Università degli Studi di Verona

Department of
Computer Science

PhD School of
Sciences and Engineering

PhD in
Computer Science
S.S.D. INF/01

With funding by
Cariverona

Cycle / year
XXXII / 2016

Title of the doctoral thesis

Saliency-based approaches for multidimensional explainability of deep networks

Doctoral Student
Dott. Marco Carletti

Coordinator Prof. Massimo Merro
Tutor Prof. Marco Cristani

Saliency-based approaches for multidimensional explainability of deep networks
Marco Carletti - Tesi di Dottorato - Verona, 22 Aprile 2020

Contents

List of Figures

List of Tables

Abstract

In deep learning, visualization techniques extract the salient patterns exploited by deep networks to perform a task (e.g. image classification) focusing on single images. These methods allow a better understanding of these complex models, empowering the identification of the most informative parts of the input data. Beyond the deep network understanding, visual saliency is useful for many quantitative reasons and applications, both in the 2D and 3D domains, such as the analysis of the generalization capabilities of a classifier and autonomous navigation.

In this thesis, we describe an approach to cope with the interpretability problem of a convolutional neural network and propose our ideas on how to exploit the visualization for applications like image classification and active object recognition.

After a brief overview on common visualization methods producing attention/saliency maps, we will address two separate points: firstly, we will describe how visual saliency can be effectively used in the 2D domain (e.g. RGB images) to boost image classification performances: as a matter of fact, visual summaries, i.e. a compact representation of an ensemble of saliency maps, can be used to improve the classification accuracy of a network through summary-

driven specializations. Then, we will present a 3D active recognition system that allows to consider different views of a target object, overcoming the single-view hypothesis of classical object recognition, making the classification problem much easier in principle. Here we adopt such attention maps in a quantitative fashion, by building a 3D dense saliency volume which fuses together saliency maps obtained from different viewpoints, obtaining a continuous proxy on which parts of an object are more discriminative for a given classifier. Finally, we will show how to inject this representations in a real world application, so that an agent (e.g. robot) can move knowing the capabilities of its classifier.

CHAPTER 1

Introduction

Machine-learning technology has become part of many aspects of modern society: it is increasingly present in almost any consumer products and services, from social and media filtering to retail web-services, from recommendation systems to autonomous driving machines, from image classification to text-to-speech recognition. Such applications are often related to a particular class of techniques called *deep learning*. In the last decades, building an intelligent pattern recognition or machine learning system required considerable expertise in the application domain in order to design the best feature extractor that is capable to transform the raw data to a suitable data representation from which the learning system, *ie.* an image classifier, could detect or classify significant patterns in the input. Representation learning is part of the AI panorama, allowing a machine to be fed with raw data and to automatically discover the sub-optimal representations needed for a specific task, such as classification or detection. Deep-learning is a representation learning method with multiple levels of representation, obtained by stacking non-linear modules capable to transform the representation at one level into a representation at a higher and somehow more abstract level. With the composition of enough non-linear transformations, in theory any complex functions can be learned and described as a neural network. Considering the classification tasks, the topmost layer of a neural architecture aims to suppress irrelevant variations in the raw input in order to amplify those aspects that are important for the final discrimination process. Each layer is then responsible of focusing on the significant information only, extracting and generating a new representation of the data coming from the previous step. For example, an image comes in the form of an array of colored pixels; the first layers of a neural classifier typically extract the low-level structural patterns of this image, such as edges and colors. These structures are then combined into motifs by the central part of the network, regardless of minor changes in the edge orientation and position. The final part may assemble motifs into larger structures corresponding to familiar objects that are eventually associated to a specific category by the very last layer of the network. The key aspect of deep learning is that these layers of features are learned from data using a general-purpose learning procedure [8, 63]: features are not designed by human engineers but the machine *learns* an optimal representation of the data to solve a particular task. This allows deep learning solving problems the community struggled in for many years: it has turned out to be very good at discovering intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government. Deep learning success is increasing day by day because it requires very little engineering by hand, so it can easily take advantage of increases in the amount of available computation and data.

Deep learning has become popular in the last decade thanks to its astonishing results in a wide range of applications, from natural language processing to healthcare, from robotics to mobile. However, this technique suffers from three main drawbacks. First, training a neural network requires a significant amount of memory and computational power, especially when the network is deep (*ie.* it consists of many stacked layers). For example, in 2012 the first deep network for image classification, known as AlexNet [56], was trained in two weeks using a 2xGPU machine while in 2019 a Generative Adversarial Network [52] has reached astonishing results in a week on an NVIDIA DGX-1 with 8 Tesla V100 GPUs. It comes immediately clear that building an AI-system is a costly affair.

The second problem is the huge amount of data a network usually requires to converge to a global optimum, as it learns progressively. Although several big datasets like ImageNet and ShapeNet [56, 116, 15] have become popular, only big firms are able to collect and label such big data collections. Furthermore, data availability for some sectors is limited or sparse, with an inevitable damping in the application of deep learning in those fields. The third main drawback related to deep learning is the lack of transparency of the latent space, that is it is tough to understand *why* a model behaves in a specific manner: although the formulation behind the training of a neural network is relatively simple to understand, the highly dimensional latent space (*ie.* the number of weights of a model) makes impossible to remove the mystery behind the output of the backpropagation optimization [8, 63].

In this work, we face the important problem of the weights visualization of a deep neural network: we make use of state of the art approaches, with particular attention to [32, 119, 91], to get robust 2D attention maps that we call *saliency maps* (Figure 1.1). The main contribution of this research is the exploitation of such images in order to improve the effectiveness of pre-trained classification models both in the 2D [37] and 3D domains [85, 110]. In particular, this work proposes a novel approach to visualize the informative parts of the input data in a compact way, as explained in the next sections.

1.1 Visualizing deep architectures

Individuating the visual regions exploited by a deep network for making decisions is important: this allows to foresee potential failures [46] and highlight differences among diverse network architectures [122, 119, 125, 120]. Historically, neural networks have been thought of as black boxes, meaning that their inner workings were mysterious and inscrutable. Recently, the scientific community has started proposing different classes of methods to open those boxes and better understand what each element (*ie.* neuron) of a neural network has learned and thus what it does represent. This is the goal of the *visualization* strategies that could be divided in three main categories: gradient based, generative methods and perturbation based techniques.

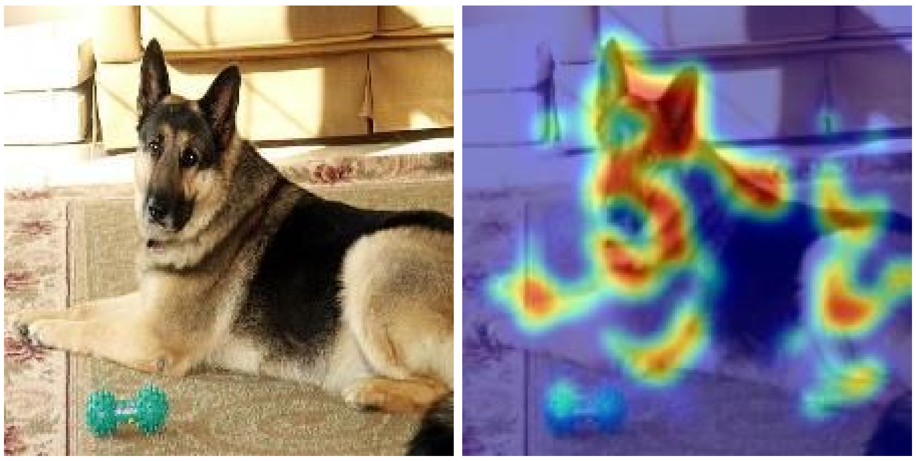


Figure 1.1: Saliency map example. Left: RGB reference image taken from "German Shepherd" ImageNet class. It is passed as input to the AlexNet image classifier [56] trained on the ImageNet dataset. Right: smooth saliency map extracted with the algorithm proposed in [32]. Hot regions highlight the areas of the input image that the classifier uses to predict the class. Best view in colors.

Early work [23, 122, 119, 120], belonging to the first and second class of approaches, individuate those images which activate a certain neuron the most. Other approaches consider the network as a whole, generating dreamlike images bringing the classifier to high classification scores [94, 120, 75]. The most studied type of visualization techniques, however, highlights those salient patterns which drive a classifier toward a class [28, 32, 91, 123, 67, 121] or against it [125] through smooth saliency maps. These are examples of perturbation-based approaches. Despite those works, no prior study investigated whether these salient patterns are systematically related to precise semantic entities to describe an object class. In fact, the previous visualization systems analyze single images independently, and no reasoning on multiple images from the same class is carried out. In other words, these approaches are not able to reveal if a network has captured an object class in all of its local aspects. It would be of great importance for interpretation of deep-architectures to be able to understand, for example, that AlexNet when classifying the class "german shepherd" is systematically very sensible to the visual patterns representing the nose, the eye and the mouth, so that the absence of one or all of these patterns in an image will most probably bring to a failure. At the same time, knowing that GoogleNet has understood also the tail (in addition to the previous parts) can add a semantic explanation of its superiority w.r.t. AlexNet. Detailed description of such categories is in the Chapter 2.

1.1.1 Regression

Despite the rising of deep neural networks is strongly associated to the image classification task, it has quickly become clear that deep learning applicability spreads in many different domains and could be applied to face other tasks, such as regression, segmentation and data generation.

In this section, we introduce our preliminary work on regression: we present a novel framework to directly estimate body fat percentage from depth images of human subjects and to visually evaluate salient points of the body shape related to the fat distribution. Measurements of body composition (*eg.* fat or lean tissue percentage) are very important in many applicative contexts, like medical diagnosis or sports science. Approximate but reasonable fat estimations are based on regression over anthropometric measurements [96, 42], which however are noisy with large inter-observer errors [54]. This brings to the common practice of using simplified formulas like the US army body fat calculator [1], whose estimates are far from being accurate. 3D body scanning can be used to make fat estimation automatic [34, 36], requiring, however, expensive equipment and time-consuming procedures. Whole body scanner solutions can be now provided with software tools for estimating body fat from measured girths (see, for example, `www.styku.com`), but they are close, expensive systems. Estimating anthropometric parameters from depth maps is at its infancy, with few related works. Nguyen et.al. [73] estimate body weight from RGB-D images using regression on hand crafted features; Pfitzner et.al. [78] add thermal camera data. These methods, however, are tested on a small number of subjects, and no publicly available data.

Our goal is to estimate fat percentage from depth images, that can be acquired with fast and cheap sensors (Microsoft Kinect, Asus Xtion, Intel Realsense, etc.), avoiding manual body measurements or full-body scans.

Following the main path of the thesis, our framework is based on regression over deep networks, currently trained and tested on depth images of real subjects with body composition assessed with a DXA scan. For this purpose, we created a novel, publicly available dataset including front and back depth images of a set of subjects with specific features (active young men or professional sportsmen) with associated ground truth fat values estimated with dual-energy x-ray absorptiometry (DXA) scanning. These depth images were obtained with depth rendering of an available dataset of whole body scans, simulating low-cost depth sensor acquisitions. As described in Chapter 3, we customized the ResNet architecture [41] to estimate fat percentage values directly from the front/back scans, achieving promising accuracy [12, 13]. In the experiments, presented in Chapter 4, we also demonstrate that, using a custom perturbation-based procedure for analyzing deep networks, it is possible to highlight, on subjects depth images, the specific body areas related to fat accumulation (typically neck, shoulders, hip, and abdomen) and those characterizing skinny subjects (chest and abdomen), as shown in Figure 1.2.

The data allows to use deep learning, performed on a ResNet-50 [41] architecture, here customized for regression. The choice is motivated by two main reasons: first, the approach avoids the necessity to find accurate body landmarks that are typically required to obtain handcrafted features for regression. Furthermore, we exploit the CNN architecture to derive an additional visualization tool, showing the salient body parts influencing the regression results. Our visualization method is inspired by two perturbation-based approaches applied to classification networks [120, 125], here customized for regression. On our depth images, it is able to highlight the areas crucial to characterize high and low-fat body shapes.

Beside the creation of a novel dataset, the main contributions of the work on regression is twofold: first, we demonstrated the feasibility of a depth-based

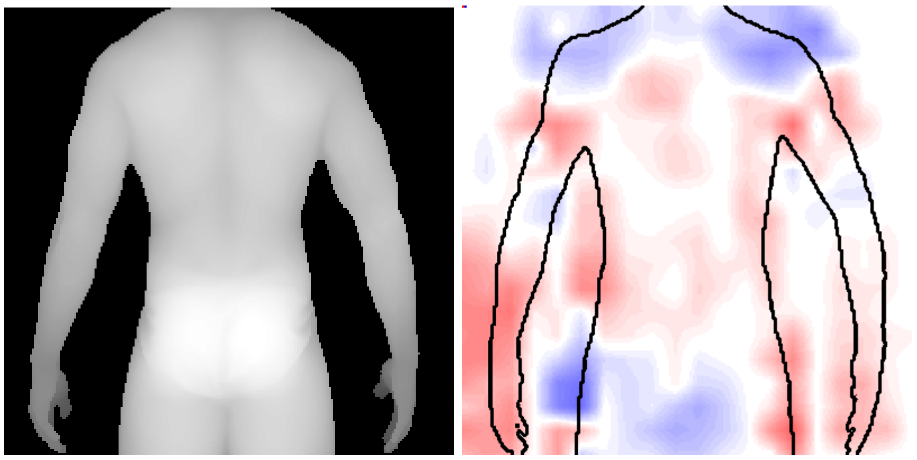


Figure 1.2: Depth rendering example of a single subject and the salient regions extracted on the torso depth image of the same subject. Blue regions are mostly related to those body parts responsible for higher fat values, while red areas refer to skinny subjects. Fat percentage value of this subject: 13%.

regression model for fat percentage estimation. Second, through the use of a visualization tool we made possible the analysis of fat related body features (*ie.* which body parts are responsible of the final fat value).

1.1.2 Classification

Deep learning has spread primarily for its impressive results on image content recognition, proposing many different neural architectures year after year. Furthermore, the availability of enormous dataset such as ImageNet [56] allowed the scientific community going deep in the classification problem. For the above reasons and our preliminary results on regression (see Section 1.1.1 and Chapter 3), we decided to investigate this field, also inspired by the fact that all the visualization approaches for deep networks proposed so far have been applied on common classification architectures [125, 119, 32].

2D domain

In the first part of this work we aim to discover a way to exploit the output of such saliency maps. In particular, we firstly face this problem in the 2D domain: we present the first visualization approach which employs analysis of multiple images within an object class to provide an explanation on what has been understood by a network in terms of *visual parts* to form an object class [37]. In practice, our approach takes as input a trained deep network and a set of images, and provides as output a set of image clusters, or *summaries*, where each cluster is representative of an object visual part. Our visualization approach is composed by two phases. In the first phase, a crisp image saliency map is extracted from each test image, indicating the most important visual patterns for a given class. Important visual patterns are those that if perturbed in an image lead to a high classification loss. The perturbation masks are found by an optimization process borrowed from [32] and made sparse to provide binary values which results to a so called crisp mask. In facts, most literature on visualization provide smooth masks where higher values mean higher importance in the region [122, 125, 23, 32, 91, 123, 67, 121]. In this work, however, we empirically demonstrate that our proposed crisp mask brings to higher classification loss w.r.t. smooth mask by incorporating a model to remove noisy patterns. Crisp mask on the other hand, facilitates further computations in the formation of the summaries. In the second phase, the connected components, i.e. *regions*, of the crisp masks are grouped across the image employing the affinity propagation algorithm [33], where the similarity measure is borrowed from the proposal flow algorithm [38]. This allows for example to cluster together the wheel regions of different images from the car class, which together with other region clusters, facilitate interpretation of the class. In the experiments, presented in Chapter 6, we show that our summaries capture clear visual semantics of an object class, by means of an automatic tagger and a user study. In addition, we show that the number of summaries produced by our approach is correlated with the classification accuracy of a deep network: the more the summaries, the higher the classification accuracy as demonstrated for AlexNet, VGG, GoogleNet, and ResNet in our experiments. Finally, we demonstrate that the summaries may improve the classification ability of a network, by adopting multiple, specific specialization procedures with the images of each summary.

3D domain

The encouraging results on the 2D domain suggested us to proceed with our research. In particular, we followed the intuition that our preliminary analysis could be extended to the 3D domain, with a focus on depth information: nowadays, it is plenty of cheap depth sensors with reasonable resolution [12, 13]. The usage of such sensors has moved our attention to the robotics field, in which we have seen the active recognition problem as the perfect opportunity to verify the potential of our intuition about saliency. One of the most challenging task is the active recognition of unknown entities in a scene: for example, a robot should be able to navigate the environment in order to see the objects of interest from different perspectives, so that the recognition task becomes easier. The active classification of objects in a scene is a fundamental skill that has to be embodied into autonomous systems to safely operate in real world environments. More frequently than ever, robotics platforms need to manipulate and interact with objects in order to perform specific tasks. In this context, active vision plays a fundamental role in analyzing the objects in the scene given the current observation [2, 68, 77, 79, 86]. This task was normally accomplished by running an object classifier (either in 2D or 3D) to localize and classify objects in the image and then to actuate the robotic platform to grasp and manipulate them [7, 27, 60, 77, 100]. However, when scenes are cluttered (a likely occurrence), objects are often incorrectly classified since they are possibly occluded by other elements in the environment. To overcome this problem a successful strategy is to move the active sensor to the Next Best View (NBV) which eases the classification task [4, 79, 51, 24].

Assistive robots have attracted increasing attention from both academics and industries [16] with the objective of supporting daily life, in particular of people with disabilities. Another important task could be related to warehouse management, in which robots are desired to substitute human to interact or manipulate with entities in the physical world to perform specific tasks such as grasping and placing [30, 59, 88, 93]. One essential capability to achieve such tasks is to be able to perceive the environment with both geometric ("where are the objects?") and semantic understanding ("what are the objects?"). However, object recognition in unstructured and cluttered 3D environment can be very challenging with only single-shot-based method [81]. Moreover, perceiving objects positions and their pose is not a trivial task even with a single object in an uncluttered environment. Multiple objects in arbitrary positions exacerbate the problem, possibly leading to catastrophic failures if the object detector/pose estimator makes wrong decisions about the scene structure. Occlusions can contribute greatly to the loss of accuracy of vision systems which are fundamental to correctly recognise and estimate the 6D pose of objects. As a consolidated experimental fact, even the best single object classifiers decrease their performance when consistent occlusions appear [47]. In this context, active vision plays a fundamental role in best analysing the objects in the scene given the current observation and deciding which next view might have the chance to achieve a better recognition score [2, 68, 77, 79, 86]. In particular, when scenes are cluttered, objects are often incorrectly classified since they are possibly occluded by itself or other elements in the environment. Active Object Recognition (AOR) becomes a promising strategy for actively covering more viewpoints which eases the classification task [4, 24, 51, 79].

Following this idea, we proposes an active strategy for 3D object classification which can iteratively selects viewpoints where salient object parts, useful for obtaining a successful classification, are clearly visible.

Our approach is applied to scenarios where multiple objects are present and occlusions are likely to occur: the proposed pipeline is sketched in Figure 1.3 and is briefly detailed in the following. A depth frame is acquired from a sensor from a given viewpoint with known camera pose: this information can be given by an encoder of a robotic arm having pre-calibrated the camera or directly estimated by a SLAM approach. Given the depth frame, the point cloud is first pre-segmented to remove the background and then object 3D instances are preselected by clustering the point cloud. Each cluster is potentially an object that is sent to an object classifier (CNN-based) which provides a probability score over the objects classes and assigns the most confident one to the object. Given the object classification output, a 6D pose detector related to the class with the maximum score provides the object 3D position and mesh in a reference space for all the instances. We then fit to each detected 3D object a pre-learned *saliency volume* [85], *ie.* we map at each voxel of the detected object a value indicating how much that voxel is crucial for increasing the classification score.

In such context, knowing what is important for the classifier is crucial to intelligently plan the movement of the acquisition device: understand what a classifier, for example a neural network [56, 58], considers important in the raw input data could be exploited to lead the selection of the machine navigation. Saliency analysis, as described in this chapter, is probably the most exhaustive and understandable way to enlighten one of the insidious drawbacks of deep learning.

Saliency Volumes The study of saliency volumes is the starting point of one of the major contributions of this thesis. Here, we propose the first AOR approach which does open the box of the classifier, understanding its capabilities, injecting thus a *recognition self-awareness* in the planning process. Technically, we model the camera planning as a Partially Observable Markov Decision Process (POMDP), in line with the most successful non-myopic techniques in the literature [2, 39, 77]. In particular we design a novel observation model (that is, how the planning process manipulates the information gathered from the scene) that exploits deep network visualization techniques [32, 119]. Deep visualization approaches provide *qualitative* explanations on why a particular classifier succeeds or fails in classifying 2D images: we are interested in those approaches which provide saliency maps over the input images [32, 125]. Here we adopt such saliency maps in a *quantitative* fashion, by building a 3D dense saliency volume which fuses together saliency maps obtained from different viewpoints, obtaining a continuous proxy on which parts of an object are more discriminative for a given classifier. This volume is injected in the POMDP observation model; in this way, the robot can move knowing the capabilities of its classifier. We call our model *Recognition Aware*-POMDP (RA-POMDP): detailed description of such model is presented in Chapter 5. As an example, if the robot knows that its classifier is effective in distinguishing between a motorbike and a bike by looking for the presence of an engine, it will move the camera in a side view, where the engine can be easily spotted, instead of focusing on frontal views, which intuitively may also work, but not for that specific classifier. Experiments have been carried out on ObjectNet3D [116], a dataset of 3D models of objects belonging to 85 semantic classes, exploiting as sensor a depth camera, with simulated and real robots, showing dramatic improvement against state-of-the-art approaches. Ablation studies show the effectiveness of our RA-POMDP against alternative models.

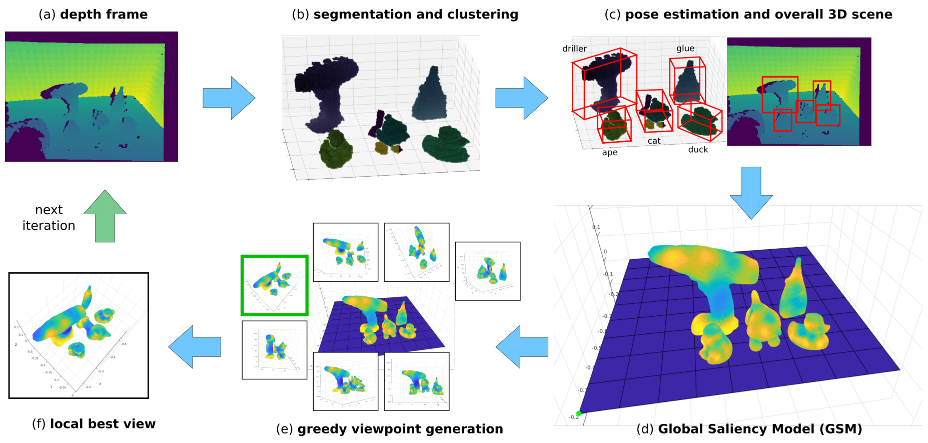

(a) depth frame
(b) segmentation and clustering
(c) pose estimation and overall 3D scene
driller
glue
ape
cat
duck
next iteration
(f) local best view
(e) greedy viewpoint generation
(d) Global Saliency Model (GSM)

Figure 1.3: Active recognition method overview. (a, b) The input depth frame is acquired and segmented in order to build and cluster the 3D point cloud of the scene. Each cluster represents an object. (c) The 6D poses of the detected objects are estimated through a pose detector and (d) the saliency volumes of the classes are aligned to generate a Global Saliency Model (GSM). A saliency volume highlights the regions of an object the classifier uses to guess the class of an object. In the end, (e) a greedy strategy is used to select the next best view, that is the neighbor view which maximizes the projection score of the volumes.

If most salient 3D parts are visible, our claim is that there will be higher chances that the object would be correctly classified. To prove this, we consider all the saliency volumes for all the objects to create a global 3D representation of the scene, that is a novel representation of the scene from the current position of the sensor. From the current viewpoint, we evaluate the contribution of the detected objects by projecting their visible parts and the associated saliency scores on the depth image plane. We evaluate the relevancy of these scores using different approaches (*ie.* weighted sum, median or other statistics) aiming to compute a unique representation of the scene, that is our Global Saliency Model (GSM) that will be used for computing the next best view for recognition. Occlusions are implicitly considered in the projection step, since given the putative 3D object representation and camera viewpoint, a z-buffer indicates if an occlusion is happening for a specific voxel. We then compute the GSM for local neighboring viewpoints by projecting the 3D scene into depth maps given by simulated camera positions and we move to the best one which maximises the GSM as seen in the selected view. The approach iterates until it finds for a given cluster a probability score higher than a threshold. The final output of our approach is a 3D volumetric representation of the scene, with the respective object 6D poses estimated and a score providing the recognition confidence.

1.2 Contributions

This thesis faces the problem related to the complexity of understanding a deep neural network. Deep learning has become popular in the last years thanks to its astonishing results in huge variety of fields, from medicine to robotics. However, one of the main criticisms against deep learning is that neural network are often considered as black boxes. In this work, we make use of visualization methods to understand the inner representation of a convolutional neural network, aiming to use these results to improve the performance of a regressor and classification system, both in the 2D and 3D domains. In particular, we aim to exploit the results of visualization to allow a recognition system reaching a new level of self-awareness.

Despite the interest of the scientific community, the problem of deep network visualization is still at it infancy with a valuable but limited literature. Visualizing deep architectures becomes crucial in those fields where knowing the reasons of the decision process is mandatory. A clinician should be aware about the prognosis of a disease if it is estimated by a machine, so it is compulsory to understand what the learning system has considered in its decision process.

An autonomous driving system decides whenever it is time to steer or not: debugging such a system becomes almost impossible if no insight about the inner structure of the system is available. Due to the above reasons, we realized that the exploitation of the saliency approaches might be a potential solution to the visualization problem and could be used to improve the performance of a neural network. The hypothesis was verified by solving three sub-problems, in order to simplify the analysis of these modern techniques. We firstly propose a preliminary evaluation of the applicability of deep learning on regression: a simple yet effective framework was proposed to estimate the fat percentage of a subject (*ie.* male athletes) and to visualize which body parts are responsible of the prediction. The succeeding analysis is mainly related to classification: we chose to face the visualization problem in the 2D domain, which provides much more mature deep architectures, in particular for image recognition [56, 95, 99, 41], a richer literature and more widely used datasets to compare with [56]. Finally, we moved our first steps to the more challenging domain of 3D, exploiting the availability of cheap depth sensors like the Microsoft Kinect and Asus Xtion.

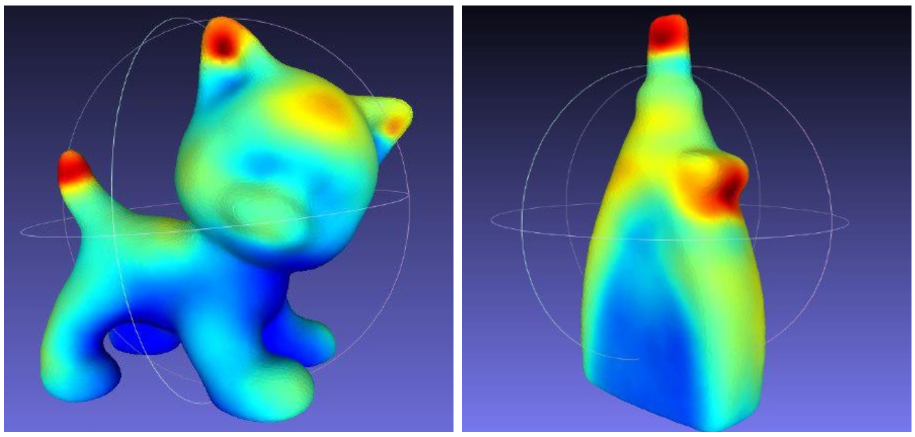


Figure 1.4: Samples of saliency volumes built from the PointNet architecture [81]. Red areas are informative for the object classifier: when these regions are visible, the classifier is most likely to estimate the correct class label with high confidence. Best view in colors.

In conclusion, in this thesis we aim to investigate and promote the applicability of saliency maps of deep architectures in order to increase interpretability. The main contributions of this thesis could be divided in two main blocks, one for the 2D domain, in particular for the classification of RGB images, and one for the 3D domain, where the classification problem is applied to point clouds and depth information. The first part of the thesis [37] offers the following contributions: we introduce the first deep network saliency visualization approach to offer an understanding of the visual parts of an object class which are used for classification. The framework consists of a model for crisp saliency masks extraction built upon the proposed model by [32]: it generates a set of visual summaries by grouping together crisp salient regions of commonly repetitive salient visual parts among multiple images within a same object class. In the second part, we propose to expand the concept of saliency map by generating a 3D proxy, *ie.* the saliency volumes (see Figure 1.4). Each volume is a spatial representation of the importance of different regions of an object. This representation could be injected into an active recognition system in order to facilitate the next best view selection problem.

1.3 Thesis outline

In deep learning, visualization techniques extract the salient patterns exploited by deep networks for image classification, focusing on single images; no effort has been spent in investigating whether these patterns are systematically related to precise semantic entities over multiple images belonging to a same class, thus failing to capture the very understanding of the image class the network has realized. This thesis goes in this direction: we face the visualization problem focusing on the 2D and 3D domains with particular attention to image classification and active object recognition.

Regression In Chapter 3, we propose our first step toward deep learning for regression and visualization: a simple yet effective approach for human body-fat estimation [12, 13] has been presented, in which we show how saliency analysis is fundamental to explain the reasons behind the decision process of the regressor (see Chapter 4).

Calssification (2D) We presents our visualization framework [37] which produces a group of clusters or *summaries* in Chapter 5.1: each group is formed by crisp salient image regions focusing on a particular part that the network has exploited with high regularity to decide for a given class. The approach is based on a sparse optimization step providing sharp 2D image saliency masks that are clustered together by means of a semantic flow similarity measure. The summaries communicate clearly what a network has exploited of a particular image class, and this is proved through automatic image tagging and with a user study. Beyond the deep network understanding, summaries are also useful for many quantitative reasons: their number is correlated with ability of a network to classify (more summaries, better performances), and they can be used to improve the classification accuracy of a network through summary-driven specializations.

Calssification (3D) In the second part of this work, we propose an active object recognition framework that introduces the *recognition self-awareness* [85, 110], which is an intermediate level of reasoning to decide which views to cover during the object exploration. As a matter of fact, autonomous agents that need to effectively move and interact in a realistic environment have to be endowed with robust perception skills. Among many, accurate object classification is an essential supporting element for assistive robotics. However, realistic scenarios often present scenes with severe clutter, that dramatically degrades the performance of current object classification methods. Due to the above reasons, it become clear that the application of saliency results in the 3D domain is not trivial, so we opted to face the active vision problem by dividing our scenario in two sub-tasks.

Firstly, we evaluate the effectiveness of the saliency volumes in a simple scene composed by one single object, which 6D pose is known in advance [85]. This is built by learning a multi-view deep 3D object classifier, as explained in Chapter 5.3. Subsequently, a 3D dense saliency volume is generated by fusing together single-view visualization maps, these latter obtained by computing the gradient map of the class label on different image planes. The saliency volume indicates which object parts the classifier considers more important for deciding a class.

Finally, the volume is injected in the observation model of a Partially Observable Markov Decision Process (POMDP) [2, 57]. In practice, the robot decides which views to cover, depending on the expected ability of the classifier to discriminate an object class by observing a specific part. For example, the robot will look for the engine to discriminate between a bicycle and a motorbike, since the classifier has found that part as highly discriminative. In order to validate the approach, experiments (Chapter 6.3) are carried out on depth images with both simulated and real data, showing that our framework predicts the object class with higher accuracy and lower energy consumption than a set of alternatives.

Secondly, in Chapter 5.4 we applied the same framework in a random cluttered scene, where multiple objects are presents. In particular, we present an active multiple 3D object classification framework estimating the label of multiple objects in a single cluttered scene. The framework uses depth information, and moves the sensor following a Next-Best-View paradigm. At each iteration, the system localises and assigns a probable class label to each object; therefore, a correspondent salient volume is aligned with each classified object. Salient volumes are models that indicate which part of a given class is discriminant (i.e. the handle of a cup) for a specific classifier. The presence of multiple salient volumes is exploited to create a global saliency model, indicating which camera pose will observe the most salient parts, thus providing the next best view. The framework, operating on a real robot, starts from a random position and is capable of deciding the class of 6 objects in the scene in 2.74 steps in average, where random approaches score 4% less detection accuracy in 20 steps. The system has been tested on real objects obtained by considering the LINEMOD benchmark [43] and on a real setup using a robotic arm.

A preliminary step before injecting the saliency volumes is presented in Chapter 5.2: this work presents an active vision approach that improves the accuracy of 3D object classification through a Next Best View (NBV) paradigm to perform this complex task with ease. The next camera motion is chosen with the criteria that aim to avoid object self-occlusions while exploring as much as possible the surrounding area. An online 3D reconstruction module is exploited in our system in order to obtain a better canonical 3D representation of the scene while moving the sensor. By reducing the impact of occlusions, we show with both synthetic and real-world data that in a few moves the approach can surpass a state-of-the-art method, PointNet with single view object classification from depth data. In addition, we demonstrate our system in a practical scenario where depth sensor moves to search and classify a set of objects in cluttered scenes.

1.4 Publications

Part of this thesis has been published in conference proceedings and the authors list order of the papers reflects the contribution each person carried to the results. The former analysis on deep learning for regression led to two papers presented at the International Conference on 3D Vision (3DV) [12] and the Conference and Exhibition on 3D Body Scanning and Processing Technologies (3DBODY.TECH) [13]. The main contributions of this work are related to the visualization of deep neural architectures and they have been presented at the British Machine Vision Conference (BMVC): the 2D approach presented in Chapter 5.1 has been published in [37]. The study of saliency volumes presented in Chapter 5.3 has been published in [86], whereas the preliminary evaluation of the active recognition framework has been accepted to the International Conference on Computer Vision workshop (ICCV) [110]. A minor contribution on 3D object pose estimation has been presented at the industrial session of the International Conference on Image Analysis and Processing [19].

CHAPTER 2

State of the art

The last decade is scattered of a rich deep learning literature. In this chapter we propose a structured overview of the works related to visualization techniques and a brief insight about active recognition methods. In particular, the first part of the chapter describes gradient based and perturbation based visualization approaches for image classifiers. That is, those methods that are able to highlight which area of an input image is used by a neural classifier to guess the image class. The second part of the chapter could be divided into two parts: firstly, an introduction of active recognition and object detection; the second part highlights some modern pose estimators and active methods for 3D navigation.

2.1 Visualizing deep neural networks

Visualization approaches can be categorized mainly into *local* and *global* techniques. *Local* techniques focus on the understanding of single neurons by showing the filters or their activation [119]. Under this umbrella, *input-dependent* approaches select the images which activate a neuron the most [122, 120, 23]: in other words, those techniques search for the image in the dataset which prediction confidence is higher. Despite the simplicity of the approach, the output of such approach requires the final user to individuate which patterns could be of interest for the task. *Global* approaches however, capture some general property of the network, as like the tendency in focusing on some parts of the images for the classification. These methods consider the network as a whole, generating dreamlike images that bring the classifier to high classification scores [125, 94, 32, 91, 123, 66]. Such approaches are given a single image as input, and output a smooth saliency map in which the areas important for classification into a certain class are highlighted (see Figure 2.2). Global approaches are mostly *gradient-based*, computing the gradient of the class score with respect to the input image [120, 23, 91, 66].

In this thesis, the proposed approaches described in Chapters 5.1 and 5.3 fall into the global category, that we can further subdivide in three subcategories that partially overlap each other: gradient-based, generative and perturbation-based.

Some other types of gradient-based approaches add activations to the analysis, obtaining edge-based images with edges highlighted in correspondence of salient parts [91]. Another type of deep visualization highlights those salient patterns which drive a classifier toward a class [28, 32, 91, 123, 67, 121]. Notably, the technique of [125] individuates also the pixels which are *against* a certain class: we make use of this approach in Chapter 3 to show which human body parts are responsible of fat estimation.

Generative approaches generate dreamlike images bringing the classifier to high classification scores [94, 72, 75]. In particular, the work of [75] and [119] are heavily built on generative-based *local* representations, which are somewhat difficult to interpret, making the forecasting of the performance of the network against new data particularly complicated. Figure 2.1 shows some example of the output of generative visualization techniques.

Perturbation-based approaches edit an input image and observe its effect on the output [125]. In this case, the general output of the model is a saliency map showing how crucial is the covering of a particular area, that can be a pixel [122, 32] or superpixel-level map [84]. In all of the previous cases, the outputs are single masked images.

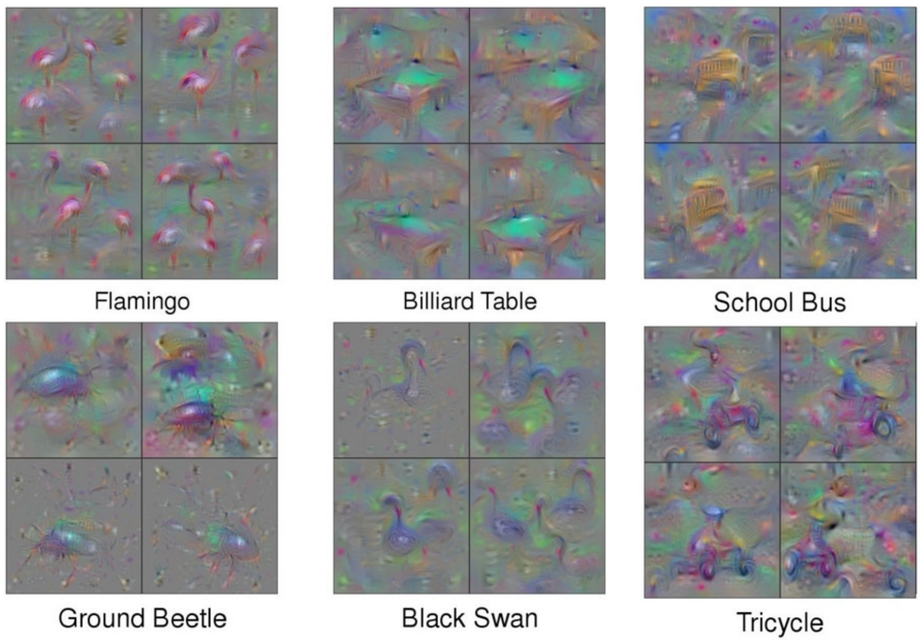
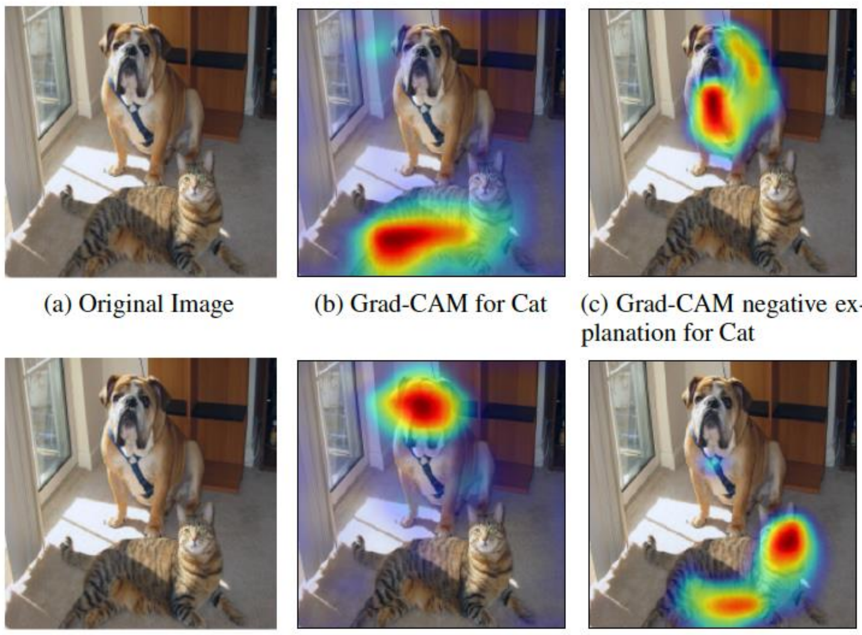


Figure 2.1: Synthetically generated images that maximally activate individual neurons in a Neural Network. They show the pattern each neuron has learned to look for. In this examples, the neurons are from a neural classifier uses to classify images. Similar images can be build for all of the hidden neurons network. Image source: [119].

Our first attempts (introduced in Sections 1.1.1 and 1.1.2) are also perturbation based, since they look for crisp portions of images that if perturbed, maximally distract the classifier. However, unlike aforementioned models where the user has to interpret multiple saliency maps to explain the behavior of a particular classifier on a particular class, our proposed approach by providing visual summaries from the saliency maps, facilitates the interpretation task for the user. In some cases, it is possible to find approaches using strong supervision in the form of annotated saliency maps in order to train CNN [109, 64, 61, 76, 62]. However, this class of approaches is far from the objective of this work, since we aim to generate saliency maps describing what the classifier has learned without any external supervision.

The output of a visualization approach is merely used to unveil hidden patterns the network is more sensible to, and it is constrained to 2D images. This


(a) Original Image (b) Grad-CAM for Cat (c) Grad-CAM negative explanation for Cat

(d) Original Image (e) Grad-CAM for Dog (f) Grad-CAM negative explanation for Dog

Figure 2.2: Example images of the Grad-CAM technique proposed in [91]. Hot regions highlight those parts of the input image that are mostly related to a specific class.

thesis exploits the recent work of [32] to generate smooth saliency maps that are used to individuate class-specific patterns. This method considers deep architectures as black-boxes, which allows to run the method on almost every image classification model.

2.2 Active object recognition

In this section we will review the current state-of-the-art in Active Object Recognition (AOR) with focus on those methods that are related to our overall method pipeline, clustered in three main subjects such as: active object recognition, depth-only object detection, single shot object pose estimators, and visualization of deep networks for saliency maps, multiview/3D object classification and motion policies for active classification.

Motion policies for active classification

Recent works on AOR model the problem with reinforcement learning techniques, such as Partially Observable Markov Decision Processes (POMDP) solved by means of point-based algorithms [2, 86], Monte Carlo tree search [77], Self-Organising Maps [6] or Belief Tree Search [68]. Despite presenting very advanced planning strategies, all these method neglect to focus on how the pose affect the classifier performances. Moreover, POMDP-based approaches suffer from two main problems: 1) intractability in the continuous search space, and 2) the observation model is usually very expensive to compute. A complementary strand of works focus on selecting the NBV within a finite set of candidates by maximizing the the unknown volume explored [4], the information gain [79], the conditional entropy [51], or exploiting unsupervised features learned from depth-invariant patches using sparse autoencoders [24]. All these methods are limited by the fact they require the 3D model of the object to search. To relax this hypothesis, Wu et.al. [115] propose a feature-based model to compute the NBV in 2D space by predicting both visibility and likelihood of feature matching in a mobile robot setup; the limitation of this approach is that it can only handle very simple object geometries. Several AOR approaches assume that the location of the target object is known and do not cope with the problem of detecting the most relevant regions of the environment to be explored. This is the case of [31, 55, 85, 86, 103, 107] where a single object in the environment in a known position was considered. In this thesis we address the case of both single and multiple objects in the scene, handling occlusions and providing an iterative solution to the next-best-view selection problem by accounting for all the objects in a single step.

Depth-Only Object Detection

Traditional 3D methods have been largely limited to instance based recognition or recognition of specific classes of shapes characterized by limited intra-class variations [10, 70]. To overcome the major difficulties for RGB object recognition – namely the variability of textures, illumination, shape, viewpoint, clutter, occlusions and self-occlusions – the use of depth maps instead of traditional images recently gained popularity. Song and Xiao [97] propose to train an SVM classifier on hand crafted 3D features extracted from synthetic images and use a sliding window strategy for object localization. Bo et.al. [10] use an aggregation of handcrafted local kernel descriptors into object level features and train an SVM classifier to perform instance and multi-class recognition. Nakashika et.al. [70] proposes a method which works with depth, RGB and RGBD information. When only depth is available, the method extracts local features based on a histogram of oriented normal vectors, encodes them using locality constrained linear coding and uses spatial pyramid matching, a hierarchical extension of the tradition bag-of-features, to infer the closest image and its corresponding class. Both [10] and [70] show that methods using only on the depth channel for performing object detection, perform worse than when they have RGB or RGBD information available. However, in challenging illumination conditions, depth information is known to be more reliable than RGB.

Single Shot Object Pose Estimation

In this thesis, our active system uses object pose estimator in the pipeline for correctly creating a global saliency model, that is a novel saliency-based representation of the scene needed to succesfully select the next action to perform. To this end we briefly review the state of the art and the choice of the 6D pose detector used in our method. For an updated and more detailed view of 6D pose detectors performance we refer the reader to the Benchmark for 6D Object Pose Estimation (BOP) by Hodan et.al. [45] and [19]. BOP is a RGBD dataset (collection) composed of a challenging combination of objects, scenes and light conditions. It provides a common ground to evaluate the current state of the art methods for single shot pose estimation based on RGBD data. Without being exhaustive (check [45] for an overview), along with classical methods in the literature [43, 11, 44, 48], best results come along with the use of point-pair features [106, 25]. Deep-learning based approaches mostly focus on the use of RGB data, occasionally relying on depth for further refinement. In both BB8 by Rad and Lepetit [83], and Single Shot Pose (SSP) by Tekin et.al. [101], a neural network is trained to regress the image coordinates of an object's 3D bounding box, being the 6D pose recovered through a standard perspective-n-point method. In SSD-6D [53] 2D bounding boxes candidates are extracted in the first stage. The location and size of the bounding boxes allows to build pool of admissible poses, which are further refined based on contour samples, and being the one with lowest projection error picked as the best one. Given best performance matched with code availability, our active recognition method uses SSP [101] as the multi-object pose estimation engine of the pipeline.

Multiview/3D object classification

Several approaches have been proposed for 3D object classification, and can be categorised as view-based, voxel-based and point-cloud-based techniques.

In view-based methods each 3D shape is represented by a set of frames generated from multiple viewpoints. Traditional approaches train one classifier for each viewpoint, assuming that the object lies in a known pose; these methods mostly use hand-engineered features like Fourier descriptors [17], local Gabor filters [26] and Fisher vectors [89]. In [98], a CNN model is trained to extract features from each available view, while a pooling layer fuses these features together and passes them to a subsequent NN architecture for classification. RotationNet [50] is a CNN-based model that takes as input a partial set of multiview RGB images and jointly estimates object's pose and category. All these methods are low-dimensional in the input space, computationally efficient and fairly robust to 3D shape representation artefacts such as holes and noise; despite that, they miss the ability to generalise over the complex 3D shape of an object.

In voxel-based methods, depth data are represented in a fixed-size 3D space in form of occupancy maps. In VoxNet [69] each voxel is assumed to have a binary state, occupied or unoccupied, while in ShapeNets [115] the 3D shape is represented as a probability distribution of binary variables; both of them use a 3D CNN architecture to perform object recognition. These methods suffer from the rigid space representation that limits the expressiveness of the input data; this limitation is overcome by point-based methods.

PointNet [81] learns a set of spatial features of each point independently and then accumulates the features by a symmetric function (*ie.* max-pooling layer). This model has a relatively simple architecture that takes as input the complete point cloud of an object and performs single object recognition and part segmentation. Subsequent models have shown that the classification performance can be further improved by considering the neighbourhoods of points rather than treating points independently due to a better leverage on the local structure features [82, 92]. Despite working very well on single objects where the whole point cloud is available, these methods suffer when part of the point cloud is missing, *ie.* in case of occlusions. In this work (specifically in Chapter 5.2) we leverage the power of point cloud representation by proposing a strategy to intelligently explore the environment and acquire data to fill the gaps in the 3D representation of objects.

CHAPTER 3

Saliency for Regression

In the first part of our research, we mainly focused on regression: we compared different approaches to estimate body fat percentages from simple depth images that can be captured by low-cost sensors. We implemented two frameworks, one based on hand-crafted features, using simple image processing methods to estimate directly from images a set of body measurements (e.g. areas, lengths girths), and one based on Convolutional Neural Networks, applying a direct regression from the grayscale maps representing the body depth, based on a pretrained networks.

With these frameworks, we evaluated the fat percentage predictions obtained with the different methods on depth images of 350 subjects with known body composition estimated with a DXA scanner. Depth images were generated by extracting the z-buffer from the renderings of the 3D body scan models acquired on the group of subjects. In our validation experiments, describe in Chapter 6, we evaluated the effect of different simulated acquisition setups, parameters settings, different image preprocessing and data-augmentation procedures and the addition of priors on height and weight on the prediction accuracy. Furthermore, since the dataset used is composed of professional sportsmen and a control group, we evaluated also the ability of both frameworks of predicting the sport practiced by the subjects with a cross-validation experiment. In specific, we propose a customized ResNet50 regressor to evaluate the whole body fat percentage of the subjects directly from the depth acquisitions. Using the same input data, we also set up a neural classifier to predict the sport category of the athletes.

Despite the limited numbers of subjects and the restricted variability of body types (all males, Caucasian, with a small number of obese), the results obtained are promising and can be considered a first step towards the development of quick and cheap body fat estimation tools that can be extremely useful for sport, health and fitness applications.

3.1 Depth image data

We demonstrate our anthropometric analysis tool on a novel dataset (*Sports-Depths*) built on top of an archive of 3D body scans of a set of 350 subjects who performed also DXA scanning for body composition assessment. The body scan archive was provided by the Department of Neuroscience, Biomedicine and Movement (DNBM) of Verona.

The subjects represent a narrow but homogeneous excerpt of population, being 250 professional male sportsmen and a control set of 100 healthy young males. The average age of the group is 26.5 (7.8 std.), and the whole body fat percentage spans from 6.64 to 36.78. During the scanning, subjects wore close-fitting underwear, and the same condition is expected in the depth sensor acquisition protocol. The distribution of the whole body fat percentage, estimated with the DXA scanner in the group is portrayed in Figure 3.1. The distribution has a mode around 10%, and has a long right tail, that actually stops at limited fat values, since we deal with sportmen.

From the 3D full scans, we obtained simulated depth scans by performing depth renderings using Blender [9]. Simulation helps to get controlled data, which have been recently shown to be highly beneficial in deep learning for modeling humans [105]. Rendering options were set in order to simulate the spatial resolution and field of view of a commercial depth sensor (Asus Xtion Pro), and an acquisition protocol where subjects stand in front of the sensor at

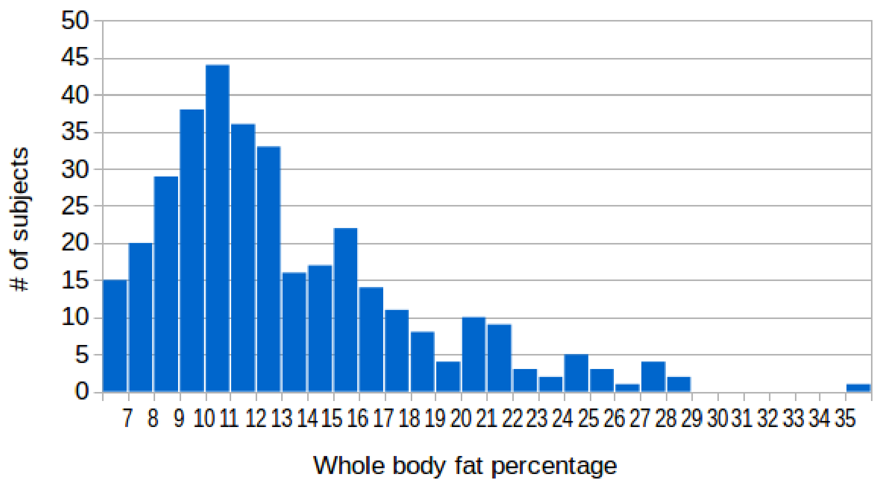


Figure 3.1: Distribution in terms of whole body fat percentage of the dataset.

a distance allowing a full acquisition of the body size (the *Front* data) and then rotate 180 degrees to have the back side acquired (the *Back* data).

The goal of our CNN-based fat estimator is to provide a method for automatically measure whole body fat percentage from a inexpensive and quick range scan. As the previously described dataset was composed of simulated depth scans, even if performed on real body geometries, and even if the rendering was made with a resolution matching the specifications of the current generation of cheap depth sensors, it would be clearly better to work on directly acquired depth images.

We were not able to collect a large dataset of real depth scan with associated DXA assessed body composition, but, in order to demonstrate that the rendered images are good to train an online fat estimator from depth scans, we were able to acquire a small test set with an Asus Xtion Pro sensor. We captured, using this sensor, depth sequences for 15 professional soccer players (only front views) with known whole body fat percentage, measured in the same day with a DXA scanner. This small dataset allow us to test the feasibility of the body fat percentage estimation on real images using a network trained on the SportsDepths images.

3.1.1 Depth image preprocessing

For the image-based fat regression, depth images are preprocessed with a simple automatic tool removing the floor (if present) and then cropping and resizing the maps. Output image size is 224×224 pixels, fitting the input size of the pre-trained ResNet-50 used in our method, and cropping is such that the body shape covers the full image height.

Similarly cropped and resized images for all the subjects available in the original study are included in the publicly distributed SportsDepth dataset and referred to as the *Full-Body* depth images. Examples are shown in Figure 3.2 left (front and back).

Our automatic pre-processing procedure provides also a differently cropped or resized 224×224 pixels image output, similarly used in our tests with the ResNet50 regressor, and referred to as the *Torso* version of the dataset. This version is cropped by identifying the location of the neck and of the pubic region with simple heuristics, and mapping the vertical range between the two landmarks in the image height, as shown in Figure 3.2 on the right. We compare in the experimental tests the results obtained with the two different croppings. In principle the torso cropping could provide a better analysis of the region where body fat is maximally accumulated, and could reduce the effect of varying pose, but neck and legs could encode relevant information as well and measurements on them are included in the classical body fat regression formulas.

Another preprocessing step performed before the regression consists in mapping the original 12 bits into an 8 bit scale with z resolution of 3mm, putting the closest point to 255 and decreasing values for larger depths.

This depth resolution actually matches the one provided by an acquisition performed with low cost depth sensors at a distance of about 1-1.2 m. [20].

Figure 3.3 shows full size and torso cropped images obtained from the an Asus Xtion pro acquisition, noisier, but quite similar to those created with the whole body scan rendering.

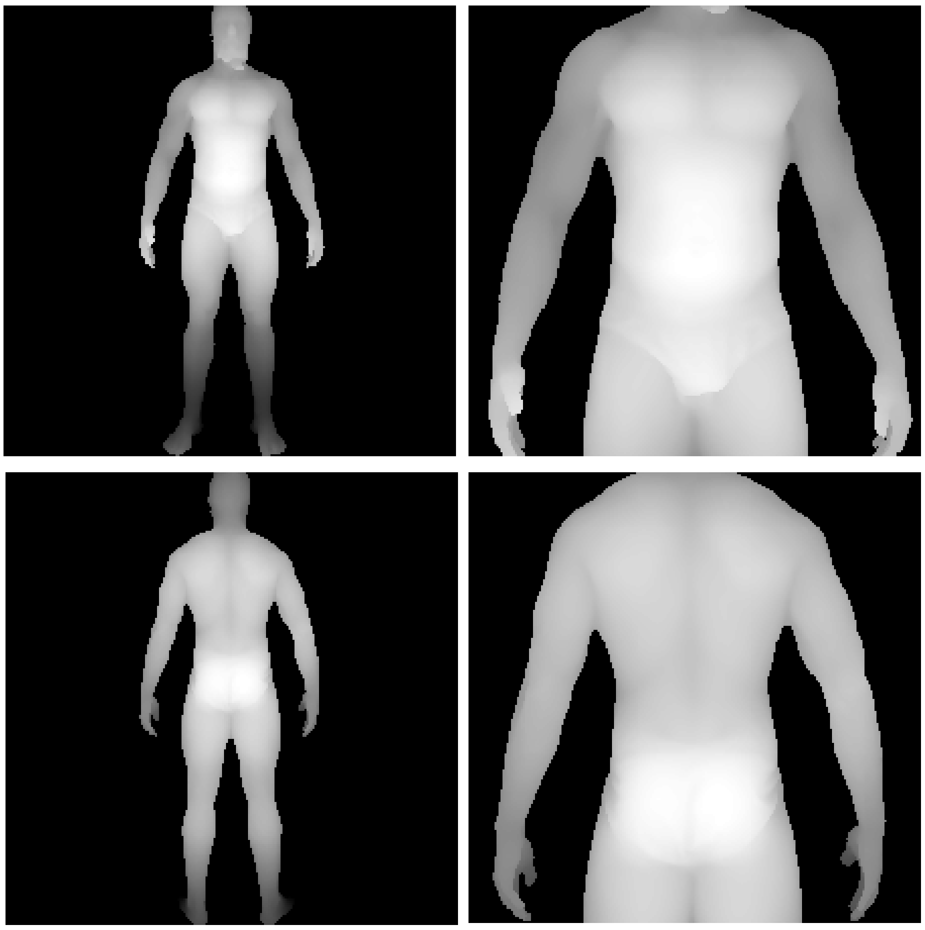


Figure 3.2: Depth renderings examples of a single subject. From left: Full body frontal view, Torso frontal view, Full body back view, Torso back view. Fat percentage value of this subject: 13%.

3.2 Regression

The ResNet architecture [41] is well-known for performing extremely well in classification tasks. We convert a ImageNet-pretrained ResNet-50 model into a regressor by removing the last classification layer, substituting it with a fully connected one, mapping the convolutional features directly to the raw fat index. The Euclidean loss function is employed (Smooth L1 loss) with a batch size of 16. ADAM optimizer is run with learning rate 0.0001, reduced by a factor 2 every 5 epochs, and weight decay 0.0005. An input image is indicated as x, its ground truth fat percentage y and the estimation is $\hat{y}$. A single regressor

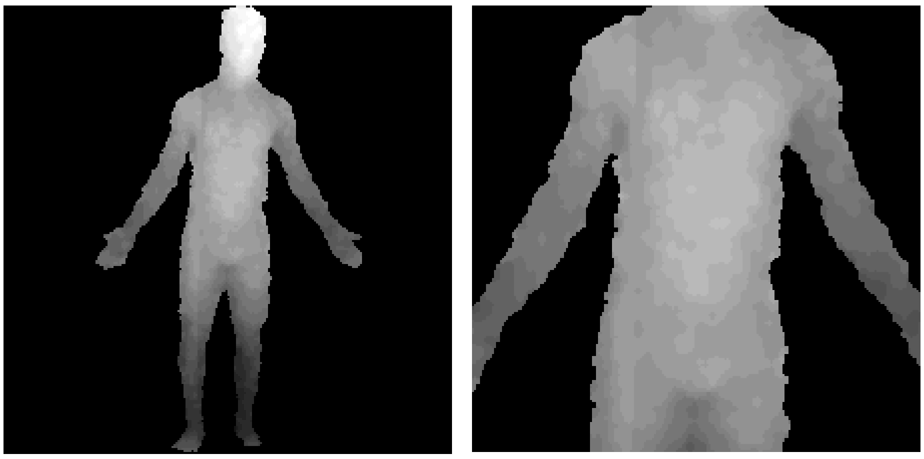


Figure 3.3: Depth maps (Full-Body and Torso) acquired with the Asus Xtion depth sensor.

is trained on the Front data (providing $\hat{y}_{Front}$ estimates), another one on the Back version. We also fuse the information of the two regressors: we estimate the train error as the average of the standard error SEE $\epsilon = \|\hat{y} - y\|$ on all the samples of each view, giving $\bar{\epsilon}_{Front}$ and $\bar{\epsilon}_{Back}$. These average errors are normalized and turned into weights:

$$w_{Front} = 1 - \frac{\bar{\epsilon}_{Front}}{\bar{\epsilon}_{Front} + \bar{\epsilon}_{Back}} \tag{3.1}$$

and similarly for w_{Back} so that the *combined* regression on a given sample is $w_{Front} * \hat{y}_{Front} + w_{Back} * \hat{y}_{Back}$.

We compare the result of the image-based regression with the classic measurement-based regression approach. The body scan dataset exploited for this work are not associated with a set of measurements as done by specialized anthropometric suites commercially available. However, we were able to apply on them the automatic measurement code proposed in [36], extracting a set of 14 simple measurements from the automatically segmented triangulated mesh. This set is composed of global (volume, surface area, height) and localized geometrical features (approximate Trunk Volume, Maximal average trunk section radius, Maximal anterior-posterior distance, Maximal trunk width, Minimal trunk width, Maximal trunk section area, two shape roundness indicators, maximal forearm, calf and thigh section radii). In [36] these measurements were used to estimate

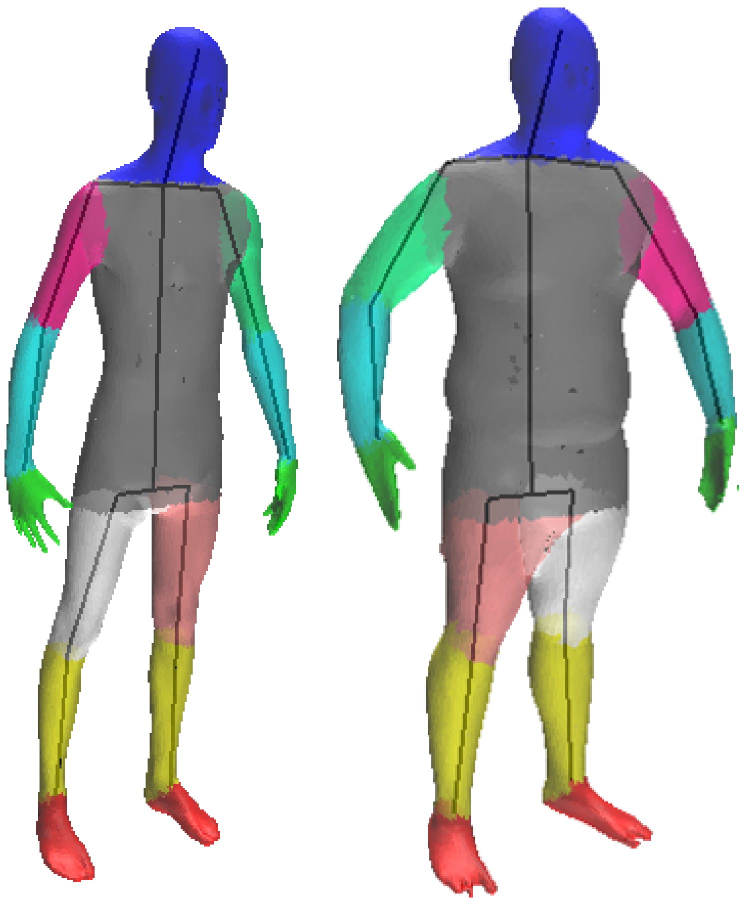


Figure 3.4: Rendered 3D scans from the athletes dataset (left: fat percentage 10.5, right: fat percentage 25.5), processed with the method described in [36] to estimate pose, segment body parts and evaluate anthropometric measurements.

body fat percentages on a set of obese and control subjects. Using the same framework, exploiting the authors' code to estimate the measurements and performing linear regression to derive fat predictors, we calculated the whole body fat percentage values from the 3D body scans of our dataset, applying the same crossvalidation procedure of the depth-based evaluation.

Figure 3.4 shows example results of the automatic mesh segmentation and pose estimation performed on two models of our dataset.

3.3 Visualization

Deep learning triggered the development of *visualization* approaches, aimed at understanding at the image level the regions more salient for a particular network architecture for classification [120, 94, 32, 125]. In our case, having visualization capabilities on regression would allow to understand, given an image, which portions of the human body have been used to estimate the fat percentage, opening to intriguing commercial perspectives.

In this part of the thesis, we customize a well-known visualization technique for explaining classifiers [120] for the regression case. In [120], a small patch of a test image is covered, creating a corrupted image; the corrupted image is then sent to the trained classifier, and the change in classification confidence (if any) is backprojected on the image, in the central location of the patch as a new pixel value. This process is repeated in order to cover all the image pixels, resulting in a saliency map that indicates which parts bring high variations. Here we apply the same idea, backprojecting instead the signed difference of the estimated fat percentage with respect to the ground truth, resulting in a saliency map of negative and positive values, similarly to [125], where negative (positive) values indicate a underestimation (overestimation) error. Here we adopt a 23×23 patch, giving interpretable feedbacks in terms of saliency, and a sufficient spatial resolution. High absolute values of this map, highlight the regions that are particularly important in the image for a correct fat percentage evaluation.

CHAPTER 4

Experiments: Regression

To demonstrate the feasibility of the depth based fat percentage estimation, we designed a 10-fold crossvalidation test, subdividing the whole dataset in different training/test subsets, and collecting all the test results to obtain fat estimates for all the subjects. We analyze then the correlation of the estimates with the associated DXA values, and the Bland-Altman plots showing the accuracy variation across the fat percentage range.

The experiments show that a reasonable fat estimation is achievable using the depth images (Section 4.1). The standard error of estimate is lower than that obtained with a measurement-based approach [36]. The trained regressor is also used to estimate fat percentage on the real depth images acquired with Asus Xtion sensor (Section 4.2). Subsequently, visualization results are detailed for both the Front and Back versions of the main dataset (Section 4.3).

4.1 Crossvalidation experiment

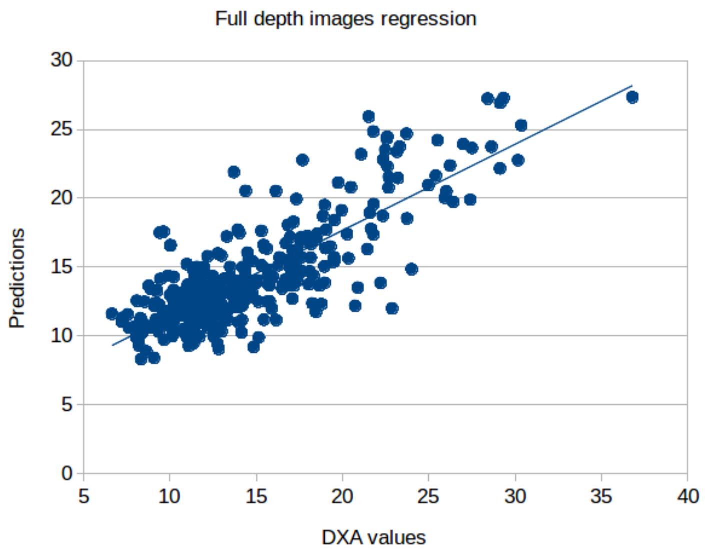
We divided the full dataset in 10 parts and for each subset we estimated the values of the whole-body fat percentage by training our ResNet50 on the images of all the remaining subjects (10-fold crossvalidation). To keep the train and test set similarly balanced, we created the tests set by sampling uniformly the subject list sorted by fat percentage.

In each training phase, in order to address the fat index unbalance (already discussed in Section 3.1), we perform non-uniform data augmentation, taking the samples out of the $[10, 18]$ fat percentage range, augmenting them with noise injection and horizontal flipping with a rate of 2 with respect to the inlier samples. This bring us to a training procedure converging in 16 epochs, less than 10 minutes on a NVIDIA GeForce GTX 1080 machine.

We performed the crossvalidation procedure for the regression from single images (front/back) and the combination of the two and for the full body and torso versions. At the end of the procedure we obtain fat percentage estimates for all the subjects in the database and all the methods, exploiting maximally the limited number of subject data and trying to compensate the fat distribution unbalance. Table 4.1 shows the SEE using the Full-Body and the Torso data, in the Front/Back version, and with the combined regression. As visible, error is lower on Torso data: this can be explained by the higher resolution of the data and the limited effect of pose variations. Combined regression gives a better performance, since more information is jointly considered. In general, the error is reasonable and actually lower than the one obtained with the measure-based framework in [36] (see Table 4.1).

The error is also similar to or lower than the ones reported in the literature for other popular estimation techniques that are mostly more expensive and time consuming [112, 21], whose application to our data was not possible since no free code is available. Plots in Figure 4.1 aggregate the Front and Back-based estimates vs reference DXA values of the fat percentage, in the Full-Body and Torso versions, showing a good and statistically significant Pearson correlation.

Figure 4.2 shows the Bland-Altman plots portraying for the Full-Body and Torso versions, the distribution of the differences between the estimates and the reference DXA values of the fat percentage. One can note that the estimates are not biased and the average error is constant across the fat range with only few outliers lying outside the lines of agreement. The difference of the means of DXA and the estimates are not statistically significant. Red lines show the so-called lines of agreement, showing the error range where most of the differences



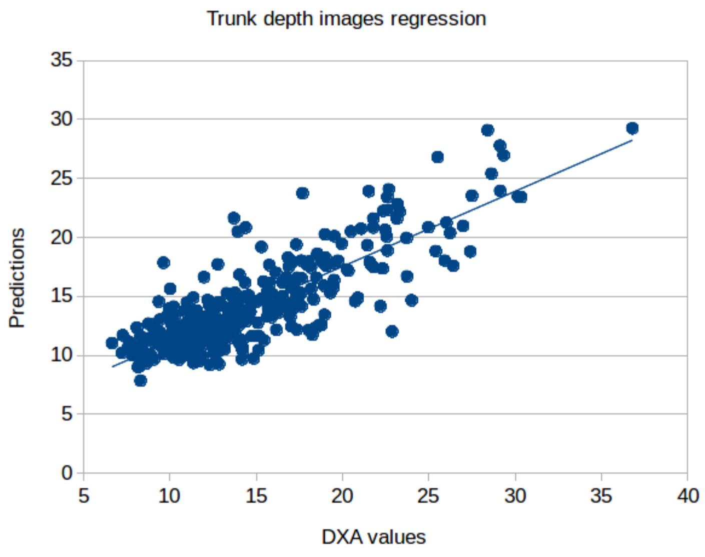


Figure 4.1: Top: Fat percentage predictions obtained combining Front and Back Full-Body depth images (10-fold crossvalidation) vs DXA reference values. Bottom: same plot created with the predictions estimated on Torso images.

(95%) fall. As discussed earlier, discrepancies are not negligible, but similar to those obtained with other methods currently used in the practice.

The better accuracy provided with the Torso images is probably due to the better $x - y$ resolution of the body shape encoding, compensating the lack of

Method	SEE	Pearson
Front full depth	2.34	0.79
Back full depth	2.35	0.78
Front torso depth	2.18	0.82
Back torso depth	2.24	0.80
Combined full	2.20	0.80
Combined torso	2.05	0.83
Measurement based [36]	2.70	0.71

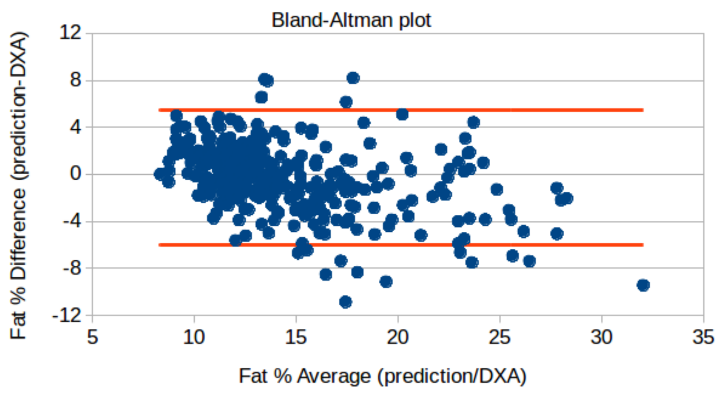
Table 4.1: Standard errors of estimates of fat percentages obtained (vs. DXA reference values) and Pearson correlation scores of fat percentage predictions obtained with the different depth images and with measurement-based regression (see text).

information about legs and arms. It is worth noting that the pose of the subjects during the body scan was not constant and, therefore, arms and legs positions are heavily varied in the rendered images. Our depth image based fat regression seems reasonably robust against these variations despite further analysis have to be performed, but we expect that a fixed pose in the acquisition protocol might result in an increased accuracy.

Figure 4.3 and Figure 4.4 show that with the measure-based approach of [36] the distribution of the errors is similar, the average difference is also close to zero and not varying with the fat percentage, but errors are larger. We applied the method as described in the paper, performing linear regression on the set of measurements and performing the crossvalidation test. The relatively poor performances could be due to the limited number of measurements and also to the varying pose of the subjects.

4.2 Results on Asus Xtion Pro Imagery

On these data, the average SEE is 2.04% for the Torso image and 2.49% for the Full-Body image. These values are reasonably small and actually quite close to the one obtained in the crossvalidation procedure performed on the SportsDepths data. Considering that no noise reduction strategies exploiting



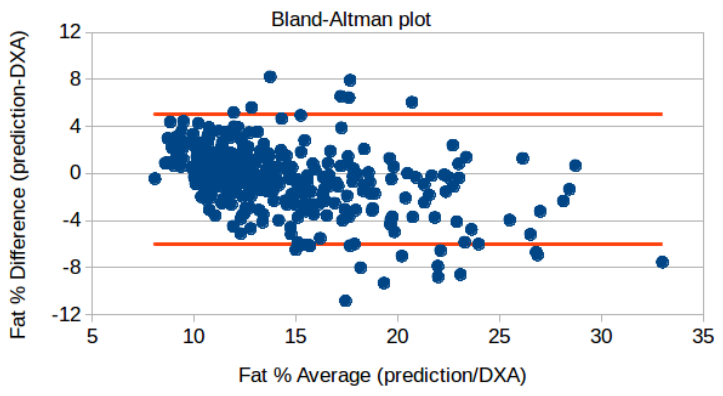


Figure 4.2: Bland-Altman plots showing the differences between fat estimations (10-fold crossvalidation) and DXA values vs. the mean of the two quantities. Top: Full-Body estimates. Bottom: Torso estimates.

the time sequences have been performed and that we use only the front view, these results seems to support our claims about the feasibility of a cheap and reliable fat estimation tool based on depth images. Another thing to be taken into account is that novel low cost sensors are now approaching the market (e.g. Realsense D400, Asus Xtion 2), promising increased quality and depth resolution.

4.3 Visualization Results

In Figure 4.5 we show our saliency maps estimated on full and torso depth images of a low fat and high-fat subject. Following Section 3.3, the saliency maps

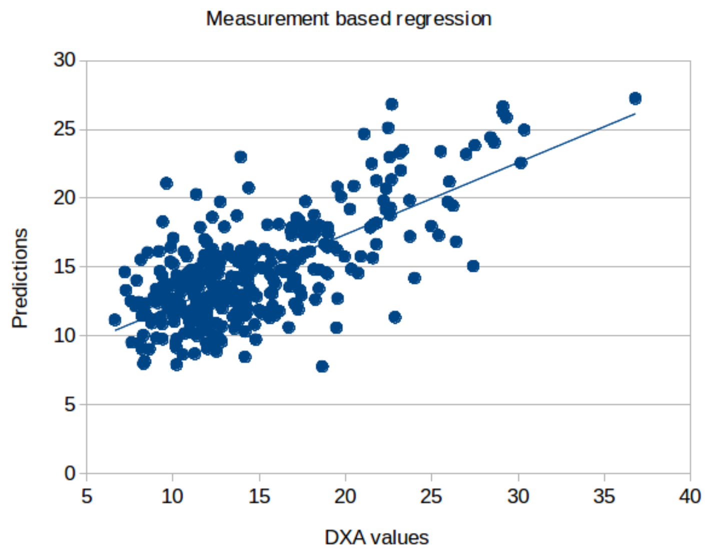


Figure 4.3: Fat percentage predictions obtained with the measurement-based method in [36] vs DXA estimated reference values.

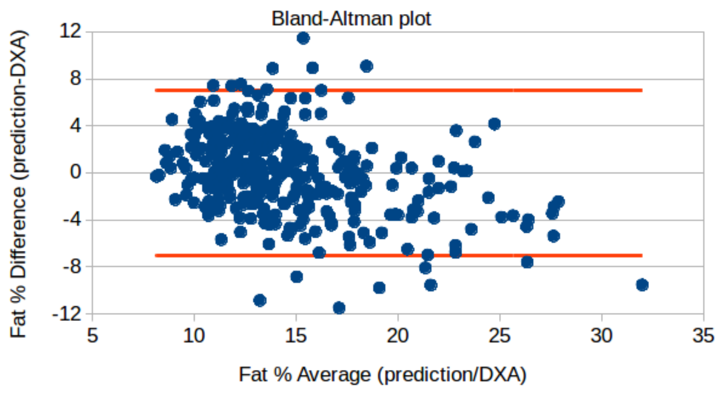


Figure 4.4: Bland-Altman plots for measurement based fat percentage estimations (10-fold crossvalidation) and DXA reference values.

enhance regions that are most important to keep the regression results close to the real value. Masking, that is, ignoring blue areas we would obtain a relevant underestimation of the fat percentage, masking the red one an overestimation. We note some interesting facts: while the most important regions to characterize well the fat subjects are the shoulder area together with hip and lower abdomen,

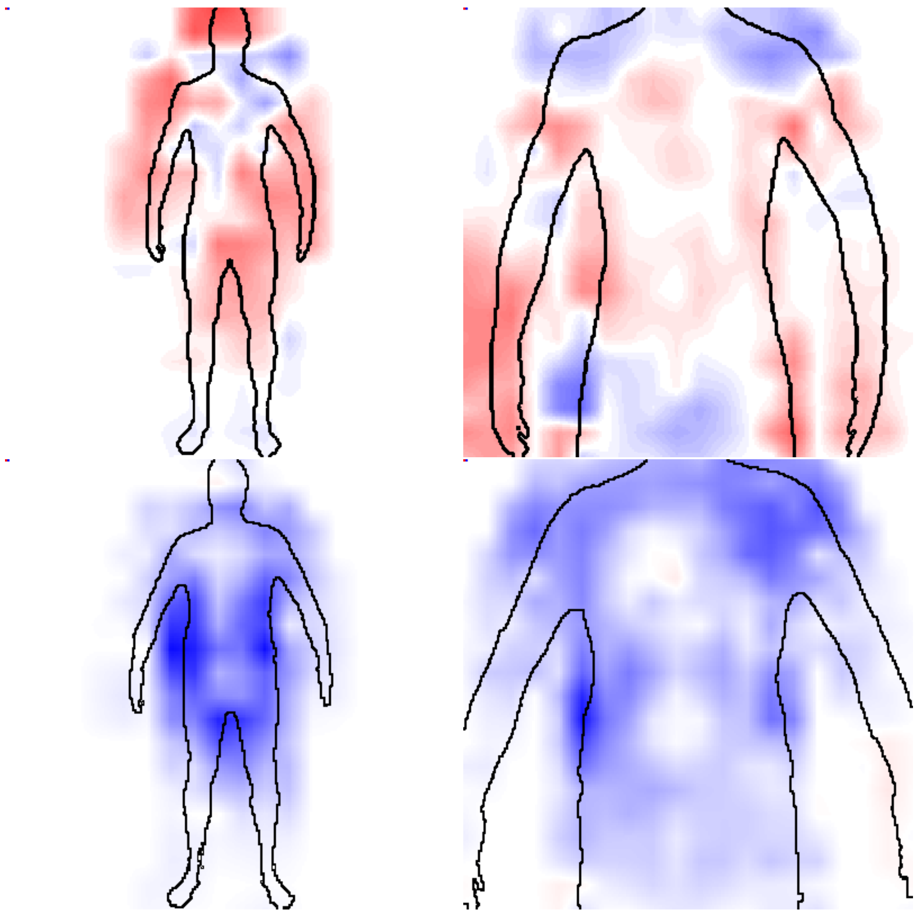


Figure 4.5: **Output of the visualization tool**. Top left: salient regions extracted on the full depth image of a low fat subject (8.2 %) with the body silhouette superimposed. Top right: salient regions extracted on the torso depth image of the same subject. Bottom: salient regions extracted on the full depth image of a fat subject (30.1 %) with the body silhouette superimposed.

for the correct characterization of skinny subjects head and arms are important as well. It seems that, for the high fat percentage subjects, the visualization tool acts as a sort of fat accumulation localizer, while in thin subjects the proportions of the body shape are evaluated.

This is even more evident looking at the accumulation of the maps estimated for the full body depth of the 50 thinnest and the 50 fattest subjects (Figure 4.6), showing clearly the relevance of the head and arms regions for the estimation of skinny subjects' fat, while for the fattest ones only torso, abdomen and, in a lighter way, legs are important.

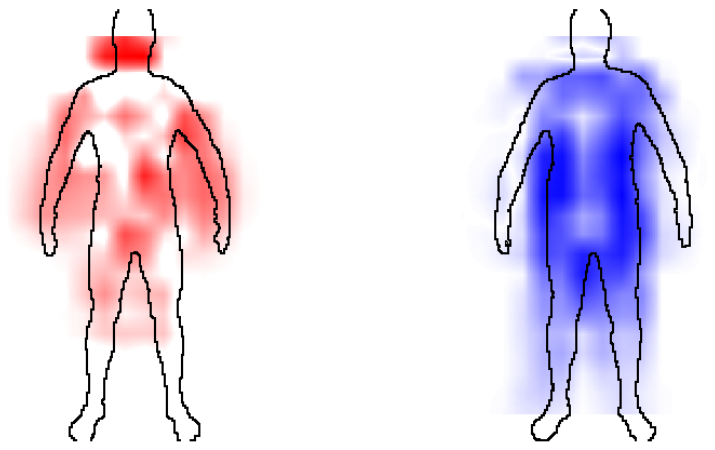


Figure 4.6: Accumulated saliency maps for the full body fat estimation. Left: 50 subjects with lowest fat percentage. Right: 50 subjects with highest fat percentage. It is here evident that, while for fat subjects the accuracy of the estimate is not influenced by head and arms regions, this is true for the thinnest ones.

4.4 Discussion

We succeed in showing that a simple tool for body fat percentage estimation from simple low-resolution depth maps is feasible. The errors obtained on our SportsDepths dataset are reasonably small (smaller than those obtained with a simple measurement-based techniques) and the early results on low cost depth sensor data are encouraging.

A mandatory step to substantiate our claims will be the creation of larger datasets including non-fit subjects with wider fat variation, trying to balance the dataset and reaching an order of magnitude more in terms of number of samples. To develop a generic application for body fat estimation from depth sensor data, it is necessary to train the system with images of both men and women and subjects belonging to different ethnic groups as the fat distribution patterns varies within the different subset.

We expect that the non-linear regression approach based on convolutional network, would be able to cope well with multiple functions linking shape to fat percentage and a balanced training set could improve the estimation accuracy.

Another interesting fact is that, knowing the additional labels (e.g. sex, body type, etc.) the network could be trained to estimate simultaneously labels and body fat. We believe that the proposed SportsDepths dataset is quite interesting for research purposes and we decided to make it available for the scientific community. Depth images are available at the web address `www.andreagiachetti.it/SportsDepths`, with associated metadata, including fat percentages and the label related to the sport practiced by each subject.

The reasons why we consider the dataset interesting are many:

- It is the very first case where such kind of dataset is disseminated. Acquiring reference body fat values with DXA or hydrodensitometry is expensive. Companies developing fat estimators from body scanner data must clearly collect similar data to validate their applications, but they do not distribute them to the scientific community.

- Using this public benchmark, CNN-based regression methods can be tested and improved, and also fat estimation application can benefit of these improvements.

- A depth based fat estimation application represents for sure an attractive commercial perspective: cheap stereo sensors are invading the market, while other inexpensive solutions of fat estimation based on bioelectric impedance, skinfold or anthropometric measurements are not reliable or require time consuming and/or expensive procedures.

- Metadata related to the sports activities are also quite interesting for practical applications, as the analysis of body features related to optimal performances is of particular interest in sports science [74].

CHAPTER 5

Saliency for Classification

In this chapter we present the two main steps of our research. In particular, we focus on the classification task, where our major contribution is proposed. Firstly, in Section 5.1 a novel framework for visualizing 2D patterns is described: we extract a saliency map for each input image aiming to cluster hotter regions, *ie.* significant areas for classification, into visual summaries. The second part of the chapter is related to the AOR problem. After introducing our approach to solve the active recognition system through the ICP algorithm in Section 5.2, we apply the same visualization strategy to 3D: saliency maps are mapped onto the reference mesh of a target object, so that an active recognition framework is able to select the next view according to the performance of the classifier exploiting such novel *saliency volumes*, as described in Section 5.3. Lastly, we present an extension of the 3D saliency volumes application by injecting the saliency volumes in a similar pipeline (Section 5.4) showing our preliminary results in this challenging scenario consisting of multiple objects.

5.1 Understanding CNNs by Visual Summaries

The results of the preliminary analysis on 2D image recognition led us to a novel data representation, that is a compact visualization of salient areas, grouped as **visual summaries**, where each summary is a cluster of a recurrent pattern.

Our method could be subdivided in two phases: *mask extraction* and *clustering*. The former captures the visual patterns that are maximally important for the classifier: if a detail is systematically present in the images, it will be selected to form a new cluster, *ie.* a visual summary. The latter part organizes the selected visual patterns into summaries: the distance among all patterns is computed and exploited by a clustering procedure to generate the visual summaries, the main contribution of this portion of the thesis.

Visual summaries are then evaluated from several point of views: through a user study, we demonstrated the visual summaries produce a compact representation of the data, which is easier for human users to understand and analyze. Finally, the first attempt to use the visual summaries was driven: the main classification model has been flanked by a set of supporter classifiers, that is a SVM classifier has been trained for each visual summary. The detailed description of such framework is organized as follow: in Section 5.1.1 we describe the procedure based on [32] to compute the image-specific saliency maps. The generation of the visual summaries, as also the overall pipeline of the method, is described in Section 5.1.2.

5.1.1 Mask extraction

Let us define a classifier as a function $y = f(x)$ where x is the input image and y is the classification score vector, in our case the softmax output of the last layer of a deep network. generating an output image in a global fashion.
Our starting point is the gradient-based optimization of [32]. In that method, the output of the optimization is a mask $m : \Lambda \rightarrow [0, 1]$ with the same resolution of x, in which higher values mean higher saliency. The original optimization equation (Eq. (3) of [32]) is

$$m = \underset{m \in [0,1]^\Lambda}{\operatorname{argmin}} f_c(\Phi(x; m)) + \lambda_1 \|1 - m\|_1 \qquad (5.1)$$

where $\Phi(x;m)$ is a perturbed version of x in correspondence of the non-zero pixels of m, in which the perturbation function Φ does blurring:

$$[\Phi(x;m)](u) = \int g_{\sigma_0 m(u)}(v-u)x(v)dv \tag{5.2}$$

with u a pixel location, $m(u)$ the mask value at u and σ_0 the maximum isotropic standard deviation of the Gaussian blur kernel g_{σ_0}, $\sigma_0 = 10$. The function $f_c(\cdot)$ is the classification score of the model for the class c: the idea is to find a mask that perturbs the original image in a way that the classifier gets maximally confused, rejecting the sample for that class. The second member of Eq. (5.1) is a L1-regularizer with strength λ_1, which guides the optimization to minimally perturb the pixels of the input image. The authors of [32] suggested also a total variation (TV) regularizer $\sum_{u\in\Lambda} \|\nabla m(u)\|_\beta$, in which the sum operates on the β-normed partial derivatives on m, calculated as the difference of the values of two contiguous pixels according to the direction.

We contribute here by adding a sparsity regularizer $\sum_{u\in\Lambda}|1-m(u)|m(u)$ enforcing sparsity [102] in the values of the mask m, making it binary. This regularizer has been designed to start working after a certain number of iterations, so we can get a rough version of the mask before starting to optimize its crisp version, in line with the MacKay's scheduler of [65]. The final version of the optimization is thus:

$$\begin{aligned} m = \operatorname*{argmin}_{m\in[0,1]^\Lambda} & f_c(\Phi(x;m)) \\ & + \lambda_1 \|1-m\|_1 \\ & + \lambda_2 \sum_{u\in\Lambda} \|\nabla m(u)\|_\beta \\ & + \lambda_3 \sum_{u\in\Lambda} |1-m(u)|m(u) \end{aligned} \tag{5.3}$$

with λs and β values set to $\lambda_1 = 0.01$, $\lambda_2 = 0.0001$, $\lambda_3 = 0$ and $\beta = 3$ during the first 300 iterations. We then modified the parameters to $\lambda_2 = 1$, $\lambda_3 = 2$ for the next 150 iterations. At the end of the mask extraction stage, each image x_i, $i = 1...N$ of a given class becomes associated to the corresponding mask m_i.

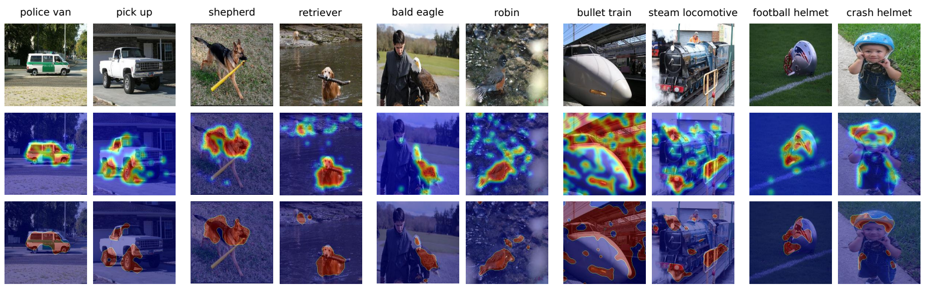


Figure 5.1: Qualitative analysis of the masks. First row, original image from different Imagenet classes. Second line, heatmaps computed with the method proposed by [32]. Third line, crisp masks computed with our optimization procedure. Best in colors.

5.1.2 Clustering

Each saliency mask m_i can be analyzed by considering its connected components $\{r_j^{(i)}\}_{j=1\ldots J_i}$ called here *regions*. Some of the regions are to be clustered together across multiple images of the same class to form the visual summaries of that class. The idea is that each region represents an articulated visual item composed by *parts*, and a summary is an ensemble of regions exhibiting at least a common part. A graphical sketch of the procedure is shown in Fig. 5.2.

In our implementation, object proposal technique [104] is employed to extract the parts of the regions. Next, the proposal flow technique [38] is incorporated to cluster the regions. Indeed, object proposals have been found well-suited for matching, with the proposal flow exploiting local and geometrical constraints to compare structured objects exhibiting sufficiently diverse poses [38].

Our procedure begins by considering the whole images of a class without resorting to the regions, in order to account as much as possible of the context where regions are merged. Given a class, all of its N images are processed; from image x_i, the set of object proposals P_i is extracted. Next, all of the images are pairwise matched adopting the proposal flow algorithm. Each pair of images $< x_i, x_j >$ will thus produce a $M_i \times M_j$ matrix Q_{ij}, with M_i indicating the number of object proposals found in image x_i. Each entry of the matrix $Q_{ij}(k,l)$ contains the matching compatibility between the k-th and the l-th object proposal of the images x_i and x_j, respectively.

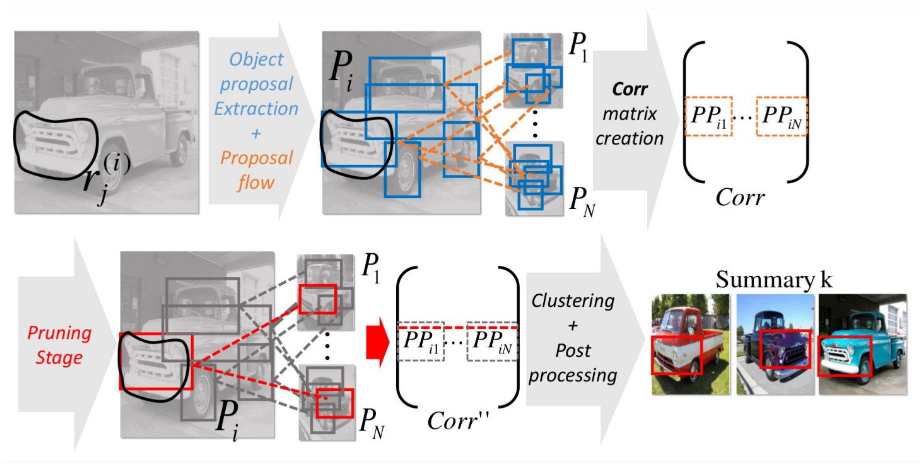


Figure 5.2: Sketch of the clustering phase of our proposed method. The pipeline starts with region proposal computation and Proposal Flow-based matching. The region proposals are pruned using overlap measurement on the saliency maps. The resulting matrix of compatibility values is then used as input for a clustering algorithm.

After this step, all the object proposals of all the pairs of images are combined together into a $N_P \times N_P$ matrix $Corr$, where $N_P = \sum_{i=1\dots N} M_i$ is the total number of object proposals. A given row of $Corr$ will contain the matching score of a particular object proposal with all the remaining object proposals. $Corr$ could be very large but can made easily sparse by thresholding the minimal admissible matching score.

At this point, we refer to the image regions $\{r_j^{(i)}\}$ extracted earlier and select from $Corr$ all of the object proposals that overlap sufficiently with a region (overlap ratio higher than 75%). In the case of two overlapping proposals, one of them is removed if the ratio between the two areas is less than a certain threshold (2 in this work). The pruning stage leads to the $Corr''$ matrix.

The matrix $Corr''$ is considered as a similarity matrix, and the Affinity Propagation clustering algorithm is applied [33] on top of it. Affinity Propagation requires only one parameter to be set (making parameter selection easier) and it is able to discover the number of clusters by itself. The resulting clusters are ensembles of parts which, thanks to the proposal flow algorithm, should consistently identify a particular portion of an articulated object, thus carrying a clear visual semantics. Next, post-processing is carried out to prune out unreliable clusters. To this end, Structural Similarity Index (SSIM) [111] is applied to all the pairs of a cluster, discarding it as inconsistent if the median value of SSIM for that cluster is lower than a threshold based on the global median of SSIM within the whole class (90% in this work). This has the purpose of removing obvious mistakes in the clusters, caused by the variety of different poses that the proposal flow has not been able to deal with. Experimentally we found that in some cases of objects oriented in opposite directions, like cars towards right and left, proposal flow did not work properly providing erroneously high matching scores, as for some complex not rigid objects like animals in drastically different poses.

All the parts of a valid cluster are highlighted in red and shown surrounded by the regions they belong to; this eases the human interpretation and provides a summary (see an excerpt in Fig. 1.1). An explanation is provided for each image class using a different number of summaries, depending on the number of valid clusters that have been kept.

5.2 Active 3D Object Classification

The natural evolution of the work presented in Section 5.1 is the application of visualization techniques to the 3D domain. We propose to face the active object recognition problem in two steps: the first, exposed in this section, considers only the geometrical information of the scene, *ie.* the occupancy of the objects is taken into account as the only significant information to decide the next operation to perform. In the second step, we inject the saliency volumes computed as described in Section 5.3 as a proxy for classification. The objective of this research is to verify the robustness of the volumes representation in a difficult scenario where occlusions may occur.

Let us consider a scenario where N objects belonging to the set of classes $\mathcal{C}$ are distributed in an area and a robotic platform equipped with depth sensor with known camera pose acquires a set of depth frames at each camera movement [1]. Our goal is to assign the correct label to each of the N objects using a finite set of camera moves. The overall view of the proposed active 3D object classification pipeline is depicted in Figure 5.3. We assume to have a CAD model for each object class. The number of objects in the scene is also assumed to be known to simplify the process of point cloud segmentation.

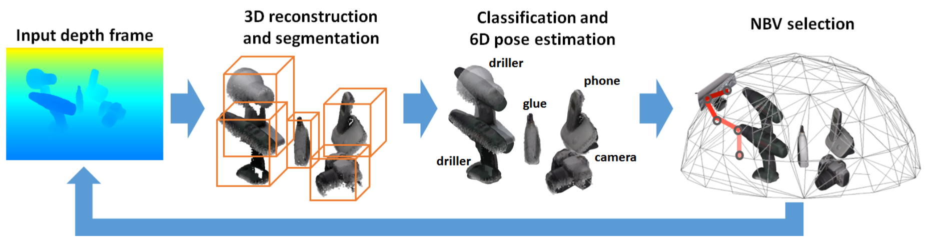

Input depth frame
3D reconstruction and segmentation
Classification and 6D pose estimation
driller
glue
phone
driller
camera
NBV selection

Figure 5.3: Overall view of the proposed active 3D classification system with a typical scenario with a set of objects occluding each other. The system reconstructs the scene and performs classification and pose estimation with refinement using geometric alignment. Given the 3D scene, the system chooses a next best view that maximises the visible object surfaces while avoiding already visited areas in order to achieve a better 3D classification. Best view in colors.

At each time step the sensor acquires a depth map of the scene and we isolate the foreground by first truncating the depth within a predefined distance and then removing the plane where the objects are lying on, *ie.* $z = h$ where h is the height of table surface w.r.t. the world coordinate. The foreground is then segmented in object candidates (see Section 5.2.1) and each segment is processed by a PointNet [81] model to generate class candidates. For each segment, we take into account the top class candidates for further refinement in the pipeline. Given the segmented point cloud (for each segment) and the class candidates, we use DenseFusion [108] to provide an initial pose estimation (see Section 5.2.2). Then we use the Iterative Closest Points (ICP) algorithm [5] to perform geometric refinement among the top candidate labels and updates the segment label as well as its pose. Lastly, we select the NBV that maximises

[1]Computing a reliable camera pose while the sensor is moving is not the focus of this work. The sensor's pose can be given by direct kinematics if the camera is equipped on a robot, by a SLAM approach if hand-held, or by integrating both information.

the visibility of all objects (accounting for occlusions) while avoiding already visited areas. We approximate the visibility by projecting the corners of the bounding cuboid of each segment (with perceived label and pose) onto the image plane. To account for occlusions, we project the segments sequentially with a z-buffer from the closest to the farther from the sensor. The number of pixels within the convex hull bounded by the projected corners of the object is used to approximate the visibility of each segment. We keep moving the sensor until the camera positions reaches a fixed number of steps.

5.2.1 Reconstruction and segmentation

Let the observation input be $\mathcal{O}(t) = \{\mathcal{D}(t),\ \mathbf{v}_W(t)\}$ at a time instance t, where $\mathcal{D}$ refers to a depth map and $\mathbf{v}_W(t) = \{\mathbf{R}_W(t),\ \mathbf{t}_W(t)\}$ refers to a viewpoint pose characterised by rotation matrix $\mathbf{R}(t)$ and 3D translation $\mathbf{t}_W(t)$ in the world coordinate. We acknowledge that there are a few popular real-time SLAM methods with RGB-D stream, such as KinectFusion [71] or ElasticFusion [113, 114], that perform dense reconstruction without the prior knowledge of camera poses. While in our setting, since the camera poses are available thanks to the known robotic kinematics and hand-eye calibration, we opt to a volume integration method using the Truncated Signed Distance Function (TSDF) [124].

At each time t, the depth image $\mathcal{D}(t)$ is registered to the previously reconstructed point cloud (if $t \neq 0$, otherwise we use the first depth image as a reference) using the camera viewpoint pose $\mathbf{v}_W(t)$ to produce a canonical volumetric representation of the scene. The reconstructed scene is firstly truncated within a cubic space of interest then segmented by means of DBSCAN algorithm for unsupervised clustering [29]. However, due to object's self-occlusion and inter-object occlusions, there is a tendency to oversegment. We further apply K-means clustering [40] on the center points to force the generation of N clusters. In such way, we produce a set of point cloud segments $\mathcal{S}(t) = \{S^1(t), \dots,\ S^N(t)\}$, where each segment corresponds to an unknown object.

5.2.2 Object detection and pose estimation

Classification with partial reconstruction. We build our classifier based on PointNet architecture, which directly takes unstructured point clouds as input [81]. The classification network firstly applies input and feature transformations in order to manage unordered point clouds, where transformation functions are trained as multi-layer perceptron (MLP) networks. Secondly, it aggregates points features by max pooling. Final global features, that are the shape features of the input point cloud, have been fed to a MLP to extract the classification scores for the number of classes $M = |\mathcal{C}|$.

At run time, we provide each point cloud segment $S^j(t)$ after zero-mean and normalisation into our fine-tuned PointNet classifier (check Chapter 6 for details) at each time step t. The classifier returns a score vector $\mathbf{z}^j(t)$ over all the classes in $\mathcal{C}$ such that $\mathbf{z}^j(t) = \{z^j_1(t), \ldots, z^j_M(t)\}$, $\Sigma^M_1 z^j_m = 1$. With a max pooling strategy, each segment $S^j(t)$ has a prior class prediction $C^{j-}(t)$ given by:

$$C^{j-}(t) = \underset{m \in \mathcal{C}}{\operatorname{argmax}}\, \mathbf{z}^j(t) \tag{5.4}$$

In addition to the most likely class prediction, we also cache the top most likely classes, $\mathcal{C}^{j-}(t)$ for further refinement. We use the superscript "$-$" to indicate all the classes/poses estimation prior to the refinement stage.

In order to obtain the initial pose guess at $t = 0$ for each segment, we make use of a CNN-based pose estimator, DenseFusion [108] that takes as inputs the cropped depth images corresponding to an object class. More in details, for each segment $S^j(t)$, we input to DenseFusion the class prediction $C^{j-}(t)$ and the segmented observation $\mathcal{O}^j(t)$ obtained via camera projection using the precalibrated camera intrinsics and the viewpoint pose $\mathbf{v}_W(t)$, and obtain the prior pose estimate $\mathbf{p}^{j-}_C(t)$ for each segment in the camera coordinate. By applying coordinate transformation, $\mathbf{p}^{j-}_W(t) = \mathbf{p}^{j-}_C(t)\mathbf{v}_W(t)$, we obtain the prior pose estimate for each segment in the canonical 3D representation (*ie.* the world coordinates).

Note that at the initial steps of the algorithm, the pose obtained can be inaccurate, in particular for the translation, due to wrong classification predictions. We further refine $\mathbf{p}^{j-}_W(t)$ by anchoring the estimated translation vector to the centroid of each segment. Although the centroid of each segment does not necessarily coincide with the centroid of its corresponding object's canonical volume, such rough translation fixation can facilitate a reasonable pose initialisation for the geometric refinement as described in the next section.

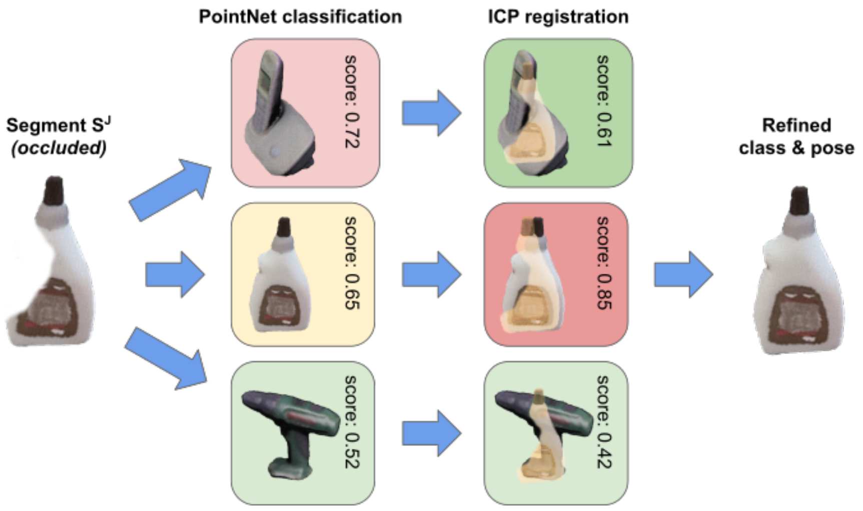


Figure 5.4: An example for the geometric class and pose refinement procedure. A reconstructed segment S^j is classified with top three candidate classes (highest classification confidence on top). By aligning the segment with the CAD models of the three candidate classes using ICP, the class and object pose can be refined by selecting the best aligned model. Best in colors.

Geometric class and pose refinement. Since the 3D classifier can provide unreliable class prediction, especially at early stages, we propose a new refinement strategy using a geometrical approach as sketched in Figure 5.4.

First, each point cloud segment $S^j(t)$ will be aligned with each top-ranking classified object using the associated 3D CAD model. Let $O^m \in \mathcal{C}^{j-}(t)$ be the 3D model of a candidate object in the point cloud. The estimated pose at the previous step $\mathbf{p}_W^{j-}(t)$ is used as an initialisation for the Iterative Closest Point (ICP) method [5]. After registration, we have a set of registered points, $S_m^{j,R}(t)$ and $O_m^{j,R}(t)$, belonging to the segment point cloud $S^j(t)$ and the object model O^m respectively. Together with the registered points, ICP also provides the pose transform $\Delta\mathbf{p}_m^j(t)$ that is applied to align the object model to the point cloud segment.

We quantify how well two point clouds are registered, given possibly misclassified classes, by the ratio of the registered points over the total number of points of the two point clouds. Let $\rho^j_m(t)$ be the registration ratio of the segment and a candidate object model, and we can compute $\rho^j_m(t)$ as:

$$\rho^j_m(t) = \frac{|S^{j,R}_m(t)|}{|S^j(t)|}\frac{|O^{j,R}_m(t)|}{|O^m|}, \tag{5.5}$$

where the operation $|\cdot|$ gives the number of points of a point cloud. The value $\rho^j_m(t)$ is larger when the 3D reconstruction of the object is complete and when the candidate class prediction is the correct one. As a result, we can obtain the refined object class at time step t as:

$$C^j(t) = \underset{O^m \in \mathcal{C}^{j-}(t)}{\operatorname{argmax}} \ \rho^j_m(t). \tag{5.6}$$

The refined object pose at time step t will be updated as:

$$\mathbf{p}^j_W(t) = \Delta\mathbf{p}^j_m(t)\mathbf{p}^{j-}_C(t). \tag{5.7}$$

5.2.3 Next Best View selection

The Next Best View (NBV) is selected among a set of candidate viewpoints $\mathcal{V}_W(t) = \{\mathbf{v}^1_W(t), \ldots, \ \mathbf{v}^I_W(t)\}$ at time step t. The candidate viewpoints $\mathcal{V}_W(t)$ are defined as those viewpoints that lie within the circular range of the current viewpoint $\mathbf{v}_W(t)$ with radius r. The movement aims to achieve more complete object reconstruction, while avoiding to revisit the areas that the camera has already visited.

We devise a utility $U_i(t)$ for each candidate viewpoint $\mathbf{v}^i_W(t) \in \mathcal{V}_W(t)$ that encourages high object surface visibility and punishes revisiting same areas. Let $O^j_i(t)$ be the function that quantifies the visibility of a segment $S^j(t)$ at viewpoint $\mathbf{v}^i_W(t)$. We approximate the visibility for segments by replacing each segment with its 3D model of the predicted class aligned by the estimated pose as given in the previous section. The corners of the bounding cuboid of each aligned object model are projected onto the image plane. The projection is then performed sequentially using a z-buffer, *ie.* an ordered vector of the distance between the segments to the camera, in order to account for occlusions. In this way, $O^j_i(t)$ is defined as the number of visible pixels within the convex hull bounded by the projected corners for each segment.

The visibility utility at viewpoint $\mathbf{v}_W^i(t)$ that accounts for all segments is defined as the sum of visibility of each segment:

$$O_i(t) = \sum_{S^j(t) \in \mathcal{S}(t)} O_i^j(t). \tag{5.8}$$

In order to penalise the system to visit already seen areas, we further introduce a weight $W_i(t) \in [0, 1)$ for each candidate viewpoint. The value of $W_i(t)$ is designed to be smaller if the candidate viewpoint is closer to the viewpoint that has been visited, otherwise the value will be larger. Let the set of previously visited viewpoints be $\mathcal{V}_W^{'}(t) = \{\mathbf{v}_W(\tau) | \forall \tau \in [1, t)\}$, and let $D(\tau, t)$ be the distance between the candidate viewpoint and a visited viewpoint at a previous time step τ. We make use of the normalised Gaussian distribution over the distance $D(\tau, t)$ in order to compute $W_i(t)$, such that:

$$W_i(t) = \max_{\mathbf{v}(\tau) \in \mathcal{V}'(t)} 1 - G\left(D(\tau, t), \mu, \sigma\right) \tag{5.9}$$

where $G(\cdot)$ defines the normalised Gaussian distribution function with the mean $\mu = 0$ and the standard deviation $\sigma = \frac{r}{2}$, where r is the radius that defines the set of candidate view points.

The final utility $U_i(t)$ combines the visibility utility $O_i(t)$ and the weight $W_i(t)$. Because the value of $W_i(t)$ is constrained within [0, 1) while $O_i(t)$ is not constrained, for fair combination, we normalise $O_i(t)$ by its maximum value within $\mathcal{O}(t) = \{O_i(t) | \forall \mathbf{v}_i(t) \in \mathcal{V}(t)\}$, *ie.* $N(O_i(t)) = \frac{O_i(t)}{\max(\mathcal{O}(t))}$. The utility $U_i(t)$ is therefore computed as:

$$U_i(t) = W_i(t) N\left(O_i(t)\right). \tag{5.10}$$

The NBV $\mathbf{v}_W(t)$ is selected as the one providing the highest utility:

$$\mathbf{v}_W^*(t) = \operatorname*{argmax}_{\mathbf{v}_W^i(t) \in \mathcal{V}_W(t)} U_i(t). \tag{5.11}$$

Once the NBV is selected, we can move the depth sensor in the desired position if no stop condition is satisfied and acquire a new measurement $\mathcal{O}(t+1) = \{\mathcal{D}(t+1),\ \mathbf{v}_W(t+1)\}$ and iterate the procedure until the method converges.

5.3 Self-awareness for active object recognition

The application of saliency to the 2D domain and the preliminary active recognition framework gave us promising results. For these reasons, we have moved to 3D, aiming to introduce the concept of saliency in an active recognition system. In particular, our contribution is a novel representation of 3D data, such as point clouds and meshes, that we call **saliency volumes**. A saliency volume is a 3D mesh which vertex values depends on the importance of their position. In other words, the higher a vertex values is, the more important is the region the vertex belongs, as shown in Figure 1.4.

Here, we present a novel active recognition system to explain the building procedure of a saliency volume and its application in a real world scenario. As a first step through the usage of visualization methods into 3D, we face the AOR problem considering only one object which pose is known. A challenging scenario with multiple objects and occlusions is presented in Section 5.4. The following section is organized as follow: we define the problem of active recognition in Section 5.3.1 while Sections 5.3.2 and 5.3.3 describe the active pipeline in which the volumes are injected.

5.3.1 Problem formulation

We consider a multi-class classification scenario, where the class labels belong to the finite set $\mathcal{C}$. A single instance of a given class is a 3D object which is located with a standard pose in the centroid of a spherical workspace V_ρ of radius ρ where a robotic arm can move in the upper hemisphere (since the object lies on a solid floor). The centroid of the object coincides with the centroid of V_ρ. A depth sensor is mounted on the end-effector of the robotic arm[2]. The robot can move and acquire a depth image at each time step, until it stops and provides the object class. The decision whether to move or to stay, and in case where to go, is taken by minimizing an energy function that combines the cost for moving the robot and acquire a new view, E_M, and the cost for an incorrect

[2]We use a depth sensor to simplify the creation of the 3D dense saliency volume. In any case, RGB-D data can be also considered, and will be the subject of future work.

classification, E_c:

$$E = E_M(x, x') + \lambda\, E_C(c, \hat{c}) \tag{5.12}$$

where c and $\hat{c}$ are the predicted and correct classes respectively, x and x' are two generic locations in the 3D space, and λ is a constant value. The process iterates over time and it stops when the cost for moving the robot becomes higher than the cost associated to the classification error. In this work, we define the classification cost as a constant value that uniformly penalizes incorrect classification:

$$E_C(c, \hat{c}) = \begin{cases} 0 & c = \hat{c} \\ 1 & \text{otherwise} \end{cases} \tag{5.13}$$

while the movement cost takes into account two different terms: one related to geometrical properties, *ie.* the length of the trajectory from x to x', and the second related to the kinematics of the robot itself:

$$E_M(x, x') = len(x, x') + man(q') \tag{5.14}$$

Here q' is the configuration of the robot's joints when the sensor is positioned at x', and $man(q')$ represents the *manipulability measure*, *ie.* an estimate of the capacity of change in position and orientation of the robot end-effector given a joint configuration. Formally this measure is defined as:

$$man(q) = \sqrt{\det\left(J(q)\ J^T(q)\right)} \geq 0 \tag{5.15}$$

where $J(\cdot)$ is the Jacobian operator, and $man(q) = 0$ coincides with a singular configuration. No other active object recognition approach considers the manipulability measure. To minimize Eq. 5.12, we consider the POMDP framework, explained in the following section.

5.3.2 Our model RA-POMDP

We restrict the moving sensor to stop and look towards the centroid of the sphere V_ρ at a finite set of viewpoints $\mathbf{x}_i \in \mathcal{X}_\rho \subset V_\rho$: this allows us to ensure that the sensor is always pointing to the object of interest. From now on, the robot iterates over the following steps:

1. decide if moving to a new position is convenient (otherwise the class label has to be provided and the process stops)
2. retrieve the best move from the POMDP policy
3. move and acquire a new image
4. pdate the belief state of the POMDP

The process starts with the robot in a random but known position.

Partially Observable Markov Decision Process. A Markov Decision Process (MDP) is a finite state process that relies on the Markov property: the transition between any pair of states depends only on the last state and the action that leads to the new state, and not on any earlier state. POMDP is a generalization of a MDP where the agent cannot directly access the whole state of the system, but it has to maintain a probability distribution over the set of all the possible states based on a set of observations and observation probabilities.

A POMDP is a 6-tuple (S, A, T, R, Ω, O), where S is a finite set of states, A is a finite set of actions, $T : S \times A \to S$ is the transition function defining the probability of state change upon application of a given action, $R : S \times A \to \mathbb{R}$ is the reward function that represents the reward granted to the system after having reached the new state with the given action, Ω is a finite set of observations, and O is the probability distribution of the observations according to the states and the actions.

At each time step, given a current state $s \in S$, the agent receives an observation $o \in \Omega$ with probability $O(s, o) = Pr(o \mid s)$. Depending on this observation and the current state, the agent takes an action $a \in A$, which causes a transition to state s' with probability $T(s, a, s') = Pr(s' \mid s, a)$. Finally, the agent receives a reward r equal to $R(s, a)$. Then the process repeats.

In our RA-POMDP formulation, the state s at time t is a pair $\langle \mathbf{x}_t, c \rangle$, where $\mathbf{x}_t \in \mathcal{X}_\rho$ is the viewpoint (here assumed to be measurable), and $c \in \mathcal{C}$ is the (hidden) class of the object in the scene. In other words, we assume the robot as knowing its position at each time step. Relaxing this constraint means that the robot, after each movement, has to check its position w.r.t. a reference system. In this work we ignore this aspect which could be considered as future work. This assumption is reasonable since the motion is deterministic (we have the set $\mathcal{X}_\rho$ of finite viewpoints), thus the set of actions is to move between viewpoints, and the transition function only affects the viewpoint part of the state (the object class label does not change) and has the form:

$$T(s,a,s') = \begin{cases} 1 & a \text{ is “move from } s \text{ to } s'\text{”} \\ 0 & \text{otherwise} \end{cases} \tag{5.16}$$

The observation o corresponds the output $\mathbf{z}_t$ of the static classifier, while the observation model $O(s,o)$ is generated at training time as described in the next section.

Solving a MDP means to find an optimal policy mapping from a state to an action that maximizes the expected total reward. However, since in a POMDP the state is partially observable, the concept of *belief* has to be taken into account. A belief is a probability distribution over all the states $s \in S$. A POMDP policy π maps a belief b to a prescribed action a. A policy π induces a value function $V_\pi(b)$ that specifies the expected total reward of executing the policy π starting from b. The goal for the robot is to choose the optimal policy π^*, *ie.* the policy that maximizes the associated value function: $V^* = \mathbb{E}\left[\sum_t r_t\right]$. This problem is usually computationally intractable, but approximate solutions have been proposed in the literature.

In this work we use SARSOP approach [57] that finds the best policy iteratively by sampling points in the belief space and pruning away the non optimal candidates. Starting from an initial distribution b_0, at every iteration the belief is updated using the formula:

$$b'(s') = \alpha\, O(s',o) \sum_{s\in\mathcal{S}} T(s,a,s')\, b(s) \tag{5.17}$$

where α is a normalization constant and all the new beliefs are guaranteed to be reachable from b_0. In this setup, we introduce a novel observation model $O(s',o)$.

5.3.3 Observation model: 3D dense saliency volume

Our RA-POMDP supposes that a 3D object classifier is trained before to operate on the robot. At training time, we generate a set of synthetic depth images $\mathcal{D}$ by projecting artificial 3D models on a simulated depth camera located in a set of viewpoints uniformly distributed on the surface of a sphere centered at the centroid of each model (Fig. 5.5a). Note that, differently from [2], with our approach these training viewpoints are conceptually unrelated to the viewpoints $\mathcal{X}_\rho$ used at testing time, as well as to the radius of the viewsphere V_ρ.

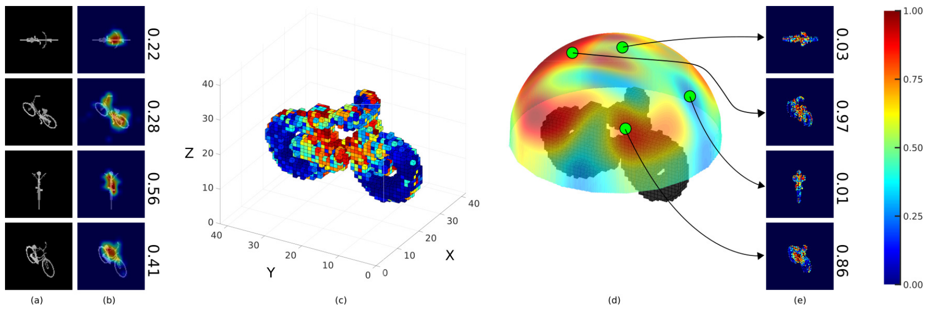


Figure 5.5: Overview of the RA-POMDP observation model construction: (a) depth maps; (b) saliency maps of the deep classifier; (c) 3D volume of the object; (d) observation model with (e) saliency maps on generic views, whose pixel summation is then mapped on the hemisphere.

The output of the classifier is a distribution $\mathbf{z}$ that returns the probability of the object to belong to each class in $\mathcal{C}$: $\mathbf{z} = [z_1 \dots z_{|\mathcal{C}|}]$, $\sum_{c=1}^{|\mathcal{C}|} z_c = 1$. Our goal is to understand how the classifier uses the input to classify, *ie.* which depth image regions have been considered more important to decide a particular class. First, for each viewpoint specific depth image $\mathcal{D}_\mathbf{x}$ we compute a 2D visualization map ($\mathcal{S}^2_\mathbf{x}$) as in [32]. We do so by learning a mask whose perturbation (usually by blurring its corresponding pixels) drifts the classifier away from deciding the correct class, causing a drop in the related entry of $\mathbf{z}$ (Fig. 5.5b). The mask is dense and each of its pixel intensities is proportional to the classification drop occurred when masking it. The 2D mask is then mapped to the 3D volume by means of the associated depth map. The 3D model is now discretized into a finite set of voxels, and to each voxel we associate a 3D saliency score $\mathcal{S}^3(v)$

computed as the median value of the saliency of all the points lying inside it (Figure 5.5c). Alternatively, if the dense CAD model of the target object is available, the voxel values could be mapped onto the 3D mesh obtaining a continuous saliency volume. In our research, we applied both the building approaches without noticing any significant difference but the smoothness of the representation.

We are now ready to define the observation model O for the POMDP planner. We assign to each viewpoint a cumulative score (Figure 5.5d) by averaging the 3D saliency value of all the voxels in the field of view of the camera (Fig. 5.5e). This viewpoint specific score is then normalized in order to guarantee that for each class the observation model is in the range $[0, 1]$. Mathematically, the observation model returns an estimate of the classifier output $\mathbf{z}$ as a function of the measured part of the state ($s = \mathbf{x}$), and, given the transition function of Eq. 5.16, we can express it as a function of the next state:

$$O(s', o) = Pr(\mathbf{z} \mid s') = \alpha \sum_{i=1}^{N_v} \mathcal{S}^3(v_i)\, \eta(v_i, \mathbf{x}') \tag{5.18}$$

where α is a normalization factor, N_v is the total number of voxels, and $\eta(v_i, \mathbf{x}')$ is a visibility function that returns 1 if the i-th voxel v_i is visible from $\mathbf{x}'$, and 0 otherwise.

5.4 Global Saliency Representation

Here, we combine what we described in Sections 5.2 and 5.3: we inject the saliency volumes built in Section 5.3.3 in out active recognition pipeline and the POMDP formulation has been wiped out in favour of a simpler framework. The research we present in this part is a preliminary evaluation of the impact our saliency representation (*ie.* the saliency volumes) in a real scenario, where occlusions and multiple objects might be present generating a realistic and challenging scenario.

We consider a table-top scenario with an unknown number of objects, where a robot with an RGB-D sensor (*ie.* a Kinect) mounted on its end-effector is required to recognise all the object instances. To do that, the robot can move and acquire data from different viewpoints. Our active object recognition framework consists of the following three main parts:

1. single-shot object detection and pose estimation;
2. global observation model;
3. next-best-view selection.

Let us consider a scenario in which an unknown number of objects belonging to the set of classes $\mathcal{C}$ are distributed in an area and a robotic arm equipped with an depth sensor in an eye-in-hand configuration, *ie.* the sensor is mounted on the end-effector of the robot. At each time step we know the sensor's pose, since it is provided by the direct kinematics of the manipulator.

At training time we learn a CNN model working on depth images to predict the semantic class an object belongs to. For each class we consider a prototype model on which we learn a 3D saliency volume representation that embeds how much a specific region of that model is important to guarantee a correct class prediction. Regions of the model with high saliency are parts of an object that are more discriminative for the given classifier.

We start acquiring a depth map of the scene from the initial viewpoint, which is fed to a CNN-based object detection module that takes care of segmenting the scene to generate object proposals, predict the semantic class of each segment, and estimate the 6D pose of each object.

These information are used to generate the Global Saliency Model (GSM): a synthetic model of the observed scene where the detected objects are substituted by the saliency volumes of the respective object classes. The next best view is the neighbor view which maximizes the projection score of the GSM. The robot moves to the selected viewpoint and the process iterates until the detector outputs a confidence level averaged over all the objects in the scene higher than a fixed threshold. A graphical overview of the framework is shown in Figure 5.6.

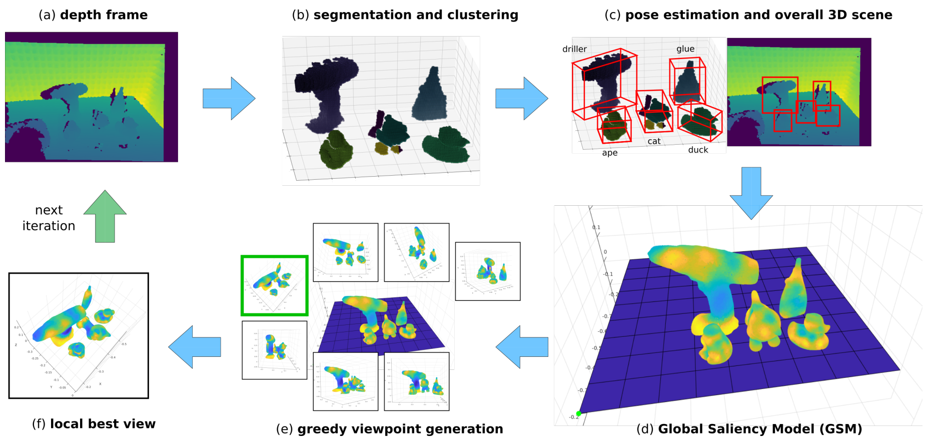

(a) depth frame
(b) segmentation and clustering
(c) pose estimation and overall 3D scene
driller
glue
ape
cat
duck
next iteration
(f) local best view
(e) greedy viewpoint generation
(d) Global Saliency Model (GSM)

Figure 5.6: Active recognition method overview. (a, b) The input depth frame is acquired and segmented in order to build and cluster the 3D point cloud of the scene. Each cluster represents an object. (c) The 6D poses of the detected objects are estimated through a pose detector and (d) the saliency volumes of the classes are aligned to generate the Global Saliency Model (GSM). A saliency volume highlights the regions of an object the classifier uses to guess the class of an object. In the end, (e) a greedy strategy is used to select the next best view, that is the neighbor view which maximizes the projection score of the volumes.

5.4.1 Object detection and pose estimation

Given the depth map $\mathcal{D}_{\mathbf{v}}$ acquired by the sensor from the generic viewpoint $\mathbf{v}$, we segment the scene by means of DBSCAN [29] algorithm for unsupervised clustering. Each cluster obtained this way is potentially an object; thus, for each cluster j, we extract from the depth map a rectangular ROI $\mathcal{D}_{\mathbf{v}}^j$ containing all the pixels of the j-th cluster and we pass the segmented depth map to our depth classifier. Our classifier consists of the convolutional layers of an AlexNet [56] architecture, followed by one fully connected layer with a number of nodes equal to the number of classes $C = |\mathcal{C}|$.

At training time, we generate a set of synthetic depth images by projecting artificial 3D models on a simulated depth camera located in an arbitrary set of viewpoints distributed all around the model. To allow the network to generalize over real world applications, we apply a data augmentation strategy consisting of three parts: (i) we overimpose the rendered 3D models on a set of background maps randomly chosen from the LINEMOD dataset [43]; (ii) we add gaussian noise on each pixel; and (iii) we flip horizontally all the depth maps created so far. At testing time, the 3D classifier returns a probability distribution $\mathbf{z}$ over all the classes in $\mathcal{C}$ such that $\mathbf{z} = \{z_1, \ldots, z_C\}$, $\Sigma_1^C z_c = 1$. We assume a candidate ROI belongs to the class with the highest score if this is at least twice the chance level, *ie.* if $max(\mathbf{z}) > \frac{2}{C}$.

The 6D pose of all the detected objects is estimated from RGB images by means of Single Shot Pose (SSP) [101] algorithm. SSP estimates the object's pose by regressing the image location of the corners of its 3D bounding box. With the 2D-3D corner correspondences, the pose can be recovered through a standard PnP method. In order to restrict SSP's attention to the object classified in the previous step, we establish a loose 2D bounding box around it and we apply Gaussian blur outside of this region. We also specify which class of object we are interested in and we repeat this procedure for every object previously detected in the scene.

5.4.2 Global Saliency Model

We build on the concept of saliency volumes proposed in Section 5.3.3 and published in [85] to compute an observation model at scene level. Indeed, we consider the scene as a single object, given by the composition of all the objects detected in the previous step. We use the output of the object detector and pose estimator to generate a synthetic scene where each object is represented by the class prototype saliency volume.

We observe that some regions are more important than others for a classifier to predict which class an object belongs to. Starting from this assumption, we compute an object level saliency volume that encodes in a class prototype model the discriminativeness of each point of the model related to a specific classifier. For each class, we generate a set of depth images by projecting the class prototype model on a simulated camera located in a set of viewpoints. For each depth image we compute a visualization map by learning a perturbation mask that is responsible for a decay in the classification score when trying to predict the correct class [32]. The mask is dense and each of its pixel intensities is proportional to the classification drop occurred when masking that particular pixel. The viewpoint specific 2D masks are then back-projected on the class prototype model to generate the class-specific saliency volume (see Figure 5.7).

Finally, we build a synthetic scene in which all the detected objects are substituted by the saliency volumes, positioned in the workspace according to the position and orientation returned by the pose estimation module. To account for the uncertainty of the classifier to predict the object class, the saliency volume of each object j is weighted by the classification score of that specifc class:

$$\text{GSM} = \cup_{j=1}^{N_{obj}} P\left(\hat{c}_j \mid \mathcal{D}_{\mathbf{v}}^{j}\right) \mathcal{S}_{\hat{c}_j} \tag{5.19}$$

where $\hat{c}_j$ is the class predicted by the classifier, $P\left(\hat{c}_j \mid \mathcal{D}_{\mathbf{v}}^{j}\right)$ is the classification score associated to class $\hat{c}_j$, and $\mathcal{S}_{\hat{c}_j}$ is the saliency volume of class $\hat{c}_j$ aligned according to the output of the pose estimator.

5.4.3 Next Best View selection

Our approach uses a greedy strategy to select the next viewpoint to visit within a set of neighbor points. All the viewpoints are samples of the 3D manipulability space of the robot and two points are neighbors if the robot can move from one to another is one time step, *ie.* if their distance is lower than a threshold.

We select the next-best-view that maximizes the target loss function defined as the average saliency score obtained by projecting the saliency volumes of the GSM to a camera plane located in $\mathbf{v}$. Formally this function is:

$$\mathcal{L}(\mathbf{v}) = \sum_{p} \text{GSM}(p)\ \eta(p, \mathbf{v}), \tag{5.20}$$

with α a normalization factor, p a generic point of the scene, and $\eta(p, \mathbf{v})$ a visibility function that returns 1 if the point p_j is visible from $\mathbf{v}$, and 0 otherwise.

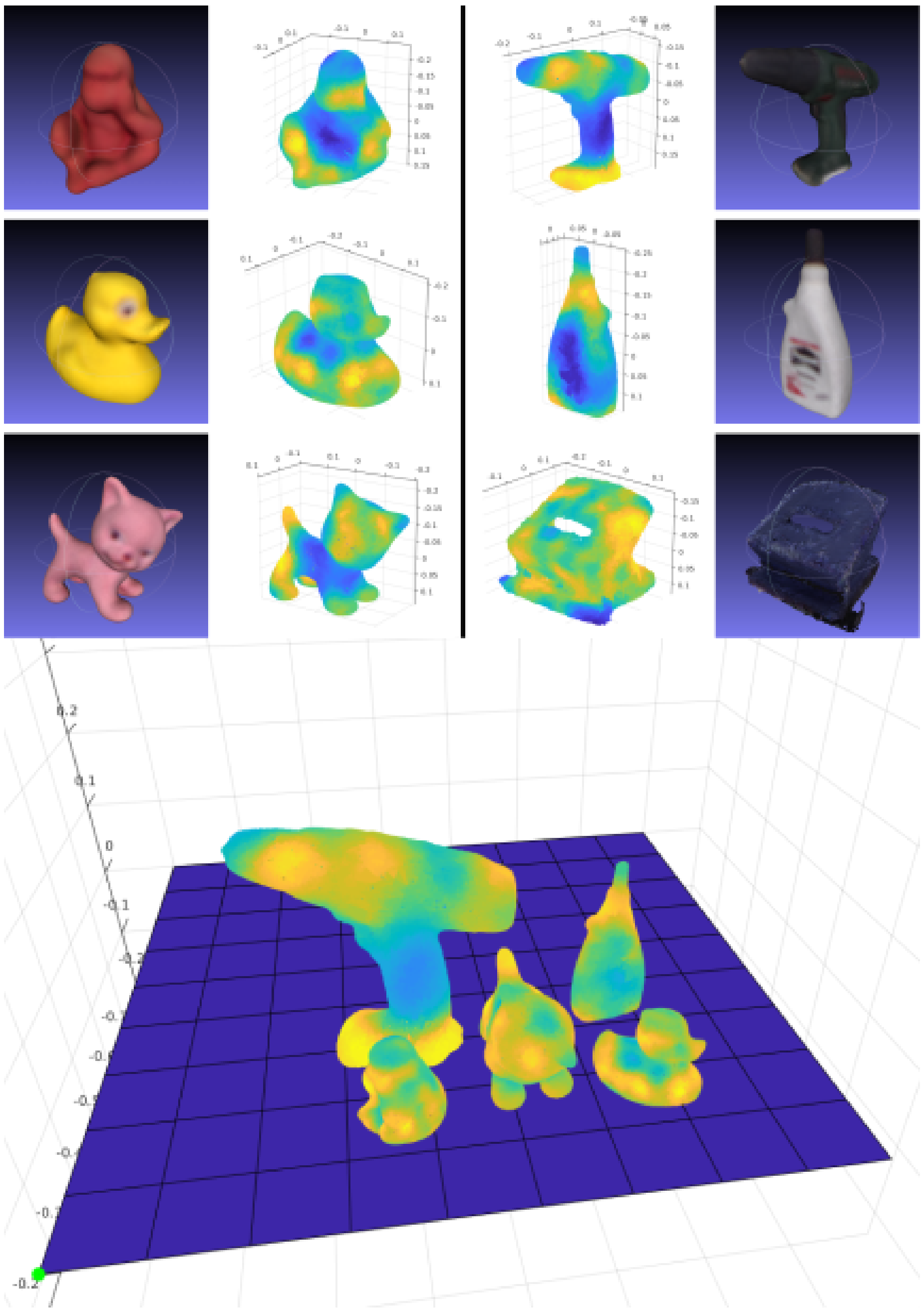


Figure 5.7: Saliency volumes of the LINEMOD subset used in the experiments. A saliency volume [86] highlights the relevant regions used by a specific classifier to discriminate the classes. The lower part of the figure shows an example of aligned saliency volumes used to compute the Global Saliency Model.

CHAPTER 6

Experiments: Classification

In order to prove our claims, we split the experimental chapter into three main sections. First section is related to 2D domain and it is related to Section 5.1. Second and third parts refer to the application of saliency analysis to 3D data, with particular attention to the active recognition problem.

6.1 Visual Summaries

For our experiments, we focus on 18 classes of Imagenet. These classes are selected considering the constraint of being adjacent in a *dense* [22] semantic space. In Table 6.1.1, adjacent classes are in subsequent rows with same background color. This constraint, brings together those classes that are adjacent to each other which provides the possibility of comparing *similar* classes along different experiments.

The set of experiments to validate our proposal is organized as follows: Sec. 6.1.1 is dedicated to show the superiority of our proposed crisp mask w.r.t. the original smooth mask [32] in terms of conciseness and expressiveness, providing higher classification drop. Sec. 6.1.2 is focused on the semantics of the summaries, showing that automatic taggers as well as humans, individuate a precise type of parts for each summary. Sec. 6.1.3 shows that the number of summaries is proportional to the classification ability of a deep architecture: the higher the number of classes the higher the classification accuracy. In Sec. 6.1.4 it is showed that summaries can be used to specialize the classifier on the visual summaries and improve the classification results.

6.1.1 Masks analysis

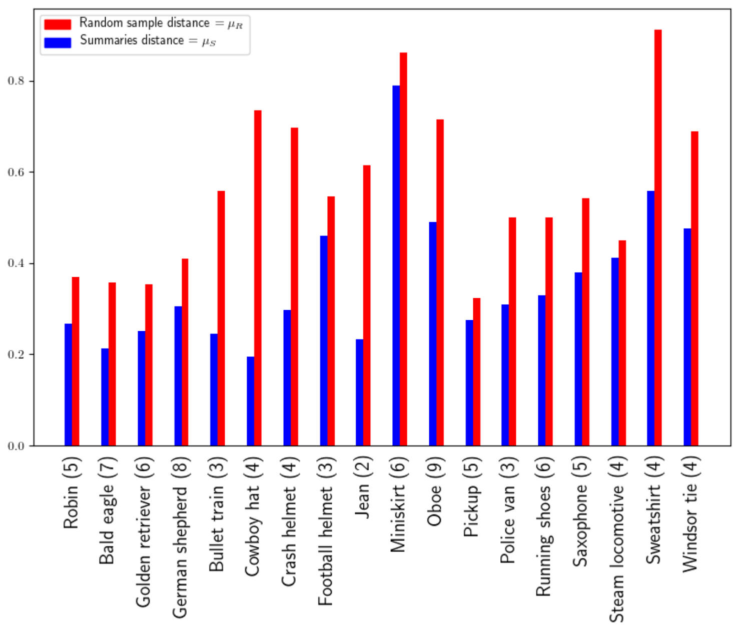


Figure 6.1: Coherency in terms of average Jaccard distance (y-axis) among the tags found with the automatic tagger, within the summaries (blue $= \mu_S$), and within a random sample of the class (red $= \mu_R$). Lower is better. The class labels come with the number of summaries found.

In this experiment the masks obtained by our approach are compared with those of the smooth mask a.k.a. IEBB [32] method employing the protocol as proposed by the authors. Given an image, the classification confidence associated to it w.r.t. the ground truth class is measured. In the case of a deep network, the classification confidence for the i-th object class is the softmax output in the i-th entry. Afterwards, the image x is blurred as explained in Section 5.1.1 by using the corresponding mask m (either the one produced by our pro-

Class Name	μ_U	Most Proposed Tag per Summary
Robin	0.12	Head, Body, Legs, Wings, Tail
Bald eagle	0.23	Head, Neck border, Eye, Beak, Face, Wing
Golden retriever	0.31	Nose, Eye, Ear, Mouth, Face, Legs, Head
German shepherd	0.22	Eye, Leg, Neck, Body, Ear, Nose, Face, Feather
Bullet train	0.38	Front train, Front glass, Train, Rails, Lights, Train body
Steam locomotive	0.56	Chimney, Front train, Wheels, Engine, Side, Window
Pick-up	0.19	Mudguard, Step bumpers, Side window, Back, Windshield, Wheel
Police van	0.17	Wheel, Police flag, Side window, Rear window, Light, Vehicle, Capote, Bumpers, Mudguard
Oboe	0.01	Body, Buttons
Saxophone	0.68	Body, Buttons, Bell
Crash helmet	0.36	Base, Side, Front, Logo
Football helmet	0.48	Front grids, Logo, Side, People
Jeans	0.01	Crotch, Pocket, Legs, Waistband
Miniskirt	0.12	Face, Waistband, Leg, Head
Cowboy hat	0.32	Ear, Face, Chin
Windsor tie	0.13	Pattern, Knot, Collar, Neck
Sweatshirt	0.31	Hoodie, Face, Arm, Laces, Wrinkles, Neck
Running shoes	0.38	Laces, Logo, Shoe side

Table 6.1: Classes from ImageNet, coherency of the summaries in terms of average Jaccard distance (μ_U) among the tags found with the user study and the set of tags collected during the user study with our approach.

posed approach or the one produced by the IEBB approach). The classification score is then re-computed after perturbation and the difference w.r.t. the score for the original image is computed. The average classification drop of a method is computed as the average score drop over the entire test set. We compare our proposal solely with IEBB, which is shown to be the state-of-the-art [32]. In addition, we compare with IEBB *thresh*, in which the smooth mask generated by IEBB is made crisp by a thresholding operation over the mask intensities. On each image the threshold is independently set to make the mask as big as the one produced by our proposed technique to ensure a fair comparison. The third column of Table 6.2 shows the classification loss of the two approaches. Notably, we succeed in improving the results, closely reaching the saturation. Interestingly, with IEBB *thresh*, the overall performance diminishes, with higher variance.

In Fig. 5.1, examples of the obtained masks using our approach and IEBB are shown. From our observations, the sparse optimization producing mask which are similar to the IEBB one. In fact, IEBB finds masks which cause a nearly complete loss. Nonetheless, our improvement gives the same importance to all of the pixels which leads to a higher classification drop, while facilitating the clustering step and consequently the final human interpretation of the summaries.

6.1.2 Analysis of the summaries

In this section of the experiments, we make use of an automated tagger [49] to show whether each summary individuates a visual semantic. For each object class, the n_i images of each single summary S_i, $i = 1, ..., K$ are tagged, providing n_i lists of textual tags (only nouns are allowed). For convenience, the tagger is constrained to provide only 8 tag for each image. This procedure is repeated on K sets R_i, $i = 1, ..., K$ of c_i random images taken from that class.

After tagging, the set of all the given tags is used to extract a one-hot vector for each image. The entry of the vector is 1 if a particular tag is given, and 0 otherwise. Synonyms tags were fused together by checking synsets of WordNet. This results to a vector of an average length of 28 entries. At this point, the n_i tag vectors of the summary S_i are pairwise compared with the

Method	%Drop (Var)
IEBB [32]	99.738365 (8.13e-4)
IEBB *thresh.* [32]	97.703865 (5.758e-3)
Ours	**99.964912**($<$ 10e-6)

Table 6.2: Mask analysis results.

Model	Summaries	Accuracy
AlexNet	5	57.1%
VGG16	5.5	72.4%
GoogleNet	6	74.5%
Resnet50	6.33	76.2%

Table 6.3: Average number of summaries for each different architecture and top-1 accuracy.

Jaccard distance, and the average intra-summary distance is computed. This is computed for each summary, and the K average intra-summary distances are further averaged, obtaining the summary distance μ_S. This process is repeated for each class. In the same way, we compute the average distance obtained with the random image subsets R_i, getting a μ_R for each class. Results are shown in Fig. 6.1. As it can be seen, on average images belonging to the same summary are closer in semantic content (i.e. lower Jaccard distance) than random images of the same class.

Since the automated tagger could only work on the entire image, we expect to have much finer grained results by focusing on the parts highlighted by the summaries. To this end we organize a user study, with the goal of giving a precise name to each of the summary, by considering the parts highlighted within. We hire a total of 50 people (35 male, 15 female subjects) with an the average age of 33 (std:8.4). Each of the users was asked to give a set of (noun) tags to each summary, by considering the entire set of regions and parts contained within. Next we check the inter/rater reliability among users toward the same summary by computing the average pairwise Jaccard distance among the obtained sets of tag. The distances over the different summaries are averaged, thus obtaining for each class μ_U which is a measure of the agreement between users expressed as the average . To name each summary, we select the tag more used among the users. Table 6.1.1 reports on the right these tags (one for each summary), together with the μ_U value. Interesting observations can be assessed: in some cases, the μ_U values are very small, but at the same time many tags are definitely more specific than those provided by the automatic tagger, indicating that the summaries individuate finer grained visual semantics that users have captured. Then, adjacent classes exhibit many common visual summaries (*german shepherd, golden retriever*).

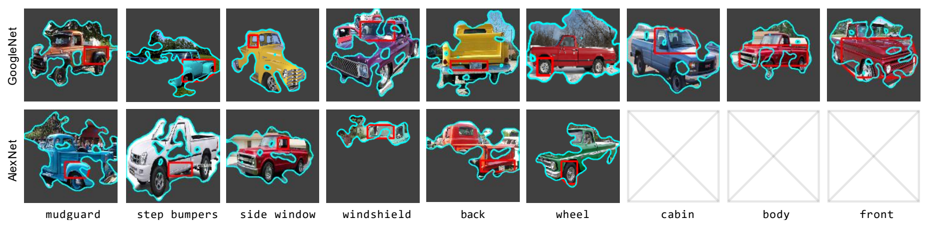


Figure 6.2: Motivating the superiority of GoogleNet against AlexNet. focusing on the *pick-up* class, our approach finds 9 summaries for the former architecture, 6 for the latter, showing that GoogleNet is capable of capturing more semantics. Best seen in color.

6.1.3 Number of summaries and classification accuracy

Another interesting question to be answered is whether the number of summaries has a role in the general classification skill of a network. To this end, we analyze four famous architectures as, AlexNet [56], VGG [95], GoogleNet [99], and ResNet [41]. For each of these architectures, the average number of summaries over the 18 chosen classes for the analysis is computed. This value is later compared with the average classification ability of each architecture in terms of accuracy over ImageNet validation dataset. The comparison results are shown in Table 6.3. Notably, from AlexNet to ResNet, as the classification accuracy rate increases, the number of summaries also rises. From this observation, we can conclude that the network classification ability is related to the the number of discriminant patterns that the network is able to recognize. This has been shown qualitatively in Fig. 6.2. We obtained similar observations with other classes and other architectures.

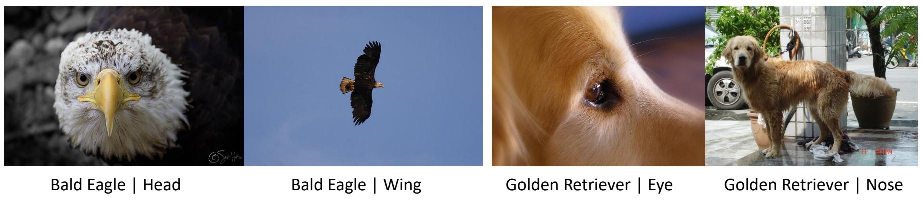


Figure 6.3: Examples of images two classes that were misclassified by the AlexNet but correctly classified by specializing the classification using SVMs trained on the summaries. The labels below are the class names and the tags associated with the summary that contributed the most to correcting the classification of each image.

6.1.4 Specializing classification with summaries

The proposed idea in this section is to improve the classification results using the images belonging to the summaries. Due to the low number of images per summary (average of 32.25), we propose to employ a linear SVM per summary instead of explicitly fine-tuning the network itself. Positive examples to train each SVM are the images belonging to that summary, and negative examples are images from other classes or from other summaries within the same class. The features used for classification are extracted from the first fully connected layer of the network. Given an image to classify, it is evaluated by all of the previously trained SVMs. The class scores vector is then obtained by selecting the highest score among the SVMs for each class. The obtained scores are used to improve the classification accuracy for a desired class by means of a convex weighted sum between the neural network classification softmax vectors and the resulting SVM class scores (normalized to sum to unity). Our experiments show that employing this approach, primarily designed to improve the classification of all the 18 classes chosen for the experiments on the AlexNet architecture, the overall classification accuracy score over all the 1000 ImageNet classes increases by 1.08% on the ImageNet validation set. Some examples of images that are classified correctly thanks to this boosting technique can be seen in Fig. 6.3.

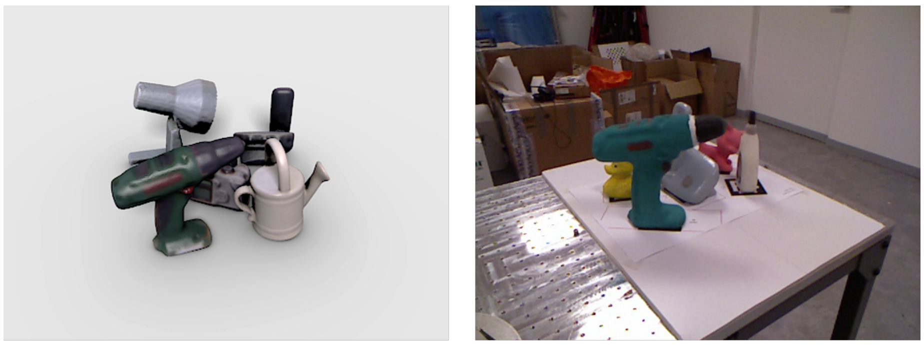


Figure 6.4: Examples of the synthetic and real scenes of our dataset, where severe inter-object occlusion can be observed.

6.2 Active 3D Object Classification

We first describe how we create a dataset with ground truth information followed by the experimental protocol. Results over several trials with both synthetic and real data will prove the effectiveness of the proposed active classification method.

6.2.1 Dataset creation

We selected 7 objects from the LINEMOD dataset [43] and generated a set of different scenarios with 5 objects in each one. In every scene, all objects lie on a planar surface defined at $z = 0$ and are placed in their *canonical pose*, *ie.* with the z axis of the model's reference frames facing up. The dataset is generated with both synthetic and real-world setup.

We generated 7 synthetic scenes where depth maps are rendered using Blender by projecting the models on 100 viewpoints uniformly distributed on a hemispherical surface (*ie.* $z \geq 0$) with the sphere's centroid coinciding with the centroid of all the objects in the scene and the radius of 1m. The rendering engine is set up to emulate a real acquisition device: the pinhole camera matrix is the same as a Kinect V1 device with resolution 640×480 pixels and focal length of 26mm. In addition, we acquired two scenes in a real-world setup with a Kinect-equipped Universal Robots UR5 using 3D-printed instances of the selected object classes. This specific robotic configuration is chosen for the experiments since it allows accurate camera poses and repeatability during the testing procedure. The arm and sensor have been hand-eye calibrated in order to obtain the correct camera viewpoint poses from the robot's encoder poses. We used ArUco fiducial markers and its off-the-shelf libraries to annotate the pose of each object and to perform hand-eye calibration of the system [87, 35]. Each scene contains 5 objects where 3 of them are to be classified and the other 2 are used to simulate generic clutter. For each scene, we acquired RGB-D observations from a pre-defined set of 138 viewpoints sampled on a hemispherical surface of radius 0.9m centered in the centroid of the objects. The viewpoints are approximately uniform on the hemisphere due to the constrained reachability of the robotic arm.

6.2.2 PointNet fine-tuning

The original PointNet is trained on the ShapeNet and ModelNet dataset which do not include most of the object classes that are supported by the DenseFusion pose estimator which is trained on YCB_Video [117] and LINEMOD dataset [43]. In order to have compatible classifiers and pose estimators, we train the PointNet network by generating a new dataset covering the classes of the LINEMOD objects. The classes are identified based on the availability of the 3D models of object instances from existing dataset (*eg.* VANDAL dataset [14]) and web resources (*eg.* 3D Warehouse[1]). For each class, we have 3D models of eight instances in the format of dense point clouds.

At training time, each point cloud contains 2500 points uniformly sampled from the object surface. Each cloud is zero-mean and normalised into a unit sphere as in the standard PointNet training. In order to be more robust to rotation, we follow the data augmentation strategy proposed by the original PointNet work [81] by randomly rotating each point cloud by an angle in the range $[0,\ 2\pi]$ along Y axis. At training phase, PointNet converges in 250 epochs on a total of 7 classes. We used Adam optimizer with learning rate of 0.001, decreasing it by a factor of 2 every 20 epochs, as suggested in the original paper [81].

[1]https://3dwarehouse.sketchup.com

6.2.3 Evaluation analysis

We first test the PointNet model by feeding it with point clouds extracted directly from depth maps at each time step, named as **PointNetSingle**, as well as with point clouds covering all the 3D shape, named as **PointNetFull**. Please note that PointNetFull uses all the available information from the scene (*ie.* complete point clouds for each object) and thus this can be considered as the upper bound that can be reached with every classifier tested.

As for the evaluation of active classification, we compare our motion strategy, **VisOcclHist**, for improving the classification performance against three baseline motion strategies: Random, VisNoOccl and VisOccl. All baseline strategies follow the same pipeline for active classification with the only difference in NBV selection. In **Random** strategy, the system randomly selects the next viewpoint to visit within the candidates that have not been visited yet. The **VisNoOccl** strategy selects the next viewpoint using the visibility score that maximizes the area of visible object surfaces without accounting for occlusions and already observed portions of the objects. The **VisOccl** strategy selects the next viewpoint using the visibility score that maximizes the area of visible object surfaces accounting for the occlusions but not for the previously observed points.

In all the cases, the next viewpoint is selected within a restricted set of points that are at a maximum distance of 0.5m from the current position. In addition, we also validate for all the baselines the classification improvement brought by the refinement module using Geometric Refinement (see Figure. 5.4) after the PointNet classifier (named with the suffix **GR**).

As for the evaluation metrics, we use standard classification measures: accuracy, precision, recall and F_1 score. We compute all these metrics for each class and then we average over all the classes, this strategy is usually dubbed as *macro-averaged.* Please note that this leads to accuracy scores higher than precision and recall since in a one-vs-all setup true negatives are usually higher than true positives [90].

6.2.4 Results discussion

Classification results are reported in Table 6.4. For fair comparison, all active classification methods stop when the maximum number of moves is reached,

Table 6.4: Object classification results after the system has reached the stop condition ($T = 10$). Values are averaged over 10 runs for each scenario with random starting points. (Best results are in bold, upper bounds are in italic.)

	Synthetic				Real			
Approach	Accuracy	Precision	Recall	F_1 score	Accuracy	Precision	Recall	F_1 score
PointNetSingle	0.80	0.32	0.40	0.34	0.71	0.37	0.41	0.38
PointNetSingleGR	0.81	0.39	0.50	0.42	0.71	0.40	0.48	0.43
PointNetFull	*0.90*	*0.64*	*0.71*	*0.67*	*0.78*	*0.50*	*0.67*	*0.56*
PointNetFullGR	*0.91*	*0.70*	*0.77*	*0.72*	*0.89*	*0.75*	*0.83*	*0.78*
Random	0.87	0.55	0.62	0.57	0.73	0.41	0.55	0.46
RandomGR	0.89	0.63	0.72	0.66	0.82	0.63	0.73	0.66
VisNoOccl	0.87	0.54	0.62	0.56	0.72	0.41	0.53	0.45
VisNoOcclGR	0.89	0.62	0.71	0.65	0.85	0.66	0.75	0.69
VisOccl	0.87	0.53	0.61	0.56	0.76	0.46	0.61	0.51
VisOcclGR	0.89	0.63	0.72	0.66	0.83	0.64	0.73	0.67
VisOcclHist	0.87	0.54	0.62	0.57	0.78	0.50	0.62	0.54
VisOcclHistGR	**0.90**	**0.64**	**0.73**	**0.67**	**0.87**	**0.70**	**0.80**	**0.73**

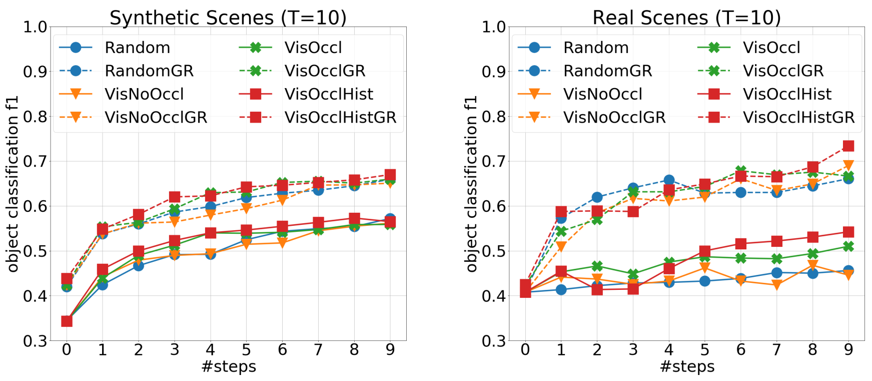


Figure 6.5: Comparison of active classification methods using both the synthetic (left) and real (right) datasets. Our method shows steady improvement with increasing number views, outperforming all the baselines.

that is $T = 10$ in our experiments; results are computed at the final step of each run. All results are averaged over 10 runs with a random starting position at each run.

The introduction of GR improves results of about 10% on F_1 score on synthetic scenes, and increases to about 20% in real scenarios. This indicates that the GR can effectively correct the classification in terms of more true positives. The improvement by GR is higher in real scenes compared to the synthetic scenes because the depth images from the real acquisition is noisier compared to the synthetic dataset, which makes the naïve PointNet classification worse. For real scenes, we also observe marginal classification improvement, in terms of the F_1 score brought by the NBV criterion when considering the occlusion on visibility and the weight for avoiding visited areas.

All experiments are performed on a Alienware Aurora Desktop with i7 core. The averaged processing time between consecutive moves for reconstruction, segmentation, geometric refinement and NBV are 0.12 s, 0.06 s, 0.88 s and 0.19 s, respectively.

Figure 6.5 shows the averaged classification performance in terms of F_1 score over time in both synthetic and real scenes. Without GR, the proposed active strategy can mostly outperform the random strategy in both synthetic and real scenes. With GR, the performance of all active strategies are boosted at each time step. To conclude, the proposed NBV strategy stands out with respect to all the baseline approaches, especially at increasing time steps as the method more efficiently covers the scene while avoiding to revisit previous viewpoints. The supplementary material shows a video of the robotic movement with various NBV criteria along with the classification performance at each step.

6.3 Self-awareness for active objects recognition

Here, we evaluate the impact of the 3D saliency volumes presented in Section 5.3. The volumes are injected into the scene representation providing a more informative scenario to the object classifier. The overall system has been built as a Partially Observable Markov Decision Process (POMDP [2]) enriched with a self-awareness module.

We evaluate two main aspects of our RA-POMDP: 1) the quality of the 3D saliency volume (Sec. 6.3.1), and 2) the recognition performances, on both simulated and real data (Sec. 6.3.5).

6.3.1 3D saliency volume

We present some results on different object classes, showing how the saliency volume changes in dependence with the type and number of classes of 3D objects considered.

As dataset, we consider the ObjectNet3D [116] dataset, commonly adopted for passive 3D object recognition. The dataset is composed of RGB images and CAD models, and we use the latter for the experiments. In particular, we select three different sets of classes, composed respectively by 2, 35 and 85 classes (the whole dataset), considering the same number of 3D models for each class in order to avoid biases in the 3D saliency volume. Specifically, we pick 5 random models from each class (the minimum number of models for a class): 3 models for training, 1 for building the 3D dense volume and 1 for the testing. Experimentally, fusing together different models of a single class in for creating the 3D dense volume was not found as particularly beneficial for the active recognition stage; in any case, we plan to further investigate this aspect as a next step.

For each 3D model, we extract depth acquisitions with the V-REP software [2], in particular simulating the real camera (Asus Xtion Pro) that has been used for the real data experiments. With V-rep, we define the set $\mathcal{X}_\rho$ of 128 views, uniformly distributed on the upper hemisphere of radius 0.6 meter.

6.3.2 3D saliency volume creation

The steps of the 3D saliency volume have been already shown in Fig. 5.5, focusing on the `bicycle` class. Once we have computed the 2D saliency of each view using [32], we build the 3D model of the target class using depth images and averaging the saliency of the views that insist on the same voxel. As for 3D classifier, we adapt an ImageNet-pretrained AlexNet architecture [56] similarly to [14]. Specifically, we substitute the three fully connected layers with a single one, mapping the convolutional features directly to the desired number of classes. For fine-tuning the classifier, we employed the ADAM optimizer with a learning rate of 0.0001 and weight decay 0.0005, batch size of 128, for a total of 4 epochs. The proposed set of hyperparameters is enough to reach a single

[2]`http://www.coppeliarobotics.com/`.

image testing accuracy of 46% over the set of 35 classes, 36% over the set of 85 classes.

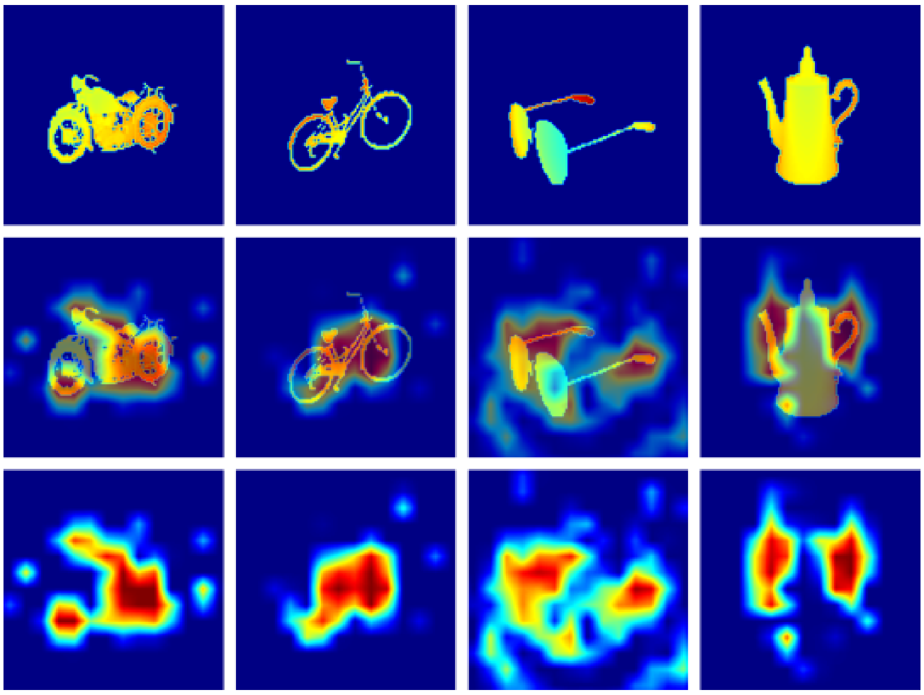


Figure 6.6: Saliency maps for the same view of different objects.

For the 3D saliency volume creation, the size of the voxels is not really crucial and goes in the range from 25 to 80. In practice, the voxelization ensures a sort of classification generalization to the other models of the class: voxels smaller than 25 lead to overfitting (bringing to low classification accuracy), higher than 80 lead to a less effective 3D dense volume.

The voxel-based representation of the saliency volumes is relatively simple and its resolution could be adjusted depending on the application needs. However, an alternative way to build such volumes is applicable if the CAD model, *ie.* a dense triangular mesh, of the target object is available. The CAD model could be updated so that each vertex is associated with a score, that is the average saliency of the reconstructed 3D points in the neighborhood of that vertex. A saliency volume built this way benefits from being smoother in its appearance (depending on the neighborhood size of the vertices, similarly to the voxel grid density), it is easier to import in any rendering engine thanks to its triangular mesh format and it could be also easily compared with other techniques which assign an importance score to the vertices.

Volumes comparison Building a saliency volume is an important step to understand what a neural classifier has learned during the training process. The output of the saliency extraction and mapping onto the object models highlights those areas that are important for the classification task, that are not necessarily correlated to any geometrical information. However, there are many geometrical approaches that are able to describe a 3D object, given its mesh, that assign a score to its vertices. The main difference between the saliency extracted from a neural classifier and a geometrical descritor is that the former is associated to a specific task, *eg.* the classification, while the latter is not necessarily associated to any *semantic* task, like shape classification, since such methods usually highlight cusps and string edges.

To prove this, we compare the saliency volumes extracted with our method on four different approaches:

1. we fine-tuned the AlexNet architecture to classify the LINEMOD [43] objects starting from their depth views as described in Section 6.3.2; saliency maps are computed for each viewpoint and mapped onto the 3D CAD models of the dataset.

2. PointNet [80, 118], that is a state of the art neural classifier specifically designed to work with 3D data, has been fine-tuned to work on 3D point clouds of the LINEMOD dataset. Saliency has been extracted by randomly removing a portion of the point clouds: the higher the loss, the higher the saliency value assigned to the points belonging to the removed region.

3. Heat Kernel Signature (HKS) [3]: this approach is related to the heat diffusion concept. Given an initial score (heat) distribution on the object surface, a kernel function is applied to compute the new distribution of the head after a certain timelapse.

4. Wave Kernel Signature (WKS) [98] computes the Laplace-Beltrami operator on the surface in order to compute the eigen-functions (used as descriptors) associated to the surface points.

Qualitative results of this test are shown in Figure 6.7. The comparison between geometrical and neural saliency extractors brings two important results: first, geometrical analysis of shapes is clearly not related to the classification task while. Second, the comparison shows that different classifiers lead to very different representations (first two columns of Figure 6.7).

6.3.3 Impact of the type of object classes

The saliency volume is the proxy of what the classifier has found as particular discriminative in a multi-class classification setup. It is thus interesting to observe the impact that different object classes have on the volume of a given class.

To this sake we consider a two-class classification problem, keeping one class fixed, `bicycle`, while changing the other class as `teapot`, `glasses`, and `motorbike`, respectively. The three cases are reported in Fig. 6.8. As visible, in the teapot case (Fig. 6.8a) the bike saliency volume is uniformly highlighted: every view serves to discriminate against the teapot. In the case of the glasses (Fig. 6.8b), the shape and relative location of the lenses resembles the shape of the tires of the bike (note that in the dataset bike and glasses have the same dimensions). As consequence the bicycle tires have less importance than the previous case. In the third case, bike and motorbike are compared (Fig. 6.8c). Here the tires have definitely less importance, while the internal framework becomes crucial (in practice, the classifier has understood the presence or absence of the engine as discriminative).

6.3.4 Impact of the number of object classes

Another important aspect is to check how the saliency volume changes while increasing the number of classes into play. The question is whether a bigger number of classes would lead to have saliency volume focusing on fewer parts. We focus on the bike class and start with two classes (bike and teapot). The saliency volume of the bike is the same than Fig. 6.8c. With 35 classes (Fig.

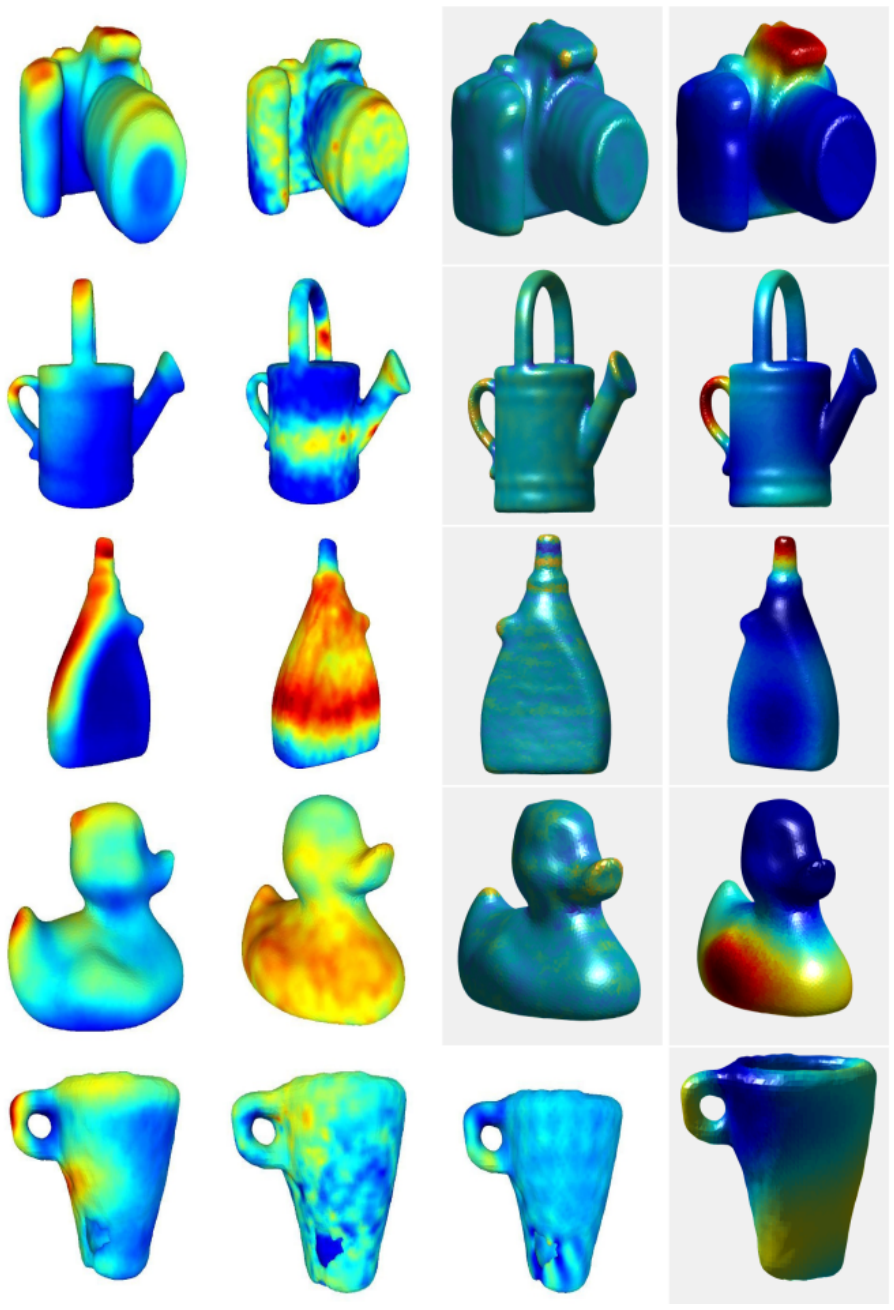


Figure 6.7: Saliency volumes computed on five objects taken from the LINEMOD dataset [43]. Each column is computed by extracting the saliency using a different approach. From left to right: PointNet, AlexNet, HKS, WKS. Part of this analysis has been reported in [18].

6.8d) the general aspect of the volume does not change, and with with 85 classes (Fig. 6.8e) the most important part remain the framework, with the tires that become quite irrelevant, due also to the presence of other classes having the tires (`car`, `wheelchair`).

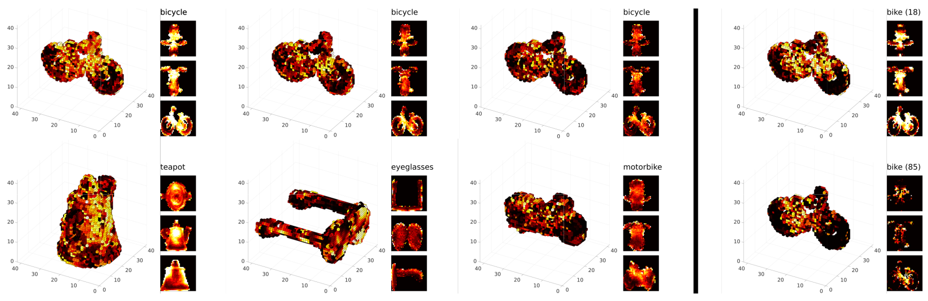


Figure 6.8: On the left, 3D dense saliency volumes when changing type of classes in a two class problem: bike VS teapot (a), bike VS glasses (b) and bike VS motorbike (c). On the right, when increasing the number of classes: 35 classes (d) and 85 classes (e).

6.3.5 Recognition results

We show here the recognition performances of our RA-POMDP, comparing against four competitors, on simulated and real data:

Static. The standard (passive) recognition baseline approach: we take a single observation from the starting viewpoint and predict the class.

Random. A random walk on the viewsphere, avoiding to revisit viewpoints. At each step, the next viewpoint to visit is selected at random among the 8 nearest neighbours.

VP-tree [2]. Like ours, this method is based on a non-myopic POMDP implementation, but in this case the classifier (a vocabulary tree) is treated as a black box, with no saliency computation therein.

Classifier. In this case we change our RA-POMDP model by closing the box of the classifier (the deep network), that is, without any saliency assumption.

As robotic platform, we use a Panda arm, from Franka Emika GmbH[3]. This is a 7 d.o.f. manipulator that can move in a workspace of about 855mm, perfectly suited for our scenario. The motion planner is implemented in the *MoveIt!* framework, part of the Robotic Operating System (ROS). We use OMPL (Open Motion Planning Library) as motion planner library and *Trac-IK* as kinematic solver. In the simulations, we simulate the RGB-D camera acquisition with *V-Rep*, which has set to emulate an Asus Xtion Pro Live[4].

6.3.6 Simulated data

Simulated data exclude all the variability related to the depth sensor acquisition (sensor noise mainly). For time reasons we are able to test 35 classes out of 85 (the list is in the additional material), repeating the experiment for each class 5 times by sampling randomly the initial position over the 64 $\mathbf{x}_i$ points $\in \mathcal{X}_\rho$.

Table 6.5 shows the results of the simulations considering the average classification accuracy (*Accuracy*). The other considered quantities are the *Belief* (the belief entry $b(s)$ of the output class s in the belief distribution b, see Eq. 5.17), the *Number of steps* (the number of atomic movements performed by the robot), the *Distance* (geodesic distance covered by the camera); the *Energy* (energy spent by the robot as in Eq. 5.12). These numbers should be considered together with Fig. 6.9, showing the process of exploration of the robot in terms of steps done and updated belief (in this case, the bicycle class has been reported as representative of the various approaches, for the sake of visualization[5]). As visible, RA-POMDP achieves the highest accuracy and the lowest energy spent. Fig. 6.9 shows that RA-POMDP has the steepest gradient, meaning that at each step the belief has a considerable improvement. The Classifier approach shows the importance of opening the box of the classifier: in practice, in this approach the observation model has a scattered volume of classification values to consider as proxy, which do not communicate which part has been actually important to be visualized for a successful classification, resulting in an extremely discontinuous observation model: the low Belief in fact says that at some point the next

[3]`https://www.franka.de/panda`

[4]`http://xtionprolive.com/asus-3d-depth-camera/asus-xtion-pro-live`

[5]doing an average over the classes here would mean to average trajectories of different lengths, resulting in a confused visualization.

useful point to move is too far and expensive and the process stops. Fig. 6.9 shows a flatter gradient. VP-tree has even lower performances, dictated by the scarcer classifier taken into account, and (Fig. 6.9) a longer process of belief update, due to the fact that at each step the classifier is not so discriminative. Please consider that [2] represents the actual state of the art as for active object classification in a very similar scenario (despite it has been proven with less classes, with another dataset). The random classifier has remarkable accuracy performance and cost, but the low belief value means that it has been arrived in a configuration where the next move costs too much than the expected belief positive update: this is due to the fact that random positions do not take into account of the energy and manipulability constraints. Random is better than VP-tree because of a better classifier. Fig. 6.9 for the random classifier shows an obviously irregular belief update.

Approach	Accuracy	Belief	Steps	Distance	Cost
Static	0.36	0.344	–	–	–
Random	0.85	0.584	3.938	**0.492**	7.837
VP-tree [2]	0.40	0.318	13.350	1.677	10.253
Classifier	0.65	0.567	**3.879**	0.788	8.005
Saliency	0.95	0.743	4.713	1.005	6.885
RA-POMDP	**1.00**	**0.771**	4.688	0.814	**6.498**

Table 6.5: Quantitative results on simulated data. Average values on objects belonging to 35 classes and 5 initial positions for each class.

6.3.7 Real data

We also tested our framework on a real setup. An RGB-D sensor Asus Xtion Pro Live has been mounted on the end effector of a Franka Panda robotic arm and the exploration has been carried out on 4 object classes. Due to the setup constrains, we focused on objects that can reasonably lie on a tabletop: `cup`, `eyeglasses`, `bottle`, and `scissors`. Results are comparable with the ones in the simulated scenario, with RA-POMDP achieving the highest accuracy, followed by Classifier and Random walks. RA-POMDP is in general able to

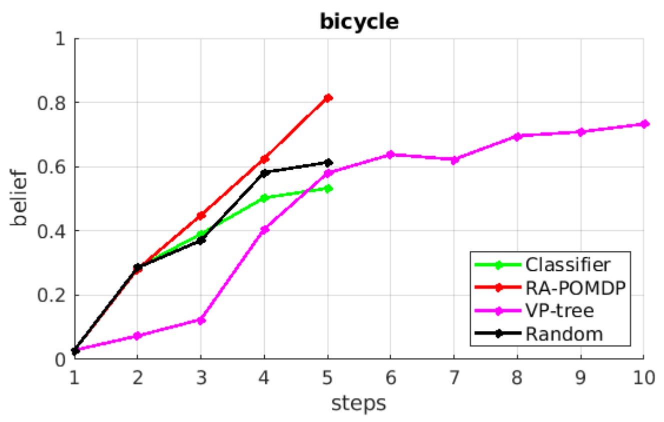


Figure 6.9: Belief evolution for class bicycle during the exploration.

generate a correct prediction sooner than most of the competitors, spending less energy during the exploration and with a higher confidence (belief).

6.4 Global Saliency Representation

We first describe how we create our real dataset with ground truth information followed by the relative experimental protocol. Results over several trial will provide information related to the active classification method.

6.4.1 Real world dataset

In order to evaluate our active classification system, we created a new real world dataset containing representative of 10 different scenes with a set of up to 6 objects lying over a planar surface. Objects arrangement was spatially different to simulate different possibilities for occlusions (see Figure 6.4 for some examples of scenarios). Since training was based on the original LINEMOD dataset [43], the objects are 3D printed instances the same classes used in that dataset. Subsequently, we used a Universal Robot (UR5) to acquire data with synchronized poses (poses of both the camera and the robot's end-effector) using a Kinect v1 mounted on the end effector of the arm. The robot arm motion was manually pre-programmed in order to obtain 150 views overall around the scene and then replicating exactly the same motion for the remaining nine scenes. In this way we can simulate different viewpoints and motion strategies for exhaustive evaluation of our and the baselines approaches. Finally we also annotate the object poses in 3D so having a reference related to the correct object poses.

In every scene, all objects are placed on top of a planar surface approximately defined at $z = 0$ and the objects are placed in their "natural pose", which correspond to the z axis from the model's reference frames facing up. Since we rely on SSP's pose estimates for picking the next best view, the model catalog used is the same as in [101], a subset of the LINEMOD's provided models which excludes the symmetric cases.

6.4.2 Evaluation protocol

Regarding the evaluation, our aim is to understand the goodness of our motion policy, called **Active**, for improving recognition scores. We test our method against three baseline approaches, **Full Random**, **Local Random**, **Noisy Saliency**. With the **Full Random** strategy we plan to verify that our **Active** method provides a motion strategy which is successful by reaching the best recognition score in fewer moves. Such baseline method randomly selects the viewpoint to visit, considering only unseen views. Furthermore, the process of selecting the next view does not take into account the current classification confidence, that is the update function is not considered during the entire movement, and neither the distance between the views. The process stops when we reach the maximum number of moves. Then we keep the viewpoint that generated the highest average classification score. As for evaluation metrics, we provide standard classification measures precision, recall and F_1 measure.

The **Local Random** pipeline randomly selects the next viewpoint on a local basis, *ie.* the set of candidates from which to sample is limited on the neighbours of the starting viewpoint (15 cm in our experiment). With the **Local Random** strategy we verify that our active method selects a local motion strategy which is successful given the maximization of the GSM. The stop criteria is the same of the previous fully random process.

The **Noisy Saliency** baseline method is identical to our **Active** method pipeline but we inject gaussian noise (zero mean, standard deviation equal to one) in the saliency volume of each object. This baseline demonstrates that the proposed Active approach is resilient to the learned saliency maps. For instance, coarse saliency maps are expected when intra-class object geometry changes. The stopping criteria is the same as our active method to allow a direct comparison.

The starting point of the camera is a random position in the allowed kinematic space of the robot. We set an upper bound of 20 viewpoints to visit for all the methods. We then evaluate a total of 50 sequences, divided equally over the available 10 scenes.

It is worth noting that no kinematic constraints are considered in any of these baseline approaches apart for the **Noisy Saliency**, our proposed method is the only one that respect the constraints of the robotic arm. For all the methods tested, the starting point of the camera is a random position in the allowed kinematic space of the robot.

6.4.3 Experimental evaluation

In this section we provide the evaluation for the proposed active approach and the baseline methods.

Approach	Confidence	Steps	Precision	Recall	F1 score
Full Random	0.82 (0.10)	20 (–)	0.75 (0.12)	0.46 (0.11)	0.56 (0.14)
Local Random	0.82 (0.14)	20 (–)	0.76 (0.17)	0.45 (0.15)	0.55 (0.14)
Noisy Saliency	0.83 (0.14)	2.94 (1.12)	0.74 (0.17)	0.45 (0.14)	0.54 (0.13)
Active	**0.86** (0.10)	**2.74** (0.89)	**0.79** (0.15)	**0.50** (0.15)	**0.60** (0.13)

Table 6.6: The Table shows the confidence in the object classification, number of movements required by the system to reach the stop condition, Precision and Recall of the detected classes.

Table 6.6 shows the evaluation of the motion planning performance of our method compared with the baseline algorithms **Full Random**, **Local Random** and **Noisy Saliency**. The **Full Random** motion is performing reasonably well at the expense of several moves but this is due to the fact that it can span views very far apart from each other (i.e. no constraints over the kinematics of a robotic arm). The **Local Random** strategy provides a mechanic similar to our active method but with a simplistic random choice of the next view. Performance decrease consistently since there is a strong limitation of the view

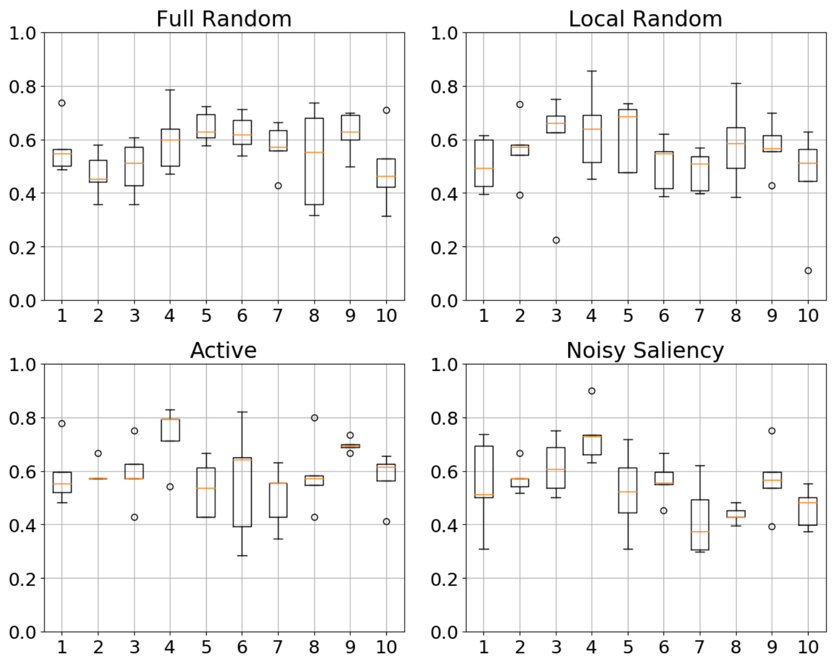


Figure 6.10: F1 score boxplots of the active system and the baselines.

span given the unpredictable local motion selection. Our active method instead shows that the introduced GSM is fundamental to move the viewpoint towards the increase of the detection score.

Boxplots in Figure 6.10 show in more details the test results distribution for the 10 scenes. A boxplot has a rectangle to represent the second and third quartiles, with a vertical orange line inside to indicate the median value. The lower and upper quartiles are shown as horizontal lines either side of the rectangle. Circles represents outlier data outside the lower and upper quartiles intervals. In particular we see that the **Full Random** approach has experiments with less outliers than other approaches. Notice that that with 20 steps we have a good sampling of the viewing sphere (the overall number of views is 150).

The **Local Random** approach is performing similarly than the random approach but have far more gross negative outliers given by the constraint on the viewpoint local choice. The boxplot related to the **Active** approach shows a higher variability given by the fact we are choosing the next best view by maximising the GSM at each viewpoint. Regardless of this, the results are in general better than all the other methods given the very limited number of average steps (2.7). Both positive and negative outlying results are present but the negative effect is far limited than the **Local Random** method. Finally, the **Noisy Saliency** just show a degradation of results but similar behaviour as the **Active** method.

CHAPTER 7

Conclusions

In the last decade, deep learning has reached impressive results in numerous fields. However, this approach suffers from several drawbacks. Training a neural network requires a considerable amount of computational power, making deep learning often a very expensive business. Furthermore, the enormous amount of required data to properly train a big network (*ie.* an architecture with multiple stacked layers) could be an insurmountable limit for some tasks, such as in the medical field where limited labeled data is available. Luckily, some datasets like ImageNet and ShapeNet [56, 116, 15] have become popular thanks to big organizations and research groups but the data availability is still a problem nowadays. Tha major drawback related to deep learning is the lack of transparency of the feature space. The mathematical formulation of the optimization process (*ie.* the backpropagation algorithm) and the relations among the layers and neurons is quite simple. However, the highly dimensional latent space (*ie.* the number of weights of a model) makes impossible to clearly understand the inner weights structure of a neural network [8, 63].

This research faced the problem of the weights visualization of a deep neural network. Thanks to state of the art approaches to get robust 2D attention maps that we call *saliency maps*, such as [32] and [119], we have proposed two novel visualization methods: the **visual summaries** and the **saliency volumes**. The main contribution of this research is the exploitation of such images in order to improve the effectiveness of pre-trained classification models both in the 2D [37] and 3D domains [85, 110]. In particular, this work proposes a novel approach to visualize the informative parts of the input data in a compact way.

7.1 Visual Summaries for Image Classification

In the first part of this work we discovered a way to exploit the output of 2D saliency maps. We have presented the first visualization approach which employs analysis of multiple images within an object class to provide an explanation on what has been understood by a network in terms of *visual parts* to form an object class [37]. Our approach takes as input a trained neural network and a set of images, generating a set of image clusters, that we call *summaries*, where each cluster is representative of an object visual part.

This novel approach is composed by two phases. Firstly, a crisp saliency map is computed for each test image so that only the most important ares are highlighted for a specific class. The per-pixel saliency score is based on the work of Fong et.al. [32]: this perturbation-based approach assigns a score to each pixel according to the variation they lead to the final classification loss. In the second phase, we connect the salient regions using the proposal flow algorithm [38]; then, we cluster the regions using the affinity propagation algorithm [33]. The output of such pipeline is a set of clusters, *ie.* **visual summaries**, associated to those patterns that are systematically present in the image data.

We performed a user study and made use of an automatic tagger to demonstrate our summaries capture clear visual semantics of an object class. Furthermore, the number of summaries generated by our approach is correlated with the classification accuracy of a deep network: the more the summaries, the higher the classification accuracy. We demonstrated this for the most common classification architectures AlexNet, VGG, GoogleNet and ResNet. Finally, the summaries are proved to be helpful improving the classification accuracy of a network, by adopting multiple, specific specialization procedures with the images of each summary.

Our approach is the first visualization system which considers multiple images at the same time, generalizing about the visual semantic entities captured by a deep network. Contrarily to the standard visualization tools, advantages of our proposed approach can be measured quantitatively, the most important of them is that of improving the original network by training additional classifiers specialized on recognizing the visual summaries. The future perspective is to inject the analysis of the summary in the early training of the deep network, and not only as a post processing boosting procedure.

7.2 Saliency Volumes for Active Recognition

Assistive robots have attracted increasing attention from both academics and industries [16] with the objective of supporting daily life and the essential capability is to be able to perceive the environment. However, object recognition in cluttered scenes can be very challenging using single-shot-based method [81, 101, 117]. Also, estimating objects positions is not trivial even with a single object in an simple (*ie.* uncluttered) environment. Multiple objects and occlusions generate a challenging scenario, possibly leading to failures if the object detector and pose estimators make wrong decisions about the scene structure. As the literature presents, even the best single object classifiers decrease their performance when occlusions appear [47]. In this context, active vision is fundamental in analysing the scene and deciding which Next Best View should be chosen to achieve an higher recognition score [2, 86]. When scenes are cluttered, objects are often misclassified due to self-occlusions or odd poses. Active Object Recognition is an interesting strategy for actively covering more viewpoints in order to ease the classification task.

In this thesis, we have proposed an active strategy to 3D object classification aiming to select the best viewpoints where salient object parts are visible.

As our former attempt, we propose the first AOR approach which does visualize what a classifier has learned giving a sort of *recognition self-awareness* in the planning procedure. Inspired by [2], we have initially modelled the planning as a Partially Observable Markov Decision Process: we have designed a novel observation model in order to exploit deep network visualization methods. Such approaches help explaining why a particular classifier succeeds in its task. We have been particularly interested in those approaches providing saliency maps over the input images [32]. Then, we have adopted such saliency maps building a **saliency volume** which fuses together 2D saliency maps, generating continuous proxy for a certain classifier. This volumes are used as observation model in our active framework.

Our approach has been applied to scenarios where single or multiple objects are present and occlusions are likely to occur. A depth frame is acquired from a sensor from a given viewpoint with known camera pose, so that a point cloud is first pre-segmented to remove the background and then object 3D instances are pre-selected by clustering the point cloud. Each cluster is potentially an object that is sent to an object classifier (CNN-based) which provides a probability score over the objects classes and assigns the most confident one to the object. Given the object classification output, a 6D pose detector related to the class with the maximum score provides the object 3D position and mesh in a reference space for all the instances. We then fit to each detected 3D object a pre-learned saliency volume [85].

Here, we have demonstrated that if most salient 3D parts are visible the target object is likely to be correctly classified with higher confidence. In the case of challenging scenarios, *ie.* where multiple objects are present, we have considered all the saliency volumes at once to generate a global 3D representation of the scene, as described in Section 5.4. The contribution of a specific viewpoint is computed by projecting the visible parts of the volumes on the camera image plane. Then, the relevancy of these scores is computed aiming to estimate a unique representation of the scene, *ie.* our Global Saliency Model, that is used for deciding the next view for recognition. The approach iterates until it finds for a given viewpoint a confidence score higher than a threshold.

Our active vision approach has firstly improved 3D object classification through shape reconstruction using depth data. Experimental results in both synthetic and real scenarios with severe occlusions have shown that both geometric refinement and NBV improve the object classification performance compared to the method using only a single depth frame, while approaching the performance achieved by the naïve PointNet with complete object point cloud. We performed a detailed study to demonstrate the effectiveness of each NBV criterion with/without the geometric refinement. Both the consideration of occlusion and view-point history in NBV brings marginal improvement while the geometric refinement improves up to about 20% in terms of F_1 score.

However, active object recognition approaches traditionally considered the classifiers as black boxes, planning robot trajectories to feed them with maximally different images. With RA-POMDP we drastically change this point of view, considering that classifiers build internal representations in which some visual patterns are more important than others. Leveraging deep visualization approaches, by means of a 3D dense saliency volume, we extract this knowledge, exploiting it to plan maximally effective sensor trajectories.

We presented an active recognition algorithm that is able to autonomously find the best view for improving recognition of objects in a cluttered environment. Real experiments over a ground truth generated by a robotic arm have shown increased performance against different baseline motion policies. In particular, even if starting from random positions, our active method is capable of deciding the class of 6 objects in the scene just with 2.74 steps in average, where random approaches score 4% less detection accuracy in 20 steps. This major speedup might be further enhanced by including memory in the NBV motion policy or to exploit contextual information as given by the depth information from the scene.

7.3 Waeknesses

Saliency extraction benefits from a recent literature that is, however, far from being robust. Most of the state of the art methods are very sensible to the hyperparameters and even a small variation in the settings might lead to significant changes in the results. Since our major contributions are based on such techniques, the extraction of the visual summaries and saliency volumes is fragile and the variation strongly affects the output of the proposed pipelines. This could be the reason, and it is particularly true for the active recognition frameworks, of the small gap among the approaches compared in Table 6.6 and shown in Figure 6.10.

7.4 Future works

As future work, we plan to relax the proposed method to address 3D object recognition with arbitrary number of objects arranged in more complex scenes. It is also of our interests to improve the method with a learnt metric for NBV by encoding the status of object reconstruction.

Due to the unsatisfactory results presented in Section 6.4, the robustness of saliency map extraction must be evaluated in order to make the visual summaries computation and saliency volumes building more stable and predictable.

The results reached so far promote our idea, still at its infancy, as a promising approach with the possibility to easily embed new features, *eg.* working at different scales: saliency could be extracted at different level of details, allowing to highlight different details and to extract visual patterns in a coarse-to-fine fashion.

Bibliography

[1] U. Army. Standards of medical fitness. *Washington, DC: Department of the Army*, 2011.

[2] N. Atanasov, B. Sankaran, J. Le Ny, G. J. Pappas, and K. Daniilidis. Non-myopic view planning for active object classification and pose estimation. *IEEE Transactions on Robotics*, 2014.

[3] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *IEEE international conference on computer vision workshops (ICCV workshops)*, 2011.

[4] J. E. Banta, L. R. Wong, C. Dumont, and M. A. Abidi. A next-best-view system for autonomous 3D object reconstruction. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 2000.

[5] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *TPAMI*, 1992.

[6] G. Best, J. Faigl, and R. Fitch. Online planning for multi-robot active perception with self-organising maps. *Autonomous Robots*, 2018.

[7] G. Biegelbauer and M. Vincze. Efficient 3d object detection by fitting superquadrics to range image data for robot's object manipulation. In

Robotics and Automation, 2007 IEEE International Conference on. IEEE, 2007.

[8] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[9] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2017.

[10] L. Bo, X. Ren, and D. Fox. Depth kernel descriptors for object recognition. In *IROS*. IEEE, 2011.

[11] E. Brachmann, A. Krull, F. Michel, S. Gumhold, Jamie, Shotton, and C. Rother. Learning 6d object pose estimation using 3d object coordinates. In *ECCV*, 2014.

[12] M. Carletti, M. Cristani, V. Cavedon, C. Milanese, C. Zancanaro, and A. Giachetti. Analyzing body fat from depth images. In *International Conference on 3D Vision (3DV)*, 2018.

[13] M. Carletti, M. Cristani, V. Cavedon, C. Milanese, C. Zancanaro, and A. Giachetti. Estimating body fat from depth images: Hand-crafted features vs convolutional neural networks. In *Conference and Exhibition on 3D Body Scanning and Processing Technologies*, 2018.

[14] F. M. Carlucci, P. Russo, and B. Caputo. A deep representation for depth images from synthetic data. In *ICRA*, 2017.

[15] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[16] K. Charalampous, I. Kostavelis, and A. Gasteratos. Recent trends in social aware robot navigation: A survey. *Robotics and Autonomous Systems*, 2017.

[17] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung. On visual similarity based 3d model retrieval. *Computer Graphics Forum*, 2003.

[18] G. Consolaro, M. Carletti, and M. Cristani. Analisi di salienza di nuvole di punti 3d per applicazioni di machine learning. *Master thesis, University of Verona*, 2019.

[19] F. Cunico, M. Carletti, M. Cristani, F. Masci, and D. Conigliaro. 6d pose estimation for industrial applications. In *International Conference on Image Analysis and Processing*, 2019.

[20] W. Darwish, S. Tang, W. Li, and W. Chen. A new calibration method for commercial rgb-d sensors. *Sensors*, 2017.

[21] S. Demura, S. Sato, and T. Kitabayashi. Percentage of total body fat as estimated by three automatic bioelectrical impedance analyzers. *Journal of physiological anthropology and applied human science*, 2004.

[22] J. Deng, A. C. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *ECCV*, 2010.

[23] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. In *CVPR*, 2016.

[24] 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.

[25] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *CVPR*. IEEE, 2010.

[26] M. Eitz, R. Richter, T. Boubekeur, K. Hildebrand, and M. Alexa. Sketch-based shape retrieval. *ACM Transactions on Graphics*, 2012.

[27] S. Ekvall, F. Hoffmann, and D. Kragic. Object recognition and pose estimation for robotic manipulation using color cooccurrence histograms. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*. IEEE, 2003.

[28] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 2009.

[29] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, 1996.

[30] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *arXiv preprint arXiv:1806.09266*, 2018.

[31] S. Foix, G. Alenya, J. Andrade-Cetto, and C. Torras. Object modeling using a tof camera under an uncertainty reduction approach. In *ICRA*, 2010.

[32] R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *ICCV*, 2017.

[33] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 2007.

[34] T. N. Garlie, J. P. Obusek, B. D. Corner, and E. J. Zambraski. Comparison of body fat estimates using 3d digital laser scans, direct manual anthropometry, and dxa in men. *American Journal of Human Biology*, 2010.

[35] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and R. Medina-Carnicer. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, 2015.

[36] A. Giachetti, C. Lovato, F. Piscitelli, C. Milanese, and C. Zancanaro. Robust automatic measurement of 3d scanned models for the human body fat estimation. *IEEE journal of biomedical and health informatics*, 2015.

[37] M. Godi, M. Carletti, M. Aghaei, F. Giuliari, and M. Cristani. Understanding deep architectures by visual summaries. In *BMVC*, 2018.

[38] B. Ham, M. Cho, , C. Schmid, and J. Ponce. Proposal flow. In *CVPR*, 2016.

[39] M. Hanheide, C. Gretton, R. Dearden, N. Hawes, J. Wyatt, A. Pronobis, A. Aydemir, M. Göbelbecker, and H. Zender. Exploiting probabilistic knowledge under uncertain sensing for efficient robot behaviour. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[40] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 1979.

[41] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE CVPR*, pages 770–778, 2016.

[42] V. H. Heyward, D. R. Wagner, et al. *Applied body composition assessment.* Human Kinetics, 2004.

[43] 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.

[44] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab. Dominant orientation templates for real-time detection of texture-less objects. *CVPR*, 2010.

[45] T. Hodan, F. Michel, E. Brachmann, W. Kehl, A. G. Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T. Kim, J. Matas, and C. Rother. BOP: benchmark for 6d object pose estimation. In *ECCV*, 2018.

[46] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. In *ECCV*. Springer, 2012.

[47] E. Hsiao and M. Hebert. Occlusion reasoning for object detection under arbitrary viewpoint. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[48] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *PAMI*, 1999.

[49] G. Kadrev, G. Kostadinov, and P. Ruskov. Expansion of a cnn-based image classifier's scope using transfer learning and k-nn. technical report. In *2016 IEEE 8th International Conference on Intelligent Systems (IS)*, 2016.

[50] A. Kanezaki, Y. Matsushita, and Y. Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *CVPR*, 2018.

[51] V. Karasev, A. Chiuso, and S. Soatto. Controlled recognition bounds for visual learning and exploration. In *NIPS*, 2012.

[52] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[53] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. SSD-6D: Making RGB-Based 3d detection and 6d pose estimation great again. In *ICCV*, 2017.

[54] M. Kouchi and M. Mochimaru. Errors in landmarking and the evaluation of the accuracy of traditional and 3d anthropometry. *Applied ergonomics*, 2011.

[55] S. Kriegel, C. Rink, T. Bodenmüller, A. Narr, M. Suppa, and G. Hirzinger. Next-best-scan planning for autonomous 3d modeling. In *IROS*, 2012.

[56] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

[57] H. Kurniawati, D. Hsu, and W. S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems*, 2008.

[58] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 2015.

[59] A. Leeper, K. Hsiao, E. Chu, and J. K. Salisbury. Using near-field stereo vision for robotic grasping in cluttered environments. *Experimental Robotics, Springer Tracts in Advanced Robotics*, 2014.

[60] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 2015.

[61] G. Li and Y. Yu. Visual saliency based on multiscale deep features. In *CVPR*, 2015.

[62] G. Li and Y. Yu. Deep contrast learning for salient object detection. In *CVPR*, 2016.

[63] S. Liang. Why deep neural networks for function approximation. *CoRR*, arXiv preprint arxiv:1610.04161, 2017.

[64] N. Liu and J. Han. Dhsnet: Deep hierarchical saliency network for salient object detection. In *CVPR*, 2016.

[65] D. J. MacKay. Probable networks and plausible predictions—a review of practical bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 1995.

[66] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015.

[67] A. Mahendran and A. Vedaldi. Salient deconvolutional networks. In *ECCV*, 2016.

[68] M. Malmir and G. W. Cottrell. Belief tree search for active object recognition. In *IROS*, 2017.

[69] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015.

[70] T. Nakashika, T. Hori, T. Takiguchi, and Y. Ariki. 3d-object recognition based on llc using depth spatial pyramid. In *ICPR*. IEEE, 2014.

[71] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.

[72] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, 2015.

[73] T. V. Nguyen, J. Feng, and S. Yan. Seeing human weight from a single rgb-d image. *JCST*, 2014.

[74] K. Norton, T. Olds, S. Olive, and N. Craig. Anthropometry and sports performance. *Anthropometrica*, 1996.

[75] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev. The building blocks of interpretability. *Distill*, 2018.

[76] J. Pan, E. Sayrol, X. Giro-i Nieto, K. McGuinness, and N. E. O'Connor. Shallow and deep convolutional networks for saliency prediction. In *CVPR*, 2016.

[77] T. Patten, W. Martens, and R. Fitch. Monte Carlo planning for active object classification. *Autonomous Robots*, 2018.

[78] C. Pfitzner, S. May, and A. Nüchter. Neural network-based visual body weight estimation for drug dosage finding. In *Medical Imaging: Image Processing*, 2016.

[79] C. Potthast and G. S. Sukhatme. A probabilistic framework for next best view estimation in a cluttered environment. *Journal of Visual Communication and Image Representation*, 2014.

[80] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, 2018.

[81] C. R. Qi, H. Su, M. Kaichun, and L. J. Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[82] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Proc. of the International Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[83] M. Rad and V. Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *ICCV*, 2017.

[84] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016.

[85] A. Roberti, M. Carletti, F. Setti, U. Castellani, P. Fiorini, and M. Cristani. Recognition self-awareness for active object recognition on depth images. In *BMVC*, 2018.

[86] A. Roberti, R. Muradore, P. Fiorini, M. Cristani, and F. Setti. An energy saving approach to active object recognition and localization. In *IECON*, 2018.

[87] F. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and Vision Computing*, 2018.

[88] A. Saxena, L. Wong, M. Quigley, and A. Y. Ng. A vision-based system for grasping novel objects in cluttered environments. In M. Kaneko and Y. Nakamura, editors, *Proc. of International Symposium Robotics Research (ISRR)*, 2011.

[89] R. G. Schneider and T. Tuytelaars. Sketch classification and classification-driven analysis using fisher vectors. *ACM Transactions on Graphics*, 2014.

[90] F. Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 2002.

[91] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017.

[92] Y. Shen, C. Feng, Y. Yang, and D. Tian. Neighbors do help: Deeply exploiting local structures of point clouds. *CoRR*, arXiv preprint arXiv:1712.06760, 2017.

[93] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani. Manipulation planning with probabilistic roadmaps. *International Journal of Robotics Research*, 2004.

[94] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR*, 2014.

[95] K. Simonyan and A. Zisserman. Very deep convolutional networks for large scale image recognition. In *ICLR*, 2015.

[96] W. E. Siri. The gross composition of the body. *Advances in Biological and Medical Physics*, 1956.

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

[98] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *ICCV*, 2015.

[99] 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.

[100] G. Taylor and L. Kleeman. *Visual perception and robotic manipulation: 3D object recognition, tracking and hand-eye coordination.* Springer, 2008.

[101] B. Tekin, S. N. Sinha, and P. Fua. Real-time seamless single shot 6d object pose prediction. In *CVPR*, 2018.

[102] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1996.

[103] L. Torabi and K. Gupta. Integrated view and path planning for an autonomous six-dof eye-in-hand object modeling system. In *IROS*, 2010.

[104] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 2013.

[105] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid. Learning from synthetic humans. In *CVPR*, 2017.

[106] J. Vidal, C.-Y. Lin, and R. Martí. 6d pose estimation using an improved method based on point pair features. In *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*. IEEE, 2018.

[107] G. Walck and M. Drouin. Automatic observation for 3d reconstruction of unknown objects using visual servoing. In *IROS*, 2010.

[108] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. *CoRR*, abs/1901.04780, 2019.

[109] L. Wang, H. Lu, X. Ruan, and M.-H. Yang. Deep networks for saliency detection via local estimation and global search. In *CVPR*, 2015.

[110] Y. Wang, , M. Carletti, F. Setti, M. Cristani, and A. D. Bue. Active 3d classification of multiple objects in cluttered scenes. In *ICCV Workshop*, 2019.

[111] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 2004.

[112] J. Wells and M. Fewtrell. Measuring body composition. *Archives of disease in childhood*, 2006.

[113] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison. Elasticfusion: Dense slam without a pose graph. In *Robotics: Science and Systems*, 2015.

[114] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, 2016.

[115] 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.

[116] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese. ObjectNet3D: A large scale database for 3d object recognition. In *European Conference Computer Vision (ECCV)*, 2016.

[117] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A convolutional neural network for 6d object pose estimation in cluttered scenes. *RSS*, 2018.

[118] D. Xu, D. Anguelov, and A. Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *CVPR*, 2018.

[119] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. In *ICML*, 2015.

[120] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[121] J. Zhang, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff. Top-down neural attention by excitation backprop. In *ECCV*. Springer, 2016.

[122] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. In *ICLR*, 2015.

[123] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.

[124] Q.-Y. Zhou and V. Koltun. Dense scene reconstruction with points of interest. *ACM Transactions on Graphics (ToG)*, 2013.

[125] L. M. Zintgracef, T. S. Cohen, T. Adel, and M. Welling. Visualizing deep neural network decisions: Prediction difference analysis. *CoRR*, arXiv preprint arXiv:1702.04595, 2017.