. Those simple features are extremely fast to compute and their combination along the path of the trees delivers high discriminative power. Furthermore, we have experimented with other types of features as well, such as HOG or gradient feature channel, but none gave any boost in performance, while at the same time being slower to compute. Pixel tests are not restricted inside a patch as in [33] and reveal global surface characteristics. At each node a random set of tests is generated and the best is chosen as the one that minimizes the Shannon Entropy of the samples at each child node, which is defined as:

$$H_{entropy} = \sum_{i=1}^{N_c} -\frac{N_i}{N} \ln(\frac{N_i}{N}) \qquad (3.2)$$

N_c is the number of classes, N_i is the number of samples of class i and N is the total number of samples reached a child node. We declare a node as leaf and stop splitting it further when a minimum number of samples reached the node or the tree has grown to a maximum allowed depth.

Inference about a previously unseen item of clothing is made by traversing its depth image towards the leafs in every tree in the forest, going left or right according to the binary test f assigned to each node. The class of the sample will be the dominant class of the average class distribution of the leaf nodes reached. We should mention that the inference time of a tree in the forest is $O(logD)$ where D is the depth of the tree and therefore is very fast in real-time recognition.

3.5 Grasp Point Estimation

Having recognized the garment, the objective is now to grasp it from two certain key-points in order to unfold it. Figure 3.2 shows the desired grasping points (arrows) for the four types of clothes we used. Our grasp point estimation is based on Hough Forests[33]. The idea is similar to random forests but with an additional property. Each training sample, apart from the label, contains some extra information which is in our case a vector containing the position of one of our desired grasping points. Thus, a training sample is now a triplet $(\mathbf{I}, c, \mathbf{p})$ where $\mathbf{p} = [p_x, p_y]$ is the position of the grasping point on the image $\mathbf{I}$. Coordinates p_x and p_y are normalized to the width and height of the bounding box of the garment for scale invariance so that $p_x, p_y \in [0, 1]$. When the grasping point is not visible, $\mathbf{p}$ is undefined and not used. A separate Hough Forest is created for each type of garment, so the classes now become two: $c_g = 0$ represents images where the grasping point is not visible and $c_g = 1$ represents images where the grasping point is visible.

The binary tests used are the same as in clothes recognition with the difference that two objective functions should now be minimized for test selection: minimizing the uncertainty about the classes and minimizing the uncertainty about the location of the grasping point of the samples in each node. The uncertainty of the classes is again measured using the Shannon Entropy:

$$H_{entropy} = \sum_{c_g \in \{0,1\}} -\frac{N_c}{N} \ln \frac{N_c}{N} \tag{3.3}$$

where N_c is the number of samples of class c and N is the number of samples reached the node. To measure the uncertainty of the location, the following quantity was used:

$$D = \sum_{samples} d(\mathbf{p_s}, \mathbf{p_M}) \tag{3.4}$$

where d is the Euclidean distance, $\mathbf{p_s}$ is the vector of a sample and $\mathbf{p_M}$ is the average vector of all samples of a node.

While training, leaf nodes store the distribution $P(c)$ of the classes along with a list $\mathbf{L_p}$ of all the location vectors of the samples reached them. When a previously unseen image traverses the Hough forest, the location vectors stored in the leaf nodes will vote for the grasping point location. These votes are accumulated into a *Hough image* and the grasping point location is estimated as the point where the concentration of votes is high (Fig. 3.4). We estimate this point by picking the maximum of the Hough image after Gaussian filtering. This can also be done using the MeanShift algorithm. The localization of the grasping point only occurs when the class recognized is 1, i.e. the grasping point is visible. After a grasp point is detected, the surface orientation in its vicinity is estimated by locally fitting a plane. This is used to create a valid grasp by moving the gripper perpendicularly to the estimated direction.

Each Hough Forest is trained for a certain garment type in order to localize only one grasping point at a time. When this point is grasped, another Hough forests is used to estimate the second one and complete the unfolding. Therefore, we have trained several Hough Forests for every occasion, i.e. for clothes hanging from the

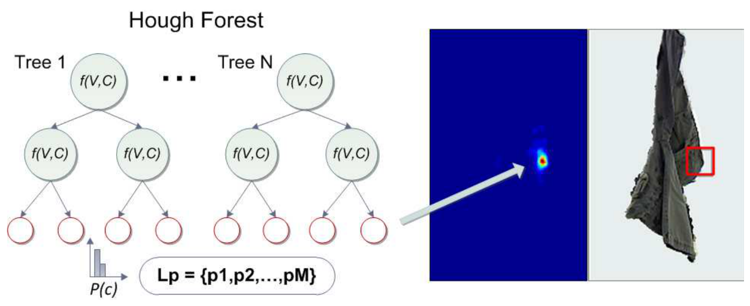


Figure 3.4: Hough forest and grasp point estimation from Hough Image

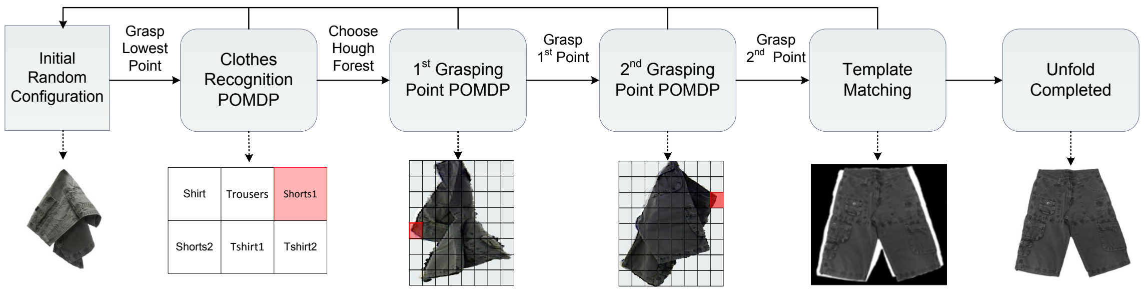


Figure 3.5: Block diagram of the unfolding procedure

lowest or from one desired grasping point. The decision about which Hough forest should be used, is based on the recognition result.

3.6 Probabilistic Action Planning

Although the one-frame recognition accuracy of our Random Forests classifier is statistically high, as we will see in Experiments Section 3.7, there are some viewpoints of clothes where their type is hardly discernible. In order to eliminate the possibility of erroneous classification, we introduce an active recognition scheme. Furthermore, we want to make our system insensitive to noisy point estimations mainly caused by the noisy depth input, introducing an active point estimation scheme as well. The idea is that the robot will rotate the garment around the gravity axis until the uncertainty about recognition or point estimation is minimized. Instead of exhaustively searching over all viewpoints we employ a probabilistic framework that will select the best action policy jointly minimizing the uncertainty and cost of manipulation.

A widely used probabilistic framework is the Partially Observable Markov Decision

Processes (MDP) which are capable of modelling the uncertainty about the current state and can find an optimal policy over the so called *belief state*. While other probabilistic planning approaches have been proposed [49] [50], we have adopted the POMDP framework because having only few states, our problem can be efficiently solved in reasonable time [51] while we take advantage of the representation power and ease of use of the framework. POMDPs have been also used in clothes manipulation by Monso et al. [52] who tried to isolate articles of clothing from a pile.

Fig. 3.5 shows the block diagram of the complete unfolding process. We have developed two different kinds of POMDPs, one for recognition and one for grasp point estimation described below. If one of those sub-tasks cannot be accomplished, the robot returns to its initial configuration by grasping the lowest point of the garment and thus enters a loop until it becomes unfolded.

3.6.1 Active Recognition

Our proposed POMDP is a tuple $(\mathbf{S}, \mathbf{A}, \mathbf{O}, \mathbf{T}, \mathbf{P}, \mathbf{R}, \gamma, \mathbf{b_0})$ where

- $\mathbf{S}$ is the set of states.
- $\mathbf{A}$ is the set of actions.
- $\mathbf{O}$ is the set of observations.
- $\mathbf{T}$ is the conditional transition probabilities.
- $\mathbf{P}$ is the conditional observation probabilities.
- $\mathbf{R}$ is the reward function over the actions and states.
- γ is the discount factor of rewards over time.
- $\mathbf{b_0}$ is the initial belief state.

The states $\mathbf{S}$ in the recognition phase are six and correspond to the six classes used in the classification $\mathbf{S} = \{S_1, S_2.., S_6\}$. The set of actions is $\mathbf{A} = \{A_{rotate}, A_1, A_2, ..., A_6\}$ where A_{rotate} means that the robotic gripper rotates the hanging garment by a degrees to take another observation, while $A_1...A_6$ is the final recognition decision being at state $S_1...S_6$ accordingly. The observations are collected from the Random Forests classifier and contain the inferred class c_{in} of the garment along with the probability $P(c_{in})$ from the averaged distribution of the leaf nodes. $P(c_{in})$ takes values in the interval $[0, 1]$ but we quantise it into five equally spaced bars for reducing the observation dimensionality. Thus, there are 30 observations, five probability bars for each of the six classes ($\mathbf{O} = \{O_{S_1,P_1}...O_{S_1,P_5}, O_{S_2,P_1}...O_{S_2,P_5}, ..., O_{S_6,P_1}...O_{S_6,P_5}\}$). The transition probabilities taking action A_{rotate} are:

$$T(S_i|A_{rotate}, S_j) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases} \qquad i, j \in \{1, .., 6\} \tag{3.5}$$

All other actions finalize the recognition process and reset the state to its initial configuration:

$$T(S_i|A_j, S_k) = b_0(S_i) \qquad i, j, k \in \{1..6\} \tag{3.6}$$

Observation probabilities $P(O|S_i)$ are only dependent on the current state and are measured experimentally from previously unseen images. Rewards are assigned in the following way: a positive reward is given to the robot when being at state S_i takes action A_i and a very negative reward when being at state S_i takes action $A_{j\neq i}$. A small negative reward is given to the action A_{rotate} in order to avoid infinite rotation. Regarding the initial probabilities, each type of garment has equal probability to be selected by the robot, however shorts and T-shirts have two possible lowest point. Thus, the initial belief state is $\mathbf{b_0} = (0.25, 0.25, 0.125, 0.125, 0.125, 0.125)$ corresponding to states $S_1...S_6$ accordingly. The discount factor γ is set to 0.99. Our objective is to increase the belief about a state before taking a final decision, taking into account the uncertainty about the result of the Random Forest classifier

$(P(O|S_i))$.

At the time period t_n, the robot is in state $s \in \mathbf{S}$ and decides to take action $a \in \mathbf{A}$. The next state of the robot will now be s' with probability $T(s'|a,s)$ and will receive the reward $\gamma^n R(s',a)$. After each action of the robot, the belief about each state has to be updated. Let $b(s)$ be the probability of the robot being at state s and o the observation of the robot after taking action a. The belief state will be updated according to the following equation:

$$b'(s') = \frac{P(o|s',a) \sum\limits_{s \in S} T(s'|s,a) b(s)}{\sum\limits_{s' \in S} P(o|s',a) \sum\limits_{s \in S} T(s'|s,a) b(s)} \tag{3.7}$$

where s' is the next state of the robot, $b'(s')$ is the new belief state over the states s' and $P(o|s',a)$ is the probability of receiving observation o after taking action a and arriving at state s'. The denominator is a normalization factor.

Solving the above POMDP generates an optimal action policy for the robot. The belief state is updated after each observation of the classifier using (3.7) and the robot decides according to the policy whether to further rotate the garment to collect more observations or take a final decision and continue the unfolding process. In case the garment is rotated more than 360 degrees, the process is restarted by re-grasping the lowest point. As we will see (Table 3.1), this active recognition dramatically increases the recognition accuracy with the cost of only a few rotations.

3.6.2 Active Grasp Point Estimation

The same idea is applied to the grasp point estimation procedure. The states now correspond to the different grasp point locations with an extra state indicating the *invisible* grasp point. Again, we quantize the image space to lower the problem dimensionality applying a 8x8 grid on the bounding box of the garment (Fig. 3.5). Thus, the set of states is $\mathbf{S} = \{S_0, S_1, ..., S_{64}\}$ where S_0 is the invisible-

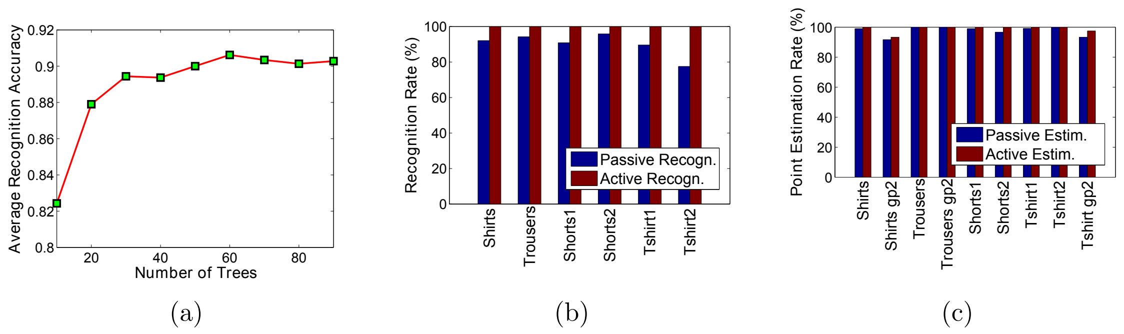


Figure 3.6: a) The average recognition rate as the number of trees increases, b) Recognition rate of each class for passive and active method, c) Grasp point estimation rate of each possible occasion. gp2 means the estimation of the 2nd grasping point while the garment is hanging from the 1st one.

grasp point state and S_1 - S_{64} correspond to the location of the grasp point on the 8x8 grid. The set of actions is similarly defined as $\mathbf{A} = \{A_{rotate}, A_1, A_2..., A_{64}\}$ where A_{rotate} should be taken when grasp point is invisible and A_i is the action of grasping the estimated point located on the ith grid square (state S_i). The observations come from the Hough Forest and contain the location of the estimated point along with the probability of the class $c_g = 1$, i.e. the probability of the point being visible. Quantising this probability into 5 equally spaced bars we get 320 different observations, 5 probability bars for each of the 64 grid locations ($\mathbf{O} = \{O_{S_1,P_1}...O_{S_1,P_5}, O_{S_2,P_1}...O_{S_2,P_5}, ..., O_{S_{64},P_1}...O_{S_{64},P_5}\}$). All the other variables $(\mathbf{T}, \mathbf{P}, \mathbf{b_0})$ are calculated experimentally from previously unseen images. A positive reward is given if the robot grasps the correct location, a smaller reward if it grasps a neighbour location and a very negative reward if it grasps any other location. In addition, if the robot decides to rotate the garment, a positive reward is given if the grasp point was invisible and a small negative reward is given if the grasp point was visible.

The solution of the unfolding POMDP gives the optimal action policy for estimating a desired point. The robot rotates the garment until a certain confidence about the location of the point is reached and then decides to grasp it. If the rotation exceeds 360 degrees the process is restarted by picking again the lowest point. This probabilistic planning makes point estimation very robust and insensitive to noisy estimations.

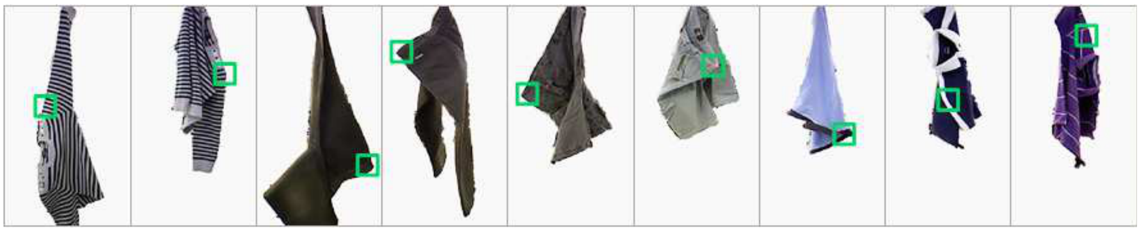


Figure 3.7: Examples of successful grasp point detection

3.7 Experiments

Robot Setup. We have tested our methods using a dual manipulator by YASKAWA[1]. We capture depth images from an Xtion depth sensor placed between the arms at a fixed height while grasping is based on custom "claw like" grippers [53]. More details can be found in Sec. A.5.1.

Training Set. Our training dataset was created using 24 regular-sized clothes of various sizes and fabric types, 6 of each category. In order to cover a variety of possible clothes configurations, each garment was grasped 20 times from each lowest point and 40 images were captured while the garment was rotating, covering the viewpoints of all 360 degrees. The final database contains 28,800 depth images. Image labelling was done manually using fiducial markers over key points to facilitate the task.

Testing Set. Testing was based on a dataset containing only novel items (not in the training dataset). The clothes used for testing were 3 per category (12 total). For measuring accuracy, we used 240 depth images for each category (1440 in total) while the same clothes were used for evaluating the whole unfolding procedure.

Random Forest configuration. For training Random Forest trees we used 5000 random candidate tests, 70 candidate thresholds per test and 4 minimum samples per node, while no restriction on the maximum depth of trees was imposed. Figure 3.6(a) shows the average recognition rate in relation with the number of trees in the forest. We see that above 60 trees recognition rate remains the same, therefore

[1]Two M1400 Yaskawa arms mounted on a rotating base

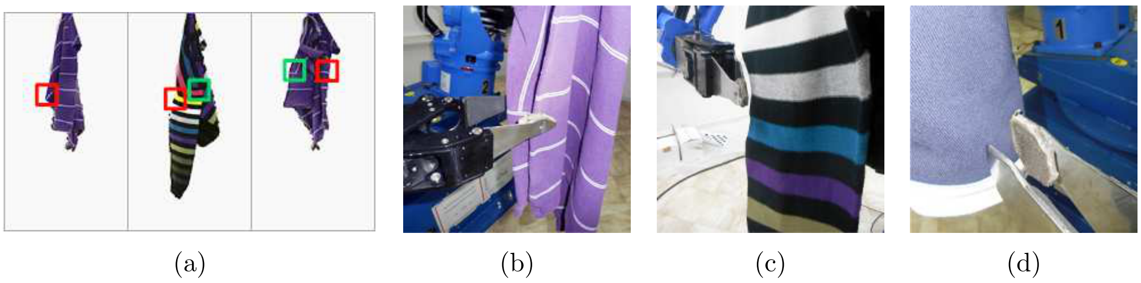


Figure 3.8: a) Failure examples of grasp point estimation (in green is the ground truth, when missing point is invisible), b) The opening of the gripper is not adjustable causing the grasp of two points, c) Inaccurate grasping because of noisy point cloud, c) Gripper cannot grasp some kinds of surfaces easily

this is the number of trees used in our forests. We used the same configuration for both random and Hough forests. Training a tree for recognition takes about half an hour while a tree for point estimation takes about 10 minutes on an Intel i7 CPU. Inference of one frame takes less than 40ms.

POMDPs. For solving our POMDPs we used the point-based SARSOP algorithm [51]. All transition and observation probabilities needed were calculated experimentally from the training set of clothes from images not used for training the forests. The rotation angle a used is 10 degrees. Moreover, reward values affect the level of confidence required for the robot in order to take a decision about an action. Having a confidence level above 0.9999 we were able to achieve 100% active recognition accuracy requiring only few rotations (Table 3.1), while also improving point estimation results (Fig. 3.6(c)).

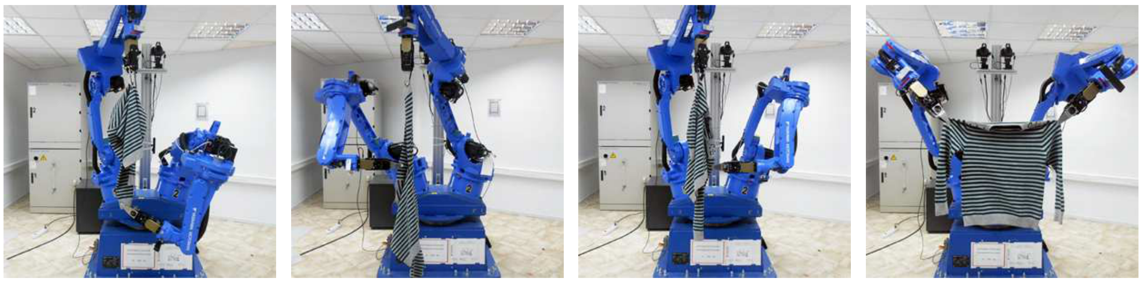
Recognition Results. It is difficult to compare clothes recognition results with other approaches as each author makes different assumptions. Fig. 3.6(b) compares results of our passive and active recognition. Active recognition achieved 100% accuracy while the one-frame (passive) recognition had 90% success rate in average.

Grasp point estimation results. Fig. 3.6(c) shows the point estimation results, while there is no other similar work to compare with. Again, an improvement of active over passive estimation is observed. Some success and failure examples of point estimations are shown in Figures 3.7 and 3.8(a).

Although 100% recognition rate gives the impression that it is a solved problem, we should mention that for research purposes, we collected a dataset of clothes that are relatively simple. The reason for this is to make the point estimation problem, which is a much harder problem, more tractable. Under this assumption the recognition task looks easy, but it shouldn't be underestimated in more general scenarios.

End-to-end unfolding results. We have conducted 120 full end-to-end unfolding experiments, using each test garment 10 times. Fig. 3.9 shows the four stages of completely unfolding a garment[2] and more qualitative results are shown in Fig. 3.10. We consider the unfolding successful when the grasped points are 10cm close to the desired points. We also implemented a shape matching algorithm [54] in order to automatically asses the final unfolded state by matching the unfolded garment to predefined unfolded templates. If the matching score is below a threshold, the process is restarted. The process is also restarted in case the garment is rotated more than 360 degrees and no recognition or point estimation occurs. If robot restarted the process more than once it was assumed as failure. 112 out of 120 experiments the unfolding was successful. Recognition achieved 100% accuracy requiring 2.4 rotations average. We encountered three types of errors caused by the gripper: a) the robot grasped two points on the garment because its opening is not adjustable, b) inaccurate alignment of the robot gripper because the plane fitting around the estimated point was affected by noisy depth sensor data, c) gripper couldn't grasp a surface because of its local shape. These errors are illustrated in Fig. 3.8(b) - 3.8(d). 15 out of 22 total errors (68%) were caused by the gripper while only 7 (32%) were caused by incorrect point estimations. However, in 14 erroneous situations, errors were perceived by the robot which restarted the process and successfully unfolded the garment. In the remaining 8 erroneous cases, robot required more than one restart and thus we considered them as failures. In no experiment the grasped point was more than 10cm away from the ground truth. Finally, the average unfolding time is 2min 23.5sec with the robot set in its safety speed mode. Results are summarized in

[2]Supplementary video: http://clopema.iti.gr/ICRA_2014/



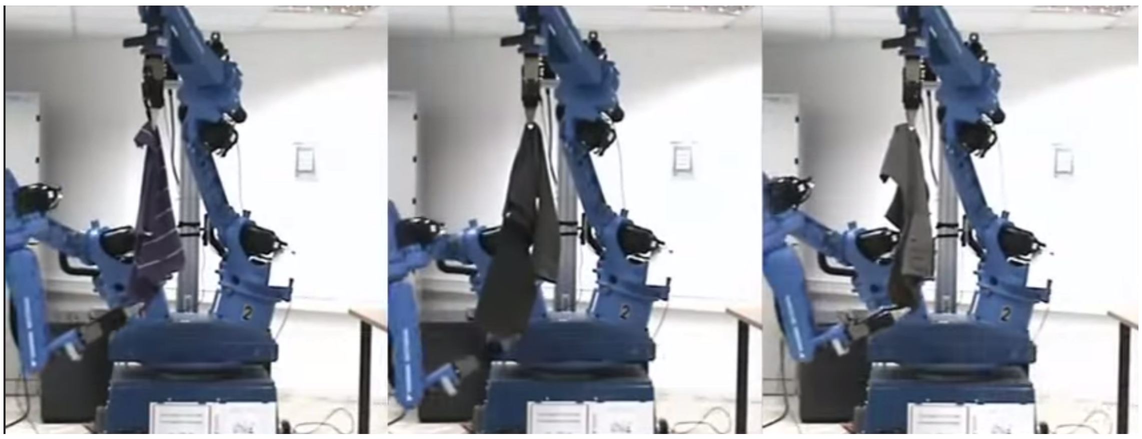
(a) (b) (c) (d)

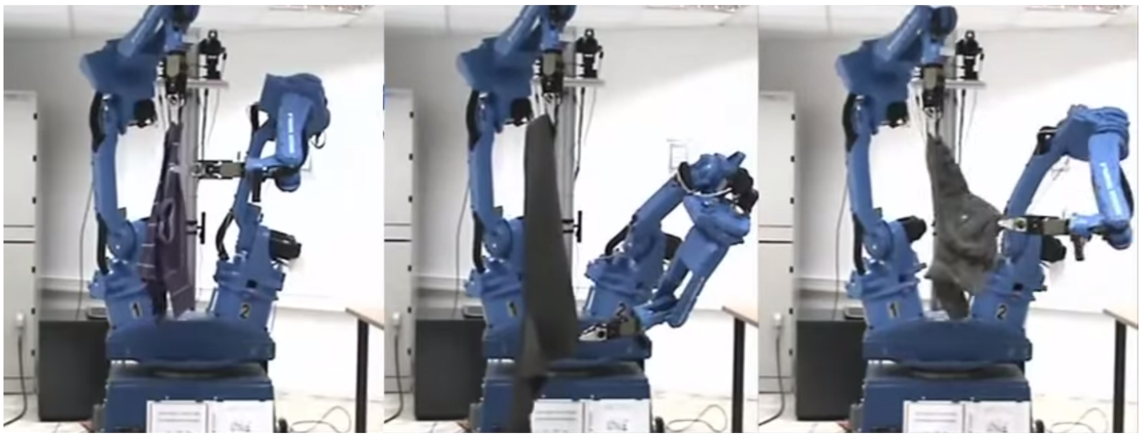
Figure 3.9: Robot unfolding a shirt: a) grasping lowest point, b) grasping 1st grasp point, c) grasping 2nd grasp point, d) spreading the unfolded garment.

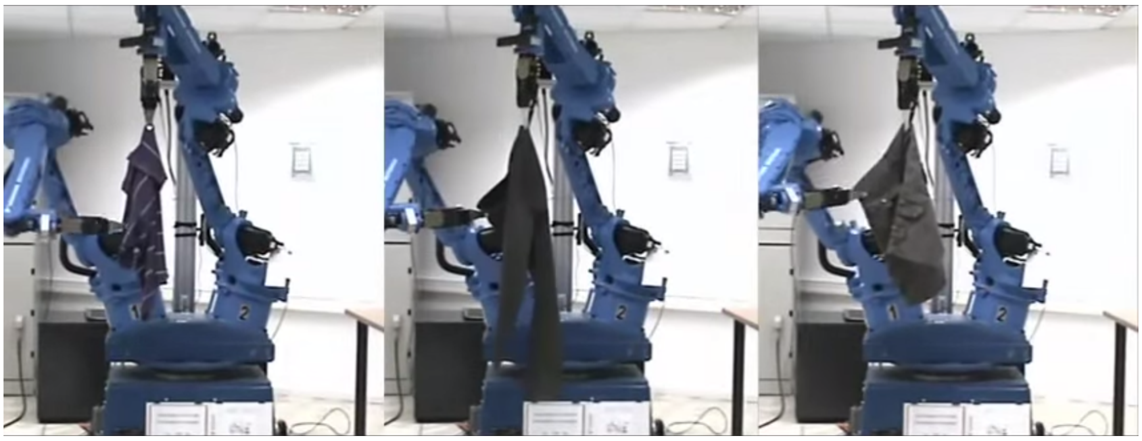
Table 3.1: Unfolding Results

	Shirts	Trousers	Shorts	T-shirts	Total	%
Experiments	30	30	30	30	120	-
Successful Unfoldings	27	30	30	25	112	**93.3%**
Successful Recognitions	30	30	30	30	120	**100%**
Avg Rotations for Recognition	0,8	1,1	2,7	5	2,4 (avg)	-
Estimation Errors	2	0	0	5	7	-
Gripper Errors	4	3	2	6	15	-
Average Time (sec)	150	136	127	161	143.5 (avg)	-

table 3.1. In the state of the art work [12], their videos show a good case scenario of unfolding shorts in about 3:20 and a more complex scenario of unfolding a T-shirt in about 4:30, which is slower than our average time in every category. Although robot parameters like motion planning affects the execution time, making it a not very reliable metric, our approach have reduced the unfolding movements to three grasps and few rotations. Such actions are generally executed faster compared to lying the garment on a table and re-grasping it quite a few times.


(a)


(b)


(c)

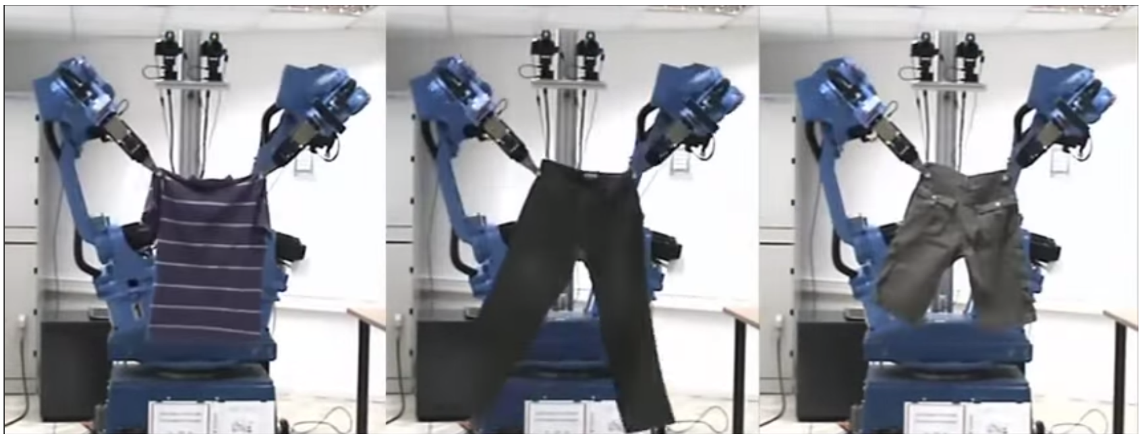

(d)

Figure 3.10: Robot unfolding different types of clothes: T-shirt, trousers and shorts. a) grasping lowest point, b) grasping 1st grasp point, c) grasping 2nd grasp point, d) final unfolding.

Chapter 4

Active Random Forests

4.1 Overview

Our Active Random Forest framework is based on standard Random Forest classifier [17]. In order to train one tree, all training samples start at the root node and keep splitting recursively into two child nodes, left and right. Splitting of samples depends on the objective function, which tries to cluster the samples, switching between a classification or regression objective. The most common metric to evaluate a split is the entropy. In order to find the best split that minimizes the objective function, there is a split function with random parameters, which is used to produce many different splits to evaluate. When samples cannot further split (due to certain criteria), the node becomes a leaf and stores a histogram of the classes of the samples arrived and / or a distribution of some continuous variables. This way many trees can be trained, all with different random parameters. During inference, a test sample passes down the tree, evaluating the split functions and branching left or right, until it reaches a leaf node. The inference outcome is the average distribution of the leaf nodes of all the trees.

Based on this formulation, we introduce another property of the trees, that is to be able to investigate another viewpoint of an object. The best way would be to investigate other viewpoints, when the current one stops being informative according to the training samples. To this end, we introduce another type of node, called *action-selection* node. This node is created when such behavior is recognized (details are included in the next subsection). In this node, the tree decides to include another viewpoint to the process of samples splitting. In order to observe another viewpoint of a garment, the gripper holding the garment can be rotated and another image can be captured from the camera that is located at the robot's base.

Active Random Forests are based on the method proposed in chapter 3. In that method however, viewpoint selection was made sequentially by taking nearby viewpoints, which is a sub-optimal solution whilst in some cases slows down the entire process. This work makes active vision faster and more efficient by the use of Active Random Forests (ARFs). With ARFs we can estimate and visit directly the desired viewpoint without passing through all the intermediate viewpoints. Furthermore, POMDPs where built upon the classifier outcome, whereas in ARFs, the classifier is designed in such a way that can take advantage all the information gathered at once before taking the final decision. This greatly improves performance. In addition, we estimate the pose of the garment in order to guide the robot's gripper to grasp a desired point, which reduced grasping errors compared to the local plane fitting techniques employed in [2]. Most importantly, our framework can be easily extended to other active vision problems.

The following subsections describe in detail both training and testing of our new trees, which include the action-selection nodes, and how the trees learn to select the next best view.

4.2 Related Work

Active vision has been studied very early in the literature. Bajcsy et al. [55] presented a nice study about the ingredients that an artificial or biological organism should have in order to act autonomously, among which are active sensory and exploratory capabilities. Tsotsos et al. [56] formulated the active vision problem as an attention problem, and described an objective function of maximizing the probability that an objects appears in a certain 3D cuboid in space. Another interesting early work was presented by Jean-Yves Herve and Yiannis Aloimonos [57] where they tackled the problem of shape from shading, when the camera is able to move. They formulated their active vision problem as trying to move to the next best viewpoint that optimizes the stability of equations to be solved for the shape from shading problem. One of the first attempts to use entropy for next best view estimation for object recognition was the work of Borotschnig et al. [58]. They used an appearance-based object representation, the parametric eigenspace, and augmented it with probability distributions in order to be able to calculate the uncertainly, i.e. the entropy, in each viewpoint making it the objective for selecting the next best viewpoint.

Active vision literature focuses mainly on finding efficient methods for selecting observations optimally while little attention is paid to the classifier which is kept simple. The majority of works adopted an off-line approach which consists of precomputing disambiguating features from training data. Schiele et al. [59] introduced “transinformation”, the transmission of information based on statistical representations, which can be used in order to assess the ambiguity of their classifier and consequently find the next best views. Arbel et al. [60] developed a navigation system based on entropy maps, a representation of prior knowledge about the discriminative power of each viewpoint of the objects. In a subsequent study, they presented a sequential recognition strategy using Bayesian chaining [61]. Furthermore, Callari *et al.* [62] proposed a model-based active recognition system, using Bayesian probabilities learned by a neural network and Shannon entropy to drive the system to the next best viewpoints. Also, Sipe and Casasent [63] introduced the probabilistic feature space trajectory (FST) which can make estimation about the class and pose of objects along with the confidence of the measurements and the location of the most discriminative view. Such methods are computationally efficient both in training and testing. On the other hand, they rely mainly on their best hypotheses based on prior knowledge which can in fact have low probabilities on a test object while features from the visited viewpoints are assumed independent in order to make the final inference.

One of the most representative works in the same direction was made by Denzler *et al.* [14] who tried to optimally plan the next viewpoints by using mutual information as the criterion of the sequential decision process. They also presented a Monte-Carlo approach for efficiently calculating this metric. Later, Sommerlade and Reid [64] extended this idea in tracking of multiple targets on a surveillance system. One drawback of this approach was that the accumulated evidence about the visited viewpoints did not affect the viewpoint selection strategy which was based on pre-computed leant actions. An improvement over this idea was made by Laporte and Arbel [15] who introduced an on-line and more efficient way of computing dissimilarity of viewpoints by using the Jeffrey Divergence weighted by the probabilistic belief of the state of the system at each time step. This work however, combines viewpoint evidence probabilistically using Bayesian update which relies on the consistent performance of the features or the single-view classifier used (in at least some viewpoints), which is generally challenging in high dimensional feature spaces like the problem of pose estimation of deformable objects. A recent work on active vision was made by Jia et al. [16] who used a similarity measure based on the Implicit Shape Model and other prior knowledge combined in a boosting algorithm in order to plan the next actions. However the employed similarity measure is not suitable for highly deformable objects such as garments, whereas the boosting strategy based on certain priors makes a minor improvement over [14] and [15]. Finally, there are some active vision applications to robotic systems in real scenarios [65, 66, 67, 68] mainly based on the previously described works, showing promising results.

4.3 Active Random Forests training

The training samples for Active Random Forests consist of tuples $(\mathbf{I}(v), c, \mathbf{g}(v), \mathbf{p}(v))$, $v \in V$, where $\mathbf{I}$ is the depth image of the garment, c is the garment type (class), $\mathbf{g}$ is the 3D position of the desired grasp point in the current camera frame, $\mathbf{p}$ is a 3D vector of the garment pose and V is the set of all possible viewpoints v of the garment (Fig. 4.1(a)). Viewpoints are discrete and equally distributed around the vertical axis, which coincides with the gripper holding the garment. If the desired point is not visible from viewpoint v then $\mathbf{g}(v)$ is undefined.

Each split node of the decision tree (Fig. 4.1(b)) stores a set V_{seen} of the already seen viewpoints and passes it to its children. At the beginning, only one viewpoint has been visited and therefore this set in the root node is $V_{\text{seen}} = \{v_0\}$. In each node, we evaluate a random set of split functions in the form $f(v, \mathbf{I}, \mathbf{I}_{\text{curv}}, \mathbf{u}_{\text{tripl}}) > t_{\text{split}}$ where t_{split} is a threshold. The first parameter v is a viewpoint selected randomly (uniform distribution) from the set of already visited viewpoints. The second and third parameters denote a feature channel for which the test should be performed. That is the original depth image $\mathbf{I}$ and mean curvature of the surface $\mathbf{I}_{\text{curv}}$ estimated from the depth image, similar to Sec. 3.4. The forth parameter $\mathbf{u}_{\text{tripl}}$ is a triplet from a set of random position triples $U = \{(\mathbf{u}_1^1, \mathbf{u}_2^1, \mathbf{u}_3^1), (\mathbf{u}_1^2, \mathbf{u}_2^2, \mathbf{u}_3^2), \ldots\}$ determining positions in the image (Fig. 3.3). The used binary tests are the same described in Section 3.4 which proved to be fast and efficient for our problem. They are summarized below:

- Two pixel test $f_1 \equiv \mathbf{I}(\mathbf{u}_1) - \mathbf{I}(\mathbf{u}_2)$, where $\mathbf{I}(\mathbf{u})$ is the depth value at position $\mathbf{u}$.
- Three pixel test $f_2 \equiv (\mathbf{I}(\mathbf{u}_1) - \mathbf{I}(\mathbf{u}_3)) - (\mathbf{I}(\mathbf{u}_3) - \mathbf{I}(\mathbf{u}_2))$.
- One pixel test $f_3 \equiv |\mathbf{I}_{\text{curv}}(\mathbf{u})|$, where $\mathbf{I}_{\text{curv}}(\mathbf{u})$ is the curvature at position $\mathbf{u}$.

We want our new decision trees to jointly perform classification, grasp point detection and pose estimation. Therefore, we apply a different quality function form for

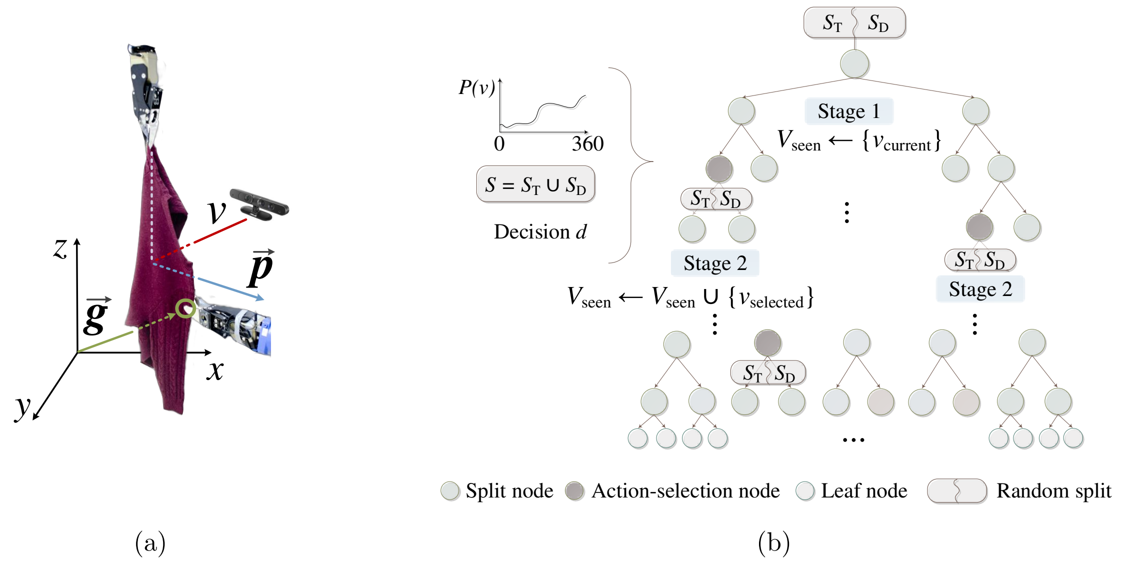


Figure 4.1: a) Visualization of grasp point vector **g** and pose vector **p**. b) Active Random Forests training procedure. Split, action-selection and leaf nodes are shown in the tree. In each action-selection node, all viewpoints are evaluated according to $P(v)$.

each objective in the split nodes in a hierarchical coarse to fine manner [69]. That is, classification is performed in the upper part of the trees, and when the classes have been discriminated, the lower part performs regression of grasp points or pose vectors for each class separately. The general form of the quality function can be written as:

$$Q = \begin{cases} Q_{\text{class}}, & \text{if } \max P(c) \leq t_c \\ Q_{\text{reg}}, & \text{if } \max P(c) > t_c \end{cases} \tag{4.1}$$

Here Q_{class} is the quality function term for classification and Q_{reg} the quality function term for regression. Specifically, Q_{class} is the *information gain* using *Shannon Entropy* and Q_{reg} is the information gain for continuous Gaussian distributions [70] weighted by the population of the nodes that a split function produces. They have the general form:

$$Q_{\text{class}} = -\sum_{\substack{child\in \\ \{\text{left,right}\}}} \frac{|S_{child}|}{|S|} \sum_{c=1}^{N_{\text{classes}}} P_{child}(c) \log_2 P_{child}(c) \tag{4.2}$$

$$Q_{\text{reg}} = -\sum_{\substack{child\in \\ \{\text{left,right}\}}} \frac{|S_{child}|}{|S|} \ln |\Lambda_q(S_{child})| \tag{4.3}$$

Here S denotes the set of samples in a node and Λ_q is the covariance matrix of the continuous variable q being optimized (more details on this can be found in [70]). The form of the quality function depends on $P(c)$, which is the probability distribution of the classes of the training samples in the node, and on the predefined threshold t_{c}, which is typically set to 0.9. This means that when samples in a node are correctly classified, then the tree switches to regression optimization. In each node, we evaluate random split tests using (4.1) and we keep the one that maximizes Q.

Apart from the standard objectives of classification and regression, we want to integrate the next best view selection into the decision trees, since this problem is closely related to the aforementioned objectives. In our problem, selecting a next viewpoint improves our system in two ways: a) it improves classification and regression accuracy, b) it optimally detects a grasp point when it is hidden in the current view. Furthermore, our approach takes into account the cost of the actions, which is currently related to minimizing the execution time of the selected movements.

The split function used until now considers only the set of visited viewpoints V_{seen}. However, below a certain tree depth, this set is uninformative and if the tree continues to split the samples further, it starts to overfit. In this case, a new viewpoint should be acquired to resolve the ambiguity. The problem now is to determine in advance when the current set of viewpoints is not informative. For this, we introduce a validation set, which is being split in parallel with the training set. The divergence of the posterior distributions between the two sets is measured in each split node. Specifically, the initial training set S is split into two equal-sized random subsets: S_{T} is the training set and S_{D} the validation set. Split functions are evaluated using only the training set and when the best split function found, both sets are split using this best split function.

For measuring the divergence of two sets, we have experimented with two alternative metrics which were tested and compared in the experiments (Section 4.5). The first is the *Hellinger distance*[71], a statistical measure defined over sets S_T^j and S_D^j as:

$$HL(S_T^j \| S_D^j) = \frac{1}{\sqrt{2}} \sqrt{\sum_{c=1}^{C} \left(\sqrt{P_{S_T^j}(c)} - \sqrt{P_{S_D^j}(c)} \right)^2} \tag{4.4}$$

when comparing the class distributions of the training set S_T^j and validation set S_D^j having C classes. $P_S(c)$ is the class probability distribution of the set S. The Hellinger distance satisfies the property $0 \le HL \le 1$ and it takes its lowest value 0 when training and validation set distributions are identical and its maximum value 1 when one distribution is 0 when the other is positive. Similarly, assuming that grasp point and vectors at node j are normally distributed variables, the averaged squared Hellinger distance over the possible viewpoints is:

$$HL^2(S_T^j \| S_D^j; \mathbf{q}) = \frac{1}{V} \sum_{v \in \mathbf{V}} 1 - \frac{\left(|\Lambda_{\mathbf{q}(v)}(S_T^j)| |\Lambda_{\mathbf{q}(v)}(S_D^j)| \right)^{\frac{1}{4}}}{|A|^{\frac{1}{2}}} \exp\{-\frac{1}{8} \mathbf{u}^T A^{-1} \mathbf{u}\} \tag{4.5}$$

where

$$\mathbf{u} = \boldsymbol{\mu}_{\mathbf{q}(v)}(S_T^j) - \boldsymbol{\mu}_{\mathbf{q}(v)}(S_D^j) \tag{4.6}$$

$\boldsymbol{\mu}_{\mathbf{q}(v)}()$ is the mean value of vectors $\mathbf{q}$ $(= \mathbf{g}(v)$ or $\mathbf{p}(v))$ in viewpoint v and A the average covariance matrix of S_T^j and S_D^j.

The other metric is the so called *Jensen–Shannon divergence* which measures the information divergence of two probability distributions and is actually a symmetric version of the *Kullback–Leibler* divergence. Measuring the class distribution divergence of training and validation sets, Jensen–Shannon divergence is defined as:

$$JS(S_T^j \| S_D^j) = \frac{1}{C} \sum_{c=1}^{C} P_{S_T^j}(c) \log \frac{P_{S_T^j}(c)}{P_m(c)} + P_{S_D^j}(c) \log \frac{P_{S_D^j}(c)}{P_m(c)} \tag{4.7}$$

where P_m is the average class distribution of S_T and S_D. Again, JS satisfies the property $0 \le JS \le 1$, where 0 indicates identical distributions while 1 indicates maximum divergence. For measuring the information divergence of our continuous variables over two sets, we substitute (4.7) with multi-variate Gaussian distributions and compute the average over viewpoints $\mathbf{V}$, which results in:

$$\begin{aligned} JS(S_T^j \| S_D^j; \mathbf{q}) = \frac{1}{2V} \sum_{v \in \mathbf{V}} \Big(\mathbf{u}^T \left(\Lambda_{\mathbf{q}(v)}(S_T^j)^{-1} + \Lambda_{\mathbf{q}(v)}(S_D^j)^{-1} \right) \mathbf{u} \\ + tr \left(\Lambda_{\mathbf{q}(v)}(S_T^j)^{-1} \Lambda_{\mathbf{q}(v)}(S_D^j) + \Lambda_{\mathbf{q}(v)}(S_D^j)^{-1} \Lambda_{\mathbf{q}(v)}(S_T^j) - 2\mathbf{I} \right) \Big) \end{aligned} \tag{4.8}$$

where $\mathbf{u}$ is defined in Eq. (4.6). More details about (4.8) can be found in [71].

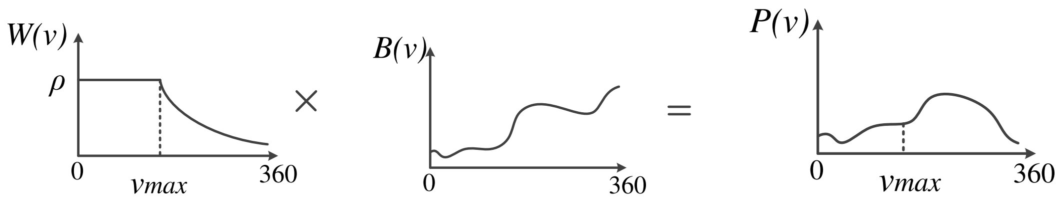
If the divergence Δ between the training and validation set is above a threshold t_Δ, the node is considered as an *action-selection node*, where an action (that is moving the holding gripper to see another view) should be taken to avoid overfitting. In this case, the split functions consider the whole set of possible viewpoints V of a garment (which are available during training) and not only V_{seen}.

To account for the execution cost of the selected actions, we assign them a cost relative to the distance from the current viewpoint (i.e. how much the gripper should be rotated), while images from the intermediate viewpoints are also captured without any additional cost. Specifically, the distance is measured as the degrees of rotation of the gripper needed to change the viewpoint. In order to weight the viewpoints according to their distance, we change the distribution from which the viewpoints v are randomly selected. The distribution of V in an action-selection node is shown in Fig. 4.2(a). The already visited viewpoints $\{1 \dots v_{\max}\}$ have a uniform distribution to be selected with probability ρ, since they are not assigned any additional cost. Viewpoint $v_{\max}$ is the furthest viewpoint seen so far in training. The next viewpoints are assigned an exponentially decreasing probability, proportional to their distance from the current view. The resulting distribution W is defined as:

$$W(v) = \begin{cases} \rho, & \text{if } v \in \{1 \dots v_{\max}\} \\ \rho \exp\left(-\frac{v - v_{\max}}{N_V}\right), & \text{if } v > v_{\max} \end{cases} \tag{4.9}$$

where N_V is the total number of quantized viewpoints.

On the other hand, the desired grasp point on the garment may be invisible in



(a) Weighted distribution (b) Visibility map (c) Final distribution

Figure 4.2: Probability distributions of viewpoint used for random test selection.

the already visited viewpoints. Therefore the next viewpoint should also make the desired point visible, apart from disambiguating the current hypotheses about the garment category or pose. The probability of the grasp point being visible can be calculated from the vectors $\mathbf{g}(v)$ in a node. The prior visibility probability $B(v)$ for each viewpoint v in node j containing samples S^j is defined as follows (Fig. 4.2(b) shows an example):

$$B(v) = \frac{\sum_{s \in S^j} b(s,v)}{\sum_{v' \in V} \sum_{s \in S^j} b(s,v')} \tag{4.10}$$

$$b(s,v) = \begin{cases} 1, & \text{if } \mathbf{g}_s(v) \text{ exists} \\ 0, & \text{if } \mathbf{g}_s(v) \text{ is not defined} \end{cases} \tag{4.11}$$

The distribution for selecting the next possible viewpoint is given by $P(v) = W(v)B(v)$. Thus, such viewpoints, which are closer to the current one and where the grasp point is more probable to be visible, are more likely to be selected. Fig. 4.2(c) shows an example of such distribution.

The next best viewpoint v_{best} in an action-selection node can now be found by randomly selecting viewpoints from the distribution $P(v)$. This time, the whole set of samples $S^j = S^j_{\text{T}} \cup S^j_{\text{D}}$ in the node j is used to evaluate the randomly generated tests in order to reduce the divergence between the previous training and validation set. However, in the child nodes of the action-selection nodes, the samples are again randomly split into training and validation sets and the process is repeated. This

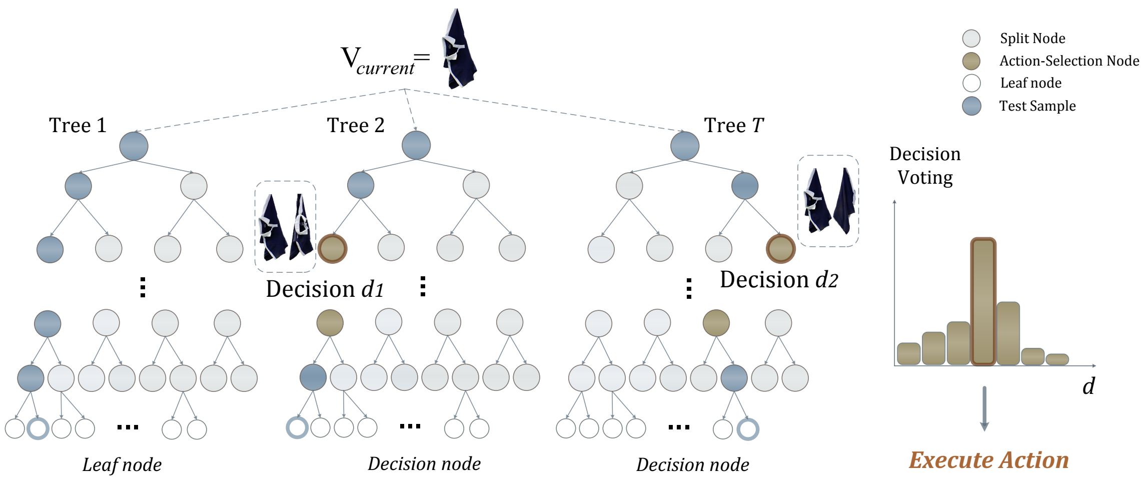


Figure 4.3: Active Random Forests inference procedure. At each action-selection node, one more viewpoint is selected to be seen, guiding the robot to rotate the garment. Best next viewpoint at the certain node has been found during training.

time, the nodes contain one more visited viewpoint, that is: $V_{\text{seen}} = V_{\text{seen}}^{\text{parent}} \cup \{v_{\text{best}}\}$. Finally, a leaf node is created when a minimum number of samples reaches the node. In the leaf nodes, we store the class distribution $P(c)$ and the first two modes of $\mathbf{g}(v)$ and $\mathbf{p}(v)$ per class (as in [72]), weighted by the class probability, for memory efficiency.

4.4 Active Random Forests inference

Inference using Active Random Forests begins with the current view of the garment hanging from its lowest point. This image starts traversing the trees, evaluating the split functions selected during training. A leaf node may be reached in some trees, however, in other trees, the image ends up in an action-selection node. An another viewpoint is therefore required to continue traversing down such tree (Fig. 4.3). Each action-selection node from any tree votes for the next best viewpoint in a similar way, how a leaf node votes in the classical Random Forests. The most voted viewpoint is then visited and another image is captured. The trees that voted for the selected viewpoint can be further traversed using the acquired viewpoint, while the remaining trees keep their votes for the next action voting.

Algorithm 1 Active Random Forests inference

```
Input: Set of pretrained trees $ARF$
       Current arbitrary viewpoint $v_{\text{current}}$
Output: Garment class $c$
        Grasp point location $\mathbf{g}$
        Pose $\mathbf{p}$
Initialize set of seen viewpoints $V_{\text{seen}} = \{v_{\text{current}}\}$
Initialize set of reached leaf nodes $Leafs = \emptyset$
while true do
    Initialize $DecisionVotes$ array to 0
    for all trees $T$ in $ARF$ do
        $node \leftarrow$ traverse tree $T$ from node $V_{\text{seen}}$
        if $node$ is leaf node then
            $Leafs \leftarrow Leafs \cup \{node\}$
            $ARF \leftarrow ARF \setminus T$
        else if $node$ is action-selection node then
            $d \leftarrow$ viewpoint decision stored in $node$
            Increase $DecisionVotes[d]$
    if $|Leafs| > N_{\text{leafs}}$ then break
    Execute action for decision $d^* = \text{argmax}_d \, DecisionVotes[d]$
    Update current view $v_{\text{current}}$
    $V_{\text{seen}} \leftarrow V_{\text{seen}} \cup v_{\text{current}}$
return average class $c$, Hough votes for $\mathbf{g}(v)$, $\mathbf{p}(v)$ from $Leafs$
```

This process stops when N_{leafs} leaf nodes have been reached. The final class is estimated by averaging the class distributions stored in the reached leaf nodes. Grasp point detection and pose estimation are made using Hough voting of the vectors $\mathbf{g}$ and $\mathbf{p}$ stored in the leafs that we reached from all the visited viewpoints. The complete inference procedure is shown in Algorithm 1. The framework is illustrated in Fig. 4.3. Parameter N_{leafs} is experimentally evaluated in Sec. 4.5.

4.5 Experimental results

4.5.1 Experimental Setup

To evaluate the ARF framework, we used our database which consists of 24 clothes, 6 of each type. Each garment was grasped by the robot gripper from each lowest point(s) 20 times to capture the random cloth configurations, collecting 40 depth images while it was rotating 360 degrees around its vertical axis. The total number of images collected is 57,600 taking into account the symmetric images as well.

Another 480 unseen images for each category were used as our test samples. The training samples consist of sets of images $\mathbf{I}(v)$ containing images of a certain garment from every viewpoint v and having every arbitrary view as the first view. The steps involved in the unfolding process using the robot are: grasp the lowest point, recognize the garment and detect the 1^{st} desired grasp point and pose, grasp desired point, search for the 2^{nd} desired grasp point and pose (no classification needed), grasp final point and unfold. In the experiments bellow, classes $c_1 - c_6$ correspond to: *shirts, trousers, shorts grasped from* 1^{st} *lowest point (leg), shorts grasped from the* 2^{nd} *lowest point (waist), T-shirts grasped from the* 1^{st} *lowest point (waist), T-shirts grasped from the* 2^{nd} *lowest point (sleeve).* We train an ARF using these classes so that the robot can recognize the cloth and grasp the first desired point, based on its pose. Furthermore, we train another ARF which is used to detect the 2^{nd} desired point and pose. The second ARF does not perform classification as it is already addressed. The second ARF is trained using images from clothes hanging from their first grasp point. Thus, we define as c_i-2 the class c_i when hanging from the 1^{st} grasp point and no classification is calculated for it. Last, We have discretized the possible viewpoints into 40 equal bins of 9 degrees each, which provides enough accuracy keeping training time reasonable(few hours). We assume a correct grasp point estimation if it is at most 10cm close to ground truth, whereas 18 degrees divergence is allowed for a correct pose estimation.

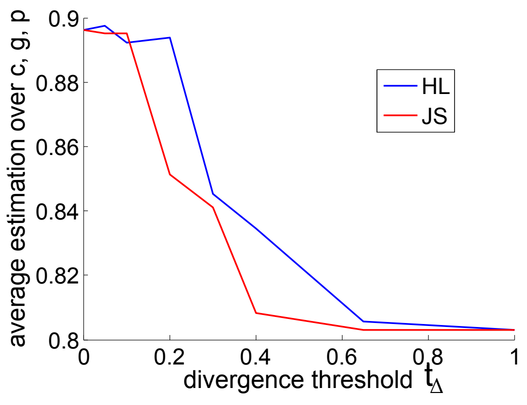
4.5.2 Parameter Analysis

An important issue in the experiments was setting up the parameters correctly. The first parameter which needs to be defined is t_Δ, the threshold of the divergence of the training and validation sets of a node, above which a new decision should be made. Fig. 4.4(a) shows the average performance of classification, grasp point and pose estimation of an ARF containing a large number of trees (discussed below) with t_Δ varying from 0 to 1 for both metrics HL and JS. When t_Δ is 0, every node in the forest becomes an action-selection node and the forest tends to overuse the possible viewpoints available for inference increasing the total number of actions required. On the other hand, when t_Δ is 1, there is no action-selection node and the forest behaves as a single-view classifier. Fig. 4.4(a) shows that when HL is used, performance starts decreasing for $t_\Delta > 0.2$ while the same happens when JS is used for $t_\Delta > 0.1$. These are the limit values for t_Δ, above which the classifier tends to behave as a single-view classifier and below which it starts using redundant actions.

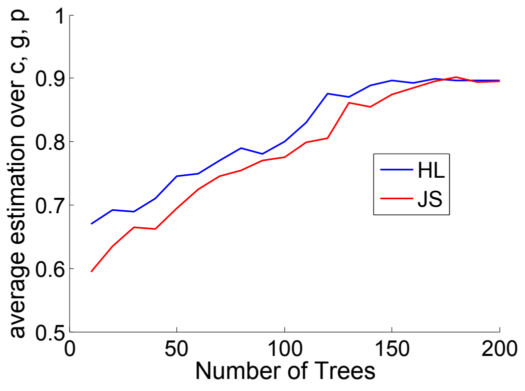
Having t_Δ defined for both of our metrics, the next parameters that should be defined are the total number of trees and the minimum number of leaf nodes N_L needed by an ARF in order to make an inference. Because ARFs have a decision voting scheme along with the leaf-node aggregation, we make the following observation: Assuming that N_x leaf nodes are sufficient to make an inference and an ARF has reached $N_x - 1$ leafs, it would be desired to have another N_x trees to vote for the next decision. Therefore, N_L is set to $N_T/2$, which is half the number of trees in the forest. Fig. 4.4(b) shows the average accuracy of our ARF, making use of the previous observation. Both metrics reach the same level of accuracy with JS requiring more trees. However, Fig. 4.4(c) shows that by using JS the forest requires significantly less movements than HL to achieve the same results. Therefore, JS was used for all the subsequent experiments.

4.5.3 Performance and Comparisons

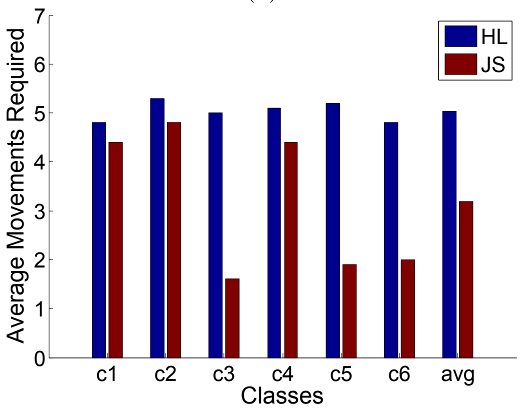
Fig. 4.4(d) shows the performance of ARF in all possible situations, with pose estimation being the most challenging objective. This figure was created without considering the weights of the actions. In the opposite case however results were very similar, thus Fig. 4.4(d) represents both scenarios. These two cases are compared in Fig. 4.4(e) which shows that weighting actions slightly increases the required



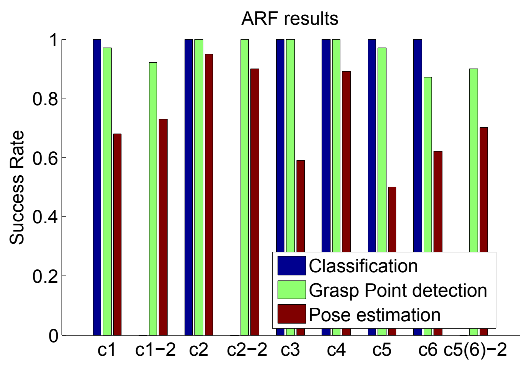
(a)



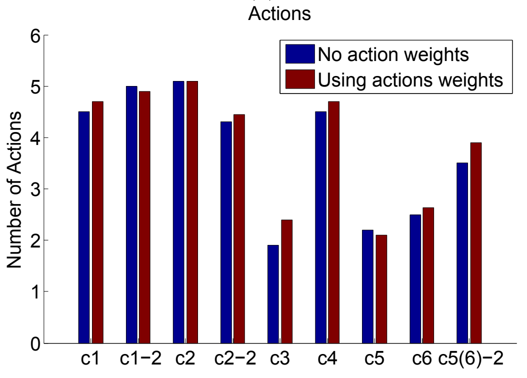
(b)



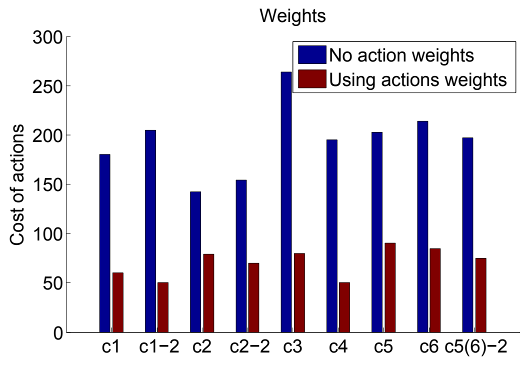
(c)



(d)



(e)



(f)

Figure 4.4: Plots from experimental results showing: a) the divergence threshold t_Δ, b) Number of trees, c) average number of movements, d) ARF success rates, e) Number of movements for weighted and non-weighted actions policy f) average cost of actions of the two policies, g-h-i) Classification-Grasp Point-Pose estimation

viewpoints needed for inference. On the other hand, in Fig. 4.4(f), the required actions in the case of considering their weights have significantly lower cost than the actions in the first case, without sacrificing accuracy. The cost of an action was considered to be the degrees of rotation the gripper required in order to reach the desired viewpoint. Fig. 4.4(f) shows the sum of the costs of all the actions needed for inference. In order to compare the ARF results, we have used two kinds of baseline methods: 1)single-view classification methods without incorporating actions; 2) active viewpoint selection methods based on a single-view method and utilizing information from entire history of selected viewpoints by updating the probability of the current state after each action. The first single-view classifier is our method described in Chapter 3 based on Random Forests, modified to perform pose estimation. The second such classifier is based on multi-class and regression SVM[73, 74]. The features used were the raw depth image of a garment and the HOG features[75] applied on the depth image. The first active vision technique used is our previous technique based on POMDP (Chapter 3), the second uses the viewpoint selection criterion proposed in [14] based on mutual information (displayed as MI) and the third uses Jeffrey Divergence metric as proposed in [15](displayed as JD). In all cases, we executed a random viewpoint selection for comparison. Finally, for a fair comparison we did not take into account the costs of actions and the visibility map (Eq. 4.10). Table 4.1 shows the results for classification, grasp point detection and pose estimation respectively. In all cases, methods based on the SVM classifier had the worst performance. In classification and point detection, the single-view classifiers have consistent good performance and therefore the active vision approaches had a positive impact on the inference. In both cases, ARF achieves equal accuracy with the best active vision technique in each case. The power of ARFs however is revealed in the pose estimation task, where they outperform previous works by almost 20%. The reason is that when dealing with such a challenging problem, the single-view inference has low accuracy producing many equally probable hypotheses. This makes classical active vision approaches perform similar to a random viewpoint

	Single		POMDP		MI		JD		Random		ARF
	SVM	RF	SVM	RF	SVM	RF	SVM	RF	SVM	RF	
Class.	0.72	0.90	0.91	0.97	0.84	0.94	0.93	0.97	0.88	0.92	**0.98**
Grasp	0.53	0.95	0.68	**0.97**	0.63	0.94	0.72	0.94	0.57	0.93	0.94
Pose	0.21	0.49	0.24	0.53	0.26	0.55	0.27	0.55	0.24	0.52	**0.71**

Table 4.1: Comparison of Active Random Forests with three state of the art methods (POMPD, MI, JD) and two baseline methods (single and random view), all of them using two different classifiers (SVM and Random Forests). Accuracies (in percents) of all compared methods were evaluated in three tasks: classification, grasp point detection and pose estimation.

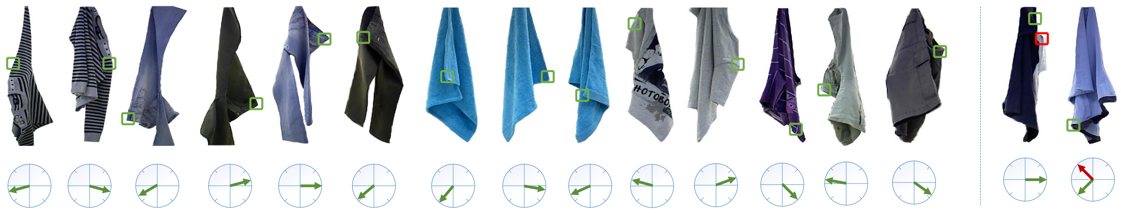


Figure 4.5: Success and failure cases (the last two) of some clothes. The arrow under each cloth indicates its pose. The first error is in grasp point detection, the second in pose estimation.

selection strategy. In contrast, ARF combines features from the most discriminant views learned in training, and thus is not so affected from single-view uncertainty. Last, for achieving all the three objectives all active vision techniques were allowed to execute at most 20 actions, above which no further improvement was noticed, even when all viewpoints were seen. In contrast, as shown in Fig. 4.4(c), ARF shows high accuracy with an average of 3.5 moves, which is significantly lower. Fig. 4.5 shows some success and failure cases using some test clothes. The failures on the right are due to wrong grasp point detection and wrong pose estimation respectively. Also our supplementary video[1] shows the whole unfolding procedure using a dual arm robot, along with comparisons of ARF with the state of the art in real scenarios.

[1]Supplementary material can be found at: http://clopema.iti.gr/ECCV-2014/

Chapter 5

3D Object Detection & Next-Best-View Prediction

5.1 Overview

Our object detection and pose estimation framework consists of two main parts: a) single shot-based 6D object detection and b) next-best-view estimation. In the first part, we render the training objects and extract depth-invariant RGB-D patches. The latter are given as input to a Sparse Autoencoder which learns a feature vector in an unsupervised manner. Learning unsupervised features with autoencoders, has the important advantage of being able to learn to discriminate both textured and texture-less object patches. In this way you dont have to hand-craft features for each such case (rgb features, depth-curvature features) as already happens in the literature, but the same pipeline can perform well with all types of objects. Using this feature representation, we train a Hough Forest to recognize object patches in terms of class and 6D pose (translation and rotation). Given a test image, patches from the scene pass through the Autoencoder followed by the Hough forest, where the leaf nodes cast a vote in a 6D Hough space indicating the existence of an object. The modes of this space represent our best object hypotheses. The second part, next-

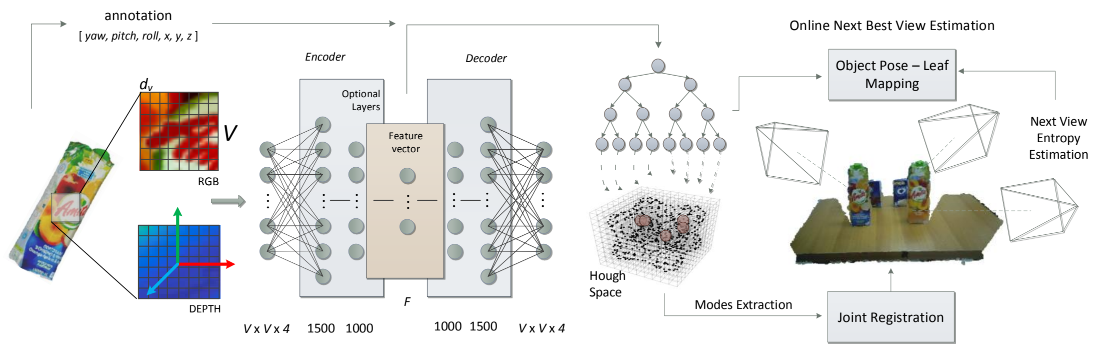


Figure 5.1: Framework Overview. After patch extraction, RGBD channels are given as input to the Sparse Autoencoder. The annotation along with the produced features of the middle layer are given to a Hough Forest, and the final hypotheses are generated as the modes of the Hough voting space. After refining the hypotheses using joint registration, we estimate the next-best-view using a pose-to-lead mapping learnt from the trained Hough Forest.

best-view estimation, is based on the previously trained forest. Using the training sample distribution in the leaf nodes, we are able to determine the uncertainty, i.e. the entropy, of our current hypotheses, and further estimate the reduction in entropy when moving the camera to another viewpoint using a pose-to-leaf mapping. Fig. A.6 shows an overview of the framework. In the following subsections, we discuss about the related work in the field, and then we describe each part in detail.

5.2 Related Work

Unsupervised feature learning has recently received the attention of the computer vision community. Hinton *et al.* [76] used a deep network consisting of Restricted Boltzmann Machines (RBMs) for dimensionality reduction and showed that deep networks can converge to a better solution by greedy layer-wise pre-training than without pre-training at all. Jarrett *et al.* [77] showed the merits of multi-layer feature extraction with pooling and local contrast normalization over single-layer architectures, while Le *et al.* [78] used a 9-layer Sparse Autoencoder to learn a face detector only from unsupervised data. Feature learning has also been used for classification[79] using RNNs, and detection[80] using sparse coding, trained with holistic object images and patches, respectively. Coates *et al.* [81] investigated different single-layer unsupervised architectures such as k-means, Gaussian mixture modes, and Sparse Autoencoders achieving state of the art results when parameters were fine-tuned. Here, we use the Sparse Autoencoders of [81] but in a deeper network architecture, extracting features from raw RGB-D data. In turn, in [82] and [83] it was shown how CNNs could be trained for supervised feature learning, while in [84] and [85] CNNs were trained to perform classification and regression jointly for 2D object detection and head pose estimation, respectively.

Object detection and 6 DoF pose estimation is also frequently addressed in the literature. Before low-cost depth cameras were available, an important method for object detection was introduced by Leibe et al. [86]. It is widely known as Implicit Shape Model, and the idea was to implicitly learn the corresponding position of the learned feature representations on the object, which makes the method learn from fewer training images. Most representative are techniques based on template matching, like LINEMOD [20], its extension [87] and the Distance Transform approaches [88]. Point-to-Point methods [89, 90] form another representative category where emphasis is given on building point pair features to construct object models based on point clouds. Tejani *et al.* [6] combined Hough Forest with [20] using a template matching split function to provide 6 DoF pose estimation in cluttered environments. They provided evidence that, using patches instead of the holistic image of the object, can boost the performance of the pose estimator in cases of severe occlusions and clutter. Brachmann *et al.* [5] introduced a new representation in form of a joint 3D object coordinate and class labelling, which, however suffers in cases of occlusions. Additionally, Song *et al.* [91] proposed a computationally expensive approach to the 6 DoF pose estimation problem that slides exemplar SVMs in the 3D space, while in [92] shape priors are learned by soft labelling Random Forest for 3D object classification and pose estimation. Lim *et al.* [27] achieved fine pose estimation by representing geometric and appearance information as a collection of 3D shared parts and objectness, respectively. Wu *et al.* [30] designed a model that learns the joint distribution of voxel data and category labels using a Convolutional

Deep Belief Network, while the posterior distribution for classification is approximated by Gibbs sampling. The authors in [25] tackle the 3D object pose estimation problem by learning discriminative feature descriptors via a CNN and then passing them to a scalable Nearest Neighbor method to efficiently handle a large number of objects under a large range of poses. However, compared to our work, this method is based on holistic images of the objects, which is prone to occlusions [6] and only evaluated on a public dataset that contains no foreground occlusions.

Hypotheses verification is employed as a final refinement step to reject false detections. Aldoma *et al.* [22] proposed a cost function-based optimization to increase true positive detections. Fioraio *et al.* [93] showed how single-view hypotheses verification can be extended to multi-view ones in order to facilitate SLAM through a novel Bundle adjustment framework. Buch *et al.* [94] presented a two-stage voting procedure for estimating the likelihood of correspondences, within a set of initial hypotheses, between two 3D models corrupted by false positive matches.

Regarding active vision, a recent work presented by Jia et al. [16] makes use of the Implicit Shape Model combined in a boosting algorithm to plan the next-best-view for 2D object recognition, while Atanasov *et al.* [95] proposed a non-myopic strategy using POMDPs for 3D object detection. Wu *et al.* [30] used their generative model based on the convolutional network to plan for the next-best-view but is limited in the sense that the holistic image of the object is needed as input. Since previous works are largely dependent on the employed classifier, more related to our work is our Active Random Forests framework (Chapter 4) which, however (similar to [96]) requires the holistic image of an object to make a decision, making it not appropriate for our patch-based method.

5.3 Single Shot-based 6D Object Detection

5.3.1 State of the art Hough Forests Features

In the literature some of the most recent 6D object detection methods use Hough Forests as their underlying classifier. In [5] simple two pixel comparison tests were used to split the data in the tree nodes, while the location of the pixels could be anywhere inside the whole object area. In our experiments, we also added the case where the pixel tests are restricted inside the area of an image patch. A more sophisticated feature for splitting the samples was proposed by Tejani et al. [6] who used a variant of the template based LineMOD feature [20]. In comparison with the above custom-designed features, we use Sparse Autoencoders to learn an unsupervised feature representation of varying length and layers. Furthermore, we learn features over depth-invariant RGB-D patches extracted from the objects, as described below.

5.3.2 Patch Extraction

Our approach relies on 3D models of the objects of interest[1]. We render synthetic training images by placing a virtual camera on discrete points on a sphere surrounding the object. In traditional patch-based techniques [21], the patch size is expressed directly in image pixels. In contrast, we want to extract depth invariant, or size-normalized 2.5D patches that cover the same area of the object regardless of the object distance from the camera, similar to [97]. First, a sequence of patch centers $c_i, i = 1..N$ is defined on a regular grid on the image plane. Using the depth value of the underlying pixels these are back-projected to the 3D world coordinate frame, i.e. $\bar{c}_i = (x, y, z)$. For each such 3D point $\bar{c}_i$ we define a planar patch perpendicular to the camera, centered at $\bar{c}_i$ and with dimensions $d_p \times d_p$, measured in meters, which is subdivided into $V \times V$ cells. Then, we back-project the center of each cell to the

[1]We obtained the 3D models of our objects using 123DCatch [35]

corresponding point on the image plane, to compute its RGB and depth values via linear interpolation[2]. Depth values are expressed with respect to the frame centered at the center of the patch (Fig. A.6). Also, we truncate depth values to a certain range to avoid points not belonging to the object. Depth-invariance is achieved by expressing the patch size in metric units in 3D space. From each training image we extract a collection of patches $\mathbf{P}$ and normalize their values to the range $[0, 1]$. The elements corresponding to the four channels of the patch are then concatenated into a vector of size $V \times V \times 4$ (RGBD channels) and are given as input to the Sparse Autoencoder for feature extraction.

5.3.3 Unsupervised Feature Learning

We learn unsupervised features using a network consisting of stacked, fully connected Sparse Autoencoders, in a symmetric encoder-decoder scheme. An autoencoder is a fully connected, symmetric neural network, that learns to reconstruct its input. If the number of hidden units are limited or a small number of active units is allowed (sparsity), it can learn meaningful representations of the data. In the simplest case of one hidden layer with F units, one input (x) and one output (y) layer of size N, the Autoencoder finds a mapping $f : \mathbf{R}^N \rightarrow \mathbf{R}^F$ of the input vectors $x \in \mathbf{R}^N$ as:

$$f = sigm(Wx + b) \tag{5.1}$$

The weights $W \in \mathbf{R}^{F \times N}$ and the biases $b \in \mathbf{R}^F$ are optimized by back-propagating the reconstruction error $||y - x||_2$. The average activation of each hidden unit is enforced to be equal to ρ, a sparsity parameter with a value close to zero. The mapping f represents the features given as input to the classifier in the next stage. We can extract "deeper" features by stacking several layers together, to form an encoder-decoder symmetric network as shown in Fig. A.6. In this case, the features

[2]The cell values calculation can be done efficiently and in parallel using texture mapping in gpu.

are extracted from the last layer of the encoder (i.e. middle layer). In experiments, we use one to three layers in the encoder part, and analyse the effect of several parameters of the architecture on the pose estimation performance, such as the number of layers, the number of features and layer-wise pre-training [76].

5.3.4 Pose Estimation

During training, we extract patches from training images of objects and use the trained network to extract features from object patches, that form a feature vector $\mathbf{f} = \{f_1, f_2, ..., f_F\}$. These vectors are annotated using a vector $\mathbf{d}$ that contains the object class, the pose of the object in the training image and the coordinates of the patch center expressed in the object's frame, i.e. $\mathbf{d} = \{class, yaw, pitch, roll, x, y, z\}$. The feature vectors along with their annotation are given as input to the Hough Forest. We propose three different objective functions: entropy minimization of the class distribution of the samples, entropy minimization of the $\{yaw, pitch, roll\}$ variables, and entropy minimization of the $\{x, y, z\}$ variables. Reducing the entropy towards the leaves, has the effect of clustering the training samples that belong to the same class and having similar position and pose on the object. More details on the computation of these entropies can be found in [70] and in Sec. 4.3. The objective function used is randomly selected in each internal node and samples are split using axis aligned random tests. The leaf nodes store a histogram of the observed classes of the samples that arrived, and a list of the annotation vectors $\mathbf{d}$. During testing we extract patches from the test image with a stride s and pass them through the forest, to reach the corresponding leaf. We create a separate Hough voting space (6D space) for each object class, where we accumulate the votes of the leaf nodes. Each vector $\mathbf{d}$ stored in the leafs, casts a vote for the object pose and its center to the corresponding Hough space. The votes are weighted according to the probability of the associated class stored in the leaf. Object hypotheses are subsequently obtained by estimating the modes of each Hough space. Each mode can be found using non-maxima suppression and is assigned a score equal to the voting weight of the mode.

Each object has its own Hough space for voting. The dimension of the Hough space is 6, 3 for object location and 3 for pose. Extracting the modes in such a high dimensional space is not efficient due to the high sparsity of the data and the computational complexity. We have experimented with three different implementations of the voting space: a) direct 6D voting, b) 3D voting in $\{x, y, z\}$, with each extracted mode subsequently voting in the 3D space $\{yaw, pitch, roll\}$, and c) voting in 2D space $\{x, y\}$ with the extracted modes subsequently voting in $\{z\}$, and the same process repeated in $\{yaw, pitch\}$ and then in $\{roll\}$. The most efficient in terms of time complexity and accuracy was the latter, which was used in our experiments. The main reason is that when computing the voting in $x - y$ space only, you can immediately localise the objects in the scene, and any false positives that may occur due to leaving out z, can be discarded in the second pass on z alone (the modes in 6D space, are also modes in any 2D or 1D projected space). After localising the objects, it is faster to vote in angle space only. The same logic applies to the angle space as well. The order of voting (i.e. $x - y$ first) in localization was chosen because the $x - y$ plane of the depth camera gives more information than any other axis-aligned plane. The order of 2D projection for the pose estimation (i.e. first voting in $\{yaw, pitch\}$ and then in $\{roll\}$) was proven experimentally that does not affect the results. Finally, the quantization of the Hough sapce is 5mm for X, Y and Z, and 1 degree for yaw, pitch and roll.

5.4 Next-Best-View Prediction

When detecting static objects, next-best-view selection is often achieved by finding the viewpoint that minimizes the expected entropy, i.e the uncertainty of the detection in the new viewpoint. There have been various methods proposed for com-

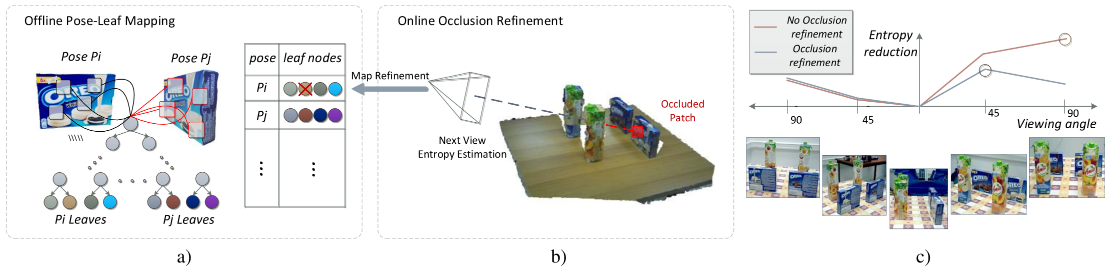


Figure 5.2: a) Offline construction of the pose-to-leaf mapping, b) Online occlusion refinement of the mapping, c) example of the effect of occlusion refinement in entropy estimation.

puting the entropy reduction [95, 96] including our Active Random Forest framework (Chapter 4). Hough Forests can facilitate the process since they store adequate information in the leaf nodes that can be used for predicting such reduction. The entropy of a hypothesis in the current view can be computed as the entropy of the samples stored in the leaf nodes that voted for this hypothesis. That is:

$$H(h) = \sum_{l_h} H(\mathbf{S}_{l_h}) \tag{5.2}$$

where l_h is a leaf voted for hypothesis h, and $\mathbf{S}_{l_h}$ the set of samples in these leaves. If the camera moves to viewpoint v, the reduction in entropy we gain is:

$$r(v) = H(h) - H(h_v) = \sum_{l_h} H(\mathbf{S}_{l_h}) - \sum_{l_{h_v}} H(\mathbf{S}_{l_{h_v}}) \tag{5.3}$$

where h_v is the hypotheses h as would be seen from viewpoint v. In order to measure the reduction in entropy, we need to calculate the second term of the right side of equation (5.3), which requires to find the leaf nodes that should be reached from the viewpoint v. Since we want to compute the reduction before actually moving the camera, we can simulate h_v by rendering the object placing a virtual camera at v, give the image as input to the forest and collect the resulting leaves. However this can be done more efficiently avoiding the rendering phase (contrary to [96]): we save offline a mapping from object poses (discrete camera views) to leaf nodes using the training data as shown in Fig. 5.2a. Given a 6 DoF hypothesis and this mapping, we can predict which leaf nodes of the forest are going to be reached if we move the camera to viewpoint v. Because of the discretization of poses in the map index, we choose the view in the mapping that is closer to the camera viewpoint we want to examine. Thus, the next-best-view v_{best} is calculated as:

$$v_{best} = \underset{v}{\operatorname{argmax}}\, r(v) = \underset{v}{\operatorname{argmin}}\, H(h_v) \tag{5.4}$$

In case of two or more uncertain hypotheses, the reduction in entropy is averaged in the new viewpoint. Also, to account for the cost of the movement, the reduction can be normalized by the respective cost.

In the general case of multiple objects present in the scene with cluttered background, we can further refine the entropy prediction to account for occlusions. In our previous formulations, we made the assumption that, from a view v the object is clearly visible. However, due to other objects present in the scene, some part or the whole object we are interested in, may be occluded (Fig. 5.2b). In this case our estimated entropy reduction is not correct. What we need to do is to exclude from the entropy calculation the samples in the leaves that are going to be occluded. More formally:

$$H(h_v) = \sum_{l_{h_v}} H(\mathbf{S}_{l_{h_v}} \setminus \mathbf{S}^{occ}_{l_{h_v}}) \tag{5.5}$$

where $\mathbf{S}^{occ}_{l_{h_v}}$ are the samples that would be occluded in viewpoint v. In order to determine this set, first, we incrementally update the 3D point cloud of the scene when the camera moves. Then, we project the $\{x, y, z\}$ coordinates of an annotated sample of a leaf onto the acquired scene as shown from view v, and estimate if it is going to be occluded or not. Figure 5.2c shows an example of our dataset, where there are two similar objects *oreo* we want to disambiguate. From the view -90 degrees to 0 it is difficult to understand the difference, while from 45 degrees onwards the difference becomes clearer. However, from the view of 90 degrees the objects of interest become occluded. Calculating the entropy as described above, we get the true reduction in entropy which is lower than in the 45 degree case.

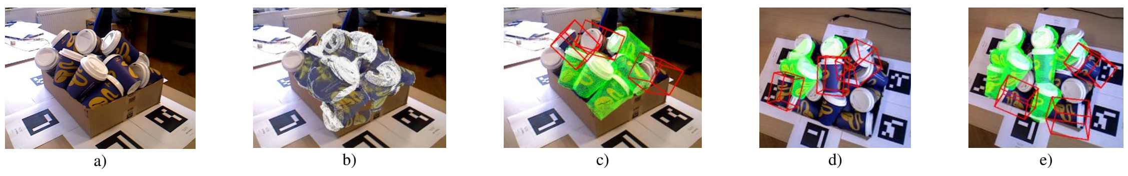


Figure 5.3: Example of hypotheses verification and active camera movement. a) Input test image, b) complete set of hypotheses overlaid on the image, c) hypotheses verification refinement, d) active camera movement, e) re-estimating hypotheses.

Another example of the complete pipeline is shown in Fig. 5.3. Given an image (Fig. 5.3a) we extract the hypotheses from the Hough voting space (Fig. 5.3b). Using the optimization described in next section 5.5 we refine this by selecting the best subset (Fig. 5.3c). The best solution does not include the objects shown in red box. However, a solution containing these hypotheses, but not well aligned with the scene due to occlusion, has a similar low cost with the best one. Being able to move the camera, we find the next-best-view as described above according to the uncertain hypotheses and change the viewpoint of the camera (Fig. 5.3d). We can re-estimate a new set of hypotheses (Fig. 5.3e) with some hypotheses still being uncertain (but keeping good ones above a threshold) and the same process is repeated.

5.5 Hypotheses verification and joint registration

Given the assumption that only one object instance exists in the scene, like in [20, 5], the pose estimation described in 5.3.4 can directly provide a solution to the problem, by considering only the Hough voting space of the object of interest and extract the most voted modes. In the general case of multiple object instances of multiple classes present in the scene, however, there may be a set of hypotheses, from which some are conflicting and some explain the scene better than others. Thus, the objective is to select the subset of all possible hypotheses produced by the pose estimator, that best explains the scene. We adopt the global optimization approach of [22], which we improve mainly by modifying the cost terms and improving the objective function and its optimization. Given a set of hypotheses $\mathbf{H}$ and a vector $\mathbf{X} = \{x_1, x_2, ..., x_N\}$

of boolean variables, which indicate that hypotheses h_i is valid or not, the objective is to find $\mathbf{X}$ that minimizes a cost function $C(\mathbf{X})$. Below we formulate the various terms of this objective function.

For each hypothesis, we render the 3D model of the object in the scene and exclude the parts that are occluded (i.e. that lay behind any scene surface) as they cannot provide any information about the scene fitting. For the remaining points p of each model, we find their nearest neighbour q in the scene and assign a fitting score, according to the following four cues:

a) local fitting: For each pair p and q there are three terms measuring their similarity: a) C_{11} is the normalized distance $\frac{||p-q||}{p_e}$, but when exceeds 1, the fitting score is not taken into account and the point is considered an outlier, b) C_{12} measures the similarity, i.e. the dot product of the normals of the points, truncated to $[0, 1]$, c) we add an extra term not in [22] defined as:

$$C_{13} = max(\frac{|R_p - R_q|}{255}, \frac{|G_p - G_q|}{255}, \frac{|B_p - B_q|}{255}) \tag{5.6}$$

that measures the similarity of p and q in color channels, with the max function used because the color significantly changes even when one of the channel changes. Finally, we set $C_1 = (C_{11} + C_{12} + C_{13})/3$.

b) inliers: Inliers are the points for which $||p - q|| \leq p_e$ and are measured as a fraction over the total visible points: $C_2 = p_{in}/p_{tot}$

c) conflicting inliers: Inliers from different hypotheses may share the same nearest neighbour in the scene. This indicates a conflict among the hypotheses that needs to be penalized. This cue C_3 is measured as a fraction of conflicting inliers over the total inliers of the involved objects.

d) clutter term: As explained in [22], we segment the scene into smooth regions and penalize the points of a region that do not belong to the same hypothesis with the other points of that region (or do not belong to any hypotheses at all). Cue C_4 is also measured as a fraction of penalized points over total points in a region.

Contrary to [22], we defined every term in the range $[0, 1]$. The final cost function $C(\mathbf{X})$ is defined as:

$$C(\mathbf{X}) = (a_1 C_1 + a_2 C_2) - (a_3 C_3 + a_4 C_4) \tag{5.7}$$

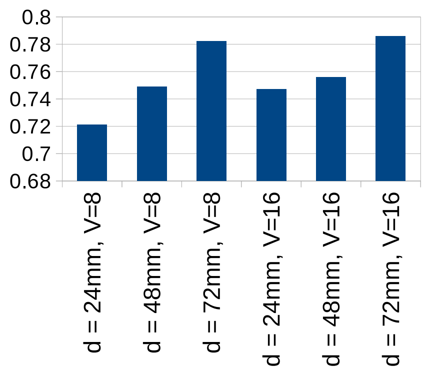
with each cue calculated only for the hypotheses that $x_i = 1$. We found that although each term has the same range, each one has a different relative importance and common range of values. Therefore, unlike [22], we put a different regularizer in each term, which is found using cross-validation. Furthermore, the solution space of the optimization can be further reduced. We observe that apart from C_3, all other cues can be computed for each object independently. If we split the set of hypotheses $\mathbf{H}$ into non-intersecting subsets H_i, then each subset can be optimized independently as well, decomposing the optimization problem into smaller ones and therefore reducing the time and complexity of the solution. Each smaller subset is subsequently optimised as in [22].

5.6 Experiments

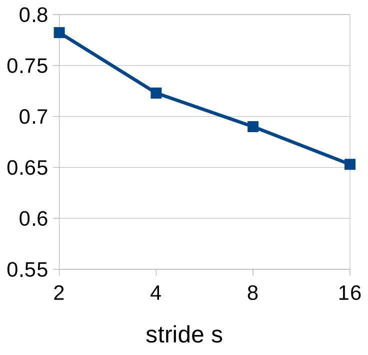
The experiments regarding the patch size and feature evaluation were performed on a validation set of our own dataset. Object detection accuracy is measured using the F1-score and is averaged over the whole set of objects. When comparing with the state of the art methods, we use the public datasets and the evaluation metrics provided by the corresponding authors. When evaluating on our own dataset, we exclude the aforementioned evaluation set.

5.6.1 Patch Size Evaluation

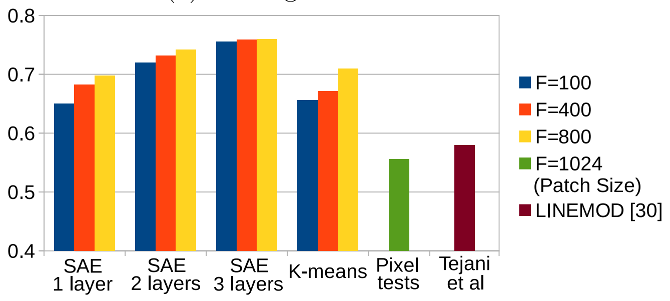
A patch in our framework is defined over 2 parameters: d_p is the actual size measured in meters, and $V \times V$ is the number of cells a patch contains, which can be



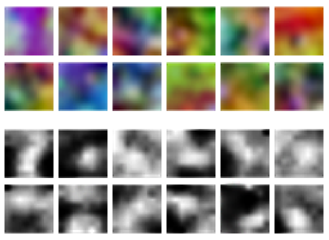
(a) Patch-grid size



(b) stride



(c) feature evaluation



(d) 1st layer filters

Figure 5.4: Patch extraction parameters

considered as the patch resolution. We used six different configurations shown in Fig. 5.4(a). The maximum patch size used was limited to the 2/3 of the smallest object dimensions. The network architecture used for patch-size experiments is 2 layers (the encoder part) of 1000 and 400 hidden units respectively. Fig. 5.4(a) shows that an increase in the patch size significantly increases the accuracy, while on the other hand, an increase of the resolution offers a slight improvement, and that comes at the expense of additional computational cost. Another important factor is the stride s during patch extraction. Fig. 5.4(b) shows that the smaller the stride the more accurate the detection becomes.

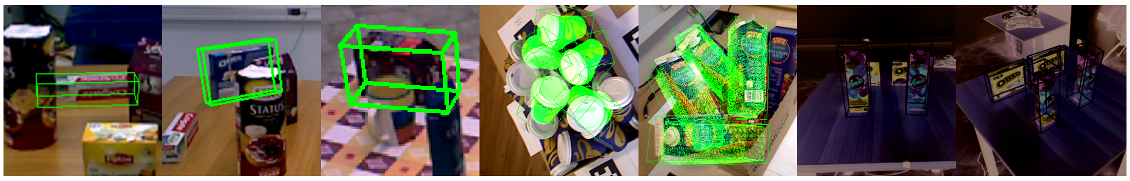
5.6.2 Feature Evaluation using Hough Forests

In order to evaluate our unsupervised feature we created 9 different network configurations to test the effect of both the number of features and the number of layers on the accuracy. We used 1-3 layers as the encoder of the network with the last layer of the encoder forming the feature vector used in the Hough Forest. We varied the length of this feature vector to be 100, 400 and 800. When we use 2 layers, the first has 1000 hidden units, while when we use 3 layers, the first two have 1500 and 1000 hidden units respectively. The patch size used for these experiments is $d_p = 48mm$ with $V = 16$, creating an input vector of 1024 dimensions. Using the same Hough Forest configuration, we evaluate three state of the art features: a) the feature of [6], a variant of LineMOD[20] designed for Hough Forests, along with its split function, b) the widely used pixel-tests [5] and c) K-means clustering, the unsupervised single-layer method that performed best in [81][3] with 100, 400 and 800 clusters. Pixel-tests have been conducted inside the area of a patch for comparison purposes, however in the next subsection we compare the complete framework of [5] with ours. Results are shown in Fig. 5.4(c). The 3-layer Sparse Autoencoder shown the best performance. Regarding the Autoencoder, we notice that the accuracy increases if more features are used, but when the network becomes deeper, the difference diminishes. However, it can be seen that deeper features significantly outperform shallower ones. K-means performed slightly better than single-layer SAE, while pixel-tests had worse performance. The feature of [6] had on average worse performance than Autoencoders and K-means, which is due to low performance on specific objects of the datasets. We further provide a visualization of the filters of the first layer learned by a network with a 3-layer encoder (Fig. 5.4(d)). The first two rows are filters in the RGB channel, where it can be seen a bias towards the objects used for the evaluation. Filters in the depth channel resemble simple 3D edge and corner detectors. Last, we have tried to pre-train each layer as in [76], without significantly influencing the results.

5.6.3 State of the Art Evaluation

In the experiments described in this subsection, we used an encoder of 3 layers with 1500, 1000 and 800 hidden units, respectively. The patch used has $V = 8$

[3]We used the K-means (triangle) as described in [81]



(a) Colgate (b) Oreo (c) Softkings (d) Coffecup (e) Juice (f) Camera (g) Joystick

Figure 5.5: Qualitative results of our framework. Image 5.5(g) is the next best view of image 5.5(f).

and $d_p = 48mm$, which was found suitable for a variety of object dimensions. The forests contain four trees limiting only the number of samples per leaf to 30. For a fair comparison, we do not make use of joint registration or active vision except when specifically mentioned.

We tested our solution on the dataset of [5], which contains 20 objects and a set of images regarded as background. The test scenes contain only one object per image, there is no occlusion or clutter, and are captured with different illumination from the training set, so one can check the generalization of a 6 DoF algorithm to different lighting conditions. To evaluate our framework we extracted the first $K = 5$ hypotheses from the Hough voting space and chose the one with the best local fitting score. The results are shown in Table 5.1 where for simplicity we show only 6 objects and the average over the complete dataset. Authors provided comparison with [20] only with one object, because they could not get meaningful results using their method. This dataset was generally difficult to evaluate, mainly because some pose annotations were not very accurate, resulting in having some better estimations from the ground truth exceeding the metric threshold of acceptance. Our method showed that it can generalize well on different lighting conditions, even without the need of modifying the training set with Gaussian noise as suggested in [5].

We have also tested our method on the dataset presented in [6], which contains multiple objects of one category per test image, with much clutter and some cases of occlusion. Authors adopted one-class training, thus, avoiding background class images during training. For comparison, we followed the same strategy. Since there are multiple objects in the scene, we extract the top $K = 10$ modes of the

Table 5.1: Results on the dataset of [5]

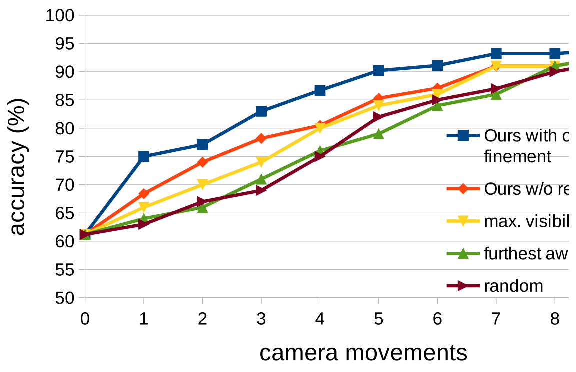
Object	[20] (%)	[5] (%)	**Our** (%)
Audiobox		**75.4**	71.5
Carry Case		**95.9**	90.7
Dish Soap		**100**	**100**
Helmet		**77.6**	74.5
Hole Puncher		**98.1**	94.3
Pump		**69.3**	67.4
Toolbox		99.5	**100**
Toy (Battle Cat)	70.2	91.8	**92.4**
Toy (Panthor)		**96.9**	94.2
Toy (Stridor)		94	**94.3**
Stuffed Cat		**98.3**	94
Duck		81.6	**87.7**
Dwarf		**67.6**	65.6
Mouse		89.1	**90.1**
Owl		60.5	**90.27**
Elephant		94.7	**96.13**
Samurai		98.5	**99.6**
Sculpture 1		82.7	**89.5**
Sculpture 2		**100**	**100**
Avg.		88.2	**89.1**
Med.		**93.0**	92.4

$\{x, y, z\}$ Hough space, and for each mode, we extract the $H = 5$ modes of the $\{yaw, pitch, roll\}$ Hough space and put a threshold on the local fitting of the final hypotheses to produce the PR curves. Table 5.2 shows the results in the form of F1-score (metric authors used) for each of the 6 objects. The results of methods [20, 89] are taken from [6].

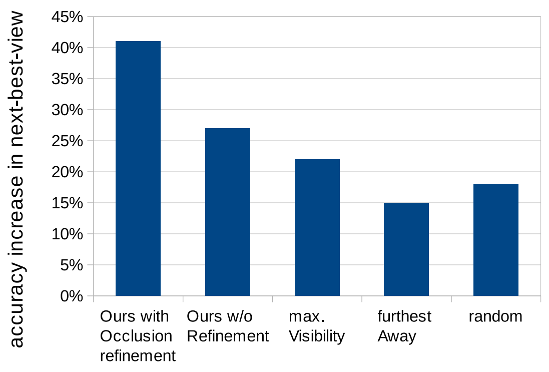
In this dataset we see that our method significantly outperforms the state of arts, especially regarding the *Camera* which is small and looks similar with the back-

Table 5.2: Results on the dataset of [6]

Object	[20]	[89]	[6]	**Our**
	F1 score			
Coffee Cup	0.819	0.867	0.877	**0.932**
Shampoo	0.625	0.651	**0.759**	0.735
Joystick	0.454	0.277	0.534	**0.924**
Camera	0.422	0.407	0.372	**0.903**
Juice Carton	0.494	0.604	**0.870**	0.819
Milk	0.176	0.259	0.385	**0.51**
Average	0.498	0.511	0.633	**0.803**



(a)



(b)

Figure 5.6: a) Results on active vision on our crowded dataset scenes, b) Results on active vision on scenes with objects arranged.

ground objects, and the Joystick, which has a thin and a thick part. Our features showed better performance on *Milk* that contains other distracting objects on it. It is evident that our learnt features are able to handle a variety of object appearances with stable performance and at the same time being robust to destructors and occluders. Note that without explicitly training a background class, all the patches in the image are classified as belonging to one of our objects. While [6] designed a specific technique to tackle this issue, our features seem informative enough to produce good modes in the Hough spaces.

We have also tested [6] and [5] on our own dataset. We also tried [20], but although we could produce the reported results on their dataset, we were not able to get meaningful results on our dataset and so we do not report them. This is mainly because this method is not intended to be used in textured objects with simple geometry. We provide results both with and without using joint object optimization. Our dataset contains 3D models of six training objects, while the test images may contain other objects as well. More on our dataset and evaluation can be found in the supplementary material. Table 5.3 shows the results on our database. The work of [5] is designed to work only with one object per image and it is not evaluated on the bin-picking dataset. Our method outperforms all others even without joint optimization, but we can clearly see the advantages of such optimization on the final performance.

Table 5.3: Results on our dataset

Object	[6]	[5]	**Our**	**Our joint optim.**
scenario 1 (supermarket objects)				
amita	26.9	60.8	**64.3**	71.2
colgate	22.8	11.1	**26.1**	28.6
elite	10.1	71.9	**74.9**	77.6
lipton	10.5	26.9	**56.4**	59.2
oreo	26.9	44.4	**58.5**	59.3
softkings	26.3	26.9	**75.5**	75.9
scenario 2 (bin picking)				
coffeecup	31.4	-	33.5	36.1
juice	24.8	-	25.1	29

5.6.4 Active Vision Evaluation

We tested our active vision method on our dataset, using two different types of scenes. One is the crowded scenario used for single-shot evaluation, and the other depicts a special arrangement of objects, one behind the other in rows, that is commonly seen in a warehouse (Fig. 5.2). All results take into account all the object hypotheses during the next-best-view estimation. We compare our next-best-view prediction with and without occlusion refinement with three other baselines [96]: a) maximum visibility (selecting a view that maximizes the visible area of the objects), b) furthest away (move the camera to the furthest point from all previous camera positions), c) move the camera randomly.

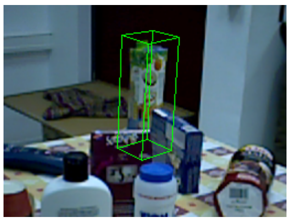
In the crowded scenario, we move the camera 10 times, measuring in each view the average pose estimation accuracy of the objects present in the scene (Fig. 5.6(a)). We see that our method without occlusion refinement slightly outperforms the maximum visibility baseline because usually the maximum reduction in entropy occurs when there is maximum visibility. Using occlusion refinement, however, we get a much better estimation of the entropy that is depicted in the performance.

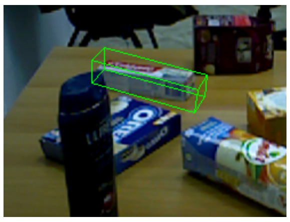
When the objects are specially arranged, we were interested in measuring the increase in accuracy only in the single next-best-view, i.e. we allow the camera to move only once for speed reasons. This experiment (Fig. 5.6(b)) makes very clear the importance of tackling occlusion when estimating the expected entropy. Our method with occlusion refinement was consistently finding the most appropriate view, whereas without this step, the next-best-view was usually the front view, with the objects behind being occluded.

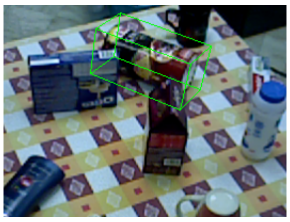
Regarding the computational complexity of our single shot approach, training 3 layers of 800 features with 10^4 patches for 100 epochs takes about 10mins on GPU. Our forest was trained with a larger set of $5 \cdot 10^6$ patches. Thanks to our parallel implementation, we train a tree on an i7 CPU in 90 mins, while [6] and [5] require about 3 and 1 days, respectively. During testing, the main bottleneck is the Hough voting and mode extraction that takes about 4-7secs to execute, with an additional 2secs if joint optimization is used for 6 objects. Other methods need about 1sec.

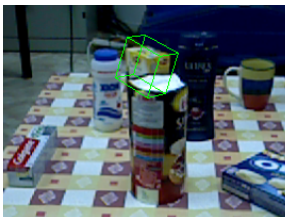
5.6.5 More Qualitative Results

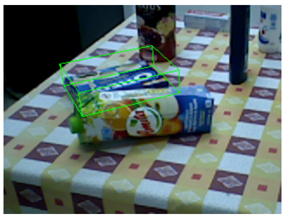
In this subsection we provide some qualitative results of our method. Fig. 5.7(a) - 5.7(f) show results of single object detection in scenario 1. Fig. 5.7(g) and 5.7(h) show examples of joint object registration using global optimization. Results on scenario 2 are shown in Fig. 5.7(i)-5.7(k) where again joint registration is used. Fig. 5.8 shows the detection results of two state of the art detectors ([6, 5]) on single view single object detection. Two objects are shown, amita and softkings. Finally, in Fig. 5.10 there are two examples of next-best-view estimation results. These scenes help in qualitatively evaluating a next-best-view strategy. However, the results of the paper regarding active vision were measured using all images in both scenarios in our dataset. More results and comparisons side-by-side with state of the art methods can be found on the video attached to the supplementary material.

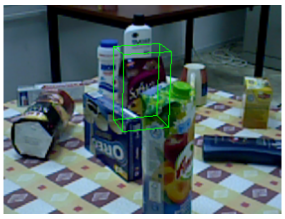

(a) Amita

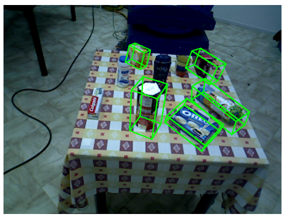

(b) Colgate

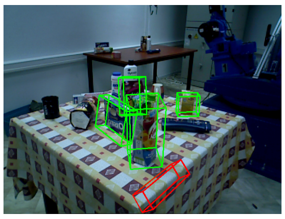

(c) Elite

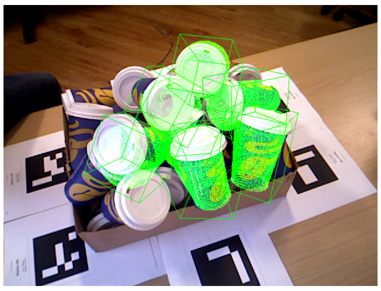

(d) Lipton

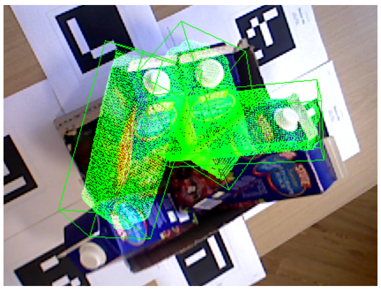

(e) Oreo


(f) Softkings


(g) Joint Registration


(h) Joint Registration


(i) Coffee cups


(j) Juice

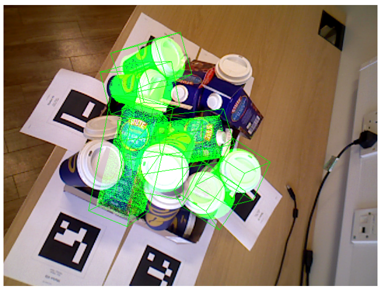
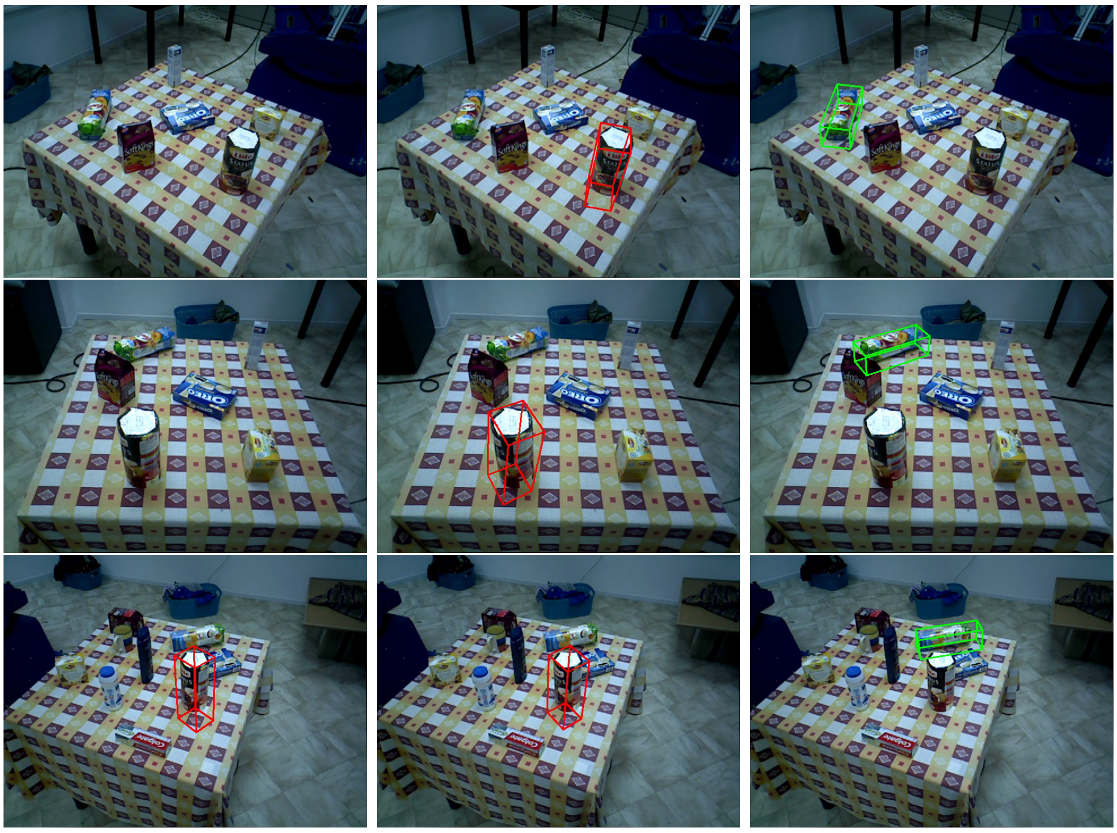

(k) Both objects

Figure 5.7: Qualitative results on our dataset. a-h) Scenario 1, i-k) Scenario 2 (bin picking)


(a) Amita Detection

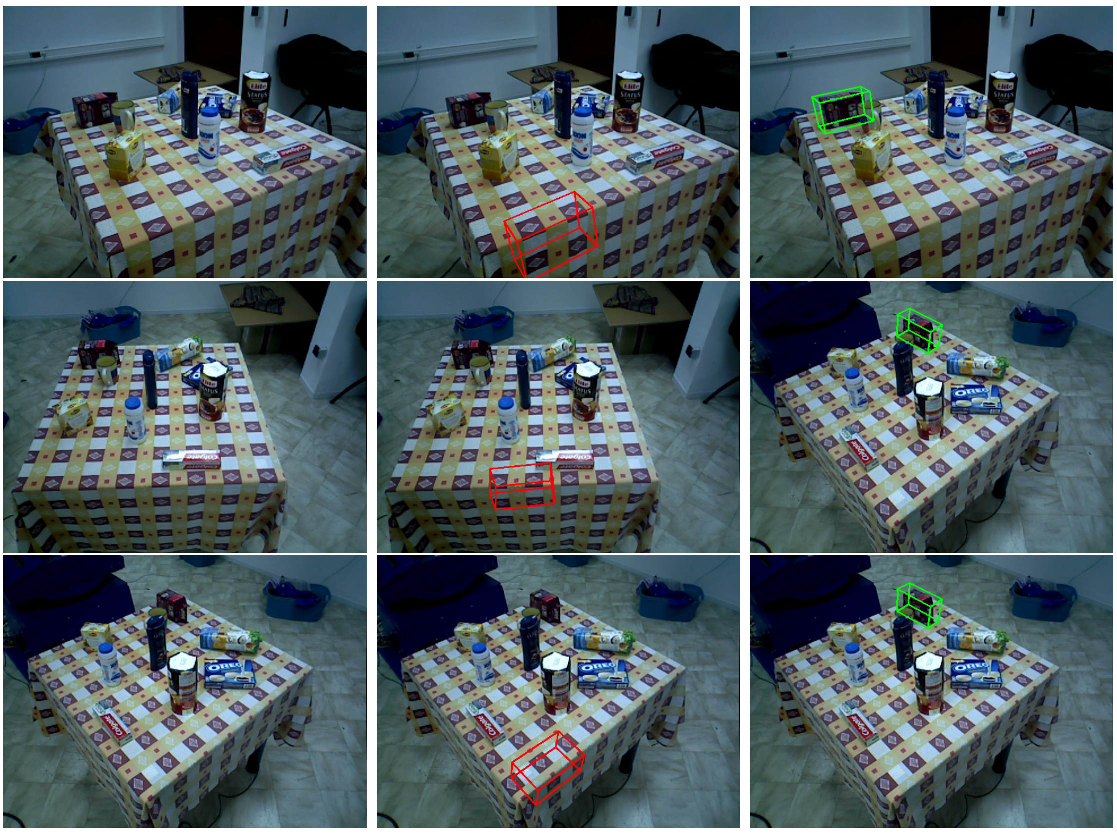

(b) Softkings Detection

Figure 5.8: Comparisons with state of the art on single object single view detection. First column is the results of [6], the second column is the results of [5] and the third column is the results of our method. When no detection is shown, the corresponding method did not produce any detection result.

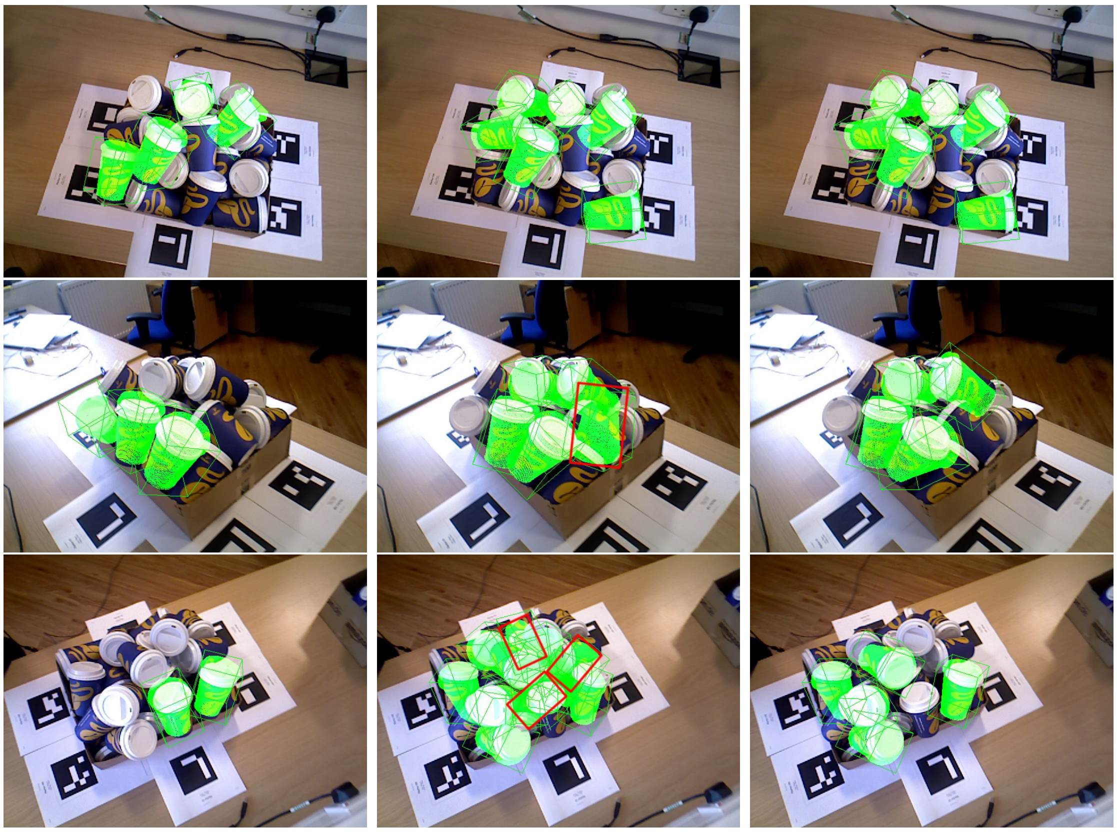
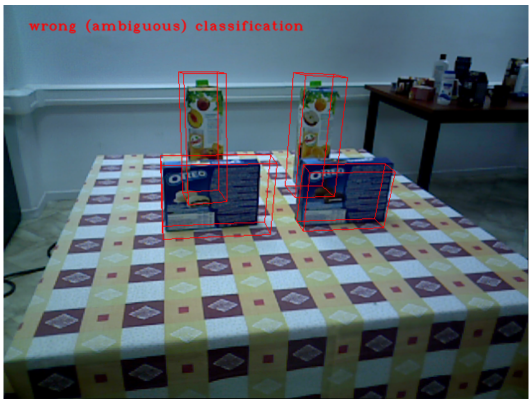
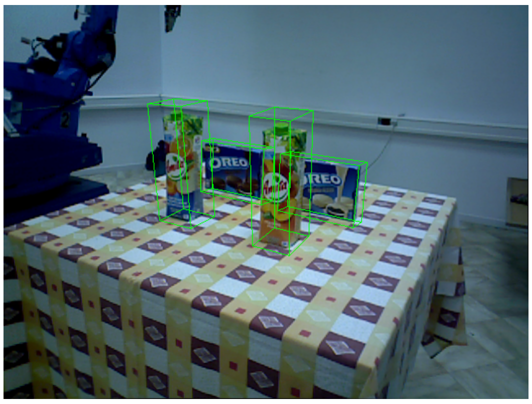


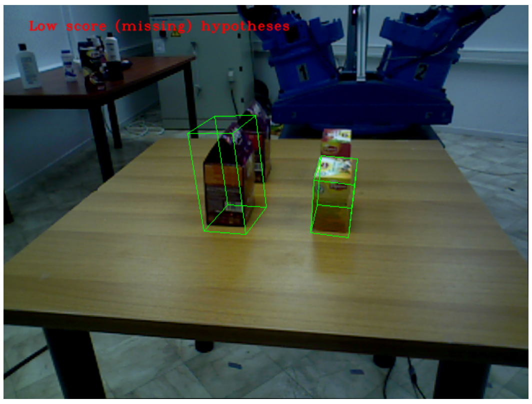
Figure 5.9: State of the art comparison in bin picking multi-instance detection. First column shows results of [6], second column shows the results of our method without using joint registration, and the third column shows the results of our method using joint registration.



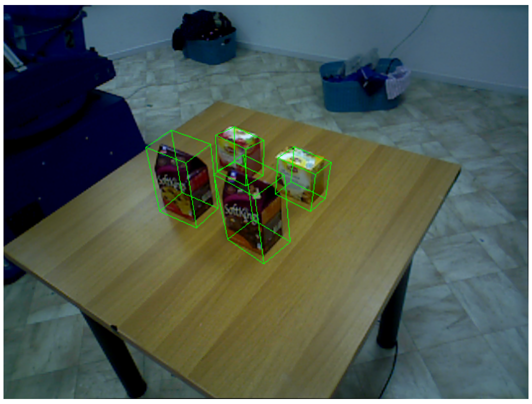
(a) Classes cannot be disambiguated from this viewpoint



(b) Next best view of (a)



(c) Hypotheses of hidden objects have low score



(d) Next best view of (c)

Figure 5.10: Active Next Best View estimation using occlusion refinement in order to disambiguate the four objects.

Chapter 6

Siamese Regression Networks

6.1 Overview

In this chapter we present our Siamese Regression Network, a network that is able to efficiently support regression over angle space for 3D object pose estimation. Regressing directly the object pose has the advantage of fast inference during testing time, avoiding the hough voting and mode estimation of our previous method (Chapter 5). However, our network works on holistic images rather than patches, which were proved to be robust to occlusions. For this reason, we introduced another objective function term to explicitly tackle occlusions, given proper training data. In the following sections, we first give an overview of related work, and then we describe our Siamese Regression Network and the contributions led to successful object pose regression.

6.2 Related work

Recognizing and detecting objects along with estimating their 3D pose has received a lot of attention in the literature. Early works made use of pointclouds to facilitate

Point-to-Point matching [89, 90], while the advent of low-cost depth sensors [26, 98] provided additional data in favor of textureless objects. As mentioned earlier in Section 5.2 Hinterstoisser et al. designed a holistic template matching method (LINEMOD) based on RGB-D data, Tejani et al. [6] integrated LINEMOD into Hough Forests to tackle the problem of occlusions and clutter, and Brachmann et al. [5, 99] employ a new representation framework that jointly maps 3D object coordinates and class labels. Hodan et al. [100] presented a method that tackles the complexity of sliding window approaches via a fast-filtering technique followed by a voting procedure for hypotheses generation, while fine 3D pose estimation is performed via a stochastic, population-based optimization scheme. In turn, in [91] exemplar SVMs are slided in the 3D space to perform object pose classification based on depth images.

Deep learning has only recently found application to the 3D object pose estimation problem. Our work described in Chapter 5 suggested using a network of stacked sparse autoencoders to automatically learn features in an unsupervised manner that are fed to Hough Forests for 6D object pose recovery and next-best-view estimation. In [101] Johns et al. employed a CNN-based end-to-end learning framework for classification of object poses in the 3D space and next-best-view prediction. In turn, in [102] a CNN was used to learn projections of 3D control points for accurate 3D object tracking, while in [103] a CNN is utilized in a probabilistic framework to perform analysis-by-synthesis as a final refinement step for object pose estimation. In [25] 3D pose estimation is performed by a scalable Nearest Neighbor method on discriminative feature descriptors learned by a CNN.

To the best of our knowledge, this work presents the first CNN-based framework for regressing object poses in the continuous 3D space. The work of Kendall et al. [104] regresses camera poses in the continuous 3D space[1] but does not offer any end-to-end learning since it makes use of the pretrained GoogLeNet[7]. The method of Sun et al. [23] offers a learning framework that learns a new face representation by

[1]camera pose estimation is the inverse of object pose estimation

joint identification-verification. The work of Dosovitskiy et al. [105] try to estimate the optical flow using CNNs and a new layer that correlates feature vectors at different image locations to help regression. However the optical flow is determined by comparing two image frames that are captured sequentially, and therefore have similar appearance making the flow estimation a slightly easier regression task for NNs. As far as feature learning for 3D object pose estimation is concerned, our work shares similar ideas with the method of Wohlhart et al. [25] that learns feature descriptors with pairs and triplets. However, we argue that our learned features are pose-guided and as experiments prove, more discriminative, which in fact suggests that they are optimized for the particular task of 3D object pose estimation.

6.3 Siamese Regression Network

6.3.1 Object Pose Estimation Using Regression CNN

We first formulate the problem of object pose estimation as a regression problem. Let $x \in \mathbf{R}^{W \times H \times 4}$ be an RGBD (4 channels) image depicting a centered object having width W and height H. Pose estimation is the problem of learning a regressor $g : \mathbf{R}^{W \times H \times 4} \rightarrow \mathbf{R}^{M}$, where M is the dimensionality of the pose representation used. For example, Euler angles require 3 angles to be defined ($M = 3$) whereas quaternions suggest $M = 4$. Regressing euler angles directly can be problematic due to multiple problems such as periodicity [106], and the non-continuous nature of the euler angle space [104]. For example, poses that are very similar visually might be far away in euler angle space, making regression harder. Therefore, similar to previous work [104] on regressing camera pose, we also use the quaternion representation, which does not suffer from the same problems.

For the task of estimating the regression function g we train a convolutional neural network (CNN). We use the simple architecture similar to [25] that consists of 2 convolutional layers, and 2 fully connected layers (we have removed the max-pooling layers as we saw that they slightly degraded the performance). On top of that, we added another fully connected layer that outputs M units to estimate the object pose. If we consider the layer just before the last regression layer as the features learned by the network, we can describe the output of the our network as:

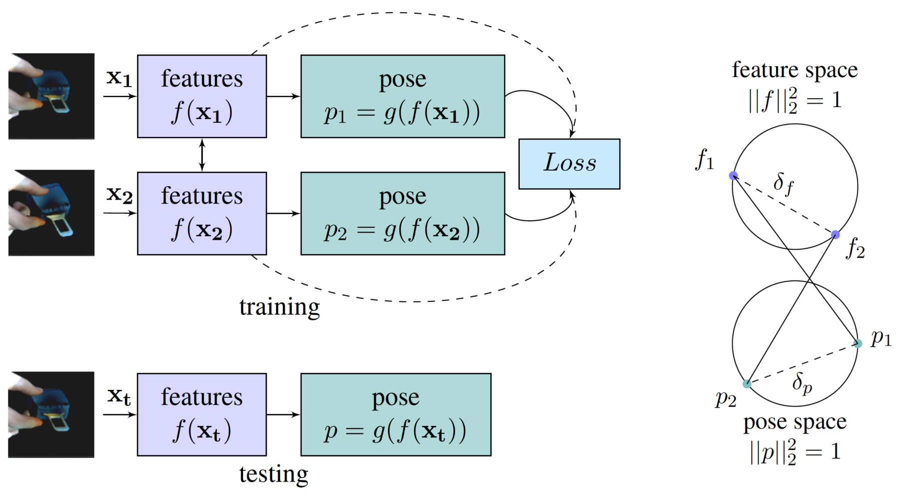
$$p = g(f(x)) \tag{6.1}$$

where $f(x)$ is the output of the feature layer, g the regression layer function and p is a pose vector returned by the network for the input image x. Given a training set that contains combinations of training samples of the form $\{x_i, y_i\}$, the most commonly used method of training a regression network is by minimizing the Mean Square Error (MSE) between the estimation $g(f(x_i))$ and the ground truth y_i and back-propagating the error. If we split the training set into mini batches of K samples each, the regression loss can be written as:

$$\ell_R = \sum_{n=1}^{K} ||g(f(x_n)) - y_n||_2^2 \tag{6.2}$$

6.3.2 Siamese Regression Objective

Previous work has shown that the feature layer f is able to learn representations that can be successfully applied in nearest neighbour matching [25] or face identification - verification [23]. However, end-to-end regression learning with CNN in angle space proved to be a very challenging task, with researchers resorting to indirect solutions, such as Nearest Neighbor template matching [25] or ad-hoc angle estimation methods like *arctan* [106]. Therefore, inspired by [23], we want to enhance the feature learning process by using additional information available during training, in order help the end-to-end regressor converge to a better minimum. Thus, our goal is to enforce



$\mathbf{x_1}$
features $f(\mathbf{x_1})$
pose $p_1 = g(f(\mathbf{x_1}))$
Loss
$\mathbf{x_2}$
features $f(\mathbf{x_2})$
pose $p_2 = g(f(\mathbf{x_2}))$
training
$\mathbf{x_t}$
features $f(\mathbf{x_t})$
pose $p = g(f(\mathbf{x_t}))$
testing
feature space $||f||_2^2 = 1$
f_1
δ_f
f_2
p_1
δ_p
p_2
pose space $||p||_2^2 = 1$

Figure 6.1: (left) Our training and testing architectures. We enforce a siamese architecture for regressing relative distance between feature and pose spaces. During testing, we extract a branch of the network, and use it for regression. (right) Illustration of our feature-guided pose regression loss. The loss seeks to associate distances in the $L2$ normalised feature space with the $L2$ normalised pose space.

a loss function in the feature layer f, in a way that the learned features are more appropriate and useful for the regression task in the last layer g.

In order to enforce a second loss function in this layer, we utilize the siamese architecture that has been very successful for learning non-linear feature embeddings with convolutional neural networks [107]. The siamese architecture consists of two (or more) branches of the same CNN that share weights and encode two inputs processed in parallel. Subsequently, a loss function can be introduced based on both outputs, which makes it possible to compare different samples of our training data passing through our network in a meaningful way.

Our study on the regression problem concluded that there is a relation between the feature and the angle space which helps a regression network layer perform much better. The relationship is the following: the euclidean distance between two sample images represented in feature space, should be maintained the same with the distance between the same samples as represented in angle space, during training. Fig. 6.1 shows an illustration of our idea. In order to enforce such relation we use a siamese network to pass through the network pairs of samples and apply an objective term on them. The pairs have the form: $\{x_1, y_1, x_2, y_2\}$ where x represent the raw input, and y the pose vector ground truth. We enforce the following loss function for feature guided regression

$$\ell_F = \sum_{n=1}^{K} \left| ||f(x_{n,1}) - f(x_{n,2})||_2^2 - ||y_{n,1} - y_{n,2}||_2^2 \right| \tag{6.3}$$

Intuitively, minimizing this loss enforces the $L2$ distance between the features in the sample pair, to be close to the $L2$ distance between the ground truth of their poses. In order to avoid weighting any of the above parts of the objective loss term, we normalize the output of the feature layer as well as the output of the pose layer to have unit norm (if using quaternions as pose representation, they already have unit norm). In fact, as we will show in experiments, this normalization has a positive effect in training angle regression even without using our extra feature term.

It should be mentioned that the siamese network is only used during training to help the regression task. During testing, only a single image produces a pose estimation without the need of providing a pair for the image.

6.3.3 Feature Guided Pose Regression

Combining the regression loss with the feature loss, we get

$$\mathscr{L} = \ell_R + \ell_F + \lambda \cdot ||w||_2^2 \tag{6.4}$$

where $\lambda \cdot ||w||_2^2$ is a term to regularise the weights of the convolutional neural network. By enforcing this loss in the proposed siamese regression network, we are able to simultaneously focus on both features that are able to work well in a nearest neighbour framework, and on the fully connected last layer that regresses the poses directly. Indeed, in our experiments we show that enforcing the feature term in the loss leads to better pose estimation in the final layer.

In Table 6.1, we describe the relationship between the two loss functions ℓ_R and ℓ_F, and the parts of the CNN weights that are updated. We note that the weights related to the feature learning, are updated using information from both losses, while the weights related with the pose regression, are only updated based on the ℓ_R loss.

6.3.4 Pose Guided Feature Learning

Despite the fact that the loss function of from Eq. 6.4 mainly aims to learn a better regressor for the pose in the final layer $p = g(f(x))$ of the convolutional network, it can be argued that the features that are learned in the $f(x)$ layer of the network can be more discriminative.

Previous work on 3D feature learning with siamese networks has focused on optimising the feature embeddings using triplets. Triplet training samples contain an anchor, a positive sample and a negative sample. The authors from [25] form the triplet by using two close views of the object as anchor and positive samples, and a view with significantly different pose as the negative. What they try to optimize is the anchor and the positive sample to be closer in feature space than the anchor and the negative one. They also use pairs of images of similar pose but different appearance and try to minimize their distance in feature space in order to learn features immune to different lightning conditions and noise.

On the other hand, our loss focuses on forcing the feature distance between a pair to be equivalent to the pose distance. Thus, it is more appropriate for a nearest neighbour framework, since the features are optimized to be relative to the pose distance. Indeed, in our experiments show that enforcing our loss from Eq. 6.4 results in features that are more suited for nearest neighbour matching.

We should note that using the objective function of [25] (eq. 6.5, x_i is similar to x_j and disimilar to x_k) instead of ℓ_F in order to help regression didn't work, with the network showing similar convergence behavior of the regression without using

Table 6.1: Our learning algorithm.

```
input: training set $\mathscr{X}$, CNN feature parameters $w_F$, CNN pose parameters $w_R$, learning rate $\eta$
for epoch e=1:N do
    sample M mini-batches from $\mathscr{X}$ using pairs with both similar and different poses
    for mini-batch b=1:M do
        $\nabla w_F = \frac{\partial \ell_F}{\partial w_F} + \frac{\partial \ell_R}{\partial w_F}$
        $\nabla w_R = \frac{\partial \ell_R}{\partial w_R}$
        update $w_R = w_R - \eta(e) \cdot w_R$
        update $w_F = w_F - \eta(e) \cdot w_F$
    end
end
output $w_R, w_F$
```

the extra term. This is a clue that our objective does indeed help regression, while at the same time regression helps building more discriminative features appropriate for pose estimation.

$$L = max(0, 1 - \frac{||f(x_i) - f(x_k)||_2}{||f(x_i) - f(x_j)||_2 + m}) \tag{6.5}$$

6.3.5 Siamese Pair Sampling

Considering a dataset of M training samples of the form $\{\mathbf{x_i}, \mathbf{y_i}\}$, there exist $\binom{M}{2}$ possible pairs to be used in the siamese training process described above. Since the number of pairs can become very large, several authors explored different techniques of sampling or mining hard negative pairs [108, 109].

Although we do not explicitly have positive and negative pairs since the training process is done in the same object, we approximate such pairs by spliting a batch of size K between pairs that are both close in the pose space, or have very large pose differences. We examined that random choice of pairs in terms of their pose difference perform inferior to our formation. Interestingly, forming batches using both similar and different pose pairs, is a factor that improves regression on its own, even without enforcing any constraints on such pairs. In the experiments we will show the relative performance gain of using well formed batches compared to regression with random batches and enforcing our sample pair objective.

6.3.6 Handling occlusions

Tackling occlusions, that is estimating the object pose when a major part of the object is missing or occluded, requires features that are robust in such conditions and one should explicitly enforce this property. We note that the form of Eq. 6.4 makes it convenient to support building such features: if we generate training samples with the object occluded, and using its annotation render a *clean* object having the same pose, we can enforce a similar term between the occluded and the clean images:

$$\ell_{oc} = \Big| ||f(x_{occluded}) - f(x_{clean})||_2^2 - ||y_{occluded} - y_{clean}||_2^2 \Big| \tag{6.6}$$

where $x_{occluded}$ and x_{clean} are images depicting occluded and clean objects respectively. Fig. 6.3 in the experiments section shows examples of such images. This term can be added to the $\mathscr{L}$ loss in order to tackle the severe occlusion problem. It should be noted that eq. 6.6 is not a new equation, but rather the equation 6.4 with additional pairs as inputs.

6.4 Experiments

Our convolutional regressor is a simple convolutional neural network with similar architecture to [25] that has the following architecture: $\{Input(4, W, H) - Conv(16, 8, 8) - Conv(7, 5, 5) - FC(256) - FC(D) - FC(4)\}$ where $Conv(N, K, K)$

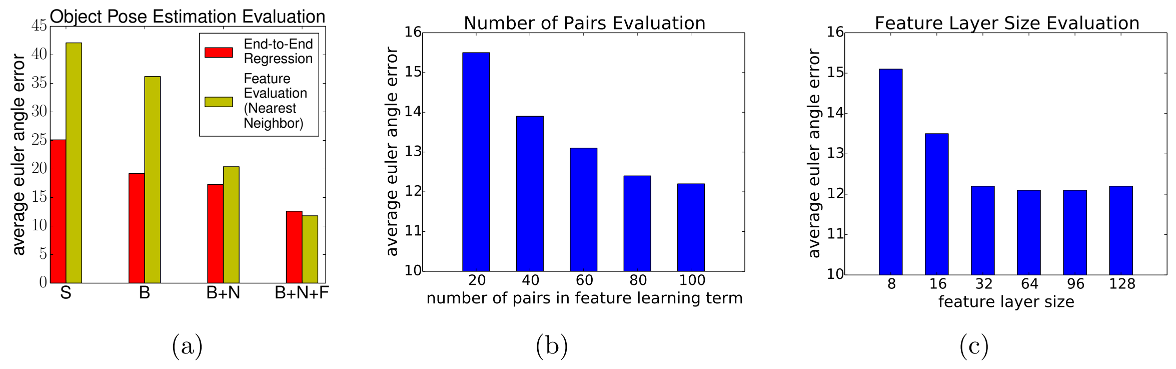


Figure 6.2: a) End-to-end regression evaluation as compared to our learned features using nearest neighbor on various network configurations: S is *simple* regression i.e. without any extra objective terms and using random batches, B is again simple regression with properly formed batch, N means that the network contains normalization layer after the feature and final layer, and F means that the network is trained using our new feature-guided pose regression objective. b) Evaluation of different number of pairs compared inside a mini-batch. c) Evaluation of the length of the feature layer.

represents a convolutional layer with N filters of size $K \times K$, and $FC(D)$ a fully connected layer with D outputs. Note that the feature layer $FC(D)$ is of variable length, something that allows a trade-off between feature extraction size and performance. We use *ReLU* as the non-linearity in all our convolutional and fully connected layers apart from the last layer that produces the pose estimation where we used *tanh*. For training the network we use the stochastic gradient method [110], with 0.9 momentum and initial learning rate of 0.01. We also decay the learning rate in each epoch, to avoid oscillations around local minima.

In order to evaluate our method we used two datasets. The first one, which is also used for our parameter analysis, is the one of LINEMOD [26]. More specifically, we worked with a variant of the RGBD images as used by [25] where the objects are centered in the image so that no localization is required. This dataset contains about 3000 training images and 1000 test images per object.

The second dataset is constructed by us and depicts a small object (a car belt) being manipulated by a human hand as seen in Fig. 6.3. The focus of such dataset is to introduce realistic occlusions that are severe and have stronger effect on the learning process than using different object instances or types. In order to construct such dataset, we recorded an RGBD video, using Asus Xtion, of a human manipulating the object, and used a particle swarm optimization tracker [111, 112] to track both the hand pose as well as the object pose. Having such information we can easily generate our needed pairs for our occlusion term (eq. 6.6). To create training and testing sets, we first subsampled the video to 3fps, and then divided the video in continuous segments, from which the 70% where used for training and the 30% for testing. Such scenario appears in autonomous learning of object manipulation by robots, where the task is being demonstrated by a human. This dataset is very challenging since human hand introduces high level of occlusion which can significantly degrade the accuracy of pose estimation. Moreover, our dataset is larger, with about 21000 training and 5000 testing images.

Regarding the evaluation metric, we use the average Euler angle error, which is the average of the absolute difference in angle (in degrees) between the estimated pose and the ground truth, measured regarding the three principal axes. Such metric is more appropriate for our regression task and matches the one used in [25] . Since we are using quaternions, we transform them in Euler angles after the estimation in order to perform the comparisons.

In the following subsections, we first evaluate different parameters of our network showing the relevant importance of each of our contributions. We also compare our siamese regression network with some base-line and state of the art methods showing the superiority of our method.

6.4.1 Parameter Evaluation

Fig. 6.2(a) shows the evaluation of the different parts of our network, starting from a simple regression network and gradually adding elements of our siamese regression network. We show both the performance of our end-to-end pose estimation and the performance of our produced features using nearest neighbor template matching. The regression network performs worse, and the performance gradually increases by just using a better formed batch, then the normalization layers and the best performance is achieved when adding our feature learning term in the objective function. Interestingly, we notice that the our features using nearest neighbor exhibit similar behavior, but the increase in performance is more significant. When using our feature learning term, it slightly outperforms the end-to-end regression. As we will see in the next subsection, the regression is more affected by overfitting regarding the small size of the linemod dataset we used for the analysis.

We also experiment with the amount of pairs required for our feature term. Fig. 6.2(b) shows the regression performance for different values of the amount of pairs used. Using a batch that contains 300 training images we see that the more pairs we use, the better the performance. However above 1000 pairs between the training images we did not get any further significant improvement.

Last, we evaluated our network on different feature sizes, shown in Fig. 6.2(c). Again we see that the more features used the better the performance achieved. Above the size of 32 however, there is not significant improvement, which is in par with what was reported in [25] .

6.4.2 State Of The Art Comparisons

Last, we performed a final evaluation of our siamese regression network compared with the method of [25] , which is the most relevant work to ours and directly comparable. This work uses triplets and pairs formed by the training samples and learn to minimize an objective function using a convolutional neural network. This objective only tries to increase the euclidean distance in feature space of dissimilar samples, while enforcing similar samples to be close. The idea behind this is to build a mapping appropriate for nearest neighbor matching with some templates.

Results are shown in Table 6.2. For comparisons we used two datasets, one as provided by [25] and on the dataset of [6] cropping only the regions with the objects centered. We see that both the end-to-end regression and our learned features outperform the previous work. One reason for this is that the objective used by [25] does not take into account the actual task objective which is the pose estimation. Our method has as ultimate goal to learn the object pose directly, and therefore constructing more appropriate features for this task. On the other hand, we see that on the small dataset of [26], nearest neighbor performs slightly better than the end-to-end regression, which is prone to overfitting regarding the dataset size. When experimenting on our larger occlusion dataset, we see that the end-to-end regression is able to converge to a better minimum. It is clear that both our features and regression significantly outperform [25]. Furthermore, our formulation gives us another opportunity to further improve the performance when we can generate synthetically the occluded and the clean image of an object. We see that by using equation 6.6 pose error decreases even further, reaching accuracy levels of the linemod dataset which does not contain occlusions. We note that method of [25] was also trained with both clean and occluded images.

Fig. 6.3 illustrates images and results of our novel hand-object occlusion dataset. From left to right columns represent: a real RGBD image; a synthetic one rendered using our tracker result; the rendered not occluded object that corresponds to the exact pose of the respective occluded real RGBD image; and our network final estimation.

Regarding our implementation, it was written in Theano. Training one epoch using Nvidia Titan X takes about 15mins for our dataset and about 20 seconds on linemod. One image of our dataset is 96×96 and takes 4ms for regression and 6ms for NN. Linemod dataset contains images of 64×64 and evaluation of an images takes about 2ms for regression and 4ms for NN.

As we have mentioned, this method is only about angle regression without localization, assuming the object is centered in the image. This assumption however makes is method unable to compare with our 6D solution presented in chapter 5. One possible extension of this method to perform localization would be a scanning

Object	**Nearest Neighbor [25]**	**Siamese Regression Network (End-to-End)**	**Siamese Regression Features + NN**	**Siamese Regression + Occlusion Term**
ape	15	12.3	**11.8**	-
benchviseblue	15.5	15.6	**13.2**	-
camera	12	10.9	**10.1**	-
can	15.5	14.5	**12.3**	-
cat	14	12.1	**10.4**	-
driller	17.8	16.7	**13.2**	-
duck	13.9	13.1	**10.9**	-
holepuncher	13.2	12.9	**11.4**	-
iron	11.4	11.6	**10.2**	-
lamp	13.3	12.6	**11.1**	-
phone	18.2	12.9	**11.7**	-
average	14.5	13.2	**11.4**	-
Camera	**4.7**	7.1	5.7	-
Coffee cup	11.2	10.9	**9.5**	-
Joystick	7	11.9	**6.7**	-
Juice Carton	8.2	9.5	**7**	-
Milk	**10.4**	13.9	11.6	-
Shampoo	7.7	8.4	**6.9**	-
average	8.2	10.3	**7.9**	-
belt (occlusion dataset)	25.2	13.2	14.3	**11.8**

Table 6.2: State of the art and self comparisons of our method against the one of Wohlhart et al. [25] in the dataset of LINEMOD [26] (first 11 objects), the dataset of [6] (next 6 objects) and our novel hand-object dataset (last object - belt).

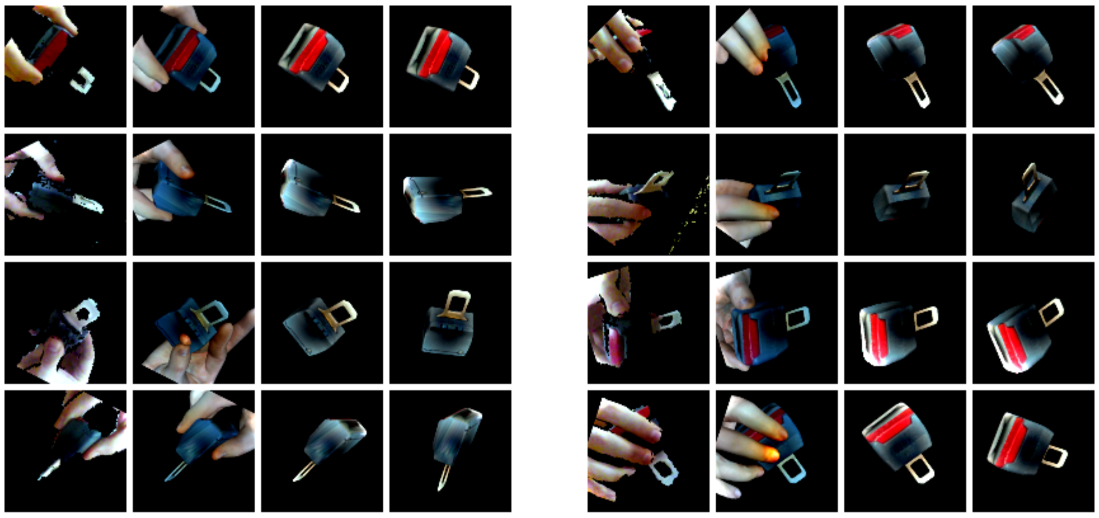


Figure 6.3: Our occlusion dataset. First column shows a real RGBD image, second column shows the synthetic image rendered using the tracking ground truth annotation, the third column shows the rendered not occluded object corresponding to the occluded image, and the forth column shows our network final estimation.

window throughout the image keeping the top hypotheses as the ones that their feature representation is closest to the template representations. However, this idea proved intractable in practice since there were many false positives, and in order to obtain meaningful results one had to keep about 100 top hypotheses (that are closest to template representations) and then run a verification step similar to our previous method. This would be intractable in terms of execution time, compared to our patch-based method which produced very accurate results when keeping only 5-10 top hypotheses. For these reasons, the combination of localization and pose estimation using a single network is left as future work.

Chapter 7

Thesis Conclusion and Future Work

In the first part of the thesis we have proposed a complete solution to the problem of autonomously unfolding an article of clothing. We used random forests for clothes classification and Hough forests for grasp point estimation in order to completely unfold four categories of clothes. Both were implemented into a POMDP framework for planning a dual manipulator optimally, enhancing the recognition and the unfolding procedure. We achieved very high recognition and unfolding success rate while our methods operate faster compared to the state of the art. The majority of our errors were caused by unsuccessful grasping of an estimated point. One reason is the noise of the Xtion depth sensor which causes inaccurate motion planning of the manipulators. The other reason is the lack of dexterity of the gripper, making the grasping of very thin or flat surfaces very difficult. The solution to the first problem would be a stereo camera with high resolution, which is planned to be used in the near future. On the other hand, a more humanoid gripper seems a more appropriate solution for clothes manipulation.

Moreover, we presented Active Random Forests, a framework for addressing active vision problems, and applied it to the task of autonomously unfolding clothes. We have focused on best viewpoint selection in classification, key point detection and pose estimation of 4 types of garments. The idea of incorporating the decision process of executing disambiguating actions inside Random Forests and combining features from multiple views outperformed classical active vision techniques, especially in the challenging problem of pose estimation of clothes. Furthermore, the required number actions is significantly reduced. This framework is also open to other actions which can be integrated like zooming to a particular region or any kind of interaction with the object. This direction is left as future work.

Last, with the contributions of our collaborators we built a complete robotic setup that was able to fold clothes that are randomly placed on a table (see Appendix A). This pipeline consisted of grasping a garment from a pile of clothes, unfolding it, spreading it on a table and then folding it. Our work is the first attempt to combine all sub-problems together in a unified framework that was integrated in a robot that was able to fold a variety of different types of clothes. This framework advanced the state of the art in both accuracy and speed and opens the road towards domestic robotic solutions that one day will help people with their housework activities. However, our solution is just the beginning and not a complete applicable solution. Our future work should focus mainly in three aspects: a) accuracy, b) speed and c) robot size. Even if we achieved state of the art performance in terms of both accuracy and speed, both of them need to be substantially improved for an industrial solution. Current speed is about 8 mins for folding a piece of clothing, which is not comparable with human performance of a few seconds. Furthermore, our current robot used was specifically designed for our particular project. Even if its size is very large exceeding humans, it is still not flexible enough to manipulate garments, especially on the table. This suggests that the robotic hardware needs to evolve further as well, in order to make the solution applicable to our everyday lives.

We have also presented solutions of rigid object detection and pose estimation, which can successfully tackle the most difficult scenarios of realistic scenes, such as heavy occlusions and background clutter. Our first approach is based on 2.5D patches and unsupervised feature learning, making use of Hough Forests for the final pose estimation. We further proposed an active vision method that utilizes the already trained forest in order to infer the next best view in case of hypotheses ambiguity. Such technique is extremely important for robots navigating a place and gives them the ability to further improve their object detection capabilities. We conducted extensive evaluation on challenging public datasets, including a new one depicting realistic scenarios, using various state of the art methods. Our framework showed superior results, being able to generalize well to a variety of objects and scenes. As a future work, we want to investigate how different patch sizes can be combined, and explore how convolutional networks can help in this direction.

Furthermore, we presented Siamese Regression Networks that are able to directly perform regression using holistic images avoid the hough voting and mode estimation steps. Particulartly, it is a convolutional network that is able to perform object pose regression in angle space directly, by enforcing distance similarity in feature and pose space among the training samples. Such network is able to learn more discriminative features that are optimal for the pose regression task, which outperform state of the art. Last, our feature-guided pose estimation can be easily modified to learn features that are robust to occlusions. In order to demonstrate the complete capacity of our algorithms and the problems that are able to tackle, with introduced 2 new datasets. One depicts everyday objects with severe occlusions and a bin-picking scenario with many objects stacked together in a bin. The other presents a hand-object manipulation scenario where is human hand is the main occluder. Those datasets are inspired by realistic problems of robotic applications (see Appendix B). As a future work, we would like to investigate how this network can be extended in order to simultaneously tackle object localization, as well as object classification.

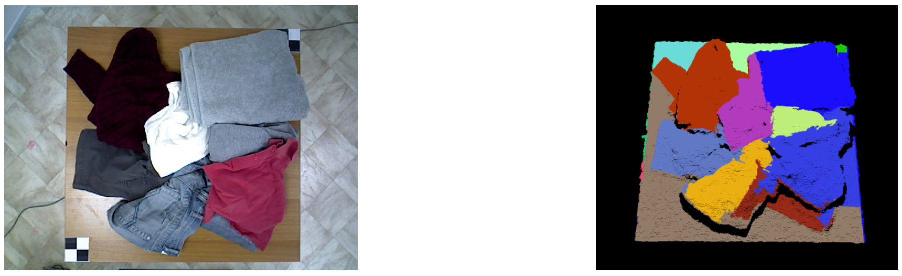
Regarding performance, we believe that our patch based approach has achieved great accuracy that can be used to real robotic solutions. However, each of the part consisting the complete framework requires time making the end-to-end detection slow. Although detection time per object is about 1-3sec, it makes the robot not fluent enough when searching for multiple objects. Our Siamese Regression Network was developed to overcome this issue, by directly regressing in pose space. However, neural networks require a lot of data in order to be trained efficiently, while it is crucial to include real images in the training set of images. This makes training less flexible and time consuming, since our patch-based approach could be trained by using only a CAD model of an object. For this reason, another direction for future work would be to combine the positives of both frameworks in a unified solution.

Appendix A

End-to-end Garment Folding

A.1 Folding Pipeline Overview

In this chapter we will present a complete pipeline for folding a pile of clothes using a dual-arm robot. The presented pipeline comprises of the following parts: isolating and picking up a single garment from a pile of crumpled garments, recognizing its category, unfolding the garment using a series of manipulations performed in the air as described in the previous chapters, placing the garment roughly flat on a work table, spreading it, and finally folding it in several steps. The pile is segmented into separate garments using color and texture information and the ideal grasping point is selected based on the features computed from a depth map. The recognition and unfolding of the hanging garment is performed in an active manner, utilizing our framework of Active Random Forests to detect grasp points, while optimizing the robot actions. Spreading action estimation is based on the detection of deformations of the garment contour. The perception for folding is based on fitting polygonal models to the contour of the observed garment, both spread and already partially folded. We have conducted several experiments on the complete pipeline producing very promising results. To our knowledge, this is the first work addressing the complete unfolding and folding pipeline on a variety of garments, including T-shirts,



(a) Input RGB image (b) Segmented point cloud

Figure A.1: a) The original RGB image and b) the point cloud with the individual segmented regions shown in various colors.

towels, and shorts.

The work of grasping, spreading and folding garments has been made by other researchers we have collaborated with and is not a direct output of the work conducted in this thesis. However our work solved a major part of the pipeline, the garment unfolding. Thus, we won't describe in detail those techniques, but rather give an overview of this novel pipeline along with the experimental results on a variety of clothes. More information can be found on our journal publication [113].

A.2 Grasping from Table

The first step of the pipeline is picking up a garment from the pile of crumpled garments. We propose a generic, robust and fast technique for grasping and picking up a single item of clothing from a pile containing multiple clothes. Fig. A.1(a) shows an example of such pile. We use depth image to detect 3D folds on the surface of the garment, which are the most suitable grasping points even for humans. Ramisa *et al.* [11] also use 3D local features, but their *wrinkling filter* may also detect false points such as concave regions. We also propose measures for assessing the graspability of the features, which consider the gripper geometry. On the contrary, the existing approaches usually apply only blind grasping.

Our aim is to detect candidate grasp points located along the folds of the garment. The method employs a rectified depth image **I**. The rectification is based on

RANSAC detection of the dominant plane corresponding to the table surface. The depth image containing the distance of 3D points to the estimated plane is computed and used as our input. Starting from the point with the strongest response, we delete points in its vicinity which have similar scale and orientation. Since folds may be considered as ridges on the 3D surface of the garment, differential geometry techniques based on surface curvature could be used to detect them. However, we have found in practice that the input images are too noisy for robust estimation of surface normals and / or second order derivatives needed by this approach. Filtering and approximation techniques may also be computationally expensive. The proposed technique is based on the *detection of curvilinear structures* in grayscale images, originally proposed in [114]. Indeed, folds may be seen as 2D lines on the image plane with a bell-shaped profile. We use the multiscale filtering technique proposed in [114] for the detection of such *ridge points*. Briefly, this consists in filtering the depth image $\mathbf{I}$ with the following non-linear filter:

$$\mathbf{I}_\sigma(\mathbf{u}) = \min\left\{\mathrm{Pos}\left((\mathbf{E}_\mathrm{l} * \mathbf{I})(\mathbf{u})\right), \mathrm{Pos}\left((\mathbf{E}_\mathrm{r} * \mathbf{I})(\mathbf{u})\right)\right\} \tag{A.1}$$

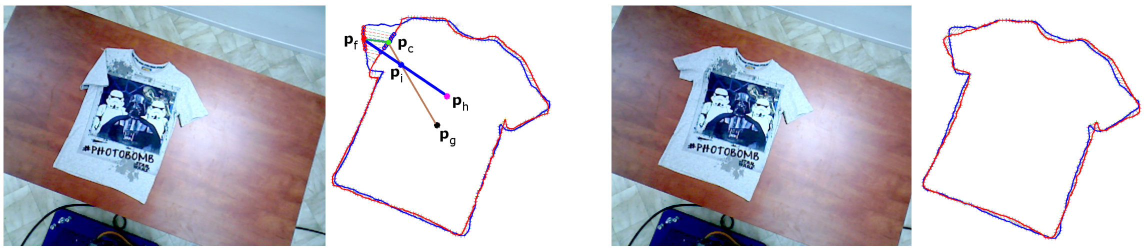
We define $\mathrm{Pos}(x) = x$ for $x > 0$ and $\mathrm{Pos}(x) = 0$ for $x \leq 0$. The operator $*$ denotes *convolution*. The filters $\mathbf{E}_\mathrm{l}$, $\mathbf{E}_\mathrm{r}$ are separable and steerable 2D filters consisting of a *derivative of Gaussian filter* (DoG) applied perpendicularly to the line direction and shifted by σ, followed by a *Gaussian filter* applied along the line direction.

In practice, for efficiency reasons, instead of DoG filtering, we first filter the images with Gaussian kernels and then compute *Sobel* responses for the horizontal and vertical direction. For a given scale σ, the line orientation is computed locally as the eigenvector of the *Harris operator*. In order to determine the scale (and thus a measure of the width of the fold), we compute $\mathbf{I}_\sigma$ over a sequence of scales, selecting the scale with the highest response for each pixel. We refer the reader to [114] for a justification and rational behind this approach. *Non-maxima suppression* of responses across the estimated line directions is applied to obtain thin skeletal lines of detect ridges. A further pruning procedure is applied to the resulting set of points to obtain a sparse set of candidate ridge points.

In order to grasp only one garment from a pile, a segmentation algorithm that takes account of color and texture information is necessary. To perform segmentation, we first extract *Gabor features* by convolving the RGB image with Gabor filter banks, created using multiple frequencies and orientations. The magnitude of the features is then used as a dissimilarity measure in a *graph-based segmentation* algorithm [115]. A sample of the segmentation output and the corresponding point cloud is shown in Fig. A.1(b). The segmentation output is subsequently combined with the results of the previous step to determine the best grasping point. In particular, we reject any regions that do not contain any candidate point. We also reject the candidate points that are too close to region boundaries. For the remaining regions and points, we sort them according to the highest (the closest one to the camera) grasp candidate point contained within their boundary. The final list of grasp candidates will contain points from the top (highest) region sorted by graspability, points from the second highest region etc. Obviously, using texture information to segment different items has the limitations of oversegmenation (i.e. garments mixing several textures) or erroneously merging items with similar texture. Nevertheless, the proposed approach reduces significantly the probability of grasping two items at the same item.

A.3 Garment Spreading on Table

Once the unfolded garment is placed on the work table (Fig. A.2(a)), it is examined in order to decide whether it is adequately spread-out for folding. This is extremely unlikely for most garments, since when unfolded they are grasped by only two points, resulting to deformations due to gravity. While experimenting with various garment types, only in case of simple geometries such as towels or shorts grasped by their waist, the robot was able to place them flat on the table.



(a) Deformed T-shirt (b) Deformed contour (c) Fully-spread (d) Contour spread

Figure A.2: Spreading algorithm applied on a T-shirt: a) the unfolded T-shirt having its right sleeve slightly deformed, b) resulting configuration after spreading. c) The deformed contour (red) is matched to the garment template (blue). The spreading actions are planned based on the detected deformations. d) No deformation is detected after spreading and therefore the T-shirt can be folded.

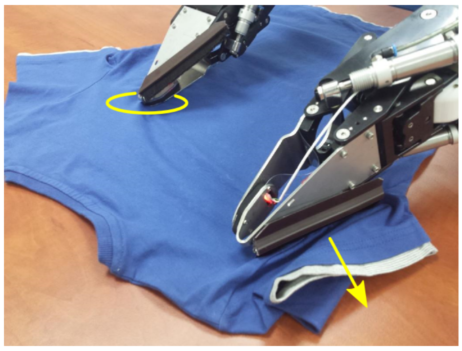


Figure A.3: Brush tool is attached to the gripper and moved in the direction shown by the arrow. The other arm is holding the T-shirt to prevent it from sliding.

Thus, a novel method is proposed for bringing the unfolded garment into a spread-out configuration, in case it is still deformed when it is placed on the work table. This method is used to bridge the gap between unfolding and folding, which is of significant importance when the complete pipeline is executed. Our approach is based on the measurement of the deformation between the outer contour of the examined garment and the template garment corresponding to the type recognized by the unfolding module (e.g. T-shirt, towel, shorts). Then, in case a deformation is detected, a spreading action is executed by the robot (Fig. A.2(b)). The spreading action consists of one arm pressing the garment towards the table in order to prevent sliding, while the other hand is swiping with a small brush in a suitable direction (Fig. A.3). After spreading, the resulting configuration is checked again for deformations and the procedure is repeated if necessary.

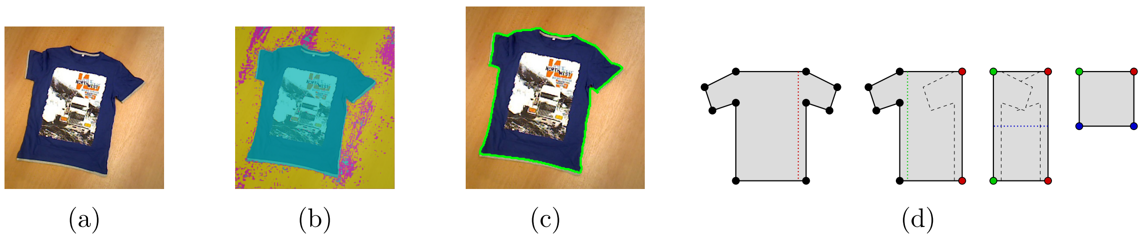
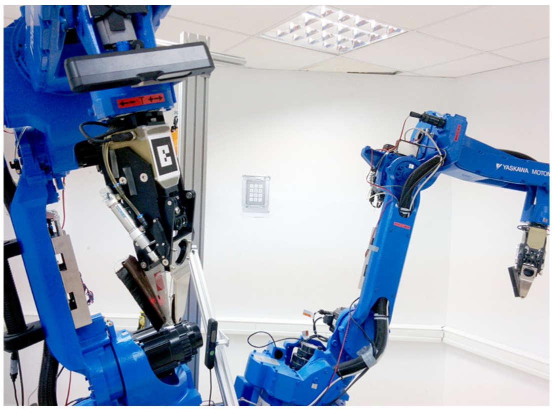


Figure A.4: Pixels of the a) input image are used to initialize b) trimap for GrabCut algorithm. The trimap consists of foreground (plotted in cyan), background (yellow) and unknown (magenta) pixels. c) The garment contour (green) is extracted from the binary segmentation mask. d) Incremental creation of folded models for a short-sleeved T-shirt. The original vertices are being replaced by new vertices denoting endpoints of the individual folds (plotted in various colors)

A.4 Garment Folding

The final step of the pipeline is folding of the garment that has been unfolded and spread on the table. Since the garment category is already known, only its pose needs to be estimated. We propose a robust method for visual detection of the garment pose from a single image. The method is not using any prior information from the previous unfolding stage except the known garment category. It can thus be used separately and independently upon the pipeline, as described in [116, 117]. Once the garment pose is recognized, a single folding move is planned and executed. The vision procedure is then repeated to check the garment pose before performing the next fold.

The perception procedure can be split into several steps. It starts with a single image of the spread garment. The garment location in the image is determined by color segmentation. The garment contour is extracted from the segmentation mask. The contour is simplified by approximating it with a polygon. The simplified polygonal contour is matched to a polygonal model for the particular category of clothes. Vertices of the matched model determine locations of the important landmark points found on the garment contour, e.g. corners, shoulders, armpits or crotch. The identified landmarks are then utilized for planning of the folding moves.


(a)

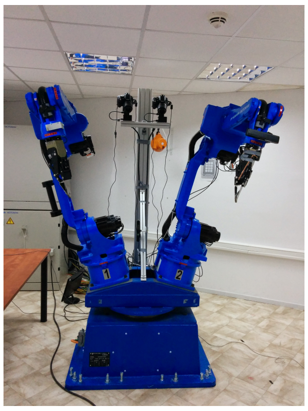

(b)

Figure A.5: a) Detail of the robot arm equipped with custom jaw gripper and combined color and range sensor used for active vision. b) complete view of the robot

A.5 Experiments

A.5.1 Testbed description

The methods described in this chapter were implemented and tested on two identical dual-arm robot testbeds located in CTU Prague and CERTH Thessaloniki. The robot is composed mainly from standard industrial components. Its body consists of two Motoman MA1400 robotic arms mounted to R750 turn-table. The components are controlled by two DX100 controllers working as master and slave. The arms are attached jaw-like grippers developed by the CloPeMa consortium [53]. Fig. A.5 shows a detailed view of the robot.

The robot is equipped with several sensors. There are three combined RGB and depth cameras ASUS Xtion PRO attached on the robot, two on the wrists and one on the base. They are the only sensors used for our current task. Furthermore, a head comprising of two Nikon D5100 cameras for stereo vision [118] and corresponding pan/tilt units is mounted on the robot base. The grippers are equipped with tactile sensors and photometric stereo intended mainly for material sensing. The wrists comprehend force and torque sensors.

The robot control system is built on ROS (Robot Operating System) [119] in the Hy-

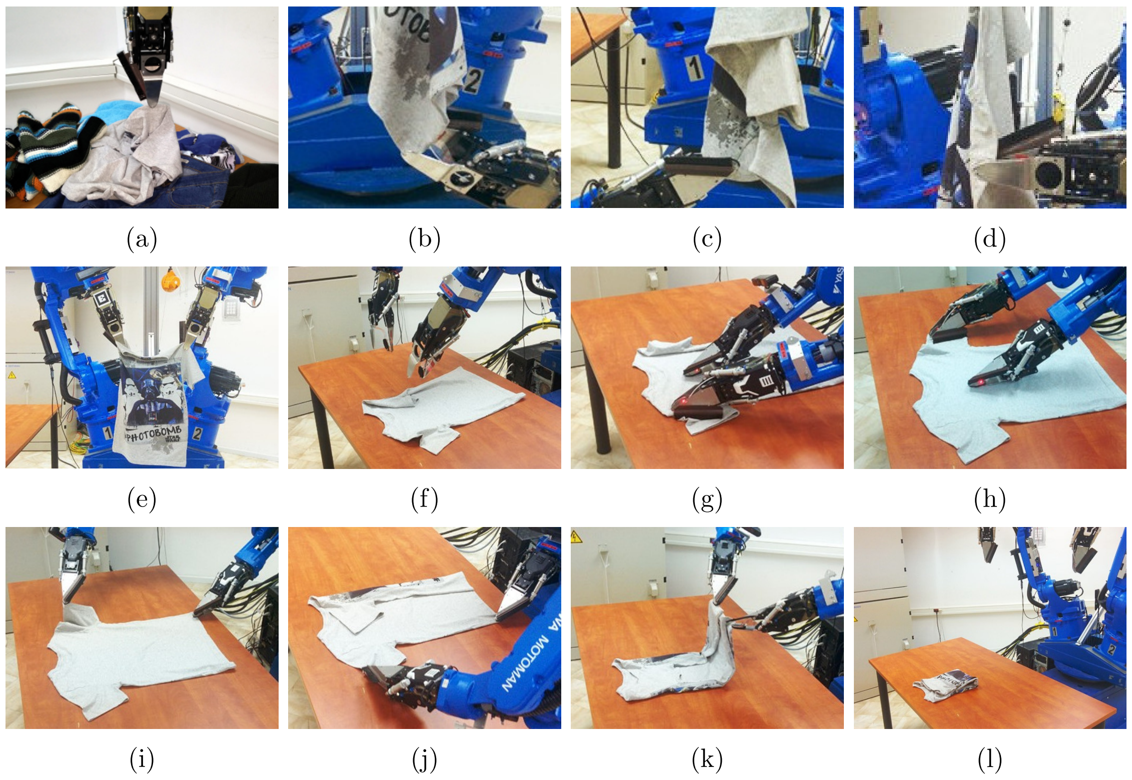


Figure A.6: The complete folding pipeline of a T-shirt. a) The optimal grasping point is selected and the T-shirt is lifted from the table. b) Its lowest point is regrasped to reduce number of possible configurations. c) The first desired point is grasped with the free arm, the lowest point is released and d) the second desired point is grasped. e) The T-shirt is unfolded by stretching the grasped points. f) The unfolded T-shirt is layed on the empty table. g) Both left and h) right sleeve are spread using the brush tool if necessary. i) The folding procedure comprehends folding right and j) left side of the T-shirt before k) performing the final fold. l) The result is the fully folded T-shirt.

dro version. The basic functionality of moving the arms and reading positions from their joints is provided by MotoROS package, which is delivered by the manufacturer. We also utilize MoveIt package and OMPL (Open Motion Planning Library) [120] for motion planning.

A.5.2 Performance

The performance of the complete pipeline was evaluated on various garments, including those used for testing of the spreading module. One example is shown in Fig. A.6. However, we have not used long-sleeved shirts and trousers, as in spreading, because of the limited workspace of the robot that does not allow the folding of such long garments. We conducted 8 trials for each of 4 garments of each category

(T-shirts, shorts, towels), yielding a total of 96 trials. The complete pipeline was successful in 72 trials, yielding a success rate of 79 %. This overall rate is lower than performance of each individual stage, because failures can occur in different stages of the pipeline and affect the complete process. More specifically, 7 times the towel was classified as T-shirt, 2 wrong grasp points were detected and grasped for shorts and 11 unsuccessful foldings occurred for T-shirts, from which 2 occurred because the garment was not well spread on table after unfolding. The results are summarized in Table A.1. The most challenging garment type is a T-shirt, which presents the poorest success rate of 66 %, despite the corrections applied by the spreading module. The stages of the complete unfolding process are depicted in Fig. A.6. Our video[1] shows the complete folding of one garment per category.

The execution of the whole pipeline takes approximately 8 minutes on average, with the robot operating in a moderate speed for safety reasons. This can be compared to [13] where a complete pipeline for folding towels took approximately 20 minutes per towel on average. Most time is spent by the actual movement, not by perception or reasoning. Picking up a garment from a pile takes about 1 second to calculate the correct grasp point, whereas the robot completes the grasp in about 20 seconds. For garment unfolding, every image captured from Xtion takes about 30-40ms to be analyzed. On average, 5 images from different viewpoints are required to classify a garment and estimate the grasp point and pose. The whole process of finding and picking two grasp points per garment takes about 130secs for the robot to execute. Regarding spreading of the garment, point estimations requires about 10 seconds, whereas each spreading step is executed in about 50 seconds by the robot. The spreading process is executed at most 3 times. Pose estimation of the garment being folded, which is performed prior to folding and then repeated after each fold, takes 2–5 seconds, depending on the garment type. The segmentation takes 0.8 seconds on average, contour extraction and simplification 0.5–3.5 seconds, and model fitting 0.1 seconds on average. One fold is performed in approximately 30 seconds.

[1] https://www.youtube.com/watch?v=8TsLkpPsdKo

	Shorts	T-shirts	Towels	Total
Successful / all trials count	30 / 32	21 / 32	25 / 32	**76 / 96**
Success ratio [%]	94	66	78	**79**

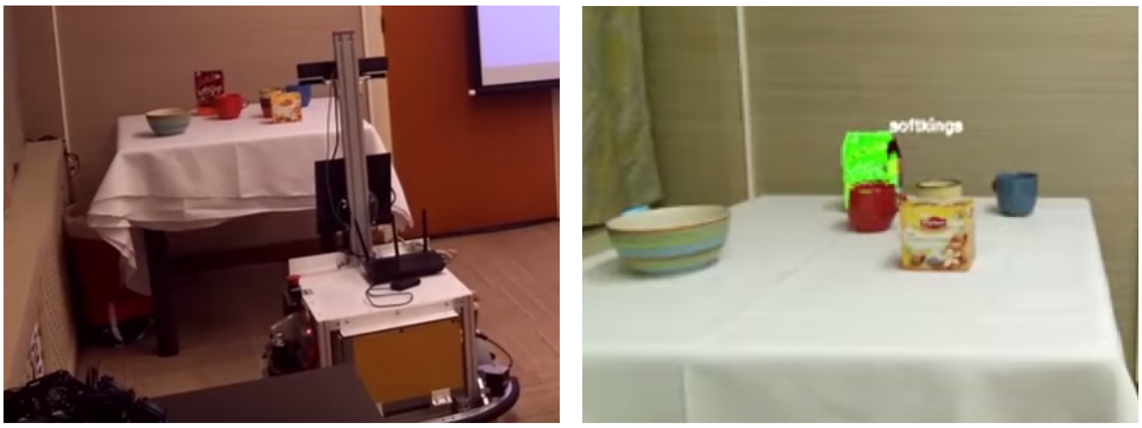
Table A.1: Overall results of experiments testing the complete pipeline including grasping, category recognition, unfolding, spreading and folding.

As discussed in section 3.2, we should also mention here that our robot has a limited working space when folding on table, therefore it is currently impossible to work with larger garments. Also, generalizing to many different types of garments needs collecting a large dataset of clothes and capturing many training images from each garment in order to train our models. This is however a tedious work and one cannot always guarantee that every possible garment shape appears in the dataset. Therefore as a future work it would be ideal if we can learn our models from simulation data using a graphics library, which can also make the process of adding new garments much easier.

Appendix B

Practical Applications of our Object Detector

Our solution for object detection and pose estimation has been implemented and used by 2 Horizon 2020 European projects, *RAMCIP* and *SARAFun*. RAMCIP stands for *Robotic Assistant for MCI Patients at home* and the project is developing a robotic solution for the elderly and those suffering from Mild Cognitive Impairments and dementia at home. Our object detector is used for their object detection needs at home, mainly for food boxes or medicine. Fig. B.1(a) shows a prototype of the robot that will be used looking at a table with various objects, and Fig. B.1(b) shows our detection result (we can notice the performance under severe occlusion). SARAFun means *Smart Assembly Robots with Advanced Functionality* and the project is related to robotic assembly of the parts of various objects such as mobile phones, while the robot will be able to learn from a human, by teaching by demonstration. This project inspired our hand-object dataset that can be seen in Fig. 6.3.



(a) RAMCIP robot

(b) object detection result

Figure B.1: RAMCIP robot and object detection demonstration

Bibliography

[1] “Euler angles.” https://en.wikipedia.org/wiki/Euler_angles.

[2] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 987–993, May 2014.

[3] A. Doumanoglou, T.-K. Kim, X. Zhao, and S. Malassiotis, “Active random forests: An application to autonomous unfolding of clothes,” in *ECCV*, 2014.

[4] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim, “Recovering 6d object pose and predicting next-best-view in the crowd,” in *CVPR*, 2016.

[5] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, “Learning 6d object pose estimation using 3d object coordinates,” in *ECCV*, 2014.

[6] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, “Latent-class hough forests for 3d object detection and pose estimation,” in *ECCV*, 2014.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *CVPR*, 2015.

[8] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE Multimedia*, vol. 19, no. 2, pp. 4–10, 2012.

[9] F. Osawa, H. Seki, and Y. Kamiya, "Unfolding of Massive Laundry and Classification Types," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, pp. 457–463, 2007.

[10] S. Miller, M. Fritz, T. Darrell, and P. Abbeel, "Parametrized shape models for clothing," in *Proc. ICRA*, 2011.

[11] A. Ramisa, G. Alenya, F. Moreno-Noguer, and C. Torras, "Using depth and appearance features for informed robot grasping of highly wrinkled clothes," in *Proc. ICRA*, 2012.

[12] M. Cusumano-Towner, A. Singh, S. Miller, J. O'Brien, and P. Abbeel, "Bringing clothing into desired configurations with limited perception," in *Proc. ICRA*, 2011.

[13] 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 *Proc. ICRA*, 2010.

[14] J. Denzler and C. M. Brown, "Information theoretic sensor data selection for active object recognition and state estimation," *PAMI*, 2002.

[15] C. Laporte and T. Arbel, "Efficient discriminant viewpoint selection for active bayesian recognition," *IJCV*, 2006.

[16] Z. Jia, Y.-J. Chang, and T. Chen, "A general boosting-based framework for active object recognition," *BMVC*, 2010.

[17] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[18] M. Martinez, A. Collet, and S. S. Srinivasa, "Moped: A scalable and low latency object recognition and pose estimation system," in *ICRA*, 2010.

[19] J. Tang, S. Miller, A. Singh, and P. Abbeel, "A textured object recognition pipeline for color and depth image data," in *ICRA*, 2012.

[20] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *ICCV*, 2011.

[21] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky, "Hough forests for object detection, tracking, and action recognition," *PAMI*, 2011.

[22] A. Aldoma, F. Tombari, L. Di Stefano, and M. Vincze, "A global hypotheses verification method for 3d object recognition," in *ECCV*, 2012.

[23] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *NIPS*, 2014.

[24] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *Similarity-Based Pattern Recognition*, 2015.

[25] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," in *CVPR*, 2015.

[26] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *ACCV*, 2012.

[27] J. J. Lim, A. Khosla, and A. Torralba, "Fpm: Fine pose parts-based model with 3d cad models," in *ECCV*, 2014.

[28] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *CVPR*, 2015.

[29] R. Mottaghi, Y. Xiang, and S. Savarese, "A coarse-to-fine model for 3d pose estimation and sub-category recognition," in *CVPR*, 2015.

[30] Z. Wu, S. Song, A. Khosla, X. Tang, and J. Xiao, “3d shapenets: A deep representation for volumetric shapes,” in *CVPR*, 2015.

[31] L. Breiman, “Random forests,” *Machine learning*, 2001.

[32] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, “Real-time human pose recognition in parts from single depth images,” in *Proc. CVPR*, 2011.

[33] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky, “Hough forests for object detection, tracking, and action recognition,” vol. 33, no. 11, pp. 2188–2202, 2011.

[34] K. Mikolajczyk, B. Leibe, and B. Schiele, “Multiple object class detection with a generative model,” in *Computer Vision and Pattern Recognition (CVPR)*, 2006.

[35] Autodesk, *123D Catch*, 2015.

[36] S. Izadi, D. Kim, O. Hilliges, Molyneaux, *et al.*, “Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera,” in *ACM*, 2011.

[37] K. Hamajima and M. Kakikura, “Planning strategy for task of unfolding clothes,” in *Robotics and Autonomous Systems*, vol. 32, pp. 145–152, 2000.

[38] M. Kaneko and M. Kakikura, “Planning strategy for putting away laundry-isolating and unfolding task,” in *Proc. of the Int. Symposium on Assembly and Task Planning*, pp. 429–434, 2001.

[39] D. Triantafyllou and N. Aspragathos, “A vision system for the unfolding of highly non-rigid objects on a table by one manipulator,” in *Intelligent Robotics and Applications*, vol. 7101, pp. 509–519, 2011.

[40] B. Willimon, S. Birchfield, and I. Walker, “Model for unfolding laundry using interactive perception,” in *Proc. IROS*, 2011.

[41] B. Willimon, S. Birchfield, and Walker, "Classification of clothing using interactive perception," in *Proc. ICRA*, 2011.

[42] B. S. Willimon Bryan, Walker Ian, "Classification of Clothing Using Midlevel Layers," *ISRN Robotics*, 2013.

[43] Y. Kita, T. Ueshiba, E. Neo, and N. Kita, "Clothes state recognition using 3d observed data," in *Proc. ICRA*, 2009.

[44] Y. Kita, F. Kanehiro, T. Ueshiba, and N. Kita, "Clothes handling based on recognition by strategic observation," in *Proc. IRAS*, pp. 53–58, 2011.

[45] C. Bersch, B. Pitzer, and S. Kammel, "Bimanual robotic cloth manipulation for laundry folding," in *Proc. IROS*, 2011.

[46] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[47] T. Yu, T.-K. Kim, and R. Cipolla, "Unconstrained monocular 3d human pose estimation by action detection and cross-modality regression forest," in *Proc. CVPR*, 2013.

[48] B. Rosenfeld, "The curvature of space," in *A History of Non-Euclidean Geometry*, vol. 12 of *Studies in the History of Mathematics and Physical Sciences*, pp. 280–326, Springer New York, 1988.

[49] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in linear pomdps by factoring the covariance," in *Robotics Research*, vol. 66, pp. 293–305, 2011.

[50] N. Dantam, P. Koine, and M. Stilman, "The motion grammar for physical human-robot games," in *Proc. ICRA*, 2011.

[51] D. Hsu, W. S. Lee, and N. Rong, "A point-based pomdp planner for target tracking," in *Proc. ICRA*, 2008.

[52] P. Monso, G. Alenya, and C. Torras, “Pomdp approach to robotized clothes separation,” in *Proc. IROS*, 2012.

[53] T.-H.-L. Le, M. Jilich, A. Landini, M. Zoppi, D. Zlatanov, and R. Molfino, “On the development of a specialized flexible gripper for garment handling,” *Journal of Automation and Control Engineering*, vol. 1, no. 3, 2013.

[54] I. Mariolis and S. Malassiotis, “Matching folded garments to unfolded templates using robust shape analysis techniques,” in *Computer Analysis of Images and Patterns*, vol. 8048, pp. 193–200, 2013.

[55] R. Bajcsy and M. Campos, “Active and exploratory perception,” in *CVGIP: Image Understanding, 56(l):31-40*, 1992.

[56] J. Tsotsos and K. Shubina, “Attention and visual search : Active robotic vision systems that search,” in *The 5th International Conference on Computer Vision Systems, Bielefeld*, 2007.

[57] J.-Y. Herve and Y. Aloimonos, “Exploratory active vision: theory,” in *Computer Vision and Pattern Recognition (CVPR)*, 1992.

[58] H. Borotschnig, L. Paletta, M. Prantl, and A. Pinz, “Appearance-based active object recognition,” in *Image and Vision Computing*, 2000.

[59] B. Schiele and J. L. Crowley, “Transinformation for active object recognition,” in *ICCV*, pp. 249–254, 1998.

[60] T. Arble and F. P. Ferrie, “Viewpoint selection by navigation through entropy maps,” *ICCV*, 1999.

[61] T. Arble and F. P. Ferrie, “On the sequential accumulation of evidence,” *IJCV*, 2001.

[62] F. G. Callari and F. P. Ferrie, “Recognizing large 3-d objects through next view planning using an uncalibrated camera,” *ICCV*, 2001.

[63] M. A. Sipe and D. Casasent, "Feature space trajectory methods for active computer vision," *PAMI*, 2002.

[64] E. Sommerlade and I. Reid, "Information-theoretic active scene exploration," *CVPR*, 2008.

[65] J. Vogel and N. de Freitas, "Target-directed attention: Sequential decision-making for gaze planning," *ICRA*, 2008.

[66] D. Meger, A. Gupta, and J. J. Little, "Viewpoint detection models for sequential embodied object category recognition," *ICRA*, 2010.

[67] K. Welke, J. Issac, D. Schiebener, T. Asfour, and R. Dillmann, "Autonomous acquisition of visual multi-view object representations for object recognition on a humanoid robot," *ICRA*, 2010.

[68] B. Rasolzadeh, M. Bjorkman, K. Huebner, and D. Kragic, "An active vision system for detecting, fixating and manipulating objects in the real world," *IJRR*, 2010.

[69] D. Tang, T. Yu, and T.-K. Kim, "Real-time articulated hand pose estimation using semi-supervised transductive regression forests," *ICCV*, 2013.

[70] A. Criminisi, J. Shotton, and E. Konukoglu, "Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning.," Tech. Rep. MSR-TR-2011-114, Microsoft Research, Oct 2011.

[71] L. Pardo, *Statistical inference based on divergence measures.* CRC Press, 2005.

[72] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, "Efficient regression of general-activity human poses from depth images," *ICCV*, 2011.

[73] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani, *The elements of statistical learning*, vol. 2. Springer, 2009.

[74] G. Guo, Y. Fu, C. R. Dyer, and T. S. Huang, "Head pose estimation: Classification or regression?," in *ICPR*, 2008.

[75] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, vol. 1, pp. 886–893, 2005.

[76] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," in *Neural Computation*, 2006.

[77] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?," in *ICCV*, 2009.

[78] Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng, "Building high-level features using large scale unsupervised learning," in *ICML*, 2012.

[79] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng, "Convolutional-recursive deep learning for 3d object classification," in *NIPS*, 2012.

[80] L. Bo, X. Ren, and D. Fox, "Learning hierarchical sparse features for rgb-(d) object recognition," *IJRR*, 2014.

[81] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *NIPS*, 2010.

[82] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, "Matchnet: Unifying feature and metric learning for patch-based matching," in *CVPR*, 2015.

[83] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," 2015.

[84] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, *et al.*, "Deepid-net: Deformable deep convolutional neural networks for object detection," in *CVPR*, 2015.

[85] G. Riegler, D. Ferstl, M. Rüther, and H. Bischof, “Hough networks for head pose estimation and facial feature localization,” in *BMVA*, 2014.

[86] B. Leibe, A. Leonardis, and B. Schiele, “Combined object categorization and segmentation with an implicit shape model,” in *European Conference on Computer Vision (ECCV)*, 2004.

[87] R. Rios-Cabrera and T. Tuytelaars, “Discriminatively trained templates for 3d object detection: A real time scalable approach,” in *ICCV*, 2013.

[88] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, Y. Taguchi, T. K. Marks, and R. Chellappa, “Fast object localization and pose estimation in heavy clutter for robotic bin picking,” *IJRR*, 2012.

[89] B. Drost, M. Ulrich, N. Navab, and S. Ilic, “Model globally, match locally: Efficient and robust 3d object recognition,” in *CVPR*, 2010.

[90] R. B. Rusu, N. Blodow, and M. Beetz, “Fast point feature histograms (fpfh) for 3d registration,” in *ICRA*, 2009.

[91] S. Song and J. Xiao, “Sliding shapes for 3d object detection in depth images,” in *ECCV*, 2014.

[92] U. Bonde, V. Badrinarayanan, and R. Cipolla, “Robust instance recognition in presence of occlusion and clutter,” in *ECCV*, 2014.

[93] N. Fioraio and L. Di Stefano, “Joint detection, tracking and mapping by semantic bundle adjustment,” in *CVPR*, 2013.

[94] A. G. Buch, Y. Yang, N. Kruger, and H. G. Petersen, “In search of inliers: 3d correspondence by local and global voting,” in *CVPR*, 2014.

[95] N. Atanasov, B. Sankaran, J. Le Ny, G. Pappas, and K. Daniilidis, “Nonmyopic view planning for active object classification and pose estimation,” *IEEE Transactions on Robotics*, 2014.

[96] Z. Wu, S. Song, A. Khosla, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *CVPR*, 2015.

[97] M. Sun, G. Bradski, B.-X. Xu, and S. Savarese, "Depth-encoded hough voting for joint object detection and shape recovery," in *ECCV*, 2010.

[98] R. Rios-Cabrera and T. Tuytelaars, "Discriminatively trained templates for 3d object detection: A real time scalable approach," in *ICCV*, 2013.

[99] E. Brachmann, F. Michel, A. Krull, M. Ying Yang, S. Gumhold, and C. Rother, "Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image," in *CVPR*, 2016.

[100] T. Hodan, X. Zabulis, M. Lourakis, S. Obdrzalek, and J. Matas, "Detection and fine 3d pose estimation of texture-less objects in rgb-d images," in *IROS*, 2015.

[101] E. Johns, S. Leutenegger, and A. J. Davison, "Pairwise decomposition of image sequences for active multi-view recognition," in *CVPR*, 2016.

[102] A. Crivellaro, M. Rad, Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit, "A novel representation of parts for accurate 3d object detection and tracking in monocular images," in *ICCV*, 2015.

[103] A. Krull, E. Brachmann, F. Michel, M. Ying Yang, S. Gumhold, and C. Rother, "Learning analysis-by-synthesis for 6d pose estimation in rgb-d images," in *ICCV*, 2015.

[104] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *ICCV*, 2015.

[105] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazrbas, and V. Golkov, "Flownet: Learning optical flow with convolutional networks," in *International Conference on Computer Vision (ICCV)*, 2015.

[106] K. M. Yi, Y. Verdie, P. Fua, and V. Lepetit, "Learning to assign orientations to feature points," 2016.

[107] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *CVPR*, 2006.

[108] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *CVPR*, 2014.

[109] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *ICCV*, 2015.

[110] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*, 2012.

[111] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-time continuous pose recovery of human hands using convolutional networks," *ACM Transactions on Graphics (TOG)*, 2014.

[112] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect.," in *BMVC*, 2011.

[113] A. Doumanoglou, J. Stria, G. Peleka, I. Mariolis, V. Petrik, A. Kargakos, L. Wagner, V. Hlavac, T.-K. Kim, and S. Malassiotis, "Folding Clothes Autonomously: A Complete Pipeline," *IEEE Transactions on Robotics, accepted to appear in*, 2016.

[114] T. Koller, G. Gerig, G. Szekely, and D. Dettwiler, "Multiscale detection of curvilinear structures in 2-D and 3-D image data," pp. 864–869, 1995.

[115] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," vol. 59, no. 2, pp. 167–181, 2004.

[116] J. Stria, D. Průša, and V. Hlaváč, "Polygonal models for clothing," pp. 173–184, 2014.

[117] J. Stria, D. Průša, V. Hlaváč, L. Wagner, V. Petrík, P. Krsek, and V. Smutný, "Garment perception and its folding using a dual-arm robot," pp. 64–67, 2014.

[118] A. Schmidt, L. Sun, G. Aragon-Camarasa, and J. P. Siebert, "The calibration of the pan-tilt units for the active stereo head," in *Image Process. and Commun. Challenges 7*, vol. 389 of *Advances in Intelligent Systems and Computing*, pp. 213–221, Springer, 2016.

[119] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop on Open Source Software*, 2009.

[120] I. A. Şucan, M. Moll, and L. E. Kavraki, "The open motion planning library," vol. 19, no. 4, pp. 72–82, 2012.