Deep Learning for Object Detection in Robotic Grasping Contexts

Thèse

Jean-Philippe Mercier

Doctorat en informatique
Philosophiæ doctor (Ph. D.)

Québec, Canada

Deep Learning for Object Detection in Robotic Grasping Contexts

Thèse

Jean-Philippe Mercier

Sous la direction de:

Philippe Giguère, directeur de recherche
Abdeslam Boularias, codirecteur de recherche

Résumé

Dans la dernière décennie, les approches basées sur les réseaux de neurones convolutionnels sont devenus les standards pour la plupart des tâches en vision numérique. Alors qu'une grande partie des méthodes classiques de vision étaient basées sur des règles et algorithmes, les réseaux de neurones sont optimisés directement à partir de données d'entraînement qui sont étiquetées pour la tâche voulue. En pratique, il peut être difficile d'obtenir une quantité suffisante de données d'entraînement ou d'interpréter les prédictions faites par les réseaux. Également, le processus d'entraînement doit être recommencé pour chaque nouvelle tâche ou ensemble d'objets. Au final, bien que très performantes, les solutions basées sur des réseaux de neurones peuvent être difficiles à mettre en place.

Dans cette thèse, nous proposons des stratégies visant à contourner ou solutionner en partie ces limitations en contexte de détection d'instances d'objets. Premièrement, nous proposons d'utiliser une approche en cascade consistant à utiliser un réseau de neurone comme pré-filtrage d'une méthode standard de « template matching ». Cette façon de faire nous permet d'améliorer les performances de la méthode de « template matching » tout en gardant son interprétabilité. Deuxièmement, nous proposons une autre approche en cascade. Dans ce cas, nous proposons d'utiliser un réseau faiblement supervisé pour générer des images de probabilité afin d'inférer la position de chaque objet. Cela permet de simplifier le processus d'entraînement et diminuer le nombre d'images d'entraînement nécessaires pour obtenir de bonnes performances. Finalement, nous proposons une architecture de réseau de neurones ainsi qu'une procédure d'entraînement permettant de généraliser un détecteur d'objets à des objets qui ne sont pas vus par le réseau lors de l'entraînement. Notre approche supprime donc la nécessité de réentraîner le réseau de neurones pour chaque nouvel objet.

Abstract

In the last decade, deep convolutional neural networks became a standard for computer vision applications. As opposed to classical methods which are based on rules and hand-designed features, neural networks are optimized and learned directly from a set of labeled training data specific for a given task. In practice, both obtaining sufficient labeled training data and interpreting network outputs can be problematic. Additionnally, a neural network has to be retrained for new tasks or new sets of objects. Overall, while they perform really well, deployment of deep neural network approaches can be challenging.

In this thesis, we propose strategies aiming at solving or getting around these limitations for object detection. First, we propose a cascade approach in which a neural network is used as a prefilter to a template matching approach, allowing an increased performance while keeping the interpretability of the matching method. Secondly, we propose another cascade approach in which a weakly-supervised network generates object-specific heatmaps that can be used to infer their position in an image. This approach simplifies the training process and decreases the number of required training images to get state-of-the-art performances. Finally, we propose a neural network architecture and a training procedure allowing detection of objects that were not seen during training, thus removing the need to retrain networks for new objects.

Contents

Résumé ii

Abstract iii

Contents iv

List of Tables vi

List of Figures vii

Remerciements xi

Foreword xii
Included Publications xii

Introduction 1

1 Object Detection for Robotic Grasping 3
1.1 Traditional Methods 4
1.2 Deep Learning Methods 7
1.3 Deep Learning Limitations 16

2 Deep Object Ranking for Template Matching 22
2.1 Résumé 22
2.2 Abstract 22
2.3 Introduction 23
2.4 Related Work 25
2.5 Overview of the proposed method 26
2.6 Experiments 30
2.7 Conclusion 35

3 Learning Object Localization and 6D Pose Estimation from Simulation and Weakly Labeled Real Images 39
3.1 Résumé 39
3.2 Abstract 39
3.3 Introduction 40
3.4 Related Work 42
3.5 Proposed Approach 43

3.6 Weakly Supervised Learning Experiments for object detection and classification . . . 47
3.7 6D Pose Estimation Experiments . . . 49
3.8 Conclusion . . . 51

4 Deep Template-based Object Instance Detection **56**
4.1 Résumé . . . 56
4.2 Abstract . . . 56
4.3 Introduction . . . 57
4.4 Related work . . . 59
4.5 Network architecture . . . 60
4.6 Training data . . . 63
4.7 Experiments . . . 65
4.8 Discussion . . . 70
4.9 Supplementary Material . . . 71

Conclusion **81**

Bibliography **84**

List of Tables

2.1 Average ranking of objects 33

3.1 Comparative performance evaluation for the ADD metric 51
3.2 Comparative performance evaluation for the visual discrepency recall scores . . 51

4.1 Network architecture ablation study 66
4.2 Ablation study on random rotations applied to the local template orientation during training 67
4.3 Ablation study for various numbers of local templates at test time 67
4.4 Robustness towards different selection of global templates 68
4.5 Impact of the number of objects used during training 68
4.6 Quantitative comparison to the state of the art 69
4.7 Performances on the 2D bounding box metric for each object of the Linemod dataset 71
4.8 Average precision for each object evaluated on the Occluded Linemod dataset . 72

List of Figures

1.1 Typical vision-based robotic grasping workspace . . . 3
1.2 Typical output for object detection . . . 4
1.3 Typical pipeline for correspondence-based methods . . . 5
1.4 Linemod pipeline and template modalities . . . 7
1.5 Challenging conditions for computer vision approaches . . . 7
1.6 Typical CNN architecture . . . 9
1.7 Overview of R-CNN . . . 12
1.8 Overview of Fast R-CNN . . . 12
1.9 Region Proposal Network for Faster R-CNN . . . 13
1.10 Overview of Faster R-CNN . . . 13
1.11 Overview of Mask R-CNN . . . 14
1.12 Overview of YOLO . . . 15
1.13 Overview of SSD . . . 15
1.14 Zoom out data augmentation strategy example . . . 16
1.15 Overview of RetinaNet . . . 16
1.16 Research areas related to decreasing annotated data requirements . . . 17
1.17 Synthetic data generation pipeline . . . 19
1.18 Data-related limitations of CNNs . . . 20

2.1 Block diagram of our proposed method . . . 25
2.2 Image sequence showing detections as objects are removed . . . 27
2.3 Residual block and network used in the paper . . . 29
2.4 Different backgrounds used for our experiments . . . 32
2.5 Objects from the pose dataset . . . 32
2.6 Performance on the in-house dataset . . . 34
2.7 Performance on the pose dataset . . . 35

3.1 Overview of our approach for 6D pose estimation at inference time . . . 41
3.2 Overview of the proposed approach at training . . . 44
3.3 Examples of synthetic image used in training . . . 45
3.4 Ablation study . . . 48

4.1 Overview of the proposed method. At test time, our network predicts the 2D location (in an RGB image) of a target object (unseen during training) represented by templates acquired from various viewpoints. . . . 58

4.2 Our proposed architecture. In stage 1, the network learns to localize an object solely from a set of templates. Object-specific features are learned by the "object attention" and "pose-specific" branches, and are subsequently correlated/subtracted with the generic features of the backbone network. In stage 2, the network leverages the learned representation to perform different tasks: binary segmentation, center and bounding box prediction. At test time, a single "global" template is randomly selected, while several "local" templates are combined. 61
4.3 Examples from our domain randomization training set. In (a), objects are randomly placed in front of the camera and rendered using OpenGL with a background sampled from Sun3D dataset [41]. In (b) and (c), a physical simulation is used to drop several objects on a table with randomized parameters (camera position, textures, lighting, materials and anti-aliasing). For each render, 2 variations are saved: one with simple diffuse materials and without shadows (b), and one with more sophisticated specular materials and shadows (c). . . . 64
4.4 Qualitative results on the Occluded Linemod dataset 70
4.5 All 15 objects in the Linemod dataset 72
4.6 More domain randomization images generated with our simulator 73
4.7 More qualitative results on Linemod dataset 74
4.8 More qualitative results on the Occluded Linemod dataset 75
4.9 Detailed networks 76

To my friends and family

True knowledge exists in knowing that you know nothing.

Socrates

Remerciements

Tout d'abord, j'aimerais remercier Philippe Giguère, mon directeur de recherche. Tu m'as accueilli dans ton laboratoire en 2013 pour la maîtrise et je te serai pour toujours reconnaissant pour le support que tu m'as apporté et les belles expériences que tu m'as permis de vivre. Je me souviendrai longtemps des voyages qu'on a faits ensemble et je garderai également un excellent (et possiblement un peu brouillon) souvenir de nos rencontres impromptues aux Voûtes de Napoléon.

Également, j'aimerais remercier Abdeslam Boularias, mon co-directeur de recherche. Abdeslam, je te remercie de m'avoir reçu dans ton laboratoire de l'Université Rutgers au New Jersey et de m'avoir donné un gros coup de pouce dans l'avancement de mon doctorat. Durant cette visite, j'ai pu remarquer ton ardeur au travail, ce qui m'a certainement motivé à travailler encore plus fort. Somme toute, j'ai adoré mon expérience à travers laquelle j'ai pu connaître des gens extraordinaires.

Ensuite, j'aimerais remercier mes collaborateurs. Ludovic Trottier, tu as été une ressource inestimable dans mon apprentissage ainsi qu'un formidable partenaire de voyage. Chaitanya Mitash, I have enjoyed getting to know you during my visit and collaborate with you. Jean-François Lalonde et Mathieu Garon, un gros merci pour toute l'aide apportée durant le dernier droit de mon doctorat. Cette aide m'a permis de rester motiver et de finalement atteindre l'objectif. Les autres gens du DAMAS, ce fut un honneur de vous côtoyer, d'échanger des idées et d'avoir du plaisir avec vous.

J'aimerais aussi remercier l'entreprise Robotiq qui m'a offert une chance avec un stage à la fin de ma maîtrise qui s'est transformé en une collaboration de plusieurs années. Cette collaboration m'a permis de parfaire mes connaissances en plus de faire connaissance avec des personnes exceptionnelles.

Finalement, j'aimerais remercier mes amis et ma famille. Votre soutien tout au long de mon parcours m'a permis de rester les pieds sur terre durant la montagne russe d'émotions qu'a été cette expérience. Sans vous, cette thèse n'existe pas.

Foreword

There are 3 papers included in this thesis. This chapter contains, for each included paper, the authors, my authorship status, the publication status, modifications between the published version and the included one and my contributions to the project.

Included Publications

Deep Object Ranking for Template Matching

Jean-Philippe Mercier, Ludovic Trottier, Philippe Giguère and Brahim Chaib-Draa
In *Winter Conference on Applications of Computer Vision* (WACV), 2017

Authorship Status: Main author
Publication Status: Published
Modifications: Minor
Citations have been added in the abstract and the text has been formatted to one column.
Contributions: Major
For this paper, I implemented the pretrained object detection network used to generate object hypotheses and the template matching method. I also coded a program to make all the modules communicate among themselves, as the neural networks were executed on a server. I acquired the training and test images for the homemade dataset and developed the evaluation procedure. Ludovic Trottier contributed by training the classifier network and participated in the writing of the paper in the corresponding section. Philippe Giguère contributed as the main advisor of the project and also participated in the writing. Some people from the company Robotiq have also contributed as non-official advisors for the project.

Learning Object Localization and 6D Pose Estimation from Simulation and Weakly Labeled Real Images

Jean-Philippe Mercier, Chaitanya Mitash, Philippe Giguère and Abdeslam Boularias
In *International Conference on Robotics and Automation* (ICRA), 2019

Authorship Status: Main author

Publication Status: Published
Modifications: Minor
Citations have been added in the abstract and the text has been formatted to one column.
Contributions: Major
The core of the project has been done mainly while I was a visiting student at Rutgers University and finished when I was back at Laval University. My contributions for this project are the developement, training and testing of the neural network, the development of a simple simulator to generate synthetic images, annotations of real images and writing the core of the paper. Chaitanya Mitash was responsible to execute the StoCS method to evaluate the 6D pose estimation performances and also has contributed to the writing of the corresponding section in the paper. Abdeslam Boularias acted as the main advisor on the project while keeping Philippe Giguère in the loop. Both participated actively in revising my first draft of the paper.

Deep Template-based Object Instance Detection

Jean-Philippe Mercier, Mathieu Garon, Philippe Giguère and Jean-François Lalonde In *Winter Conference on Applications of Computer Vision* (WACV), 2021

Authorship Status: Main author
Publication Status: Published
Modifications: Minor
Supplementary material has been added as an additional chapter at the end of the paper.
Contributions: Major
For this project, I started from a code base developed by Mathieu Garon, as he and I had the same idea and he was more advanced than I was when we decided to join forces. At this point, the project was at a proof-of-concept stage. The network was working well to detect known objects on simple synthetic images, but was not generalizing well to unknown objects and failed to work on real images. My contributions for the project were the generation of different domain randomized synthetic training datasets, the addition of several data augmentation strategies, the coding and testing of all architecture ideas we had. I also developed the testing procedure to compare our detection performances with other state-of-the-art approaches. Mathieu Garon, Philippe Giguère and Jean-François Lalonde all acted as project advisors and participated in the writing.

Introduction

Deep Learning is a powerful way to approximate any mathematical functions by "enabling the computer to build complex concepts in terms of other simpler concepts" [24]. It has progressed significantly in the last decade with the advent of more powerful general purpose graphics processing units (GP-GPUs), bigger annotated datasets, new techniques and easy-to-use software librairies. As such, most state-of-the-art approaches in fields such as computer vision (image classification, detection, segmentation, etc.), natural language processing and speech recognition are now based on deep learning. As an example, the top-5 classification error of the winning approach on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [74], which is an image classification contest, went from 25% in 2011 to less than 3% in 2017. Notably, in 2012, a deep learning approach won the challenge for the first time. The proposed approach, AlexNet [44], won with a top-5 error of 16% while the runner-ups were in the 20-30% range. It marked a point of no return in history, as the popularity of deep learning has grown ever since.

While very popular in both research and industry, deep learning has some limitations. For instance, one of the main limitations is that models need a lot of annotated training data to obtain state-of-the-art results. For particular fields such as industrial robotics, it can be quite difficult to find annotated datasets available for download with the required set of objects. Therefore, datasets have to be acquired and annotated by people in what can be a time-consuming process, especially for tasks such as object detection (boxes have to be drawn around each object and each box has to be labeled as belonging to a category) and object segmentation (each pixel of every image has to be labeled). For example, it has been reported that the acquisition and annotation process for an object detection dataset of 1400 images was about 201 hours (10 hours for image acquisition, 185 hours for labeling and 6 hours to correct the annotations) [33]. Scaling that kind of dataset to millions of images like in ImageNet would be very difficult and ineficient. In certain niches, such as collaborative robotics, tasks to be done or object sets can change frequently, which triggers the need to retrain the network or acquire a new dataset. Consequently, coupled with the challenge of debugging and interpreting the results of deep networks, which are often referred as black boxes, many classical approaches are still used in favor of deep learning.

The main objective of this thesis is to propose new ways to simplify the use of deep learning approaches, particularly for computer vision in the context of robotic grasping. A review of these existing approaches is contained in Chapter 1. Then, the core of the thesis consists of 3 publications that are presented in chronological order. The publications, along with their contributions, are summarized below:

- Chapter 2: A cascaded approach in which a deep learning model generates prior information (ranking of probable objects in an image) to a classical template matching method. The approach allows to improve detection performances of the template matching method while keeping its interpretability.
- Chapter 3: A cascaded approach in which the proposed architecture leverages the strengths of weakly-supervised and adversarial learning to train a network from a combination of synthetic and weakly-labeled images. The proposed approach generates class-specific probability maps to initialize an iterative pose estimation method. The full approach achieves performances similar to the state-of-the-art, but with a simplified process for image acquisition and annotation.
- Chapter 4: A new object detection approach trained only on synthetic images that allows to detect new objects (unseen during training) at test time without having to retrain or finetune the network. The approach is versatile and achieves performances that are similar to other state-of-the-art approaches, but without being trained on the test objects.

Chapter 1

Object Detection for Robotic Grasping

To grasp objects sucessfully, humans require different senses such as vision and touch. In the same way, to accomplish robotic grasping, robotic arms need different sensors to perceive their environment. Visual light cameras and more recently, 3D cameras, are the most commonly used sensors for this task because of their simplicity, cheap price and all the possible applications they allow. Other sensors, such as tactile ones, can be useful by providing additional modalities [5, 10, 56]. However, they are not studied in this thesis.

In the left part of Fig. 1.1, an example of a typical vision-based grasping system is shown. This system comprises a robotic arm equipped with a gripper and a camera that can either be placed directly on the arm or in a fixed overhead position. The arm is placed alongside a worksurface on which target objects usually lie.

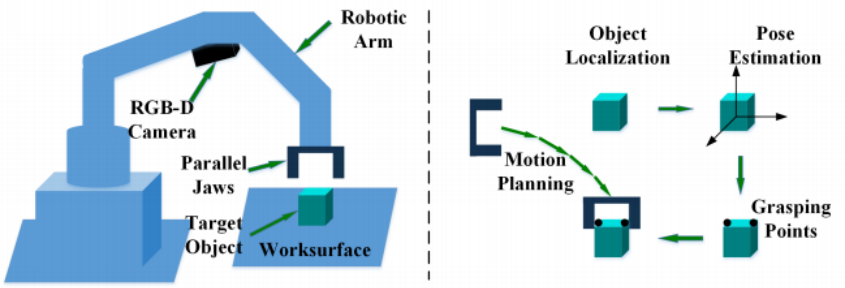


Figure 1.1 – Typical vision-based robotic grasping workspace. On the left, the system comprises a robotic arm equipped with a gripper and a camera (that can be placed directly on the arm or on a fixed overhead position) that grasps objects placed on a surface. On the right, the typical steps for grasping are shown: object localization (detection), pose estimation, grasping points estimation and motion planning to achieve the grasp. Image from [12].

The typical workflow for vision-based grasping is illustrated on the right of Fig. 1.1. Grasping can be performed either on known or unknown objects. To grasp unknown objects, systems rely on the 2D or 3D visual data to directly determine a grasping point. DexNet 2.0 is a

successful example of such an approach: it predicts the quality (probability of grasp success) from previously-determined grasp point candidates. Instead of being limited by a prior step like the prediction of grasp candidates, Levine et al. [47] predict movements of the gripper that maximizes the probability of successfully grasping an object solely from an overhead camera. While these approaches work really well, they are limited to simple tasks, since the system does not know the object it is manipulating nor its configuration in the robotic gripper. For more complex manipulation tasks, such as splitting different objects in different bins or inserting objects into others, knowing objects in advance simplifies the problem. In an offline phase, specific information such as predefined grasp points, features, templates or 3D models can be stored for each object and then used at test time to localize them, estimate their poses and grasp them.

The focus of the thesis is on object localization/detection of *known objects*. The detection task, as shown in Fig. 1.2, typically is to find rectangular regions, called bounding boxes, around objects of interest and determine which object is enclosed in the box. These objects can be classified as different generic categories (persons, cars, dogs, etc.) or can represent specific object instances (this person, this car, this dog, etc). For robotic grasping, we are more interested in the finding specific instances. Example images of the target object obtained from one or multiple viewpoints, also referred as "templates", can be used as training data in an offline phase or directly at runtime for matching. In the following sections, object detection methods are reviewed, from traditional rule-based approaches based on handcrafted features (sec. 1.1) to modern deep learning approaches (sec. 1.2) and their limitations (sec. 1.3).

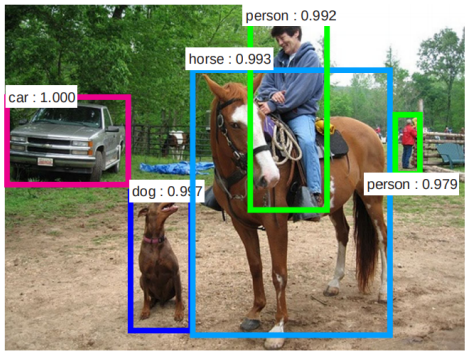


Figure 1.2 – Typical output for object detection [70]

1.1 Traditional Methods

To localize known objects in images, traditional computer vision approaches use handcrafted features to represent object characteristics. In this section, we review some traditional approaches belonging to correspondence-based and template-based approaches.

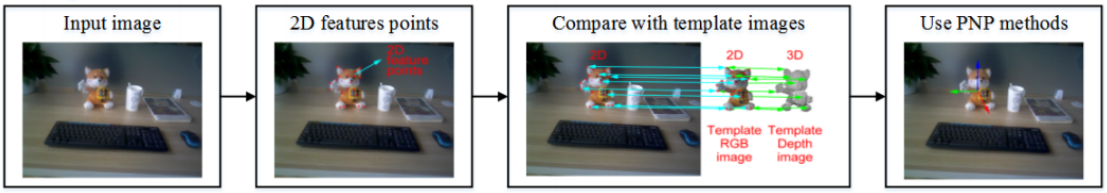


Figure 1.3 – Typical pipeline for correspondence-based methods. Feature points from the input image are matched with precomputed features of the template image for detection. PnP methods can be used to retrieve the pose of the object if 3D information about the template is available. Figure from [12].

1.1.1 Correspondence-based approaches

Correspondence-based approaches use local features to find known objects by matching small image patches with similar properties between an example image of the target object (template) and an input image. A typical pipeline for such task is shown in Fig. 1.3. The first step consists of computing local feature points. It is usually achieved by detecting interest points or "keypoints" in the image (edges, corners or regions/blobs) and then computing descriptors for each of them. Then, the next step is to find matches (corresponding features) between the template and input image and generate transformation hypotheses from the possible matches. If 3D information about the template is available and the camera intrinsic parameters are known, the 6D pose of objects can be estimated with methods solving the Perspective-n-Point (PnP) problem [20, 31, 46]. These typical steps are briefly described in more details below:

Keypoint Extraction Keypoint extraction is used to improve the matching speed by decreasing the number of points to match between source and target images (instead of matching every pixel). Keypoint extraction methods aim to be repeatable under different image transformations such as rotation, translation, scale, etc. Popular keypoint detection methods include the Canny edge detector [6], Sobel operator [81], Harris corner detector [27], Features from accelerated segment test (FAST) [72], Laplacian of the Gaussian (LoG), Difference of Gaussians (DoG) and maximally stable extremal regions (MSER) [58].

Feature calculation During feature calculation, a descriptor (feature) is computed for each detected keypoints. Descriptors are represented by caracteristics of the point and/or its neighbourhood and can be as simple as the color or gradient of the pixel. Descriptors are designed to be as distinct as possible between them (to decrease the likelihood of bad matches) and to be robust to different image transformations. In the litterature, descriptors are often compared by their dimensionality, data types (binary or floating point) and their invariance properties (scale, rotation, illumination, viewpoint, blur, etc.). Popular descriptors include the Scale-Invariant Feature Transform (SIFT) [55], Speeded Up Robust Features (SURF) [1] and Oriented FAST and Rotated BRIEF (ORB) [73].

Correspondences retrieval and rejection Correspondences are searched between extracted features from the template and input image. For fast processing of the multiple match possibilities, a library for fast approximation of nearest neighbors search such as FLANN [59] is usually used. For increased robustness and reliability of correspondences, different strategies, such as the ratio and symmetry tests, can be applied. The ratio test consists of dividing the distance of the best match by the distance of the second best match. Those with a ratio higher than a predefined threshold are rejected to avoid ambiguities. This strategy is used in SIFT [55] and eliminates approximately 90% of the bad correspondences while maintaining most of the good ones. The symmetry test rejects a correspondence if the best match of a feature in one way (e.g. template towards input image) is not the best match in the other way (e.g. input image towards template).

Transformation Hypotheses From the possible correspondences, transformation hypotheses are generated. One of the most popular approaches is Random Sample Concensus (RANSAC) [17]. In RANSAC, random triplets are iteratively selected to compute transformations. The one with the most inliers is kept as the best transformation hypothesis.

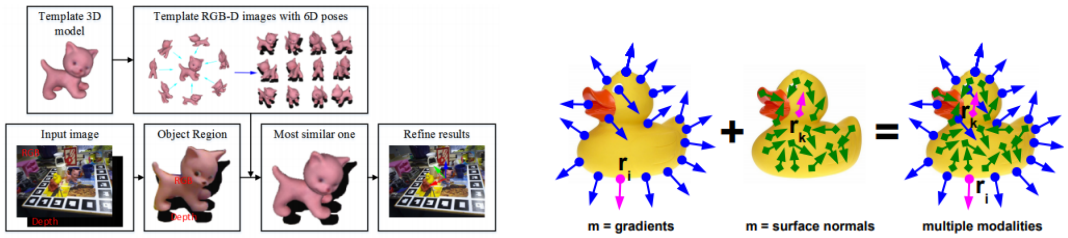
1.1.2 Template-based methods

Feature correspondence methods work really well when target objects have rich textures, but are suboptimal for textureless objects. Instead of generating a sparse set of correspondences, template-based methods produce dense sets of correspondences and thus work well on textureless objects which have few discriminating features. To create a dense set of correspondences, template-based methods typically employ a sliding window approach in which the template is slid over the input image and outputs similarities between the two at each position.

The most naïve template-based approaches try to match a template identical to what is seen in the input image. Simple comparison metrics, such as the sum of squared differences, can be used for this kind of approach, since a perfect match is expected. For cases where there are small differences (brightness, noise, etc.) between the template and the object in the image, cross-correlation or normalized cross-correlation, which output similarities for each sliding windows, are good solutions. Like many template matching methods, they are greatly limited by their sensitivity to scale and rotation changes.

Using image gradients is usually more robust to illumination changes. They are therefore a popular choice of feature for template-based methods. For instance, Histogram of Oriented Gradients (HoG) [11] can be used for detection. In [57], they trained a linear Support Vector Machine (SVM) for each object instance. This approach is relatively slow compared to more recent approaches, since at test time, each classifier needs to be computed on each image patch coming from the sliding window.

One of the most popular and recent example of template matching approaches is called Linemod [32]. As shown in Fig. 1.4, Linemod uses multiple modalities (color gradients and surface normals) for each rendered view of a 3D model to retrieve the object instance in RGB-D images. Its variant, Line-2D, only uses gradients and works on color images. Since generated templates represent specific viewpoints, a rough estimation of the object pose in the input image can be retrieved from the most similar template.

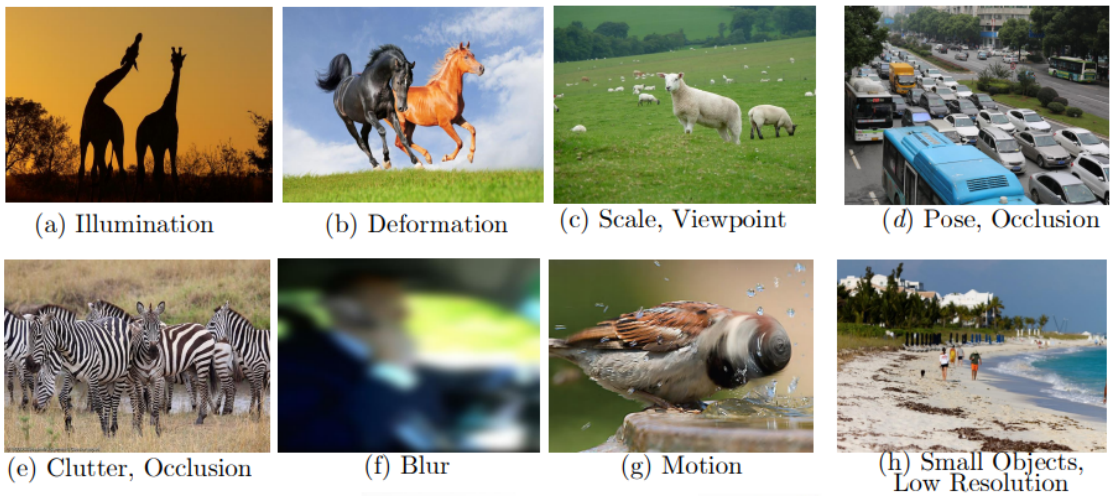


(a) Pipeline (image from [12]) (b) Template modalities

Figure 1.4 – Linemod [32] is a template matching method that uses templates generated from 3D models. Its pipeline is illustrated in (a) and the template modalities it uses are shown in (b).

1.2 Deep Learning Methods

Images can be composed of thousands or millions of pixels and be altered in nearly unlimited ways. Consequently, the appearance and shape of objects contained in images can vary wildly (see Fig. 1.5). Traditional methods, which are based on engineered features and sets of rules, tend to fail to generalize to these infinite variations.



(a) Illumination (b) Deformation (c) Scale, Viewpoint (d) Pose, Occlusion

(e) Clutter, Occlusion (f) Blur (g) Motion (h) Small Objects, Low Resolution

Figure 1.5 – Challenging conditions for computer vision approaches [51].

Deep learning, on the other hand, is a powerful and flexible machine learning technique mostly applied to high-dimensional data such as images. Importantly, they can approximate any non-linear mathematical function. As opposed to traditional methods that depend on fixed handcrafted rules, deep learning approaches learn directly from the data and tend to generalize better, as long as they are trained on a large quantity of data. In 2012, AlexNet [44] was released and won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [74] by a wide margin. Since then, most of the research and state-of-the-art approaches in the area of computer vision are based on deep learning. Many concepts used in deep learning, however, date back to much earlier than 2012. For instance, Convolutional Neural Networks (CNNs), which form the basis of most deep learning approaches applied to vision, were introduced towards the end of the 20th century [18, 45]. It was not until recently, with innovations such as general-purpose graphics processing units (GP-GPUs), bigger annotated datasets and new techniques, that they became so prevalent. In this section, we review the general concepts of CNNs (sec. 1.2.1), the most notable innovations and network architectures (sec. 1.2.2) and specific approaches for object detection (sec. 1.2.3).

1.2.1 Convolutional Neural Networks (CNNs)

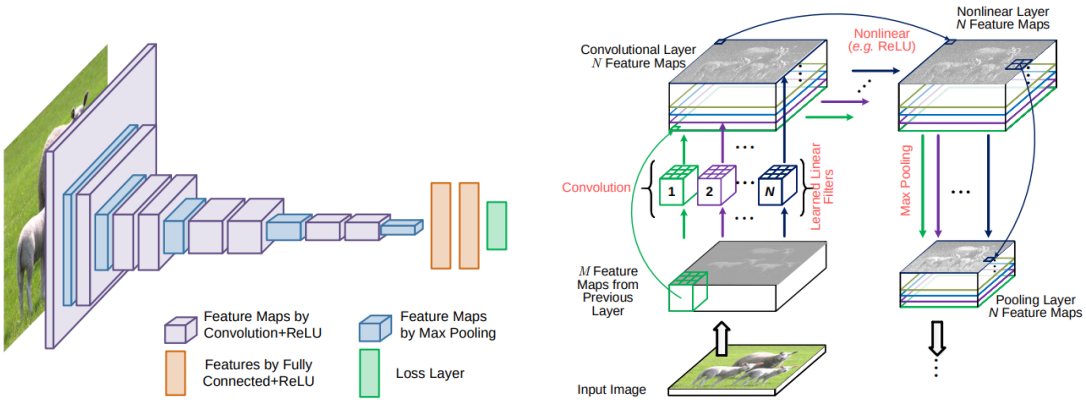
In this section, a typical CNN architecture is first described. Then, the training process is briefly explained.

Architecture

The global structure of a Convolutional Neural Network (CNN) is similar to a Multilayer perceptron (MLP): it has an input, hidden layers and an output. A typical example of a CNN architecture for the task of object classification is shown in Fig. 1.6a. The input is the image and the output is a vector of prediction scores with one score per object class. The core of the archictecture is the hidden layers that are stacked on top of each others to form a hierarchy in which complex and abstract concepts are learned from simpler concepts. As shown in Fig. 1.6b, a hidden layer is generally composed of multiple blocks that include convolution filters, activation functions and pooling. These components are briefly described below:

Convolution A convolution layer is composed of multiple convolution filters that are optimized through training. The role of the filters is to extract features at every pixel of an image, based on their neighbourhood (size of the filter). Using a sliding window approach, each filter is slided over the whole input, generating feature maps.

Activation Function Activation functions allow networks to approximate non-linear functions. The most common activation function is the Rectified Linear Unit (ReLU) [60]. Its main property is that it allows to train deeper networks without the vanishing gradient problem present with other activation functions such as sigmoid and tanh.



(a) Typical CNN Architecture (b) Typical convolution block

Figure 1.6 – In (a), a typical CNN architecture for classification is shown. The input image passes through multiple convolution blocks (convolution + ReLU + pooling), which is shown in more details in (b), to finally be processed by fully-connected layers that output a prediction vector. During training, a loss function is used so that the network can be optimized using gradients computed by backpropagation. Figures from [12].

Pooling Pooling layers create downsampled feature maps by splitting them in a certain number of regions and calculating a certain value (usually either average or maximum) for all these regions. By reducing the spatial resolution of the feature maps, pooling allows CNNs to become invariant to small translations.

For the output layer, fully-connected layers are often used. Their multiple neurons are "fully" connected (connected to every cell of the previous layer) and generate flattened outputs (vectors instead of feature maps for convolutions) of desired dimension. If used as the last layer of the network, their output (predicted class scores) is usually transformed in probabilities with the softmax function:

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{1.1}$$

where x_i is the score to be normalized and x_j are the scores for each individual class.

Training Process

The training of deep learning approaches is an iterative process in which annotated training data is sampled at each iteration. To train the network, there is an additional step called backward pass that is added after the forward pass (prediction made from the input). During this backward pass, an error is calculated using a loss function (e.g. cross-entropy for object classification) that compares the predictions made by network and the expected predictions (ground truths). Then, an optimization method such as Stochastic gradient descent (SGD) [42,

71] updates the weights of the network in attempt to minimize the error. This is achieved by computing the gradients of the loss function with respect to the weights of the CNN, a step called backpropagation, and then updating the weights according to the learning rate and the previously-computed gradients. For SGD, each weight update is based on gradients that are calculated on minibatches (small numbers of random training data). It allows fast approximations of the gradient compared to methods that calculate the gradient on the whole dataset for a single weight update.

1.2.2 Timeline of CNN architectures

At this point, general deep learning concepts have been reviewed. In this section, some of the most notable CNN architectures and their contributions are reviewed in chronological orders.

- In 2012, AlexNet [44] revolutionized computer vision by winning by almost 10% (absolute gain) the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [74], which evaluated the top-5 error rate on object classification. They combined previously-proposed ideas, such as ReLU [60] (see section 1.2.1) and regularization techniques such as dropout [34] (randomly setting to 0 the output of a neuron in fully-connected layers) and data augmentation [8, 9] (random translations, reflections, crops and RGB intensity alterations) to reduce overfitting.

- In 2014, VGGNet [79] and GoogLeNet [83] improved the top-5 error on ImageNet to respectively 6.8% and 6.7%. In VGG, they used stacked 3x3 convolutions to increase the network depth. In GoogLeNet, they introduced Inception modules and replaced the usual fully-connected layers by global average pooling. The Inception modules process the input with parallel convolutional filters of different sizes that are concatenated to generate the output features, which allows more flexibility regarding the receptive field of the network.

- In 2015, Batch Normalization [39] was introduced. By using the mean and variance of mini-batches to normalize, scale and shift layer inputs, batch normalization allows faster training and better generalization of networks.

- In 2016, ResNet [30] decreased the top-5 error on ImageNet to 3.6% by proposing skip connections which made identity mapping possible, thus allowing much deeper networks than it was previously possible with other architectures.

- In 2017, Squeeze and Excitation Networks [37] reduced the top-5 error on ImageNet to 2.3% by proposing a channel-wise feature recalibration block that scales input feature maps. This block is made of a global average pooling layer, two fully-connected layers separated by a ReLU and a sigmoid activation function.

1.2.3 Object Detection Architectures

As shown in Fig. 1.2, the object detection task is about finding rectangular regions enclosing objects of interest. Typically, object detection network architectures use generic networks such as those introduced in sec. 1.2.2 (minus the final layers) to act as backbones and stack task-specific layers on top of them. Backbones are usually pretrained on ImageNet, meaning that their weights are initialized with the final weights of a network trained for object classification on ImageNet. It is a standard practice that results in the faster convergence of the training process, according to [28]. The task-specific part of detection networks is either performed in one or two stages. Approaches belonging to both of these groups are detailed below.

Two-Stage Detectors

The two steps of two-stage approaches consist of i) *region proposal* and ii) *region classification*. In region proposal, regions which possibly contain any object, independently of their class (class-agnostic), are proposed. Then, for each region, prediction scores are computed for each class or object. In most architectures, bounding box coordinates are also regressed in this second step. Most of the popular two-stages approaches are described below:

Regions with CNN features (R-CNN) [23] To create an object detector, R-CNN (shown in Fig. 1.7) combines two popular ideas at the time: Selective Search [87] and AlexNet [44]. Selective Search is a region proposal algorithm that uses color, texture, size and shape similarities to generate proposals. In R-CNN, all proposed regions are cropped and then warped (resized) to an image of square dimensions before being passed to the pretrained CNN (AlexNet), which extracts features. The classifier from AlexNet is removed and replaced by class-specific linear SVMs classifiers to predict the object category and linear regressors to predict offsets of the bounding box. This architecture is limited, mainly because all region proposals need to be cropped and evaluated by the CNN. Training and inference time are thus quite slow. For instance, it was reported that with a VGG16 backbone, the detection time was 47s/image on a GPU [22].

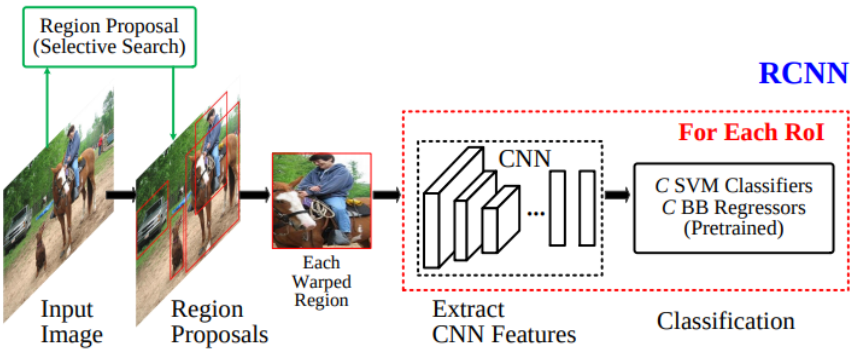


Figure 1.7 – Overview of R-CNN. Figure from [51].

Fast R-CNN [22] Fast R-CNN (shown in Fig. 1.8) improves training and inference speeds (9x faster for training and 213x faster at test-time). They indroduced Region of Interest (RoI) pooling, which replaces the image warping of R-CNN at the image level. Instead, RoI pooling resizes regions at the feature level, just before the classifier, to get fixed-length feature vectors. Furthermore, they replaced the SVM classifiers and bounding box regressors by fully-connected layers that predict softmax probabilities for classification and offsets for bounding box regression.

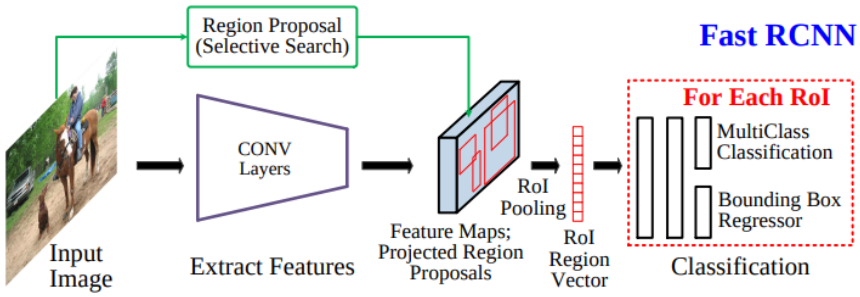


Figure 1.8 – Overview of Fast R-CNN. Figure from [51].

Faster R-CNN [70] In Faster R-CNN (shown in Fig. 1.9 and 1.10), they replace the time-consuming Selective Search algorithm by a Region Proposal Network (RPN), which allows a running rate of 5 FPS (10x faster than Fast R-CNN). The RPN shares the same backbone as the classifier and predicts objectness scores (class-agnostic, predicts foreground or background) and offsets (regression for translation and size) for bounding boxes of predefined scales and aspect ratios (anchors). As shown in Fig. 1.9, predictions are made for the k different anchors at the center of each feature maps. In the original implementation, they used 3 different anchors sizes (128, 256 and 512 pixels) and 3 aspect ratios (1:2, 1:1 and 2:1) for a total of 9 anchors.

To train the complete network, the RPN is trained in alternance with the fully connected layers used for classification and regression. During the RPN training, only 256 anchors

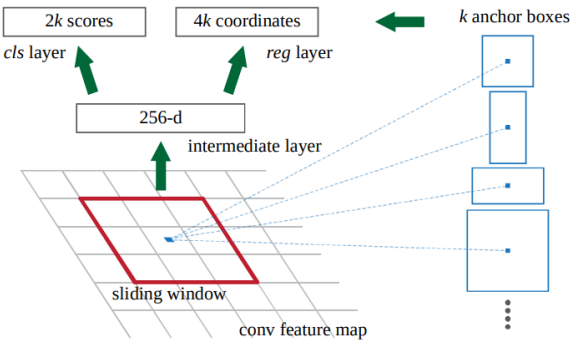


Figure 1.9 – Region Proposal Network for Faster R-CNN [70]. Multiple anchor boxes of predefined sizes and aspect ratios are used as references to predict objectness scores and offsets for each cell of convolutional feature maps. For each of the k anchors used, there are 2 outputs for objectness (foreground and background) and 4 outputs for the offsets relative to the bounding box proposals (translation in x, translation in y, width ratio and height ratio).

are sampled in a 1:1 ratio between positive and negative examples to avoid a bias towards negative examples. At test time, the predictions with the best objectness scores are used as proposals.

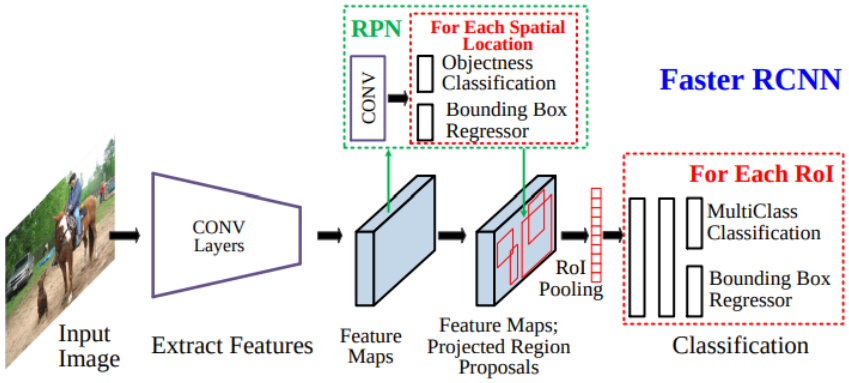


Figure 1.10 – Overview of Faster R-CNN. Figure from [51].

Mask R-CNN [29] Mask R-CNN (shown in Fig. 1.11) extends Faster R-CNN by adding a parallel branch in which segmentation masks are predicted for each region of interest. The RoI pooling operation in Faster R-CNN usually extracts feature maps of 7x7 for each region of interest. It however causes pixelwise misalignments by rounding feature map locations. In Mask R-CNN, they instead propose a quantization-free operation named RoI Align that uses bilinear interpolation to preserve spatial locations. The overhead created by the mask prediction branch is relatively small. The network can run at a rate of 5 FPS while obtaining state-of-the-art performances for detection and segmentation.

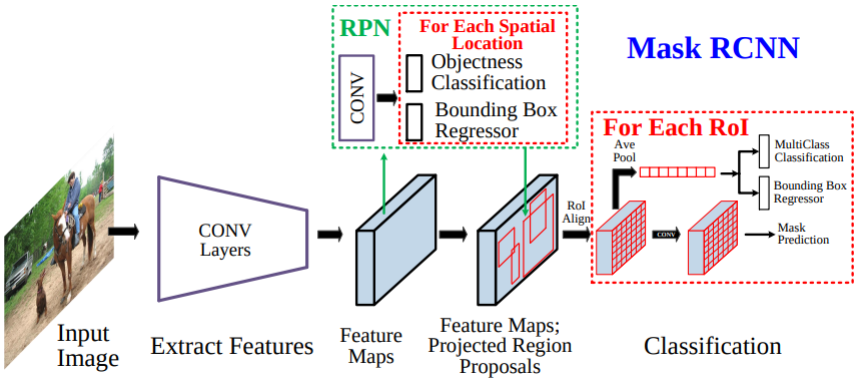


Figure 1.11 – Overview of Mask R-CNN. Figure from [51].

Single-Stage Detectors

Single stage detectors are optimized end-to-end and bypass the region proposal stage required by two-stage architectures. They are thus usually less computationally expensive, being fully convolutional. Some of the most popular approaches are described below.

You Only Look Once (YOLO) V1 [67], V2 [68] and V3 [69] YOLO (shown in Fig. 1.12) was the first proposed single-stage detector. It is significantly faster than two-stage detectors (45 FPS or 155 FPS for the smaller version) at the cost of slightly worse performances (10% drop with respect to Faster R-CNN for the mAP metric on Pascal VOC datasets). In YOLO, the image is divided in a SxS grid, for which each cell predicts bounding boxes (location and size), objectness confidence (class-agnostic) and class probabilities. Since the grid is coarse, the network struggles to detect small objects.

In V2, they replaced the backbone of V1 (custom version of GoogLeNet) by their own backbone called DarkNet19. They added Batch Normalization, removed fully-connected layers, trained their network with images of different scales and used pre-defined anchor boxes like Faster R-CNN to improve their performance. It outperforms Faster R-CNN while running at 40 FPS.

In V3, the softmax classifier is removed and replaced by logistic ones to allow multi-label classification. This way, classes do not compete between themselves. They also replaced their backbone network with a new one called Darknet-53. Compared to Darknet-19, it has more layers and they added residual (skip) connections. Additionally, inspired from Feature Pyramid Networks (FPN) [49], they predict bounding boxes at three different scales in the feature maps instead of only a single one to improve their accuracy on small objects. They use three predefined anchor boxes at each scale for a total of nine anchors.

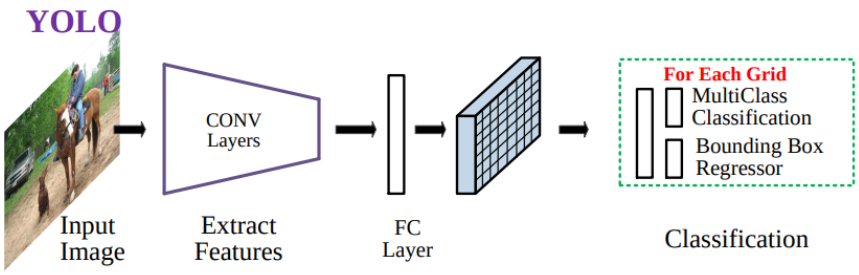


Figure 1.12 – Overview of YOLO. Figure from [51].

Single Shot Detector (SSD) [52] SSD (shown in Fig. 1.13) is a fast fully-convolutional architecture with a processing rate of 59 FPS on images of resolution 300 x 300 pixels and is as accurate as Faster R-CNN. It adds convolution layers on top of the VGG16 backbone and predicts classes and bounding box offsets on feature maps of different scales.

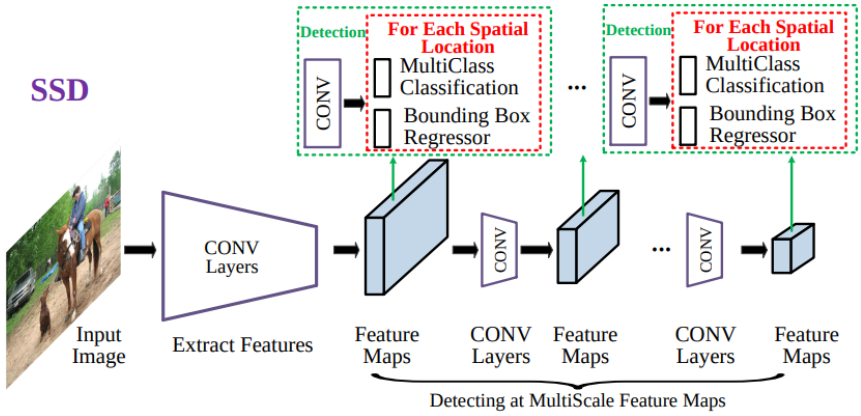
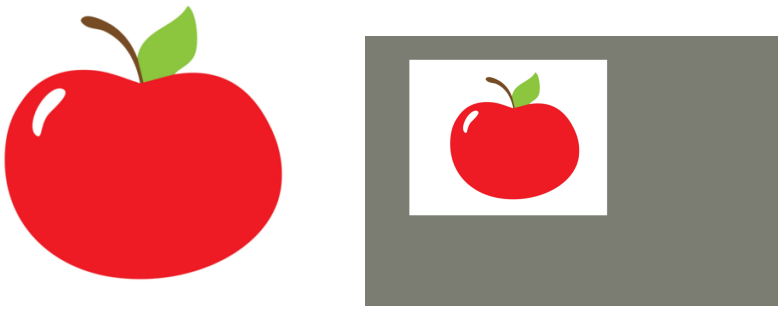


Figure 1.13 – Overview of SSD. Figure from [51].

To improve the performance of the network on small objects, they use a data augmentation trick that they call image expansion or "zoom out" that improves the mAP metric of their network by 2-3% on many datasets. In this technique, they place the original image on a new image that has up to 16x of the original resolution before resizing or cropping. As shown in Fig. 1.14, it thus creates more small objects for training.



(a) Normal image of an apple (b) Zoom out image of an apple

Figure 1.14 – Zoom out data augmentation strategy example. The original image of an apple (a) is placed in a bigger image filled with mean values and is then rescaled to the original resolution (b), thus creating a smaller apple.

RetinaNet [50] A specific problem of object detection methods is the imbalance between the number of positive (object) and negative (background) examples. To tackle this problem, they introduced the Focal loss in RetinaNet (shown in Fig. 1.15), in which the classical cross-entropy loss is modified so that easy negatives have less impact on the loss than harder ones. For the architecture, they used a Feature Pyramid Network (FPN) backbone [49] and stacked two different subnetworks on top of it for bounding box classification and regression. In FPN, predictions are made at multiple resolutions from feature maps coming from the different levels of the CNN. It is a faster way to replicate image pyramids which allows networks to detect objects at various scales. It especially improves detection performances on smaller objects, since most networks predict from the last layer, which has low spatial resolution.

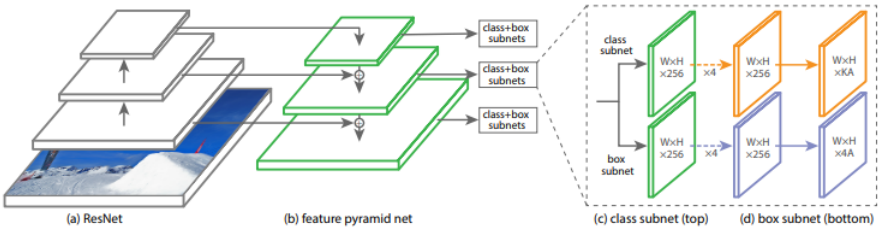


Figure 1.15 – Overview of RetinaNet. Figure from [50].

1.3 Deep Learning Limitations

Deep Convolution Neural Networks are trained to accomplish complex functions directly from labelled data. They can be limited by the data (cost of acquisition and labeling, biases, etc.) or the networks themselves (need of powerful computing ressources, lack of interpretability, etc.). Some of these limitations and different ways of overcoming them are presented in this

section.

1.3.1 Dataset Acquisition

For generalization purposes, CNNs require big annotated datasets. However, obtaining annotated data for certain domains or tasks can be a tedious process. For example, [33] reported that the acquisition and annotation process for an object detection dataset of 64 objects and 1400 images took around 200 hours. Scaling this process to generate datasets with millions of images is therefore close to impossible. Fortunately, different techniques can decrease requirements for annotated images. In Fig. 1.16, different research areas related to this topic are listed. They are further described in this section.

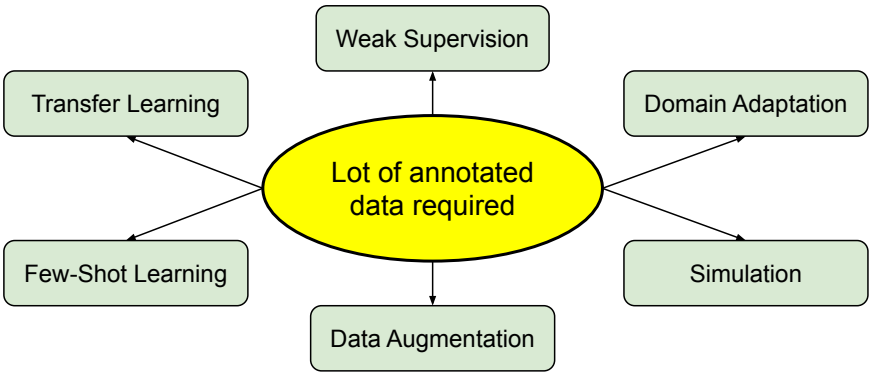


Figure 1.16 – Research areas related to decreasing annotated data requirements

Transfer Learning One of the most popular transfer learning technique is pretraining. According to [28], pretraining is used to make CNNs converge faster during training. Features from networks trained on big datasets can be transferred to networks for different and smaller datasets of related tasks. For vision-related tasks such as object classification, detection and segmentation, starting with weights trained on a big dataset such as ImageNet provide a good initialization since many learnt features are generic (especially the low-level features at the beginning of the network). The network thus requires much less training data to converge to a good solution for a new related task. In all the proposed approaches contained in this thesis, pretraining was used.

Weak Supervision The concept of weak supervision is to use a simpler supervision level to accomplish a more complex task. For instance, networks trained for classification with image-level labels can predict the location (bounding boxes) and segmentation masks of objects. By combining Fully Convolutional Networks (FCNs) [53] with different aggregation strategies such as global average pooling [94], min-max pooling [13] or peak

stimulation [95], these methods are able to train a classifier and retrieve scores for each object at every spatial locations in the feature maps. This principle is at the core of our proposed approach in Chapter 3.

Domain Adaptation When a learning-based approach is trained on data that is quite different from the test data, there is what is called a *domain gap/shift*. This discrepancy between source and target datasets may cause a significant performance drop. For example, networks trained with synthetic data, such as our approaches presented in Chapters 3 and 4, often exhibit poor performances when tested on real data. To reduce the gap, domain adaptation methods either try to modify the input directly [7, 96] or align the feature space of both domains [19, 54]. Adversarial methods based on Generative Adversarial Networks (GANs) [25] and Domain Adversarial Networks (DANNs) [19] have achieved good success of decreasing the performance gap. However, most of these methods require some data in the target domain (labelled or not). A method specific to synthetic data called Domain Randomization [85, 86], which is introduced in the next paragraph on "Simulation", has shown great success to improve the generalization performance of networks trained on synthetic images without requiring images from the target domain during training.

Simulation Synthetic images generated in simulation allow to increase the size of datasets without the need for additional hand labeling. Despite the discrepency between synthetic and real data (reality gap), it has been shown that combining synthetic images with real images increase object detection performances. For instance, [14] proposed a simple approach in which object instances are pasted on random backgrounds. They also used different data augmentation and blending techniques to help networks ignore artifacts created by the pasting operation. Also, domain randomization [85, 86] has recently gained popularity. The idea is to generate enough variations in simulation parameters (lighting, textures, object placement, noise, etc.), such as what is shown in Fig. 1.17, so that real images are seen as simple variations of the simulation images. Domain randomization is especially interesting because it outperforms models trained on realistic synthetic images [86]. It also has been shown that synthetic images generated in a context similar to the test images (settings in which the objects are present) can improve detection results [35, 64].

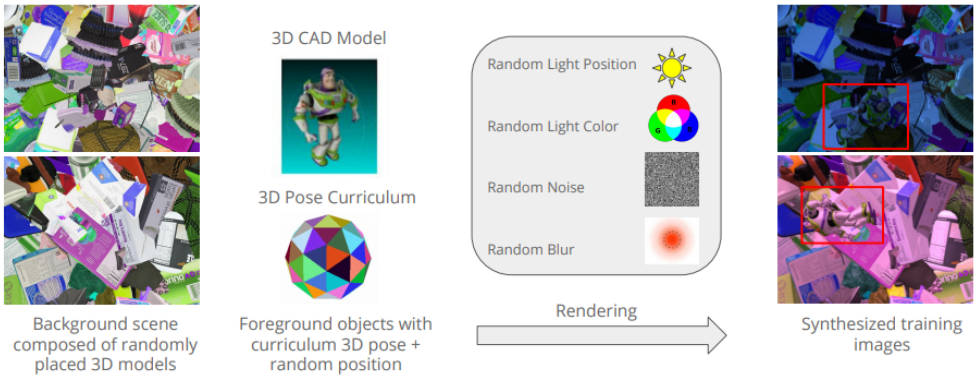


Figure 1.17 – Synthetic data generation pipeline of [33]. 3D models are rendered in different poses on top of different synthetic backgrounds and are augmented with random parameters (light position and color, noise and blur) to generate the annotated synthetic dataset for object detection.

Data Augmentation It is possible to decrease overfitting by increasing the number of training examples with data augmentation techniques [77]. Data augmentation includes, but is not limited to, basic image manipulations such as kernel filters (sharpen, blur), geometric transformations (translation, rotation, cropping, flipping) and color space transformations (lighting, noise, contrast, white balance, color jittering). It is also possible to learn data augmentation policies [97]. A special focus was given to data augmentation in Chapter 4 to maximize the number of objects seen by the network during training.

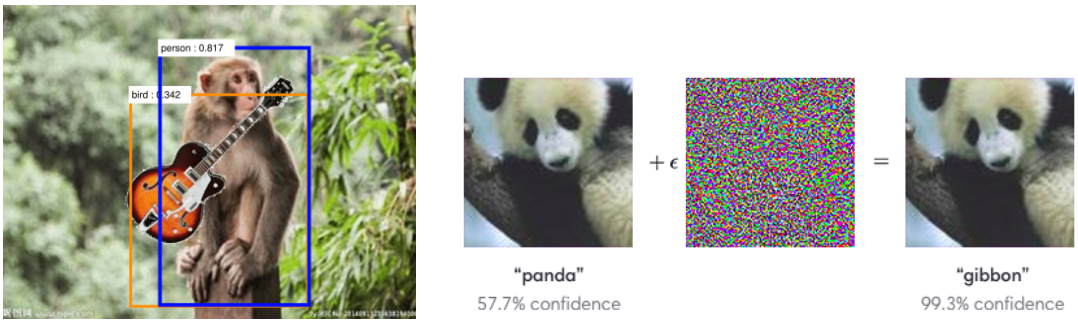
Few-Shot Learning Deep Neural Networks have a lot of parameters and require a lot of annotated examples in training to generalize well to new data. The aim of few-shot learning is to achieve the same, but only with few annotated examples. Since training with few labelled samples usually lead to overfitting, most few-shot learning methods are based on meta-learning, which is a concept of "learning to learn". A classic example of a meta-learning approach for few-shot learning is a classifier that is trained on series of training tasks with few examples of different classes. The network therefore *learns to learn* from few labelled data. Meta-learning approaches can be separated in multiple categories:

1. Based on metric learning. These methods learn to generate embeddings to distinguish between unknown classes [36, 43, 80, 82, 88].
2. Fast adaptation. These methods optimize the parameters so that they can quickly converge given new data [16, 62, 66].
3. Parameter prediction. These methods adapt network weights specifically for the classes given in input [2, 21, 41, 65].

Different approaches based on meta-learning have been proposed for object detection [15, 41, 75, 93]. However, it has been demonstrated that fine-tuning the last layer of a network outperforms most recent approaches [92]. Object tracking can also be framed as one-shot detection, with trackers initialized with a previous example of a target object. For instance, in approaches based on Siamese Region Proposal Networks [48, 91], the weights of some kernels are adjusted by the network that processes the object to track. As opposed to "fast adaptation" methods, these approaches do not need to be retrained for new data. Our proposed approach in Chapter 4 is based on this principle to detect objects that have not been seen during training.

1.3.2 Dataset Biases

The size of datasets is not the only limitation related to the data itself. However big they are, datasets are generally biased and CNNs are likely to learn their biases. For instance, in the real world, a guitar player is usually a person and most datasets are probably biased this way. In Fig. 1.18a, a guitar placed in front of a monkey is enough to change the class prediction from "monkey" to "person". The "guitar" is also detected as a "bird", most likely due to the background. It is a good example of how biases can influence predictions of object detectors. Also, very small perturbations have been shown to modify network predictions [26, 61], as shown in Fig. 1.18b. These perturbations, called adversarial examples, show not only that networks are vulnerable to different kind of attacks [38, 63], but that even though networks have good generalization performances, they do not perform as we expect for all possible inputs. Consequently, they are highly dependent of their training data.



(a) Context (image from [90]) (b) Adversarial examples (image from [26])

Figure 1.18 – In (a) is shown an extreme example where a dataset may be biased towards different contexts. The monkey, occluded by the guitar, is classified as a person and the guitar, probably because of the forest in background, is detected as a bird by the network. In (b), a small perturbation added to the input image modifies the network prediction.

Also, in scenarios where data is more difficult to acquire, obtaining data from videos can be an easier way to rapidly expand the size of a dataset. The individual frames are therefore highly correlated. An example of a dataset acquired in this manner is Occluded Linemod [3], which

we used to evaluate all of our approaches. The limitation of this dataset is not necessarily the dataset itself, but how it is used. Many state-of-the-art approaches on this dataset [4, 84, 89] randomly sample the video sequence to generate a training set. Hence, the reported results are likely biased positively, since these approaches are tested on the same image sequence that they were trained on.

1.3.3 Network-related Limitations

Networks are not only limited by their training data, but also by the network architecture themselves. Some of these architecture-related limitations are briefly described here:

Hardly interpretable CNNs solve complex non-linear mathematical functions with multiple convolution layers stacked on top of each other. They are often seen as "black boxes", because interpreting the weights of networks is quite difficult. In particular, it becomes difficult for a human to understand how a particular decision was reached by a network.

Rigidity Once a network is trained, the weights are fixed. They do not learn from their mistakes at test time and additional knowledge can not be added unless the network is fine-tuned or retrained. This process takes time: networks may take few hours to few weeks to be trained. This compares badly to traditional engineered features methods, which can work right out of the box for new objects.

Specialized hardware GPUs are necessary for training and preferable if fast inference time is important. Also, dedicated equipment such as servers with multiple nodes can be useful, especially for hyperparameter optimization.

Some solutions have been proposed for these limitations. For instance, to show the inner workings of CNNs, some work focus on visualization techniques [76, 78]. They show which parts of the input are activated and responsible for a prediction. These visualizations are useful to understand networks at a high-level, but how to debug them (eg. modify some filter weights to solve a bad prediction made by the network on a certain input without impacting the output on other inputs) is still not quite understood.

In this chapter, we reviewed background material related to object detection methods for robotic grasping, from traditional rule-based methods to recent deep learning approaches. We have also seen the data and network-related limitations of deep learning approaches. In the following chapters, we present our different approaches in which we try to tackle some of these limitations for object detection in the context of robotic grasping.

Chapter 2

Deep Object Ranking for Template Matching

2.1 Résumé

Dans le domaine de la robotique industrielle, les applications de « pick-and-place », qui consistent à saisir des objets afin de les déplacer à un nouvel endroit, sont très populaires. Pour localiser les objets à saisir avec suffisamment de précision, les approches de type « template matching » sont généralement utilisées. Cependant, dans les cas où les systèmes doivent composer avec une base de données contenant plusieurs objets, ces approches peuvent devenir dispendieuses en temps de calcul en plus d'être dépendantes de seuils difficiles à configurer. Dans cet article, nous proposons d'utiliser des réseaux de neurones en pré-traitement dans le but d'améliorer les performances de méthodes traditionnelles de « template matching ». Les résultats sur deux jeux de données démontrent que notre approche est compétitive par rapport à une approche par force brute pour retrouver les objets en plus d'être plus rapide. Ceci ouvre donc la possibilité d'utiliser des approches de « template matching » en industrie avec un nombre d'objets plus élevé.

2.2 Abstract

Pick-and-place, which consists of picking objects and placing them somewhere else, is an important task in robotic manipulation. In industry, template-matching approaches are often used to provide the level of precision required to locate an object to be picked. However, if a robotic workstation is to handle numerous objects, brute-force template-matching becomes expensive, and is subject to notoriously hard-to-tune thresholds. In this paper, we explore the use of Deep Learning methods to speed up traditional methods such as template matching. In particular, we employed a Single Shot Detection (SSD) [20] and a Residual Network (ResNet) [9] for object detection and classification. Classification scores allow the re-ranking

of objects so that template matching is performed in order of likelihood. Tests on a dataset containing ten industrial objects demonstrated the validity of our approach, by getting an average ranking of 1.37 for the object of interest. Moreover, we tested our approach on the standard Pose dataset [30] which contains 15 objects and got an average ranking of 1.99. Because SSD and ResNet operates essentially in constant time in a Graphics Processor Unit, our approach is able to reach near-constant time execution. We also compared the F_1 scores of `LINE-2D` [12], a state-of-the-art template matching method, using different strategies (including our own). The results show that our method is competitive to a brute-force template matching approach. Coupled with near-constant time execution, it therefore opens up the possibility for performing template matching for databases containing hundreds of objects.

2.3 Introduction

The automation of simple industrial operations, particularly for the small scale, fast turnaround robotics equipment, is seen as a key concept of the fourth industrial revolution (often dubbed *Industry 4.0*). A desirable characteristic for its acceptance would be the ability for a non-technical person to program advanced robotic equipment, such as robotic arms. The *Baxter* robot, from *Rethink Robotics*, exemplifies this concept, by being able to perform pick-and-place operations using intuitive programming methods. In this system, training by examples replaces the traditional and more tedious computer programming by experts.

In this paper, we explore the idea of improving both the robustness and speed of a key and traditional component used in pick-and-place solutions, namely *template matching*. We aim at exploiting recent developments in Deep Learning on object detection and classification to first detect, then sort, putative objects before performing template matching. These networks are trained by employing databases of images collected in a simple manner. This way, we stay within the philosophy of training by example. As our approach is fairly conservative (the final object localization is still performed via template matching), we believe that it might speed up the acceptance of Deep Learning approaches in the industry.

Deep neural networks, particularly the convolutional networks, have established themselves as clear winners in many classification tasks. Moreover, with the use of GPU (Graphics Processor Units), they have shown to perform classification over a thousand object categories in real-time. This speed is possible because *a)* GPUs provide massive parallel computing capabilities, up to several thousands of cores and *b)* the features are extracted only once by the neural network, and are available to all object classifiers, since the object classification is performed in the fully-connected layers at the top of the network architecture. By contrast, parallelizing template matching libraries is tedious, as it requires significant hand-crafting and tuning. Moreover, current template-matching approaches do not share the feature extraction pipeline across templates, making them inherently less efficient at exploiting the massive parallelism

of GPUs.

Template matching [8] has been used for a long time in machine vision, and has now been widely accepted in industry. It has proven to perform well in retrieving the pose of an already-identified object. However, it comes with a number of known flaws. For instance, the matching speed is dependent on the number of templates the method is trying to match. Furthermore, the final decision about whether an object has been well detected/located is based on a threshold on the matching score. Thresholds are notoriously difficult to select properly, as there is an intrinsic compromise to be made between a low threshold (which results in many false detections) and a high threshold (which has less false detections but will also reject good detections).

We propose a generic template matching pre-processing step that takes advantage of the success and properties of Deep Learning for object detection and recognition. A diagram of our approach is shown in Fig. 2.1. It consists in detecting, classifying and ranking the seen objects before using template matching. As such, our approach is completely agnostic to the subsequent step. This *cascading* philosophy is not novel. In fact, it is a common approach in vision-related tasks, in order to accelerate processing. For instance, the Viola-Jones real-time detection technique [31] uses a number of progressively more discriminative (and expensive) classifiers. Neural networks have also been employed in cascade for object understanding, notably a two-staged approach for grasp location detection in range images [18]. Deep Learning approaches might be used for initializing model-matching, such as with the tracking of body parts, as they are more robust due to their larger basin of attraction [7]. In some sense, our approach uses the same philosophy, where the coarse part (object detection and classification) is performed with a Convolutional Neural Network, while the fine-tuning of object pose estimation is completed via template matching.

A significant advantage of our proposed approach is that detection and classification modules, which are often expensive, are essentially executed in constant time. Indeed, one forward-pass of a typical deep neural network can recognize amongst a thousand objects in less than 80 *ms* [17]. On the other hand, if object recognition is to be done by template matching, one needs to run it until a sufficiently-high scoring template has been found. In this case, the average number of templates that will be tried by this template matching method should be, on average, around half of the total number of templates. As we will demonstrate, by pre-ranking templates by confidence of our object classifier before matching begins, this number tends to be significantly less. It can thus allow for fast matching, even if there is a large number of templates, as long as classification precision is sufficiently high.

Another positive effect of our approach is an increase in robustness for object classification. In some template-matching approaches, any score higher than the user-selected threshold results in a positive match and the termination of the algorithm. By using a low threshold on template

matching scores (higher recall), it results in a high number of bad detections (lower precision), as the first template tried will return a positive match. In another kind of approach, every template is processed, and the template which scored the best is selected. By pre-ranking templates in a first step, our method improves the performance of the "first-match" approach, and get similar results to the "best-match" approach, while having to process significantly less templates.

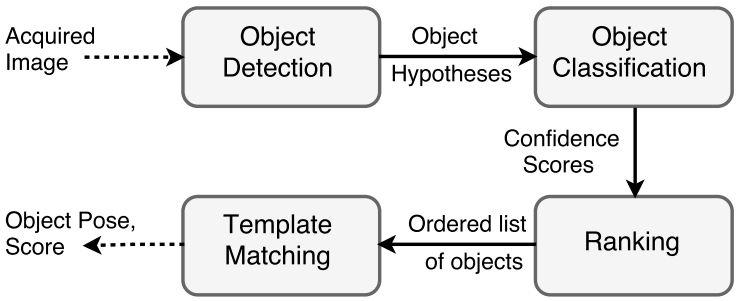


Figure 2.1 – Block diagram of our proposed method. Object detection and classification are performed by Deep Neural Networks, providing robustness and speed. Objects in the database are sorted by their classification scores before being passed to the template matching module. Objects are processed one after the other until a template results in a score higher than a user-defined threshold, upon which the algorithm terminates.

2.4 Related Work

Some approaches have been used to reduce the computational load of template matching methods, which can be heavy, especially for those that searches for the complete set of possible transformations [16, 22, 28]. In [23], they used a convolutional neural network to predict the "matchability" of templates across different transformation of the input image. They then select regions around local maximums from the computed matchability maps as templates for any selected template matching method. In [2], they employed a Support Vector Regression to find a function which is used to generate template candidates at runtime, but strictly applied to visual tracking.

Object detection methods based on keypoint descriptors work well on textured objects [3, 21]. However, by looking at specific visual patterns on surfaces, they do not perform well on untextured objects by nature, and they tend to require heavy calculations [1]. On the opposite, template matching methods work well on objects with little visual texture [12, 15, 24], provided that they are located on a distinctive background. Since a significant number of objects in an industrial context are devoid of textures (such as metallic parts that are freshly machined), template matching tends to be more appropriate for those environments.

`LINEMOD` [12] is among the state-of-the-art methods for template matching on textureless

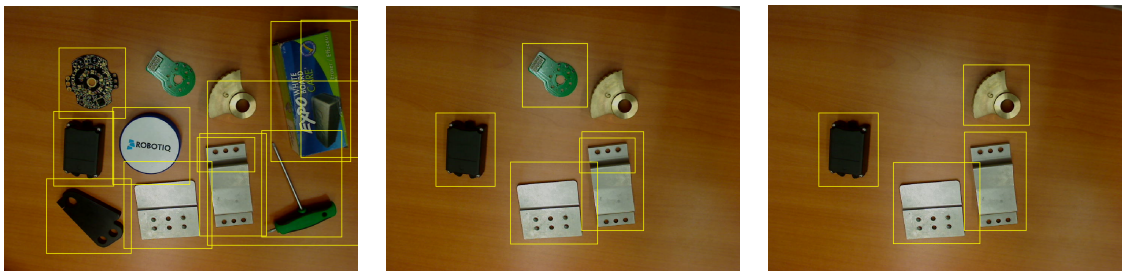
objects. It uses quantized color gradients for the 2D version (`LINE-2D`) along with surface normals for the 3D version. They use some optimizations, such as a look-up table for the color gradients and linearization of gradient response maps, to increase the speed of their method. To detect objects under varying poses, it requires a training phase in which a significantly large collection of templates needs to be acquired from different viewpoints. In practice, full 3D models are needed to generate these viewpoints. DTT-3D [24] improved the performance and speed of `LINEMOD`. In their approach, they train a SVM with positive and negative examples to learn the discriminative regions of templates, and use them only in the matching process. They also divide the templates of each objects into different clusters using their own similarity measure and the clustering method of [13], which represents a group of similar templates with what they refer as a "cluster template". On the other hand, Hashmod [15] uses a hash function on concatenated `LINEMOD` descriptors (color gradient and surface normals) to match templates. Following a sliding window approach, they compute hashed descriptor at each image locations and if the descriptor is close enough to a descriptor in the hash table (which contains the hashed descriptors for each object under a certain pose), it is considered a match. [32] followed a similar approach, but instead of using a hash function, they trained a CNN to learn descriptors for varying poses of multiple objects. Matching was performed with a Nearest Neighbor search. As opposed to the previously mentioned methods, [5] used only monocular grayscale images to retrieve the 3D pose of known objects (they still had a 3D model of the object), as methods based on depth sensors tend to fail for metallic objects due to their specular surfaces. They based their pose estimation method on the 2D reprojection of control points located on the different and manually labeled parts of the 3D models. From the 3D-2D correspondences of all control points of the different parts, they solve the PnP problem to estimate the 3D pose, which results in a method more robust to occlusions.

2.5 Overview of the proposed method

Fig. 2.1 shows a workflow of template matching including our proposed pre-processing steps, which detect and rank objects before further processing. As it can be observed, our method is highly modular. As such, it allows a drop-in replacement of any module (e.g. template matcher) if any better method becomes available. In the rest of this Section, we describe the implementation details of each of the modules.

2.5.1 Object Detection

The first step in our method is to perform object hypotheses detection. We have used the state-of-the-art *Single Shot Multi Box Detector* (SSD) [20] trained on the Microsoft COCO dataset [19], which contains more than 2 million instances of the 80 object categories. SSD is a convolutional neural network framework that can take as input images resized to 300×300 (SSD300) or 500×500 (SSD500) pixels, and outputs bounding boxes and classification scores



(a) (b) (c)

Figure 2.2 – Image sequence showing object hypotheses retrieved from our object detection module. One can notice that as the clutter in the workspace diminishes, objects previously undetected by SSD can now be detected.

of every class. On proper hardware (GPU), it can run in real-time (58 FPS for SSD300) or close to real time (23 FPS for SSD500). Objects are considered as detected when the score of a bounding box is higher than a user-specified threshold τ_{SSD}. In our experiments, we used the SSD500 version with a low threshold $\tau_{SSD} = (0.08)$. We preferred SSD500 over SSD300 as it outputs more bounding boxes, while still being sufficiently fast. An example of object detection on our object database with SSD is shown in Fig. 2.2. We can see in Fig. 2.2a that some objects are undetected. It sometimes happens that an object will be considered as part of the background in the presence of some clutter, i.e. when multiple objects are near each other. This effect tends to decrease when near objects are removed, as shown in Fig. 2.2b and Fig. 2.2c, where we can see that both the green and the gold objects were finally detected, once there was fewer objects around them. In a robotics grasping application where the goal is to empty a work cell from all objects, this is usually not a significant issue. Indeed, objects around the undetected one would eventually all be grasped. Once the undetected objects are the only objects remaining, the detection threshold can be decreased to get more object hypotheses.

2.5.2 Object Classification

To classify objects, we used a Residual Network (ResNet) [9, 11]. It has been recently demonstrated as a top contender for ImageNet object classification. For instance, [9] won the ILSVRC 2015 classification task. The key idea of ResNet is that residual mappings are easier to optimize than absolute ones. To take advantage of this fact, ResNet adds skip connections that bypass several convolutional layers using identity mappings to form shortcuts in the network, called residual blocks. With a combination of Batch Normalization (BN) [14] and MSR initialization [10], stacking residual blocks significantly improves the training efficiency by reducing the vanishing gradient degradation. This allows learning very deep networks far more easier than it is without residual connections.

Model Description

The ResNet architecture and the residual blocks are shown in Fig. 2.3, where we can formulate each residual block as follows:

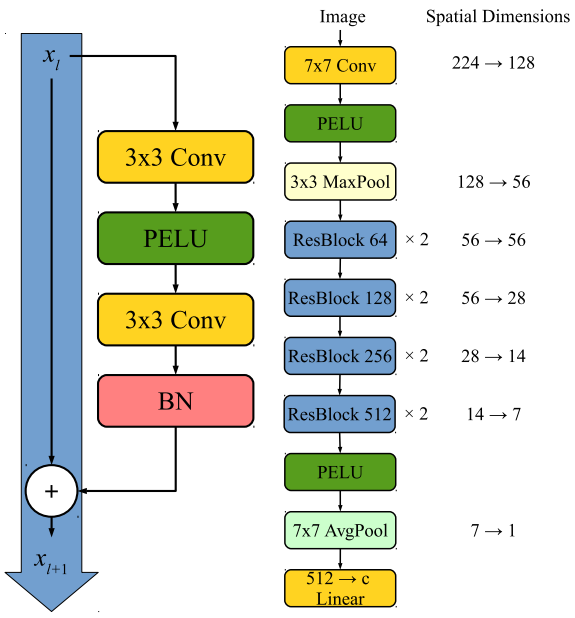
$$x_{l+1} = x_l + \mathcal{F}(x_l, W_l)\,, \tag{2.1}$$

where x_l and x_{l+1} are the input and output of the l-th residual block and $\mathcal{F}$ is the residual mapping. As seen in Fig. 2.3a, we opted for a Conv-PELU-Conv-BN residual mapping, where PELU stands for Parametric Exponential Linear Unit, an adaptive activation function that showed high performance on several image classification task over other alternatives [25, 29].

The overall 18-layers network, which has about 11.7 M parameters to be learned, is shown in Fig. 2.3b. It takes as input a 224×224 color image, and using a series of various transformations, outputs a c-dimensional vector of class conditional probabilities. The first convolutional layer contains 64 7×7 filters, which are convolved with a 2×2 stride and 3×3 zero-padding. We will denote such convolutional layer as Conv(64, 7×7, 2×2, 3×3). PELU is then applied on the resulting feature maps of the first layer, followed by a max pooling layer MaxPool(3×3, 2×2, 1×1). The input then passes through four twice-replicated residual blocks, containing respectively 64, 128, 256 and 512 filters in each of their convolutional layers. The network performs a final PELU activation, followed by a global average pooling AvgPool(7×7, 1×1) and linear layer $512 \rightarrow c$, where c represents the number of classes. When the network doubles the residual block number of filters, the input spatial dimensionality is reduced by half. This particularity allows the network to gradually pool spatial information into its feature maps and extract higher-level information. For instance, between ResBlock 64 and ResBlock 128, the input spatial dimensionality is reduced from 56×56 to 28×28 pixels. However, this poses a problem for the first ResBlock 128, as we cannot apply an identity skip connection because the input and output spatial dimensionality are different. In this case, we use as skip connection a convolutional layer with 1×1 filters with a stride of 2×2, currently known as a type B connection [11].

Training Details

In spite of the recent experiments showing that fine-tuning an ImageNet [6] pre-trained network gives good results [26], we instead opted to train our network from scratch. We relied heavily on data augmentation techniques for improving generalization performances. This powerful technique allows generating new images from existing datasets to artificially grow their sizes, and thus help the network converge at a better local optimum. During training, prior to inputting the image to the network, we applied the five following random transformation: color jittering (contrast, lighting and saturation are changed by a small amount), lighting Noise (a PCA-based noise in RGB space from [17]), color standardization (we subtracted the mean and divided by the standard deviation of each RGB channel to get a mean of 0 and



(a) Residual block (b) Residual network

Figure 2.3 – Residual block and network used in this paper

unit-variance), horizontal flip (images are randomly flipped horizontally), rotation (images are randomly rotated from 0 to 360 degrees). At test time, we simply did color standardization. All transformations were performed on the fly during training, and due to the randomness of the process, the network had effectively seen a new image at each iteration, even though our dataset only contained 50 images per object (see section 2.6.1 for more information about the dataset). For training the network, we used the hyper-parameters and the Torch implementation of [29]. Note that this number of images is much less than in the standard ImageNet database, which has between 700-1300 images per class [6].

2.5.3 Object Ranking

This simple step takes the confidence scores returned by the classification step as input, creating a list ordered by their classification scores. The most likely objects to appear in the image, according to our classifier, are therefore placed at the beginning of the list. The whole algorithm is summarized in Algorithm 1.

2.5.4 Template Matching

To match object templates, we employed the `LINE-2D` [12] algorithm, which uses the orientation of gradients to perform matching. `LINE-2D` is considered as one of the state-of-the-art template matching method on monocular color images. We used its default implementation in the OpenCV library [4], with 2 pyramid levels and 63 features. In an offline phase, we acquired, with a fixed and uncalibrated webcam, multiple templates per object under varying

Algorithm 1 Object Ranking

```
1: procedure RankHypotheses
2:     ClassScores[1..class] = [0, 0 .. 0]
3:     BoundingBoxes ← SSD(inputImage)
4:     CroppedImages ← crop(BoundingBoxes)
5:     For i in CroppedImages :
6:         scores[1..class] ← ResNet(i)
7:         For j in 1..class :
8:             if scores[j] > ClassScores[j] :
9:                 ClassScores[j] ← scores[j]
10:    ClassRanking ← sort(ClassScores)
```

poses instead of generating them from CAD models. We also did perform some data augmentation to increase the number of templates per object. In our case, we limited ourselves to scaling. For each template in our training dataset, we added a smaller (90%) and a bigger (110%) version. This was deemed sufficient, as our experimental setup mimicked a robotic work cell, where objects are assumed to be placed on a flat surface at a fixed, known distance from the camera.

At runtime, objects are matched starting with the better ranked objects coming from our pre-processing. If no template from an object has a higher score than a given threshold, the next object on the list is tried. This process is repeated until the matching score of a template is higher than the threshold τ_{score}. In this case, this template is then considered as matched and the template matching algorithm is stopped. It is important to note that we used less templates (few dozens to few hundreds) than the original implementation of [12] (few thousands). Also, we used larger image resolutions (980x720 instead of 640x480 pixels).

2.6 Experiments

In this Section, we describe the experiments we performed to test our method. In particular, we detail the training and testing datasets used, motivating our choice of acquiring our own datasets. We then present results, in the form of speed up (ranking of objects) and F1-score classification results.

2.6.1 Datasets

In-house dataset

In order to validate our approach, we acquired our own training and test datasets, much in the same way that an operator would provide training examples during the configuration procedure of a robotic work cell. This stays in line with the concept that non-experts should be able to program such systems in an intuitive manner. Moreover, it complies with the situation

where there is no *a priori* knowledge about objects, such as 3D models. Another reason for gathering our own datasets is that the majority of image datasets available in computer vision contests represents natural objects with significant intra-class variation, shown over complex background. These are not representative of the low intra-class variation typical of industrial objects, which are also generally placed on surfaces with uniform backgrounds. For our image datasets creation, we selected 10 objects that could be present in typical industrial robotic grasping applications. Care was taken to select them so that the majority had low-texture and/or were made of polished metal. The latter tend to have widely-varying appearance from viewpoints, due to specular reflections.

For the acquisition of the first dataset D_{green} (used for training purposes), our setup consisted in a fixed camera placed over a table, on which lied a green-colored cardboard. The cardboard allowed to easily segment objects from the background. For each object, a total of 50 images were captured, while being manually placed in random poses (translation and in-plane rotations). The goal was to try to capture the impact of the lighting on the appearance of the object. For instance in Fig. 2.4, one can see that the metallic part has one side slightly darker in appearance than the other (image with the green background on the left), while its surface has a more similar appearance in a different orientation (right image with the fake-wood background). For each image, ground truth images were created by our object segmentation method, which extracted image regions lying inside of the automatically generated bounding boxes. After this, a manual verification was done on all ground truth images to remove those for which the automatic segmentation failed. Specific training procedures were taken for `LINE-2D` and our ResNet, which are detailed in Sections 2.5.4 and 2.5.2, respectively.

The D_{other} dataset, used mainly for testing purposes, consisted of 20 images per object, for a total of 200 images. Like for the D_{green} dataset, objects were manually placed in random poses. The main difference is that objects were placed on a different background, seen in Fig. 2.4, namely glossy fake wood. It provided for extra challenge, as it is a different background than D_{green}, with texture and reflection of the wood. We have decided to use a different background for this other dataset for a simple reason: we wanted to estimate the impact of the change of background on the results. If we take a real scenario of an industrial user wanting to use many robotic grasping stations for multiple objects, it would be ideal for them to have a training station where they acquire the necessary information and then share this knowledge with every other grasping station. For small industrial robots, it is very likely that each grasping station will have different backgrounds. Therefore, this dataset will allow us to partially evaluate the performance of our approach for such cases.

Pose Dataset

We have also tested our method on the Pose dataset from [30]. It consists of 15 objects, for which half the images are objects that are rendered on a black background, and the other

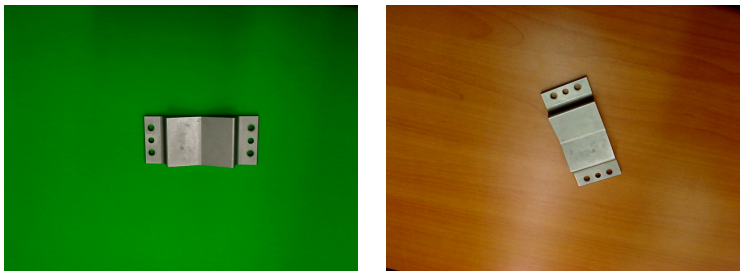


Figure 2.4 – Different backgrounds used for our experiments: D_{green} (left) and test D_{other} (right) datasets

half on a cluttered background. There are around 700 images for both backgrounds, to offer a visual sampling of each object at regular angle intervals. We used the black background images along as our training dataset, as it bears similarities with our D_{green} datasets (uniform backgrounds). Additionally, we have randomly selected around a hundred images from the images with a cluttered background to the training set of our ResNet to avoid overfitting. We have also randomly selected another hundred images from the remaining images with a cluttered background as our test dataset. Fig. 2.5 shows an overview of the objects in the dataset and an example of an object rendered on a cluttered background.

Figure 2.5 – Objects from the pose dataset [30]. The image on the left shows a number of its objects, rendered on a uniform background. On the right, an example of an object rendered on a cluttered background.

2.6.2 Results

In here, we evaluated what we consider two advantages of our approach. First, we wanted to establish the quality of our ranking, which directly translate into a speed up of matches. This is done by measuring the average rank of the ground truth object in the sorted output of our ResNet classifier. Then, we evaluate the increase in matching reliability that the ranking inherently brings, as better matches are performed on average first.

Average Ranking

We evaluated the ranking performance of our approach on the 2 datasets described in sections 2.6.1, namely our own acquired dataset D_1 and the pose dataset D_2.

We evaluated on our own dataset D_1 for two scenarios, which only differed slightly by the training dataset D_{train}. For the first experiment, the training dataset contained strictly images from the uniform green background, i.e. $D_{train} = D_{green}$. This was done in order to confirm the risk of overfitting with ResNet, if proper care is not taken when building training datasets. In the second experiment, a single image I_j per object is randomly picked from the D_{other} dataset containing the wood background, and added to our training dataset , i.e. $D_{train} = D_{green} + I_j$, with $I_j \in D_{other}$. For the first case, experiments reported an average rank of 4.05 over 10 objects on the testing dataset D_{other}, which is barely above randomness (the average rank would be 5 for an approach that uses a random order). This simply demonstrated, unsurprisingly, the overfitting of our ResNet, which has learned objects only when they were on a colored and textureless background. It then cannot generalize to objects shown on the textured background. More surprising was the fact that adding a single image from D_{other} was sufficient to nearly completely overcome this overfitting issue. Indeed, the average rank for this newly trained network was 1.37 over $D_{test} = \{D_{other} \setminus I_j\}$, very close to the results obtained directly on the training set (1.13). It therefore shows that a few examples can help to generalize to new backgrounds.

On the pose dataset D_2, we did not compute the average ranking on its training dataset consisting of the images with a black background $D_{train} = D_{black}$, since we already confirmed the risk of overfitting with dataset D_1. For the experiment on this dataset, we randomly picked 100 images $\{I_j\}$ of the cluttered background D_{other}, which is a pretty conservative number considering our results with dataset D_1, and added them to our training set, $D_{train} = D_{black} + \{I_j\}$, with $\{I_j\} \in D_{other}$. The results for the average rank of the different scenarios are recapitulated in Table 2.1.

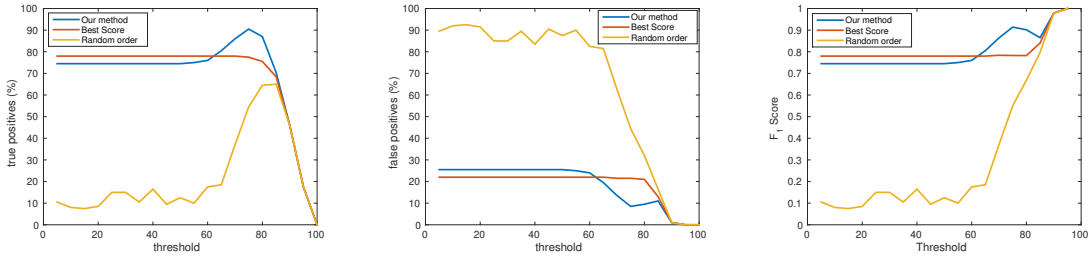
Training Set	**Avg. Rank (D_1) (over 10 objects)**	**Avg. Rank (D_2) (over 15 objects)**
D_{train}	4.05	-
$D_{train} + \{I_j\}$, with $\{I_j\} \in D_{other}$	1.37	1.99

Table 2.1 – Average ranking of objects

F_1 Score

In this section, we present the results of evaluating the performance of `LINE-2D` template matching, with and without our pre-processing approach. We used the multi-class F_1 score defined in [27] to do the evaluation. The score can be calculated as follows:

$$\mathrm{F_1score}_\mu = \frac{2 \times \mathrm{Precision}_\mu \times \mathrm{Recall}_\mu}{\mathrm{Precision}_\mu + \mathrm{Recall}_\mu} \tag{2.2}$$



(a) True Positives (b) False Positives (c) F_1 Score

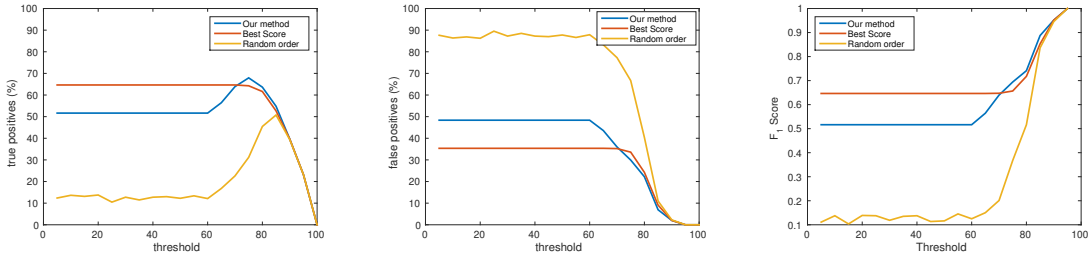
Figure 2.6 – Different performance indicators with respect to template matching thresholds on our own acquired dataset D_1

$$\text{Precision}_\mu = \frac{\sum_{i=1}^{x} tp_i}{\sum_{i=1}^{x} tp_i + fp_i} \tag{2.3}$$

$$\text{Recall}_\mu = \frac{\sum_{i=1}^{x} tp_i}{\sum_{i=1}^{x} tp_i + fn_i} \tag{2.4}$$

These last equations are valid to measure classification performance for classes C_i. The variables tp_i, fp_i, tn_i and fn_i are respectively for true positives, false positives, true negatives and false negatives, all for class C_i.

Scores are shown in Fig. 2.6 on our dataset D_1 and in Fig. 2.7 for the pose dataset D_2. These graphs show the performance of `LINE-2D` for 3 different methods: our pre-processing method, the best score and random order. For a description of our method, see section 2.5. For the best score approach, all templates are processed and the template with the highest score is considered a match. For the random order approach, a random order of objects is generated and all the templates of that particular object are processed. It is repeated until a template of the object gets a score higher than the selected threshold. In these graphs, we can see that our method clearly outperforms randomness. The effect of using our ResNet for classification is shown for low thresholds, where the score represents the classification performance of our ResNet. We therefore show that our method greatly mitigates the negative impact brought on by electing a threshold τ_{score} that is too low. Compared to the best score approach, we can see that our method performs slightly worse for lower thresholds, but gets slightly better performances as the threshold is increased. The big difference between these 2 methods is that our method performs the matching for only a few objects while the best score approach needs to process all of them.



(a) True Positives (b) False Positives (c) F_1 Score

Figure 2.7 – Different performance indicators with respect to template matching thresholds on the pose dataset D_2

2.7 Conclusion

In this paper, we proposed a generic and modular template matching pre-processing method which ranks objects by using Deep Learning for object detection and recognition. We have shown that our pre-ranking can speed up the matching of templates while increasing its reliability (in case of first-match search). More precisely, our method was able to reduce the number of objects that needed to be processed by template matching from an average of 5 objects down to 1.37 for our own dataset and from an average of 7.5 to 1.99 for the Pose dataset. This indicates that we would get near constant-time computation, for small to moderately sized databases of objects (up to 1,000). Moreover, F_1 scores indicate that we are competitive with a brute-force template matching approach, while being significantly faster.

With this project, we also hope to ease the transition of deep learning into industries by proposing to use a deep learning detector as pre-processing approach for a template matching method. Further testing in real environments and with larger object sets will be needed.

Bibliography

[1] A. Aldoma, F. Tombari, J. Prankl, A. Richtsfeld, L. Di Stefano, and M. Vincze. Multimodal cue integration through hypotheses verification for RGB-D object recognition and 6DOF pose estimation. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2104–2111. IEEE, 2013.

[2] N. Alt, S. Hinterstoisser, and N. Navab. Rapid selection of reliable templates for visual tracking. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1355–1362. IEEE, 2010.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[4] G. Bradski et al. The opencv library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.

[5] A. Crivellaro, M. Rad, Y. Verdie, K. Moo Yi, P. Fua, and V. Lepetit. A novel representation of parts for accurate 3D object detection and tracking in monocular images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4391–4399, 2015.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[7] D. Fox. Deep learning for tracking and intuitive physics. In *Robotics: Science and Systems Workshop, Are the Sceptics Right? Limits and Potentials of Deep Learning in Robotics*, 2016.

[8] K. Fu. *Sequential methods in pattern recognition and machine learning*, volume 52. Academic press, 1968.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*, 2016.

[12] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888, 2012.

[13] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab. Dominant orientation templates for real-time detection of texture-less objects. In *CVPR*, volume 10, pages 2257–2264, 2010.

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

[15] W. Kehl, F. Tombari, N. Navab, S. Ilic, and V. Lepetit. Hashmod: A hashing method for scalable 3D object detection. *CoRR*, abs/1607.06062, 2016.

[16] S. Korman, D. Reichman, G. Tsur, and S. Avidan. Fast-match: Fast affine template matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2331–2338, 2013.

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

[18] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *I. J. Robotics Res.*, 34(4-5):705–724, 2015.

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed. Ssd: Single shot multibox detector. *arXiv preprint arXiv:1512.02325*, 2015.

[21] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[22] S. Mattoccia, F. Tombari, and L. Di Stefano. Fast full-search equivalent template matching by enhanced bounded correlation. *IEEE transactions on image processing*, 17(4):528–538, 2008.

[23] A. Penate-Sanchez, L. Porzi, and F. Moreno-Noguer. Matchability prediction for full-search template matching algorithms. In *3D Vision (3DV), 2015 International Conference on*, pages 353–361. IEEE, 2015.

[24] R. Rios-Cabrera and T. Tuytelaars. Discriminatively trained templates for 3D object detection: A real time scalable approach. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2048–2055, 2013.

[25] A. Shah, E. Kadam, H. Shah, and S. Shinde. Deep residual networks with exponential linear unit. *arXiv preprint arXiv:1604.04112*, 2016.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[27] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[28] F. Tombari, S. Mattoccia, and L. Di Stefano. Full-search-equivalent pattern matching with incremental dissimilarity approximations. *IEEE transactions on pattern analysis and machine intelligence*, 31(1):129–141, 2009.

[29] L. Trottier, P. Giguère, and B. Chaib-draa. Parametric exponential linear unit for deep convolutional neural networks. *arXiv preprint arXiv:1605.09332*, 2016.

[30] F. Viksten, P.-E. Forssén, B. Johansson, and A. Moe. Comparison of local image descriptors for full 6 degree-of-freedom pose estimation. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 2779–2786. IEEE, 2009.

[31] P. Viola and M. J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, may 2004.

[32] P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3D pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3109–3118, 2015.

Chapter 3

Learning Object Localization and 6D Pose Estimation from Simulation and Weakly Labeled Real Images

3.1 Résumé

Entraîner des réseaux de neurones profonds pour des méthodes d'estimation de pose d'objets est un processus long et demandant, alors qu'il est nécessaire de fournir aux réseaux un grand nombre d'images avec la pose en six dimensions (trois pour la translation et trois pour la rotation) de chaque objet. Dans cet article, nous proposons une approche en cascade, composée d'un réseau de neurones faiblement supervisé en amont d'une méthode d'estimation de pose, qui permet de simplifier le processus d'acquisition et d'annotation d'images d'entraînement. Les performances de notre approche en estimation de pose sur deux jeux de données publics (YCB-video [1] et Occluded Linemod [2]) montrent qu'elle se classe parmi les plus performantes lorsque comparées avec les méthodes de l'état-de-l'art malgré l'utilisation d'un niveau de supervision plus faible et d'une plus petite quantité d'images réelles à l'entraînement.

3.2 Abstract

Accurate pose estimation is often a requirement for robust robotic grasping and manipulation of objects placed in cluttered, tight environments, such as a shelf with multiple objects. When deep learning approaches are employed to perform this task, they typically require a large amount of training data. However, obtaining precise six degrees of freedom for ground-truth can be prohibitively expensive. This work therefore proposes an architecture and a training process to solve this issue. More precisely, we present a weak object detector that enables localizing objects and estimating their 6D poses in cluttered and occluded scenes. To minimize the human labor required for annotations, the proposed detector is trained with a combination

of synthetic and a few weakly-annotated real images (as little as 10 images per object), for which a human provides only a list of objects present in each image (no time-consuming annotations, such as bounding boxes, segmentation masks and object poses). To close the gap between real and synthetic images, we use multiple domain classifiers trained adversarially. During the inference phase, the resulting class-specific heatmaps of the weak detector are used to guide the search of 6D poses of objects. Our proposed approach is evaluated on several publicly available datasets for pose estimation. We also evaluated our model on classification and localization in unsupervised and semi-supervised settings. The results clearly indicate that this approach could provide an efficient way toward fully automating the training process of computer vision models used in robotics.

3.3 Introduction

Robotic manipulators are increasingly deployed in challenging situations that include significant occlusion and clutter. Prime examples are warehouse automation and logistics, where such manipulators are tasked with picking up specific items from dense piles of a large variety of objects, as illustrated in Fig. 3.1. The difficult nature of this task was highlighted during the recent Amazon Robotics Challenges [3]. These robotic manipulation systems are generally endowed with a perception pipeline that starts with object recognition, followed by the object's six degrees-of-freedom (6D) pose estimation. It is known to to be a computationally challenging problem, largely due to the combinatorial nature of the corresponding global search problem. A typical strategy for pose estimation methods [2, 4–6] consists in generating a large number of candidate 6D poses for each object in the scene and refining hypotheses with the Iterative Closest Point (ICP) [7] method or its variants. The computational efficiency of this search problem is directly affected by the number of pose hypotheses. Reducing the number of candidate poses is thus an essential step towards real-time grasping of objects.

Training Convolutional Neural Networks (CNN) for tasks such as object detection and segmentation [9–11] makes it possible to narrow down the regions that are used for searching for object poses in RGB-D images. However, CNNs typically require large amounts of annotated images to achieve a good performance. While such large datasets are publicly available for general-purpose computer vision, specialized datasets in certain areas such as robotics and medical image analysis tend to be significantly scarcer and time-consuming to obtain. In a warehouse context (our target context), new items are routinely added to inventories. It is thus impractical to collect and manually annotate a new dataset every time an inventory gets updated, particularly if it must cover all possible lighting and arrangement conditions that a robot may encounter during deployment. This is even more challenging if one wants this dataset to be collected by non-expert workers. The main goal of our approach is thus to reduce such a need for manual labeling, including completely eliminating bounding boxes, segmentation masks and 6D ground truth manual annotations.

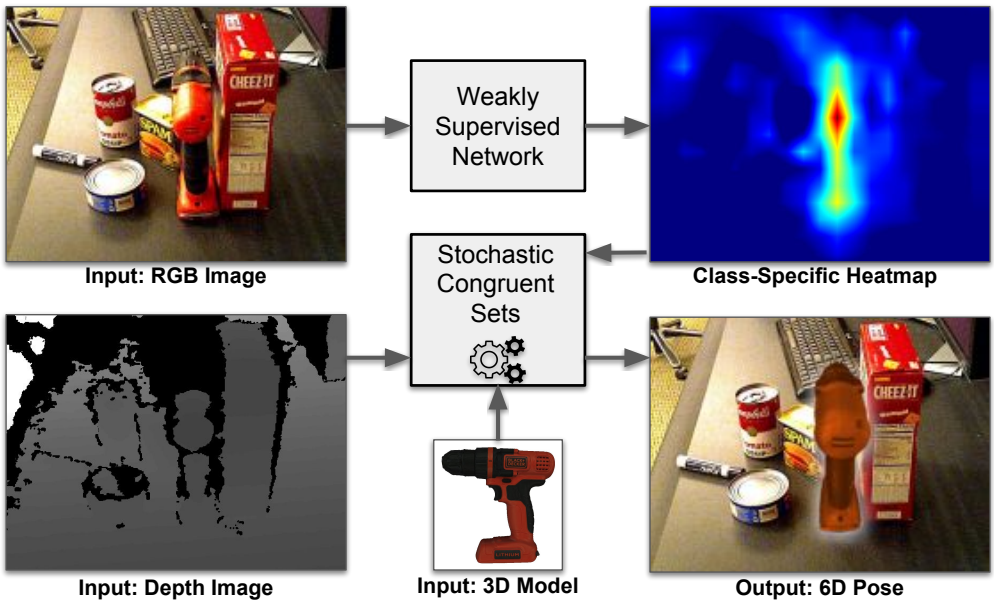


Figure 3.1 – Overview of our approach for 6D pose estimation at inference time. This figure shows the pipeline for the drill object of the YCB-video dataset [1]. A deep learning model is trained with *weakly annotated* images. Extracted class-specific heatmaps, along with 3D models and the depth image, guide the Stochastic Congruent Sets (StoCS) method [8] to estimate 6D object poses. Further details of the network are available in Section 3.5.

Our first solution to reduce manual annotations is to leverage synthetic images generated with a CAD model rendered on diverse backgrounds. However, the visual features difference between real and synthetic images can be large to the point of leading to poor performance on real objects. The problem of learning from data sampled from non-identical distributions is known as *domain adaptation*. Domain adaptation has been increasingly seen as a solution to bridge the gap between domains [12, 13]. Roughly speaking, domain adaptation tries to generalize the learning from a *source domain* to a *target domain*, or in our case, from synthetic to real images. Since labeled data in the target domain is unavailable or limited, the standard way is to train on labeled source data, while trying to minimize the distribution discrepancy between source and target domains.

While having a small labeled dataset on a target domain allows to boost performances, it may still require significant human effort for the annotations. Our second solution is to use *weakly supervised learning*, which significantly decreases annotation efforts, albeit with a reduced performance compared to fully-annotated images. Some methods [14, 15] have been shown to be able to retrieve a high level representation of the input data (such as object localization) while only being trained for object classification. To the best of our knowledge, this promising kind of approach has not yet been applied within a robotic manipulation context.

In this paper, we propose a two-step approach for 6D pose estimation, as shown in Fig. 3.1.

First, we train a network for classification through domain adaptation, by using a combination of weakly labeled synthetic and real color images. During the inference phase, the weakly supervised network generates class-specific heatmaps that are subsequently refined with an independent 6D pose estimation method called Stochastic Congruent Sets (StoCS) [8]. Our complete method achieves competitive results on the YCB-video object dataset [1] and Occluded Linemod [2] while using only synthetic images and few weakly labeled real images (as little as 10) per object in training. We also empirically demonstrate that for our test case, using domain adaptation in semi-supervised settings is preferable than training in unsupervised settings and fine-tuning on available weakly labeled real images, a commonly-accepted strategy when only a few images from the target domain are available.

3.4 Related Work

In this paper, we aim at performing object localization and 6D pose estimation with a deep network, with minimal human labeling efforts. Our approach is based on training from synthetic and weakly labeled real images, via domain adaptation. These various concepts are discussed below.

6D Pose Estimation Recent literature in pose estimation focuses on learning to predict 6D poses using deep learning techniques. For example, [1] predicts separately the object center in images for translation and regresses over the quaternion representation for predicting the rotation. Another approach is to first predict 3D object coordinates, followed by a RANSAC-based scheme to predict the object's pose [5, 6]. Similarly, [6] uses geometric consistency to refine the predictions from the learned model. These methods, however, need access to several images that are manually labeled with the full object poses, which is time-consuming to acquire. Some other approaches make use of the object segmentation output to guide a global search process for estimating object poses in the scene [8, 16, 17]. Although the search process could compensate for errors in prediction when the segmentation module is trained with synthetic data, the domain gap could be large, and a computationally expensive search process may be needed to bridge this gap.

Learning with Synthetic Data Training with synthetic data has recently gained significant interest, as shown by the multiple synthetic datasets recently available [18–23], with some focusing on optimizing the realism of the generated images. While the latter can decrease to a certain degree the gap between real and synthetic images, it somehow defeats the purpose of using simulation as a cost-effective way to create training data. To circumvent this issue, [24, 25] proposed instead to create images using segmented object instances copied on real images. This type of approach, akin to data augmentation, is however limited to the number of object views and illuminations that are available in the original dataset. Recently, [26, 27] showed promising results by training object detectors with 3D models rendered in simulation with

randomized parameters, such as lighting, number of objects, object poses, and backgrounds. While in [26] they only use synthetic images in training, [27] demonstrated the benefits of fine-tuning on a limited set of real labeled images. The last one also showed that using photorealistic synthetic images does not necessarily improve object detection, compared to training on a less realistic synthetic dataset generated with randomized parameters.

Domain Adaptation Domain adaptation techniques [12, 13] can serve to decrease the distribution discrepancy between different domains, such as real vs. synthetic. The popular DANN [28] approach relies on two classifiers: one for the desired task, trained on labeled data from a source domain, and another one (called *domain classifier*) that classifies whether the input data is from the source or target domain. Both classifiers share the first part of the network, which acts as a feature extractor. The network is trained in an adversarial manner: domain classifier parameters are optimized to minimize the domain classification loss, and shared parameters are optimized to maximize the domain classification loss. It is possible to achieve this minimax optimization in a single step by using a gradient reversal layer that reverses the sign of the gradient between shared and non-shared parameters of the domain classifier. To the best of our knowledge, the present work is the first use a DANN-like approach for point-wise object localization, a fundamental problem in robotic manipulation.

Weakly Supervised Learning We are interested in weakly supervised learning with inexact supervision, for which only coarse-grained labels are available [29]. In [14], a network was trained only with weak image-level labels (classes that are present in images, but not their position) and max-pooling was used to retrieve approximate location of objects. The proposed *WILDCAT* model [15] performs classification and weakly supervised point-wise detection and segmentation. This architecture learns multiple localized features for each class, and uses a spatial pooling strategy that generalizes to many ones (max pooling, global average pooling and negative evidence). In the present work, we push the paradigm of minimum human supervision *even further*. To this effect, we propose to train WILDCAT with synthetic images, in addition to weakly supervised real ones, and use MADA (a variant of DANN) for domain adaptation.

3.5 Proposed Approach

We present here our approach to object localization and 6D pose estimation. It is trained using a mix of synthetic and real images and only requires weak annotations (only class-presence) in both domains.

3.5.1 Overview

Figure 3.2 depicts an overview of our proposed system. It comprises *i*) a *ResNet-50* model pre-trained on *ImageNet* as a feature extractor (green), *ii*) a weak classifier inspired from

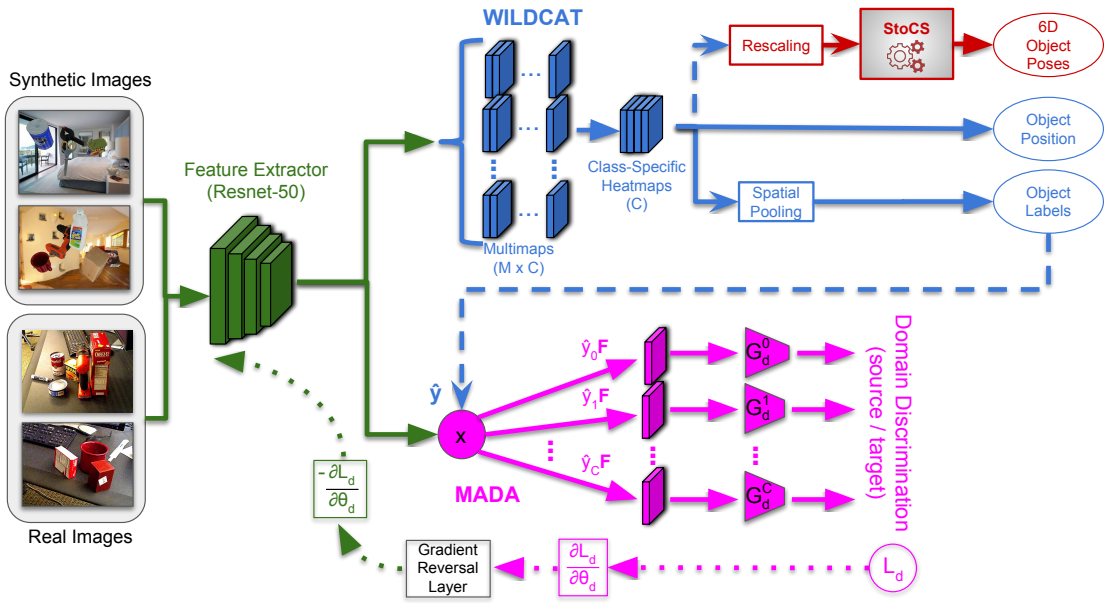


Figure 3.2 – Overview of the proposed approach for object localization and 6D pose estimation with domain adaptation, using a mix of synthetic images and weakly labeled real images.

the WILDCAT model [15] (blue), *iii*) the Stochastic Congruent Sets (*StoCS*) for 6D pose estimation (red) [8], and *iv*) the MADA domain adaptation network to bridge the gap between synthetic and real data. During the inference phase, the domain adaptation part of the network is discarded. Given a test image, class-specific heatmaps are generated by the network. These heatmaps indicate the most probable locations of each object in the image. This probability distribution is then fed to StoCS, a robust pose estimation algorithm that is specifically designed to deal with noisy localization. To force the feature extractor to extract similar features for both synthetic and real images, a MADA module (described below) is employed. MADA's purpose is to generate gradients during training (via a reversal layer) in order to improve the generalization capabilities of the feature extractor.

3.5.2 Synthetic Data Generation

For synthetic data generation, we used a modified version of the SIXD toolkit[1]. This toolkit generates color and depth images of 3D object models rendered on black backgrounds. Virtual camera viewpoints are sampled on spheres of different radii, following the approach described in [30]. We extended the toolkit with the functionality of rendering more than one object per image, and also used random backgrounds taken from the LSUN dataset [31]. Similarly to recent *domain randomization* techniques [32], we observed from our experiments that these simple modifications help transferring from simulation to real environments where there are

[1] https://github.com/thodan/sixd_toolkit

(a) (b)

Figure 3.3 – Examples of synthetic image used in training.

multiple objects of interest, occlusions and diverse backgrounds. Figure 3.3 displays some examples of the generated synthetic images that we used to train our network.

3.5.3 Weakly Supervised Learning with WILDCAT

The images used for training our system are weakly labeled: only a list of object classes present in the image is provided. In order to recover localization from such weak labels, we leverage the WILDCAT architecture [15]. Indeed, WILDCAT is able to recover localization information through its high-level feature map, even though it is only trained with a classification loss. As a feature extractor, we employ a ResNet-50 (pretrained on ImageNet) for which the last layers (global average pooling and fully connected layers) are removed, as depicted in Figure 3.2. The WILDCAT architecture added on top of this ResNet-50 comprises three main modules: a *multimap transfer layer*, a *class pooling layer* and a *spatial pooling layer*. The *multimap transfer layer* consists of 1×1 convolutions that extracts M class-specific modalities per class C, with $M = 8$ as per the original paper [15]. The *class pooling* module is an average pooling layer that reduces the number of feature maps from MC to C. Then, the *spatial pooling* module selects k regions with maximum/minimum activations to calculate scores for each class. The classification loss for this module is a multi-label one-versus-all loss based on max-entropy (*MultiLabelSoftMarginLoss* in PyTorch). The classification scores are then rescaled between 0 and 1 to cooperate with MADA.

3.5.4 Multi-Adversarial Domain Adaptation with MADA

We used the *Multi-Adversarial Domain Adaptation* (MADA) approach [33] to bridge the "reality gap". MADA extends the *Domain Adversarial Networks* (DANN) approach [28] by using one domain discriminator per class, instead of a single global discriminator as in the original version of DANN [28]. Having one discriminator per class has been found to help aligning class-specific features between domains. In MADA, the loss L_d for the K domain discriminators

and input $\mathbf{x_i}$ is defined as:

$$L_d = \frac{1}{n}\sum_{k=1}^{K}\sum_{\mathbf{x_i}\in D_s\cup D_t} L_d^k\Big(G_d^k\Big(\hat{y}_i^k G_f(\mathbf{x_i})\Big), d_i\Big), \tag{3.1}$$

wherein $i \in \{1,\ldots,n\}$, and $n = n_s + n_t$ is the total number of training images in source domain D_s (synthetic images) and the target domain D_t (real images). G_f is the feature extractor (the same for both domains), $\hat{y}_i^k$ is the probability of label k for image $\mathbf{x_i}$. This probability $\hat{y}_i^k$ is the output of the weak classifier WILDCAT. G_d^k is the k-th domain discriminator and L_d^k is its cross-entropy loss, given the ground truth domain $d_i \in \{\text{synthetic}, \text{real}\}$ of image $\mathbf{x_i}$. Our global objective function is:

$$C = \frac{1}{n}\sum_{\mathbf{x_i}\in D} L_y\Big(G_y\Big(G_f(\mathbf{x_i})\Big), y_i\Big) - \lambda L_d \quad , \tag{3.2}$$

where L_y is the classification loss, L_d the domain loss and λ has been found to work well with a value of 0.5. The heat-map probability distribution extracted from WILDCAT is used to guide the StoCS algorithm in its search for 6D poses, as explained in the next section.

3.5.5 Pose Estimation with Stochastic Congruent Sets (StoCS)

The StoCS method [8] is a robust pose estimator that predicts the 6D pose of an object in a depth image from its 3D model and a probability heatmap. We employ a min-max normalization on the class-specific heatmaps of the Wildcat network, transforming them into a probability heatmaps w_{p_i}, using the per-class minimum (w_{min}) and maximum (w_{max}) values:

$$\pi_{p_i \to O_k} = \frac{w_{p_i} - w_{min}}{w_{max} - w_{min}}. \tag{3.3}$$

This generates a heatmap providing the probability π of an object O_k being located at a given pixel p_i. The StoCS algorithm then follows the paradigm of a randomized alignment technique. It does so by iteratively sampling a set of four points, called a base B, on the point cloud S and finds corresponding set of points on the object model M. Each corresponding set of four points defines a rigid transformation T, for which an alignment score is computed between the transformed model cloud and the heatmap for that object. The optimization criteria is defined as

$$T_{opt} = arg\,max_T \sum_{m_i \in M_k} f(m_i, T, S_k), \tag{3.4}$$

$$f(m_i, T, S_k) = \pi_k(s*), if \mid T(m_i) - s* \mid < \delta_s. \tag{3.5}$$

The base sampling process in this algorithm considers the joint probability of all four points belonging to the object in question, given as

$$Pr(B \to O_k) = \frac{1}{Z}\prod_{i=1}^{4}\{\phi_{node}(b_i)\prod_{j=1}^{j<i}\phi_{edge}(b_i, b_j)\}. \tag{3.6}$$

where ϕ_{node} is obtained from the probability heatmap and ϕ_{edge} is computed based on the point-pair features of the pre-processed object model. Thus, the method combines the normalized output of the Wildcat network with the geometric model of objects to obtain base samples which belong to the object with high probability.

In the next two Sections, we demonstrate the usefulness of our approach. First in Section 3.6, we quantify the importance of each component (Wildcat, MADA) in order to train a network that generates *relevant* feature maps from weakly labeled images. In Section 3.7, we then evaluate the performance of using these heatmaps with StoCS for rapid 6D pose estimation, which is the final goal of our paper.

3.6 Weakly Supervised Learning Experiments for object detection and classification

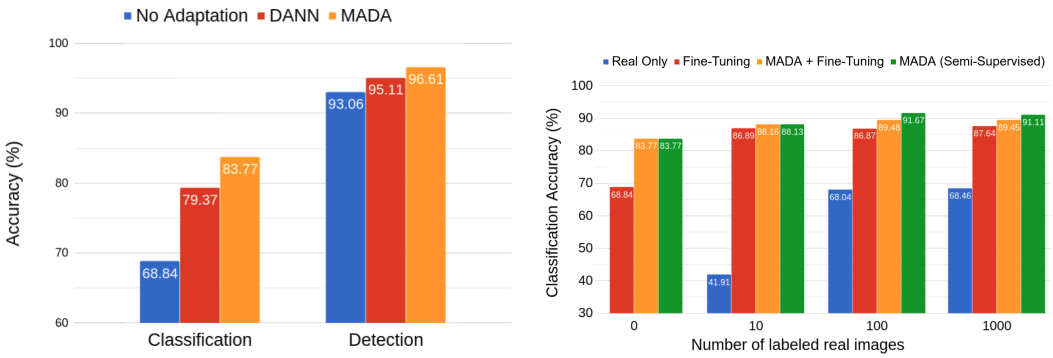
In this first experimental section, we perform an ablation study to evaluate the impact of various components for classification and point-wise localization. We first tested our approach without any human labeling, as a baseline. We then evaluated the gain obtained by employing various numbers of weakly labeled images for four semi-supervised strategies.

We performed these evaluations on the YCB-video dataset [1]. This dataset contains 21 objects with available 3D models. It also has full annotations for detection and pose estimation on 113,198 training images and 20,531 test images. A subset of 2,949 test images (keyframes) is also available. Our results are reported for this more challenging subset, since most images in the bigger test set are video frames that are too similar and would report optimistic results.

For these experiments, we trained our network for 20 epochs (500 iterations per epoch) with a batch size of 4 images per domain. We used stochastic gradient descent with a learning rate of 0.001 (decay of 0.1 at epochs 10 and 16) and a Nesterov momentum of 0.9. The ResNet-50 was pre-trained on ImageNet and the weights of the first two blocks were frozen (configuration that performed the best in initial experiments).

3.6.1 Unsupervised Domain Adaptation

For this experiment, we trained our model with weakly labeled synthetic images (WS) and unlabeled real images (UR). We tested three architecture configurations of domain adaptation: 1) without any domain adaptation module (WILDCAT model trained on WS), 2) with DANN (WS+UR) and 3) with MADA (WS+UR). We evaluated each of these configurations for both classification and detection. For classification, we used the accuracy metric to evaluate our model's capacity to discriminate which objects are in the image. We used a threshold of 0.5 on classification scores to predict the presence or absence of an object. For detection, we employed the point-wise localization metric [14], which is a standard metric to evaluate



(a) (b)

Figure 3.4 – Performance analysis. In (a), we compare classification accuracy and point-wise detection when no label on real images are available. In (b), we compare the performance of different training processes when different numbers of real images are weakly labeled.

the ability of weakly supervised networks to localize objects. For each object in the image, the maximum value in their class-specific heatmap was used to retrieve the corresponding pixel in the original image. If this pixel is located inside the bounding box of the object of interest, it is counted as a good detection. Since the class-specific heatmap is a reduced scale of the input image due to pooling, a tolerance equal to the scale factor was added to the bounding box. In our case, a location in the class-specific heatmaps corresponds to a region of 32 pixels in the original image. In Figure 3.4a, we report the average scores of the last 5 epochs over 3 independent random runs for each network variation. These results *a)* confirm the importance of employing a domain adaptation strategy to bridge the reality gap, and *b)* the necessity of having one domain discriminator G_d^k for each of the X objects in the YCB database (MADA), instead of a single one (DANN). Next, we evaluate the gains obtained by employing weakly-annotated real images.

3.6.2 Semi-Supervised Domain Adaptation

A significant challenge for agile deployment of robots in industrial environments is that they ideally should be trained with limited annotated data, both in terms of numbers of images and of their extensiveness of labeling (no pose information, just class). We thus evaluated the performance of four different strategies as a function of the number of such weakly-labeled real images:

1. Without domain adaptation:
    a) Real Only: Trained only on weakly labeled real images,
    b) Fine-Tuning: Trained on synthetic images and then fine-tuned on weakly labeled real images,

2. With domain adaptation:

    a) Fine-Tuning: Trained on synthetic images and then fine-tuned on weakly labeled real images,

    b) Semi-Supervised: Trained with synthetic images and weakly labeled real images simultaneously.

For 1.a and 1.b, we validate that using fine-tuning on a network pre-trained with synthetic data is preferable to training directly on real images. For 2.a and 2.b, we compare the performance of our approach trained with fine-tuning, and in a semi-supervised way (using images from both domains at the same time). We are particularly interested in comparing the two approaches 2.a and 2.b, since [34] achieved the lowest error rate compared to any other semi-supervised approach by only using fine-tuning.

Our results are summarized in Figure 3.4b. From them, we conclude that training with synthetic images improves classification accuracy drastically, especially when few labels are available. Also, our approach performs slightly better when trained in a semi-supervised setting (2.b) than with a fine-tuning approach (2.a), which is contrary to [34].

In this Section, we justified our architecture, as well as the training technique employed, to create a network capable of performing object identification and localisation through weak learning. In the next Section, we demonstrate how the feature maps extracted by our network can be employed to perform precise 6 DoF object pose estimation via StoCS.

3.7 6D Pose Estimation Experiments

We evaluated our full approach for 6D pose estimation on YCB-video [1] and Occluded Linemod [2] datasets. We used the most common metrics to compare with similar methods. The average distance (ADD) metric [35] measures the average distance between the pairwise 3D model points transformed by the ground truth and predicted pose. For symmetric objects, the ADD-S metric measures the average distance using the closest point distance. Also, the visible surface discrepancy [36] compares the distance maps of rendered models for estimated and ground-truth poses.

We used the same training details mentionned in section 3.6. Since the network architecture is fully convolutional, we also added an experiment for which we combined the output of the network for 3 different scales of the input image (at test time only).

3.7.1 YCB-Video Dataset

This dataset comprises several frames from 92 video sequences of cluttered scenes created with 21 YCB objects. The training for competing methods [1, 37, 38] is performed using

113,199 frames from 80 video sequences with semantic (pixelwise) and pose labels. For our proposed approach, we used only 10 randomly sampled weakly annotated (class labels only) real images per object class combined with synthetic images. As in [1], we report the area under the curve (AUC) of the accuracy-threshold curve, using the ADD-S metric. Results are reported in Table 3.1. Our proposed method achieves 88.67% accuracy with a limited number of weakly labeled images and up to 93.60% when using the full dataset with multiscale inference. It outperforms competing approaches, with the exception of PoseCNN+ICP, which performs similarly. However, our approach has a large computational advantage with an average runtime of 0.6 seconds per object as opposed to approximately 10 seconds per object for the modified-ICP refinement for PoseCNN. It also uses *a)* nearly a hundredfold less real data, and *b)* also only using the class labels. This results thus demonstrate that we can reach *fast* and *competitive* results without the need of 6D fully-annotated real datasets.

3.7.2 Occluded Linemod Dataset

This dataset contains 1215 frames from a single video sequence with pose labels for 9 objects from the LINEMOD dataset with high level of occlusion. Competing methods are trained using the standard LINEMOD dataset, which consists in average of 1220 images per object. In our case, we used 10 real random images per object (manually labelled) on top of the generated synthetic images, using the weak (class) labels only. As reported in Table 3.1, our method achieved scores of 68.8% and 76.6% (multiscale) for the ADD evaluation metric and using a threshold of 10% of the 3D model diameter. These results compare with state-of-the-art methods while using less supervision and a fraction of training data. The multiscale variant (input image at 3 different resolutions) made our approach more robust to occlusions. We did not train with the full Linemod training dataset, since the dataset only has annotations for 1 object per image and our method requires the full list of objects that are in the image. Furthermore, we evaluated our approach on the 6D pose estimation benchmark [36] using the visual discrepency metric. We evaluated our network with multiscale inference and we can see in Table 3.2 that we are among the top 3 for the recall score while being the fastest. We also tested the effect of combining ICP with StoCS. At the cost of more processing time, we obtain the best performance among the methods that were evaluated on the benchmark.

Method	Modality	Supervision	Full Dataset	AUC (ADD-S) YCB-Video	ADD-10% Occluded Linemod
PoseCNN [1]	RGB	Pixelwise labels + 6D poses	Yes	75.9	24.9
PoseCNN+ICP [1]	RGBD	Pixelwise labels + 6D poses	Yes	93.0	**78.0**
DeepHeatmaps [37]	RGB	Pixelwise labels + 6D poses	Yes	81.1	28.7
FCN + Drost et. al. [38]	RGBD	Pixelwise labels	Yes	84.0	-
FCN + StoCS [8]	RGBD	Pixelwise labels	Yes	90.1	-
Brachmann et al. [5]	RGBD	Pixelwise labels + 6D poses	Yes	-	56.6
Michel et. al. [6]	RGBD	Pixelwise labels + 6D poses	Yes	-	76.7
OURS	RGBD	Object classes	No (10 weakly labeled images)	88.7	68.8
OURS	RGBD	Object classes	Yes	90.2	-
OURS (multiscale inference)	RGBD	Object classes	No (10 weakly labeled images)	-	76.6
OURS (multiscale inference)	RGBD	Object classes	Yes	**93.6**	-

Table 3.1 – Area under the accuracy-threshold curve for 6D Pose estimation on YCB-Video dataset (ADD-S metric) and ADD metric for Occluded Linemod with threshold of 10% of the diameter (ADD-S metric for 2 objects).

Method	Recall Score (%)	Time (s)
Vidal-18 [39]	59.3	4.7
Drost-10 [38]	55.4	2.3
Brachmann-16 [40]	52.0	4.4
Hodan-15 [41]	51.4	13.5
Brachmann-14 [5]	41.5	1.4
Buch-17-ppfh [42]	37.0	14.2
Kehl-16 [43]	33.9	1.8
OURS (MS)	55.2	0.6
OURS (MS) + ICP	**62.1**	6.4

Table 3.2 – Visual discrepency recall scores (%) (correct pose estimation) for $\tau = 20$mm and $\theta = 0.3$ on Occluded Linemod, based on the 6D pose estimation benchmark [36]. MS means multiscale.

3.8 Conclusion

In this paper, we explored the problem of 6D pose estimation in the context of limited annotated training datasets. To this effect, we demonstrated that the output of a weakly-trained network is sufficiently rich to perform full 6D pose estimation. Pose estimation experiments on two datasets showed that our approach is competitive with recent approaches (such as PoseCNN), despite requiring *significantly less annotated images*. Most importantly, our annotation level requirement for real images is *much weaker*, as we only need a class label without any spatial information (either bounding box or full 6D ground truth). In this end, this makes our approach compatible with an agile automated warehouse, where new objects to be manipulated are constantly introduced in a training database by non-expert employees.

Bibliography

[1] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[2] A. Krull, E. Brachmann, F. Michel, M. Ying Yang, S. Gumhold, and C. Rother, "Learning analysis-by-synthesis for 6D pose estimation in RGB-D images," in *International Conference on Computer Vision*, 2015, pp. 954–962.

[3] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Osada, A. Rodriguez, J. Romano, and P. Wurman, "Analysis and Observations From the First Amazon Picking Challenge," *Transactions on Automation Science and Engineering*, 2016.

[4] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige, "Going further with point pair features," in *European Conference on Computer Vision*. Springer, 2016, pp. 834–848.

[5] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6D object pose estimation using 3D object coordinates," in *European Conference on Computer Vision*. Springer, 2014, pp. 536–551.

[6] F. Michel, A. Kirillov, E. Brachmann, A. Krull, S. Gumhold, B. Savchynskyy, and C. Rother, "Global hypothesis generation for 6D object pose estimation," in *Conference on Computer Vision and Pattern Recognition*, 2017, pp. 462–471.

[7] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–607.

[8] C. Mitash, A. Boularias, and K. Bekris, "Robust 6D object pose estimation with stochastic congruent sets," *arXiv preprint arXiv:1805.06324*, 2018.

[9] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker Jr, A. Rodriguez, and J. Xiao, "Multi-view self-supervised deep learning for 6D pose estimation in the amazon picking challenge," in *International Conference on Robotics and Automation*, 2017.

[10] C. Hernandez, M. Bharatheesha, W. Ko, H. Gaiser, J. Tan, K. van Deurzen, M. de Vries, B. Van Mil, J. van Egmond, R. Burger *et al.*, "Team delft's robot winner of the amazon picking challenge 2016," in *Robot World Cup*. Springer, 2016, pp. 613–624.

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

[12] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *arXiv preprint arXiv:1802.03601*, 2018.

[13] G. Csurka, "Domain adaptation for visual applications: A comprehensive survey," *arXiv preprint arXiv:1702.05374*, 2017.

[14] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Is object localization for free?-weakly-supervised learning with convolutional neural networks," in *Conference on Computer Vision and Pattern Recognition*, 2015, pp. 685–694.

[15] T. Durand, T. Mordan, N. Thome, and M. Cord, "Wildcat: Weakly supervised learning of deep convnets for image classification, pointwise localization and segmentation," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2017.

[16] V. Narayanan and M. Likhachev, "Discriminatively-guided deliberative perception for pose estimation of multiple 3D object instances." in *Robotics: Science and Systems*, 2016.

[17] C. Mitash, A. Boularias, and K. E. Bekris, "Improving 6D pose estimation of objects in clutter via physics-aware monte carlo tree search," *arXiv preprint arXiv:1710.08577*, 2017.

[18] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," *arXiv preprint arXiv:1605.06457*, 2016.

[19] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4040–4048.

[20] W. Qiu and A. Yuille, "Unrealcv: Connecting computer vision to unreal engine," in *European Conference on Computer Vision*. Springer, 2016, pp. 909–916.

[21] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3234–3243.

[22] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, "Driving in the matrix: Can virtual worlds replace human-generated annotations for real

world tasks?" in *International Conference on Robotics and Automation*. IEEE, 2017, pp. 746–753.

[23] S. R. Richter, Z. Hayder, and V. Koltun, "Playing for benchmarks," in *International Conference on Computer Vision*, 2017.

[24] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," *ArXiv*, vol. 1, no. 2, p. 3, 2017.

[25] G. Georgakis, A. Mousavian, A. C. Berg, and J. Kosecka, "Synthesizing training data for object detection in indoor scenes," *arXiv preprint arXiv:1702.07836*, 2017.

[26] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, "On pre-trained image features and synthetic images for deep learning," *arXiv preprint arXiv:1710.10710*, 2017.

[27] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," *arXiv preprint arXiv:1804.06516*, 2018.

[28] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[29] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, 2017.

[30] S. Hinterstoisser, S. Benhimane, V. Lepetit, P. Fua, and N. Navab, "Simultaneous recognition and homography extraction of local patches with a simple linear classifier." in *BMVC*, 2008, pp. 1–10.

[31] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "LSUN: construction of a large-scale image dataset using deep learning with humans in the loop," *CoRR*, vol. abs/1506.03365, 2015.

[32] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Intelligent Robots and Systems*. IEEE, 2017, pp. 23–30.

[33] Z. Pei, Z. Cao, M. Long, and J. Wang, "Multi-adversarial domain adaptation," in *AAAI Conference on Artificial Intelligence*, 2018.

[34] A. Oliver, A. Odena, C. Raffel, E. D. Cubuk, and I. J. Goodfellow, "Realistic evaluation of deep semi-supervised learning algorithms," *arXiv preprint arXiv:1804.09170*, 2018.

[35] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes," in *Asian conference on computer vision*. Springer, 2012, pp. 548–562.

[36] T. Hodan, F. Michel, E. Brachmann, W. Kehl, A. GlentBuch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis *et al.*, "Bop: benchmark for 6D object pose estimation," in *European Conference on Computer Vision*, 2018, pp. 19–34.

[37] M. Oberweger, M. Rad, and V. Lepetit, "Making deep heatmaps robust to partial occlusions for 3D object pose estimation," *arXiv preprint arXiv:1804.03959*, 2018.

[38] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 998–1005.

[39] J. Vidal, C.-Y. Lin, and R. Martí, "6D pose estimation using an improved method based on point pair features," in *International Conference on Control, Automation and Robotics*. IEEE, 2018, pp. 405–409.

[40] E. Brachmann, F. Michel, A. Krull, M. Ying Yang, S. Gumhold *et al.*, "Uncertainty-driven 6D pose estimation of objects and scenes from a single rgb image," in *Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3364–3372.

[41] T. Hodaň, X. Zabulis, M. Lourakis, Š. Obdržálek, and J. Matas, "Detection and fine 3D pose estimation of texture-less objects in RGB-D images," in *Conference on Intelligent Robots and Systems*. IEEE, 2015, pp. 4421–4428.

[42] A. G. Buch, L. Kiforenko, and D. Kraft, "Rotational subgroup voting and pose clustering for robust 3D object recognition," in *International Conference on Computer Vision*. IEEE, 2017, pp. 4137–4145.

[43] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab, "Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation," in *European Conference on Computer Vision*. Springer, 2016, pp. 205–220.

Chapter 4

Deep Template-based Object Instance Detection

4.1 Résumé

Les réseaux de neurones sont limités à un ensemble d'objets pour lesquels ils sont entraînés. Pour généraliser à de nouveaux objets, les réseaux doivent être ré-entrainés. Ce n'est pas pratique lorsque les objets d'intérêts peuvent changer souvent et rapidement, tel qu'en robotique industrielle. Dans cet article, nous proposons une approche générique de détection d'objets basée sur les réseaux de neurones pouvant être utilisée pour des objets jamais vus lors de l'entraînement. Notre approche améliore d'environ 30% les performances en détection d'objets pour la métrique « mean Average Precision » (mAP) par rapport à l'approche `LINE-2D` [11] (50.7% contre 21%), qui est l'état-de-l'art pour les méthodes de détection générique d'objets. Les performances de notre approche sont également comparables aux méthodes d'apprentissage entraînées spécifiquement pour l'ensemble d'objets de test. Notre approche est donc performante et simplifie l'utilisation en pratique pour des cas où l'ensemble d'objets à retrouver peut changer fréquemment.

4.2 Abstract

Much of the focus in the object detection literature has been on the problem of identifying the bounding box of a particular class of object in an image. Yet, in contexts such as robotics and augmented reality, it is often necessary to find a *specific* object instance—a unique toy or a custom industrial part for example—rather than a generic object class. Here, applications can require a rapid shift from one object instance to another, thus requiring a fast turnaround which affords little-to-no training time. What is more, gathering a dataset and training a model for every new object instance to be detected can be an expensive and time-consuming process. In this context, we propose a generic 2D object *instance* detection approach that

uses example viewpoints of the target object at test time to retrieve its 2D location in RGB images, without requiring any additional training (i.e. fine-tuning) step. To this end, we present an end-to-end architecture that extracts global and local information of the object from its viewpoints. The global information is used to tune early filters in the backbone while local viewpoints are correlated with the input image. Our method offers an improvement of almost 30 mAP over the previous template matching methods on the challenging Occluded Linemod [3] dataset (overall mAP of 50.7). Our experiments also show that our single generic model (not trained on any of the test objects) yields detection results that are on par with approaches that are trained specifically on the target objects.

4.3 Introduction

Object detection is one of the key problems in computer vision. While there has been significant effort and progress in detecting generic object classes (e.g. detect all the phones in an image), comparatively little attention has been devoted to detect specific object instances (e.g. detect *this particular* phone model). Recent approaches on this topic [21, 29, 40, 43] have achieved very good performance in detecting object instances, even under challenging occlusions. By relying on textured 3D models as a way to specify the object instances to be detected, these methods propose to train detectors tailored for these objects. Because they know the objects to be detected at training time, these approaches essentially *overfit* to the objects themselves: they become specialized at detecting them (and only them).

While this is a promising and active research direction, requiring knowledge of the objects to be detected at training time might not always be practical. For instance, if a new object needs to be detected, the entire training process must be started over. This implies first generating a full training dataset and then optimizing the network. Also, using a single network per object can be a severe constraint in embedded applications where memory is a limited resource.

In this work, we explore the case of training a *generic* 2D instance detector, where the specific object instance to be detected is only known at test time. The object to be found is represented by a set of images of that object captured from different viewpoints (fig. 4.1). In order to simplify the data capture setup and to facilitate comparisons to previous work on standard datasets, in this work we employ 3D models of the test objects and render different viewpoints. If a 3D model is not accessible, it would be possible to instead capture a few viewpoints of the object on a plain background.

This paper is akin to a line of work which has received somewhat less attention recently, that of template matching. These techniques scan the image over a dense set of sub-windows and compare each of them with a template representing the object. A canonical example is Linemod [11], which detects a 3D object by treating several views of the object as templates, and by efficiently searching for matches over the image. While very efficient, traditional

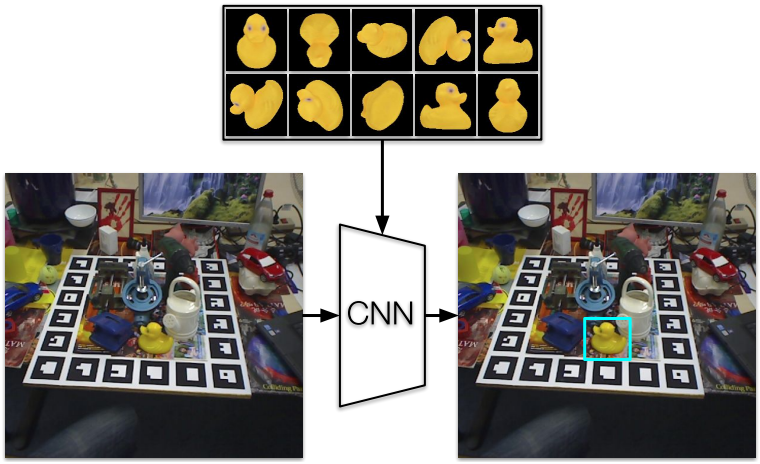


Figure 4.1 – Overview of the proposed method. At test time, our network predicts the 2D location (in an RGB image) of a target object (unseen during training) represented by templates acquired from various viewpoints.

template matching techniques can be quite brittle, especially under occlusion, and yield large amounts of false positives.

In this paper, we revive this line of work and propose a novel instance detection method. Using a philosophy sharing resemblance to meta-learning [38], our method uses a large-scale 3D object dataset and a rendering pipeline to learn a versatile template representation. At test time, our approach takes as input multiple viewpoints of any object and detects it from a single RGB image immediately, without any additional training (fig. 4.1).

Our main contribution is the design of a novel deep learning architecture which can localize instances of a target object from a set of input templates. Instead of matching pixel intensities directly such as other template matching methods, our network is trained to localize an instance from a joint embedding space. Our approach is trained exclusively on synthetic data and takes a single RGB image as input. In addition, we introduce a series of extensions to the architecture which improve the detection performance such as tunable filters to adapt the feature extraction process to the object instance in the early layers of a pretrained backbone. We quantify the contribution of each extension through a detailed ablation study. Finally, we present extensive experiments that demonstrate that our method can successfully detect object instances that were not seen during training. In particular, we report performances that significantly outperform the state of the art on the popular Occluded Linemod [3] dataset. Notably, we attain a mAP of 50.71%, which is almost 30% better than LINE-2D [11] and on par with methods that overfit on the object instance during training.

4.4 Related work

Our work is most related to two areas: object instance detection in RGB images, and 2D tracking in RGB images.

Object instance detection. Our work focuses on retrieving the 2D bounding box of a *particular object instance*. This is in contrast with well-known methods such as Faster-RCNN [31] and SSD [25] or with methods that bear more resemblance to our approach such as CoAE [18], which all provide 2D locations of *object classes*. Detecting a specific object is challenging due to the large variety of objects that can be found in the wild. Descriptor-based and template-based methods are useful in such context. Generic features such as gradient histograms [11] and color histograms [34] can be computed and then retrieved from an object codebook.

Recent progress in deep learning enabled the community to develop approaches that automatically learn features from the 3D model of an object using neural network [21, 29, 40, 43] or random forest [4] classifiers. While these methods perform exceptionally well on known benchmarks [16], they share the important limitation that training these deep neural networks requires a huge amount of labeled data tailored to the object instances to be detected. Consequently, gathering the training dataset for specific objects is both costly and time-consuming. Despite this, efforts have been made to capture such real datasets [3, 7, 12, 15, 32, 33] and to combine them together [16]. A side effect is that it confines most deep learning methods to the very limited set of objects present in these datasets, as the weights of a network are specifically tuned to detect only a single [21] or a few instances [21, 29]. The difficulty of gathering a real dataset can be partially alleviated using simple rendering techniques [13, 21, 29] combined with data augmentation such as random backgrounds and domain randomization [35, 36, 42], but still suffers from a domain gap with real images. Recently, Hodan et al. [17] demonstrated that the domain gap can be minimized with physics-based rendering. Despite this progress, all of the above methods share the same limitation, in that they all require significant time (and compute power) to train a network on a new object. This implies a slow turn-around time, where a practitioner must wait hours before a new object can be detected.

To circumvent these limitations, we propose a novel generic network architecture that is trained to detect a target object that is unavailable at training time. Our method is trained on a large set of objects and can generalize to new, different objects at test time. Our architecture bears resemblance to TDID [1] that uses a template to detect a particular instance of an object. We show in our experiments that our method performs significantly better than [1] on objects not seen during training.

Tracking in 2D images. Our work shares architectural similarities with 2D image-based tracking, for which approaches use a template of the object as input, typically identified as a bounding box in the first frame of the video. In contrast, we focus on single frame detection. Thus, we employ known viewpoints of the object acquired offline. Many of these tracking

approaches propose to use an in-network cross-correlation operation (sometimes denoted as $\star_d$) between a template and an image in feature space [5, 22, 39]. Additionally, recent 6-DOF trackers achieve generic instance tracking using simple viewpoint renders from a 3D model [8, 23, 26]. These methods are limited by the requirement of a previous temporal state in order to infer the current position. Our method takes inspiration from these two lines of work by first using the in-network cross-correlation and second, our experiments show that using renders is sufficient to *locate* a specific object instance from a single RGB image.

4.5 Network architecture

We first introduce an overview of our proposed network architecture, depicted in fig. 4.2. Then, we discuss the two main stages of our architecture: 1) correlation and 2) object detection. The correlation stage borrows from classical template matching methods, where the template of an object is compared to the query image in a sliding-window fashion. The second stage is inspired from the recent literature in class-based object detection.

4.5.1 Architecture overview

We design an architecture that receives knowledge of the object as input, computes the template correlation as a first stage, and regresses bounding boxes around the object from the correlation results in a second stage. As shown in fig. 4.2, the network takes as input the RGB query image and two types of templates: 1) a *global* template used as an object attention mechanism to specialize early features in the backbone network; and 2) a *local* template that helps extract viewpoint-related features. Each template is an RGB image representing the rendered 3D object from a given viewpoint on a black background, concatenated with its binary mask to form four-channel images. The templates are obtained with a fast OpenGL render of the object with diffuse reflectance, ambient occlusion and lit by a combination of one overhead directional light and constant ambient lighting.

4.5.2 Correlation stage

The query image is first processed by a conventional backbone to extract a latent feature representation. The global template is fed to an "Object Attention Branch" (OAB), which injects a set of tunable filters early into this backbone network such that the features get specialized to the particular object instance. On the other hand, the local template is consumed by the "Pose-Specific Branch" (PSB) to compute an embedding of the object. The resulting features are then correlated with the backbone features using simple cross-correlation operations. Note that at test time, the backbone (85% of total computing) is processed only once per instance, while the second stage is computed for each template.

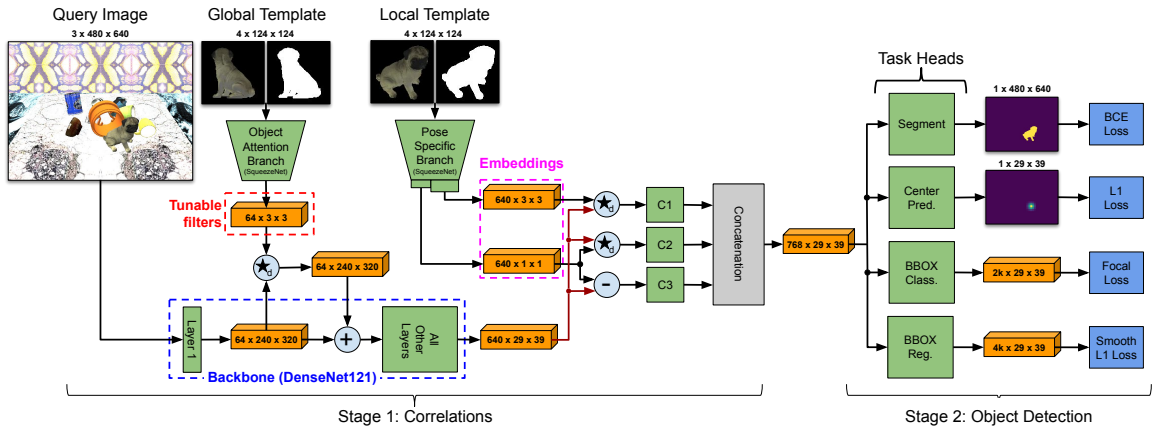


Figure 4.2 – Our proposed architecture. In stage 1, the network learns to localize an object solely from a set of templates. Object-specific features are learned by the "object attention" and "pose-specific" branches, and are subsequently correlated/subtracted with the generic features of the backbone network. In stage 2, the network leverages the learned representation to perform different tasks: binary segmentation, center and bounding box prediction. At test time, a single "global" template is randomly selected, while several "local" templates are combined.

Backbone network. The role of the backbone network is to extract meaningful features from the query image. For this, we use a DenseNet121 [19] model pretrained on ImageNet [6]. Importantly, this network is augmented by adding a set of tunable filters between the first layer of the backbone (7×7 convolution layer with stride 2) and the rest of the model. These tunable filters are adjusted by the Object Attention Branch, described below.

Object attention branch (OAB). It has been widely studied that using a pretrained backbone provides better features initialization [28]. For a task related to template matching, this however limits the feature extraction process to be generic and not specialized early on to a particular instance (e.g. it is not necessary to have a high activation on blue objects if we are looking for a red object.). Thus, a specialized branch named "Object Attention Branch" (OAB) guides the low-level feature extraction of the backbone network by injecting high-level information pertaining to the object of interest. The output of the OAB can be seen as tunable filters, which are correlated with the feature map of the first layer of the backbone network. The correlation is done within a residual block, similarly to [10]. The ablation study in sec. 4.7.3 demonstrates that these tunable filters are instrumental in conferring to a fixed backbone the ability to generalize to objects not seen during training.

The OAB network is a SqueezeNet [20] pretrained on ImageNet, selected for its relatively small memory footprint and good performance. In order to receive a four-channel input (RGB and binary mask), an extra channel is added to the first convolution layer. The pretrained weights for the first three channels are kept and the weights of the fourth channel are initialized by the Kaiming method [9]. During training, a different pose of the target object is sampled at

each iteration. For testing, a random pose is sampled once and used on all test images.

Pose-specific branch (PSB). Since an object instance can greatly vary depending on its viewpoint, a "pose-specific branch" (PSB) is trained to produce a high-level representation (*embeddings*) of the input object under various poses. This search, based on learned features, is accomplished by depth-wise correlations *and* subtraction with 1×1 local templates applied on the backbone output feature map. This correlation/subtraction approach is inspired by [1], where they have demonstrated an increased detection performance when combining these two operations with 1×1 embeddings. Siamese-based object trackers [2, 39] also use correlations, but with embeddings of higher spatial resolution. We found beneficial to merge these two concepts in our architecture, by using depth-wise correlations (denoted as $\star_d$) in both 1×1 and 3×3 spatial dimensions. The first one is devoid of spatial information, whereas the second one preserves some of the spatial relationships within a template. We conjecture that this increases sensitivity to orientation, thus providing some cues about the object pose.

This PSB branch has the same structure and weight initialization as the OAB, but is trained with its own specialized weights. The output of that branch are two local template embeddings: at 1×1 and 3×3 spatial resolution respectively. Depth-wise correlations (1×1 and 3×3) and subtractions (1×1) are applied between the embeddings generated by this branch and the feature maps extracted from the backbone. All of them are processed by subsequent 3×3 convolutions (C1–C3) and are then concatenated.

At test time, the object viewpoint is not known. Therefore, a stack of templates from multiple viewpoints are provided to the pose specific branch. Processing time can be saved at runtime by computing the templates embeddings in an offline phase. Note that the correlation between the local templates and the extracted features is a fast operation and can be easily applied in batch. The backbone network is only processed once per object instance.

4.5.3 Object detection stage

The second stage of the network deals with estimating object information from the learned correlation map. The architecture comprises a main task (bounding box prediction) and two auxiliary tasks (segmentation and center prediction).

Bounding box prediction. The bounding box classification and regression tasks are used to predict the presence and location of the object respectively (as in [24]). The classification head predicts the presence/absence of the object for k anchors at every location of the feature map while the regression head predicts a relative shift on the location (x, y) and size $(\text{width}, \text{length})$ with respect to every anchor. In our method, we have $k = 24$: 8 scales (30, 60, 90, 120, 150, 180, 210 and 240 pixels) and 3 different ratios (0.5, 1 and 2). Both heads are implemented as 5-layer convolution branches [24]. Inspired from RetinaNet [24], anchors with

an Intersection-over-Union (IoU) of at least 0.5 are considered as positive examples, while those with IoU lower than 0.4 are considered as negatives. The other anchors between 0.4 and 0.5 are not used. At test time, bounding box predictions for all templates are accumulated and predictions with an (IoU) > 0.5 are filtered by Non-Maximum Suppression (NMS). Also, for each bounding box prediction, a depth estimation can be made by multiplying the depth at which the local template was rendered with the size ratio between the local template size (124 pixels) and the prediction size. Predictions that have a predicted depth outside the chosen range of [0.4, 2.0] meters, which is a range that fits to most tabletop settings, are filtered out.

Segmentation and center prediction. The segmentation head predicts a pixel-wise binary mask of the object in the scene image at full resolution. The branch is composed of 5 convolution layers followed by $2\times$ bilinear upsampling layers. Additionally, the center prediction head predicts the location of the object center at the same resolution than the correlation map (29×39) to encourage a strong correlation. The correlation channels are compressed to a single channel heatmap with a 1×1 convolution layer.

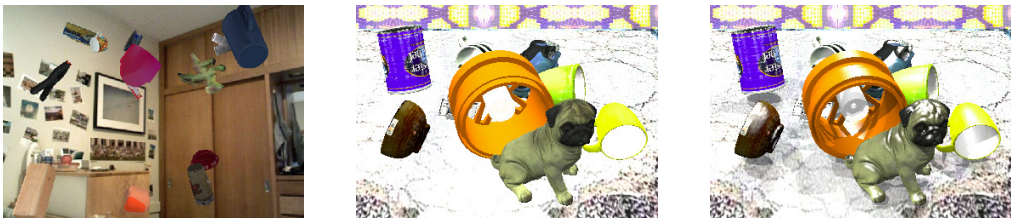
Loss Functions. The network is trained end-to-end with a main (bounding box detection) and two auxiliary (segmentation and center prediction) tasks. As such, the training loss $\ell_{\text{train}} = \lambda_1 \ell_{\text{seg}} + \lambda_2 \ell_{\text{center}} + \ell_{\text{FL}} + \ell_{\text{reg}}$, where ℓ_{seg} is a binary cross-entropy loss for segmentation, ℓ_{center} is an L_1 loss for the prediction of the object center in a heatmap, ℓ_{FL} is a focal loss [24] associated with the object presence classification and ℓ_{reg} is a smooth-L_1 loss for bounding box regression. The weights λ_1, λ_2 were empirically set to 20.

4.6 Training data

In this section, we detail all information related to the input images (query and templates) during training. In particular, we define how the synthetic images are generated and how the dataset is augmented.

4.6.1 Domain randomization training images

We rely on 125 different textured 3D models gathered in majority from the various datasets of the 6D pose estimation benchmark [16] (excluding Linemod [11] since it is used for evaluation). Our fully-annotated training dataset is generated with a physic-based simulator similar to [27], for which objects are randomly dropped on a table in a physical simulation. Every simulation is done in a simple cubic room (four walls, a floor and a ceiling) containing a table placed on the floor in the middle of the room. Inspired from the success of domain randomization [35, 36], we added randomness to the simulation parameters in order to reduce the domain gap between synthetic and real images. The following parameters are randomized: the texture of the environment (walls, floor and table), lighting (placement, type, intensity and color),



(a) (b) (c)

Figure 4.3 – Examples from our domain randomization training set. In (a), objects are randomly placed in front of the camera and rendered using OpenGL with a background sampled from Sun3D dataset [41]. In (b) and (c), a physical simulation is used to drop several objects on a table with randomized parameters (camera position, textures, lighting, materials and anti-aliasing). For each render, 2 variations are saved: one with simple diffuse materials and without shadows (b), and one with more sophisticated specular materials and shadows (c).

object materials (diffuse and specular reflection coefficients) and anti-aliasing (type and various parameters).

Renders. Our physics-based domain randomization dataset is composed of 10,000 images. To generate these images, we ran 250 different simulations with different sets of objects (between 4 and 13 objects in each simulation). In 50% of the simulations, objects were automatically repositioned to rest on their bottom/main surface to replicate a bias found in many tabletop datasets. For each simulation, 20 camera positions were randomly sampled on half-spheres of radius ranging from 0.8 to 1.4 meters, all pointing towards the table center with random offsets of ± 15 degrees for each rotation axis. For each sampled camera position, two image variations were rendered: one with realistic parameters (containing reflections and shadows) as shown in fig. 4.3-(c) and the other without, as shown in fig. 4.3-(b). Tremblay et al. [37] showed that using different kinds of synthetic images reduced the performance gap between synthetic and real images. Accordingly, we have generated an additional set of 10,000 simpler renders using OpenGL. For this, we rendered objects in random poses on top of real indoor backgrounds sampled from the Sun3D dataset [41] (fig. 4.3-(a)).

Labels. After the simulations, we kept the 6 degree of freedom pose of each object as the ground truth. We used the pose together with the 3D model to generate a visibility mask for the segmentation task, and projected the center of the 3D model in the image plane to generate the center heatmap. The ground-truth heatmap is a 2D Gaussian with an amplitude of 1 and a variance of 5 at the projected center of the object at an image resolution equivalent to the output of the network.

4.6.2 Templates

The following section describes the template generation procedure for training. We also remind the different procedure used at test time, as described in sec. 4.5.2.

For each training iteration, one of the objects from the query image is selected as the target object and all the others are considered as background. All templates are rendered with a resolution of 124×124 pixels. To render consistent templates from multiple objects of various size, we adjust the distance of the object so that its largest length on the image plane falls in the range of 100 to 115 pixels. The borders are then padded to reach the size of 124×124.

Global template (OAB): In an offline phase, 240 templates are generated for each 3D model by sampling 40 viewpoints on an icosahedron with 6 in-plane rotations per viewpoint. During training, one of the 240 templates is sampled randomly for each iteration. At test time, a single one is randomly selected for all experiments.

Local template (PSB): We apply perturbations on the orientation of the template image by sampling a random rotation axis and rotation magnitude, and adding that perturbation to the ground truth viewpoint before rendering the local template. The impact of using different rotation magnitude is quantified in table 4.2, with best performance obtained with random rotations perturbation in the range of 20–30° to the ground truth viewpoint. At test time, a stack of 160 templates rendered from 16 viewpoints is used.

4.7 Experiments

In this section, we provide details on the training procedure and on the dataset and metrics used to evaluate our approach. We also describe the various ablation studies that validate our design choices. Finally, we present an extensive evaluation against the state-of-the-art methods.

4.7.1 Training details

Our complete network is trained for 50 epochs with AMSGrad [30].We used a learning rate of 10^{-4} with steps of 0.1 at epochs 20 and 40, a weight decay of 10^{-6} and mini batches of size 6. We used 1k renders as a validation set and used the remaining 19k of the generated dataset (OpenGL and physics-based) for training. Each epoch, the network was trained for 1,300 iterations and images are sampled with a ratio of 80/20 respectively from the physics-based and OpenGL renders. Once the training was complete, the network with the smallest validation loss (computed at the end of each epoch) was kept for testing.

4.7.2 Datasets and metrics

We evaluate on the well-known Linemod [12] and Occluded Linemod [3] datasets. Linemod consists of 15 sequences of real objects containing heavy clutter where the annotations of a single object are available per sequence. Occluded Linemod is a subset of Linemod, where annotations for 8 objects have been added by [3]. Keeping in line with previous work, we only

Network	Δ performance (%)
w/o tunable filters (OAB)	-19.76
w/o auxiliary tasks	-7.73
w/o 3×3 correlation (PSB)	-5.37

Table 4.1 – Network architecture ablation study. Removing tunable filters resulted in the most notable performance drop of almost 20% while dismissing auxiliary tasks and 3x3 correlations decreased accuracy by 7.73% and 5.37% respectively.

keep the prediction with the highest score for each object and use the standard metrics listed below. We use a subset containing 25% of the Linemod dataset for the ablation studies.

Linemod. The standard metric for this dataset is the "2D bounding box" [3], which calculates the ratio of images for which the predicted bounding box has an intersection-over-union (IoU) with the ground truth higher than 0.5.

Occluded Linemod. The standard mean average precision (mAP) is used to evaluate the performance of multi-object detection. To allow for direct comparison, we regroup the predictions made for different objects and apply NMS on predictions with an IoU > 0.5. We use the same methodology as in [3]: the methods are evaluated on 13 of the 15 objects of the Linemod dataset (the "bowl" and "cup" objects are left out). Of the remaining 13 objects, 4 are never found in the images, yet those are still detected and kept in the evaluation (as an attempt to evaluate the robustness to missing objects). The mAP is therefore computed by using all the predictions on the 9 other objects left.

4.7.3 Ablation studies

Network architecture. We evaluate the importance of different architecture modules (presented in sec. 4.5). For each test, a specific module is removed and its performance is compared to the full architecture. Tab. 4.1 shows that removing the "Object Attention Branch" resulted in the largest performance drop (almost 20%). Also, removing the higher-resolution 3×3 embeddings and auxiliary tasks reduced performances by approximately 5% and 8% respectively.

Importance of local template perturbation during training. A perfect match between the template pose and the target object pose in the scene is unlikely. As such, the training procedure must take this into account by adding orientation perturbations to local templates at train time. Here, we investigated what is the desirable magnitude of such perturbations. In tab. 4.2, a random rotation of 0° represents local templates selected with the same orientation as the object in the scene. Perturbations are then added by randomly sampling a rotation axis (in spherical coordinates) and a magnitude. A network was retrained for each amount of perturbation. Tab. 4.2 illustrates that perturbations of 20–30° seems to be optimal. Networks

Random rotations	0°	±10°	±20°	±30°	±40°	±180°
Δ performance (%)	-4.33	-3.12	0	-0.42	-5.18	-16.07

Table 4.2 – Ablation study on random rotations applied to the local template orientation during training. A random rotation of 0° represent a strict training where the local template perfectly matches the ground truth object while ±180° is equivalent to a random rotation angle. A random perturbation of ±20/30° provides the best compromise.

# of templates	80	160	320	oracle
Δ performance (%)	-2.80	0	+0.03	+16.75
runtime (ms)	230	430	870	60

Table 4.3 – Evaluating the bounding box detection performance and runtime for various numbers of local templates at test time. While runtime grows linearly, the performance gain saturates around 160 templates. The oracle sets an upper bound of performance by providing a single template with the ground truth object pose as input thus greatly reducing uncertainty.

trained with too small perturbations may not be able to detect objects under all their possible configurations, resulting in small performances drop of less than 5%, and those trained with too big perturbations are more prone to false detections (the network is trained to allow for bigger differences in appearance and shape between the template and scene object), resulting in a bigger drop of 16% for rotations of 180°.

Number of local templates. The impact of providing various numbers of local templates to the network at test time is evaluated, both in terms of accuracy and speed, in tab. 4.3. Timings are reported on a Nvidia GeForce GTX 1080Ti. To generate a varying number of templates, we first selected 16 pre-defined viewpoints spanning a half-sphere over the object. Each template subsequently underwent 5 (80 templates), 10 (160 templates) and 20 (320 templates) in-plane rotations. Tab. 4.3 compares performance with that obtained with an oracle who used a template with the ground truth pose. Performance ceases to improve beyond 160 templates.

Global template selection. In tab. 4.4, we show that the object pose of the global template does not impact significantly the detection performance. For the first test, we report the average performance of 5 different evaluations in which a random global template was selected. The performance slightly improved compared to the random template used in all other tests, suggesting that the template selection in every other test was suboptimal. However, it also shows that the templates were not cherry-picked for optimal performance on the test datasets. Secondly, we show that using a template of the good object is primordial. Using empty templates (all 0's) or providing templates from another object results in a dramatic performance drop of more than 30%, thus hinting about the discriminative power of the OAB.

Global template selection	Random pose	Empty	Wrong object
Δ performance (%)	+1.21	-32.47	-38.15

Table 4.4 – Robustness towards different selection of global templates. Evaluating with different random templates of the same object (Random pose), empty global templates (Empty) and random templates of other objects (Wrong object). The OAB uses global information about the object while being invariant to the provided pose.

# of objects	15	30	63	90	125
Δ performance (%)	-31.58	-15.97	-10.42	-3.58	0

Table 4.5 – Impact of the number of objects used during training. Performance increases rapidly when few objects are used at training but plateaus as more objects are added.

Number of objects in the training set. The network was retrained on subsets of objects of the synthetic dataset. The remaining objects were considered as background clutter. Tab. 4.5 shows the performance w.r.t the quantity of objects used during training. While using few objects still performs reasonably well, more objects does improve generalization.

Similar objects in the training set. While no single object were present in both training and test sets, it is possible that the training set contained objects that shared similarities to objects in the test set. To evaluate the potential impact this might have, we removed all cups from our training set (13 were found), trained a network on the resulting set, and evaluated its performance on the test set. Doing so reduced the overall score by less than 1%, but the average performance solely on cups slightly improved (not statistically significant). This experiment demonstrates that the network does not overfit on a particular class of instances.

4.7.4 Comparative evaluation to the state of the art

We report an evaluation on Linemod and Occluded Linemod (OL) in tab. 4.6 and compare with other state-of-the-art RGB-only methods. Competing methods are divided into 2 main groups: those who do know the test objects at train time ("known objects"), and those who do not. Approaches such as [4, 17, 21, 43] are all learning-based methods that were specifically trained on the objects. On the other hand, [11, 34] and [1, 39] are respectively template matching and learning-based methods that do not include a specific training step targeted towards specific object instances. It is worth noting that even though [34] is classified as not needing known objects at training time, it still requires an initialization phase using real images (to build a dictionary of histogram features). As in [4], they thus use parts of the Linemod dataset as a training set that covers most of the object viewpoints. These methods have therefore an unfair advantage compared to our approach and Line-2D, since they leverage domain-specific information (lighting, camera, background) of the evaluation dataset.

Methods	Known objects	Real images	Linemod (2D BBox)	Occluded Linemod (mAP)
Brachmann et al. [4]	Yes	Yes	97.50	51.00
SSD-6D [21]	Yes	No	99.40	38.00
DPOD [43]	Yes	No	N/A	48.00
Hodan et al. [17]	Yes	No	N/A	55.90*
Tjaden et al. [34]	No	Yes	78.50	N/A
LINE-2D [11]	No	No	86.50	21.0
TDID correlations [1]	No	No	54.37	34.13
SiamMask correlations [39]	No	No	68.23	41.47
Ours	No	No	77.92	50.71

Table 4.6 – Quantitative comparison to the state of the art, with 2D bounding box metric on Linemod and mAP on Occluded Linemod (OL). The 2D bounding box metric calculates the recall for the 2D bounding boxes with the highest prediction score. For both metrics, predictions are considered good if the IoU of the prediction and the ground truth is at least 0.5 (0.75 for Hodan et al. [17]). The methods are separated first according to their prior knowledge of test objects and then if real images similar to the test set are used to optimize the performance. Our approach is the most robust of all methods that were not trained for the test objects, having a good score on Linemod and the best score on Occluded Linemod.

Our method is evaluated without prior knowledge of the Linemod objects. It can be directly compared with Line-2D [11] which also uses templates as input. On the standard Linemod dataset, Line-2D outperforms our method by 8.5% on the "2D bounding box" metric. The better results of Line-2D on Linemod can be explained in part by an additional and naive post-processing color-based check that rejects false positives [4] while we report the performance of our approach without any post-processing. We note that this naive approach fails if minor occlusions occurs. In contrast, our method outperforms Line-2D by almost 30% in mAP on the more difficult Occluded Linemod. Our approach also provides competitive performance that is on par or close to all other methods that test on known objects and/or have access to real images. Fig. 4.4 shows qualitative results on Occluded Linemod. We also compare our approach with TDID [1] and SiamMask [39]. We replaced their original backbones by the same architecture (DenseNet) we are using. As specified by those methods, a siamese backbone replaced our 2 branches approach (OAB and PSB). TDID uses a 1×1 embedding whereas a 3×3 embedding is used for SiamMask. All implementations were trained on the same task following the same procedure than our approach and their scores are reported in tab. 4.6. Overall, our proposed approach significantly outperforms these two baselines.

Figure 4.4 – Qualitative results on the Occluded Linemod dataset [3], showing good (green), false (blue) and missed (red) detections. For reference, the 15 objects are shown in the bottom row (image from [16]). To generate these results, all objects (except objects 3 and 7) are searched in each image.

4.8 Discussion

We have proposed a method for detecting specific object instances in an image that does not require knowledge of the object at training time. At test time, the proposed network takes multiple viewpoints of the object as input, and predicts its location from a single RGB image. Our experiments show that while the network has not been trained with any of the test objects, it is significantly more robust to occlusion than previous template-based methods (30% improvement in mAP over Line-2D [11]). It is also highly competitive with networks that are specifically trained on the object instances.

Limitations. False positives arise from clutter with similar color/shape as the object, as shown in fig. 4.4. We hypothesize that our late correlation at small spatial resolution (templates of 3×3 and 1×1) prevents the network from leveraging detailed spatial information related to the object shape. Another limitation is that the method requires 0.43s to detect a single object instance in an image (c.f. tab. 4.3), scaling linearly with the number of objects. The main reason for this is the object attention branch (OAB), which makes the backbone features instance-specific via tunable filters, which needs to be recomputed for each object. Also, while capturing a 3D model has become increasingly simpler (it takes less than 5 minutes

with commodity hardware [14]), this may not always be practical. While our experiments rely on such 3D models to allow for quantitative evaluation on standard datasets for which only the 3D model is available, obtaining multiple viewpoints of an object could also be done simply by photographing it against a uniform background.

Future directions. By providing a generic and robust 2D instance detection framework, this work opens the way for new methods that can extract additional information about the object, such as its full 6-DOF pose. We envision a potential cascaded approach, which could first detect unseen objects, and subsequently regress the object pose from a high-resolution version of the detection window.

4.9 Supplementary Material

4.9.1 Per object performances

In table 4.6, we reported the performance of our approach on Linemod [12] and Occluded Linemod [3] datasets. We extend the reported results by showing the performance of our approach on each object in tables 4.7 and 4.8. Object with their corresponding indices can be viewed in fig. 4.5.

Object ID	2D BBox metric (%)
1	89.16
2	71.50
3	94.00
4	46.88
5	92.47
6	80.75
7	82.74
8	77.19
9	63.31
10	96.89
11	89.51
12	67.83
13	87.67
14	86.39
15	42.48
Mean	77.92

Table 4.7 – Performances on the 2D bounding box metric for each object of the Linemod dataset.

Object ID	Mean Average Precision (mAP)
1	36.58
2	55.92
5	73.49
6	29.18
8	55.20
9	77.48
10	52.79
11	16.26
12	59.52
Mean Average Precision (mAP)	50.71

Table 4.8 – Average precision for each object evaluated on the Occluded Linemod dataset.

Figure 4.5 – All 15 objects in the Linemod dataset (taken from [16]).

4.9.2 Domain randomization training images

Additional examples of domain randomization images generated with our simulator are shown in fig. 4.6.

4.9.3 Qualitative results on Linemod dataset

We show examples of good and bad predictions on Linemod dataset in fig. 4.7.

4.9.4 Qualitative results on Occluded Linemod

We show additional qualitative results on Occluded Linemod in fig. 4.8 to expand results shown in fig. 4.4 of the paper.

4.9.5 Architecture details

To expand fig. 4.2 of the main paper, we show more detailed networks in fig 4.9.

Figure 4.6 – More domain randomization images generated with our simulator

Figure 4.7 – Qualitative results on Linemod dataset [12] with predictions (yellow) and ground-truths (red). The first two rows show good predictions while the last row shows examples of bad predictions.

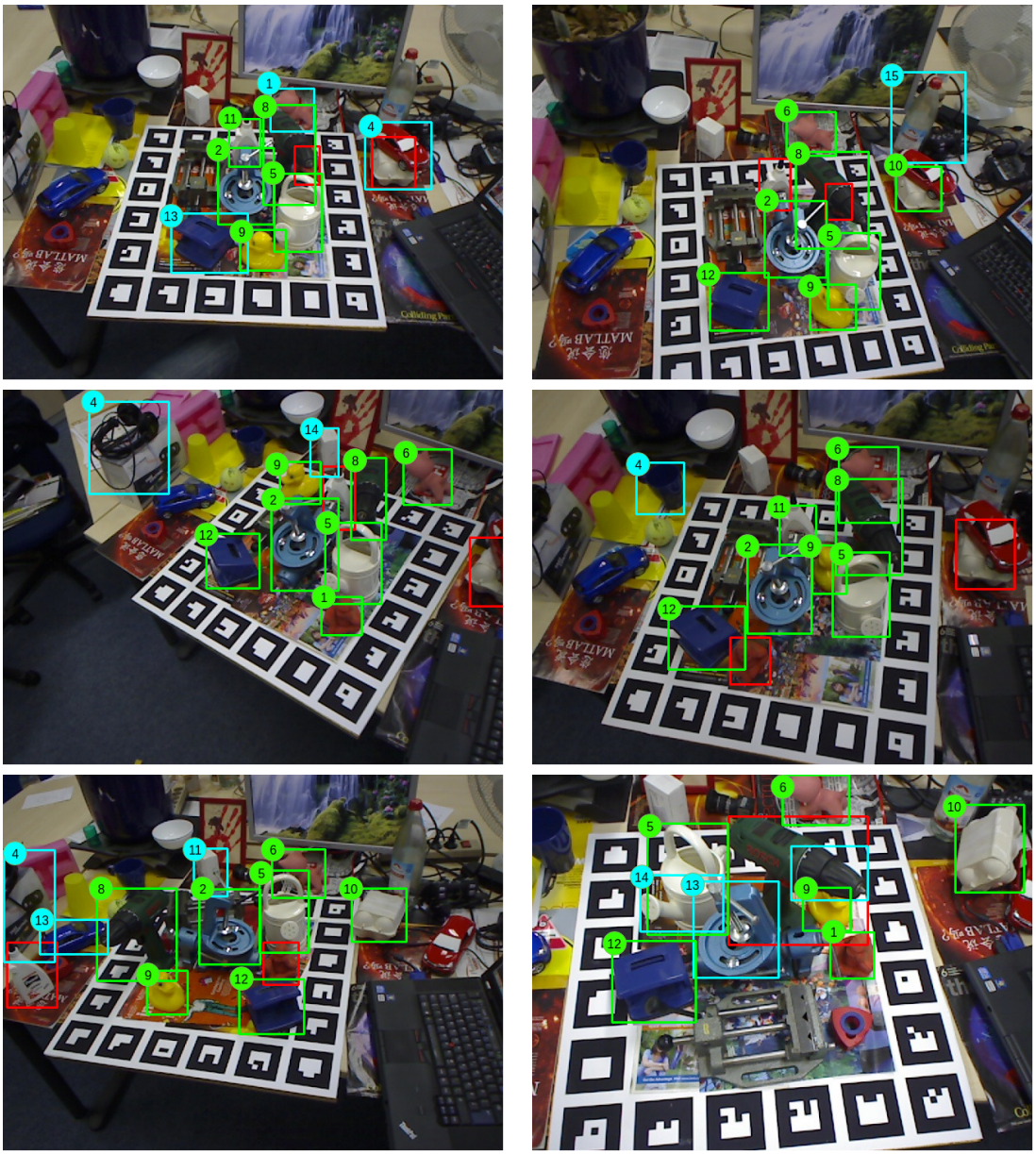


Figure 4.8 – More qualitative results on the Occluded Linemod dataset [3].

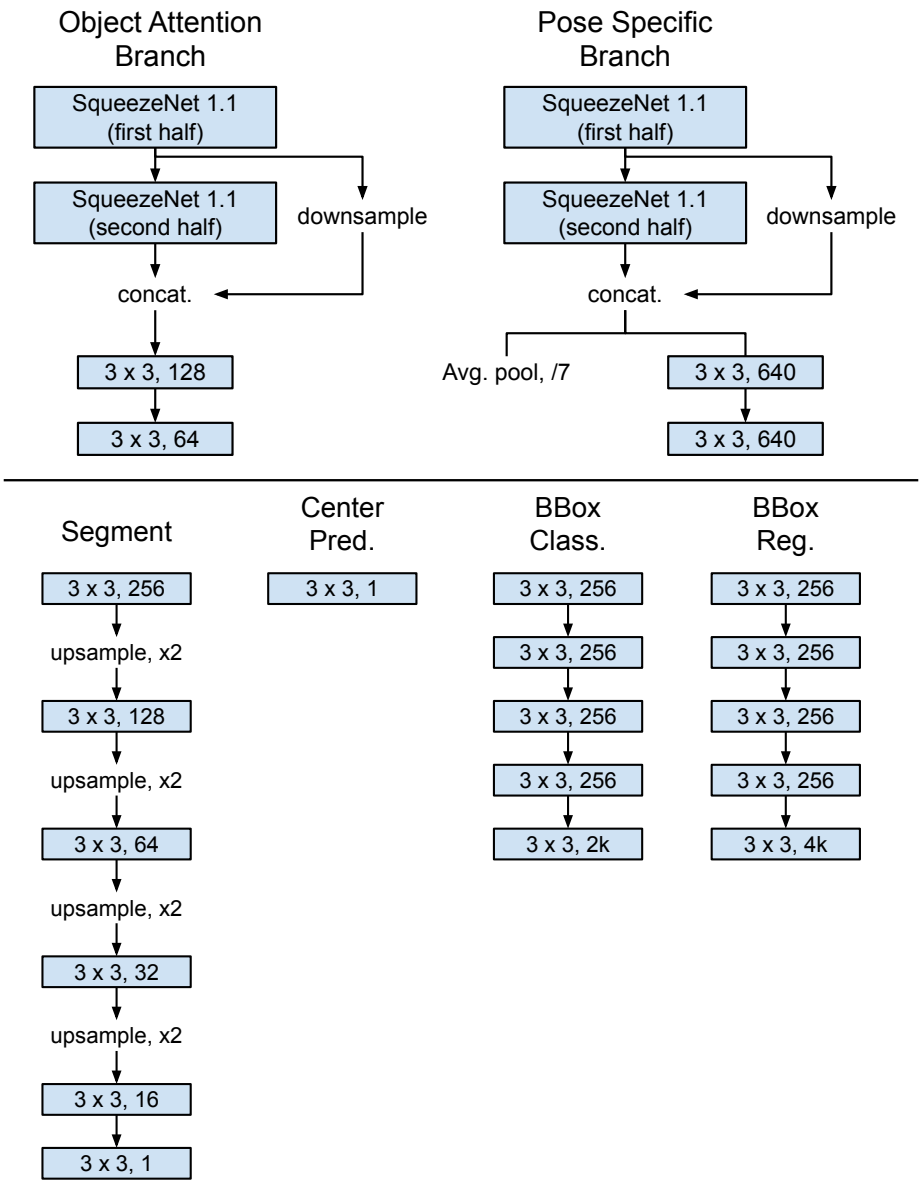


Figure 4.9 – Detailed networks

Bibliography

[1] Phil Ammirato, Cheng-Yang Fu, Mykhailo Shvets, Jana Kosecka, and Alexander C Berg. Target driven instance detection. *arXiv preprint arXiv:1803.04610*, 2018.

[2] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European Conference on Computer Vision*, 2016.

[3] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6D object pose estimation using 3D object coordinates. In *European Conference on Computer Vision*, 2014.

[4] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, et al. Uncertainty-driven 6D pose estimation of objects and scenes from a single rgb image. In *IEEE Conference on Computer Vision And Pattern Recognition*, 2016.

[5] Achal Dave, Pavel Tokmakov, Cordelia Schmid, and Deva Ramanan. Learning to track any object. *arXiv preprint arXiv:1910.11844*, 2019.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision And Pattern Recognition*, 2009.

[7] Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, and Tae-Kyun Kim. Recovering 6D object pose and predicting next-best-view in the crowd. In *IEEE Conference on Computer Vision And Pattern Recognition*, 2016.

[8] Mathieu Garon, Denis Laurendeau, and Jean-François Lalonde. A framework for evaluating 6-DOF object trackers. In *European Conference on Computer Vision*, 2018.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision*, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision And Pattern Recognition*, 2016.

[11] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-

less objects in heavily cluttered scenes. In *IEEE International Conference on Computer Vision*, 2011.

[12] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Asian Conference on Computer Vision*, 2012.

[13] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On pre-trained image features and synthetic images for deep learning. In *European Conference on Computer Vision*, 2018.

[14] Stefan Hinterstoisser, Olivier Pauly, Hauke Heibel, Marek Martina, and Martin Bokeloh. An annotation saved is an annotation earned: Using fully synthetic training for object detection. In *IEEE International Conference on Computer Vision Workshops*, 2019.

[15] Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-less: An RGB-D dataset for 6D pose estimation of texture-less objects. In *IEEE Winter Conference on Applications of Computer Vision*, 2017.

[16] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders GlentBuch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, et al. Bop: Benchmark for 6D object pose estimation. In *European Conference on Computer Vision*, 2018.

[17] Tomas Hodan, Vibhav Vineet, Ran Gal, Emanuel Shalev, Jon Hanzelka, Treb Connell, Pedro Urbina, Sudipta N Sinha, and Brian Guenter. Photorealistic image synthesis for object instance detection. *arXiv preprint arXiv:1902.03334*, 2019.

[18] Ting-I Hsieh, Yi-Chen Lo, Hwann-Tzong Chen, and Tyng-Luh Liu. One-shot object detection with co-attention and co-excitation. In *Advances in Neural Information Processing Systems*, pages 2725–2734, 2019.

[19] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision And Pattern Recognition*, 2017.

[20] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[21] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3D detection and 6D pose estimation great again. In *IEEE International Conference on Computer Vision*, 2017.

[22] Xin Li, Chao Ma, Baoyuan Wu, Zhenyu He, and Ming-Hsuan Yang. Target-aware deep tracking. In *IEEE Conference on Computer Vision And Pattern Recognition*, 2019.

[23] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6D pose estimation. In *European Conference on Computer Vision*, 2018.

[24] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision*, 2017.

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, 2016.

[26] Fabian Manhardt, Wadim Kehl, Nassir Navab, and Federico Tombari. Deep model-based 6D pose refinement in rgb. In *European Conference on Computer Vision*, 2018.

[27] Chaitanya Mitash, Kostas E Bekris, and Abdeslam Boularias. A self-supervised learning system for object detection using physics simulation and multi-view pose estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

[28] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[29] Mahdi Rad and Vincent Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth. In *IEEE International Conference on Computer Vision*, 2017.

[30] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *International Conference on Learning Representations*, 2019.

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015.

[32] Colin Rennie, Rahul Shome, Kostas E Bekris, and Alberto F De Souza. A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters*, 1(2):1179–1185, 2016.

[33] Alykhan Tejani, Danhang Tang, Rigas Kouskouridas, and Tae-Kyun Kim. Latent-class hough forests for 3D object detection and pose estimation. In *European Conference on Computer Vision*, 2014.

[34] Henning Tjaden, Ulrich Schwanecke, and Elmar Schomer. Real-time monocular pose estimation of 3D objects using temporally consistent local color histograms. In *IEEE International Conference on Computer Vision*, 2017.

[35] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

[36] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *IEEE On Computer Vision And Pattern Recognition Workshops*, 2018.

[37] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018.

[38] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016.

[39] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. In *IEEE Conference on Computer Vision And Pattern Recognition*, 2019.

[40] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6D object pose estimation in cluttered scenes. In *Robotics: Science and Systems (RSS)*, 2018.

[41] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *IEEE International Conference on Computer Vision*, 2013.

[42] Sergey Zakharov, Wadim Kehl, and Slobodan Ilic. Deceptionnet: Network-driven domain randomization. *IEEE International Conference on Computer Vision*, 2019.

[43] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Dpod: Dense 6D pose object detector in rgb images. In *IEEE International Conference on Computer Vision*, 2019.

Conclusion

In this thesis, we proposed different strategies to facilitate the use of deep learning approaches for object detection in the context of robotic grasping. We first proposed to combine the strengths of deep networks and engineered solutions in cascaded approaches as a way to improve the performances of the engineered solutions while keeping their interpretability. We also proposed different image acquisition and annotation strategies such as using simulation and weak labels to decrease the usual required effort to acquire and annotate deep learning datasets. Simulation, for instance, allows to get nearly labor-free annotation on data. Moreover, it can also improve generalization performances by enabling many conditions (lighting, backgrounds, camera parameters, object poses, etc.) that can be difficult to reproduce in the real world. It can therefore help to reduce biases that can happen when only using real images.

Some biases, however, can be worth keeping. If something is expected to be constant between training and testing conditions, overfitting on that particular aspect can improve performance. For instance, in [35], they showed an absolute improvement of 13% on the mAP metric when training with objects in the same context (objects placed in a shelf) as the test set compared to when training with objects placed out of context (not in a shelf). In our case, even when training with synthetic images, we mostly used images in similar contexts to the test settings (tabletop with small objects on it). Hence, the reported performances of our approaches could differ if evaluated in different contexts.

Also, most datasets for tabletop settings contain really few objects (10 to 30). It can therefore be relatively easy for networks to distinguish them, especially if the objects are quite different. For this reason, the reported performances on these datasets are circumstantial and may not necessarily represent how an approach would perform in the real world with a more diverse group of objects.

Limitations and Future Work

In this last part, we review some limitations of our approaches and propose potential research directions.

In our first approach presented in Chapter 2, we proposed a cascaded approach with indepen-

dent blocks for object proposals and classification. Our main motivation for using separate blocks was that it was easier and faster to acquire a dataset for object classification than for object detection, which would have required bounding box annotations. Another limitation of our proposed approach is that we did not use the location of the predictions in the ranking. The main idea of our method was to use a global object ranking to limit the number of objects that are tested with template matching approaches, which resulted in less false detections. A combination of the ranking with the location of the top prediction (or top k predictions) would likely improve the performance of our approach.

In the work presented in Chapter 3, we used a weakly-supervised network to train with classification annotations to retrieve the location of objects. One of the main limitations of the approach is that we did not use all of the available annotations in simulation (bounding boxes, pixelwise). Instead, we used the same training losses for both synthetic and real images. We tried to simultaneously train for segmentation on synthetic images, but it did not improve our results. Another limitation is the bias of the datasets that were used. For the experiments in which we trained our network with real images, we randomly sampled training images. While we did not train using the same images that were in the test set, some could look really similar since the datasets were acquired from videos. Also, both datasets used nearly identical settings (background, lighting, clutter, etc.) for most of the sequences, making it easy for networks to overfit on them. With such biases in the datasets, it is difficult to assess the real generalization performance of the approach.

Our last work in Chapter 4 was inspired from template matching approaches, but was only applied for detection. The 6D pose of objects could have been estimated from the template that contributed the most to the target object detection, but in unreported experiments, we found that it did not work well in practice. We attribute this to the fact that the "Pose Specific Branch" correlates templates with the input at the end of the network (where the spatial resolution of the feature maps is at its minimum) and that the network is not optimized for that specific task. Therefore, during training, our network does not extract any pose-related features. There are different possible solutions to retrieve the object orientation that we did not investigate, such as adding a loss for the pose. However, unlike other pose estimation methods, our approach is applied to unknown objects. Thus, a relative pose error between the pose of the template and the target object in the input image should be estimated instead of an absolute pose. There could also be an advantage of correlating features at a higher resolution or at different scales using a feature pyramid. Moreover, the templates were synthetic renders, so the object masks were perfect. If we tried to replicate our approach with real objects as templates, imperfect object masks could potentially impact the performance.

Also, in general, the appearance of objects may shift significantly under different perspectives, scales, illumination and occlusions. It makes them difficult to detect under all their possible variations. To improve the robustness of detectors, a simple solution could be to take advantage

of multiple viewpoints. For instance, different cameras could be installed at different locations around a scene or a camera mounted on a robotic arm could be moved around and acquire an image sequence. This would allow detectors to obtain more semantic information about the scene and likely obtain easier viewpoints for certain objects (bigger size, less occluded, etc.), thus improving the detector's effectiveness. This research topic is not well explored in the litterature (less than 1% of papers in recent object detection surveys [40, 51] used "multi-view" in their titles) and can be explained mostly by the lack of datasets (standard challenging datasets are mostly single-view), a recurring theme for deep learning.

Bibliography

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

[2] Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *Advances in neural information processing systems*, pages 523–531, 2016.

[3] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6D object pose estimation using 3D object coordinates. In *European Conference on Computer Vision*, 2014.

[4] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, et al. Uncertainty-driven 6D pose estimation of objects and scenes from a single rgb image. In *IEEE Conference on Computer Vision And Pattern Recognition*, 2016.

[5] Roberto Calandra, Andrew Owens, Dinesh Jayaraman, Justin Lin, Wenzhen Yuan, Jitendra Malik, Edward H Adelson, and Sergey Levine. More than a feeling: Learning to grasp and regrasp using vision and touch. *IEEE Robotics and Automation Letters*, 3(4):3300–3307, 2018.

[6] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pages 679–698, 1986.

[7] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8789–8797, 2018.

[8] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3642–3649. IEEE, 2012.

[9] Dan C Cireşan, Ueli Meier, Jonathan Masci, Luca M Gambardella, and Jürgen Schmidhuber. High-performance neural networks for visual object classification. *arXiv preprint arXiv:1102.0183*, 2011.

[10] Deen Cockbum, Jean-Philippe Roberge, Alexis Maslyczyk, Vincent Duchaine, et al. Grasp stability assessment through unsupervised feature learning of tactile images. In *2017 IEEE*

International Conference on Robotics and Automation (ICRA), pages 2238–2244. IEEE, 2017.

[11] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[12] Guoguang Du, Kai Wang, and Shiguo Lian. Vision-based robotic grasping from object localization, pose estimation, grasp detection to motion planning: A review. *arXiv preprint arXiv:1905.06658*, 2019.

[13] Thibaut Durand, Taylor Mordan, Nicolas Thome, and Matthieu Cord. Wildcat: Weakly supervised learning of deep convnets for image classification, pointwise localization and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 642–651, 2017.

[14] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1301–1310, 2017.

[15] Qi Fan, Wei Zhuo, Chi-Keung Tang, and Yu-Wing Tai. Few-shot object detection with attention-rpn and multi-relation detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4013–4022, 2020.

[16] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[17] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[18] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[19] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[20] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):930–943, 2003.

[21] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

[22] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[23] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[25] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[26] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[27] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer, 1988.

[28] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4918–4927, 2019.

[29] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[31] Joel A Hesch and Stergios I Roumeliotis. A direct least-squares (dls) method for pnp. In *2011 International Conference on Computer Vision*, pages 383–390. IEEE, 2011.

[32] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Asian conference on computer vision*, pages 548–562. Springer, 2012.

[33] Stefan Hinterstoisser, Olivier Pauly, Hauke Heibel, Martina Marek, and Martin Bokeloh. An annotation saved is an annotation earned: Using fully synthetic training for object instance detection. *arXiv preprint arXiv:1902.09967*, 2019.

[34] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[35] Tomas Hodan, Vibhav Vineet, Ran Gal, Emanuel Shalev, Jon Hanzelka, Treb Connell, Pedro Urbina, Sudipta N Sinha, and Brian Guenter. Photorealistic image synthesis for object instance detection. *arXiv preprint arXiv:1902.03334*, 2019.

[36] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015.

[37] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[38] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[39] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[40] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868, 2019.

[41] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8420–8429, 2019.

[42] Jack Kiefer, Jacob Wolfowitz, et al. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[43] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[44] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[45] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[46] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009.

[47] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

[48] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, 2018.

[49] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[50] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[51] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020.

[52] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[53] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[54] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *Advances in neural information processing systems*, pages 136–144, 2016.

[55] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[56] Shan Luo, Joao Bimbo, Ravinder Dahiya, and Hongbin Liu. Robotic tactile perception of object properties: A review. *Mechatronics*, 48:54–67, 2017.

[57] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A Efros. Ensemble of exemplar-svms for object detection and beyond. In *2011 International conference on computer vision*, pages 89–96. IEEE, 2011.

[58] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004.

[59] Marius Muja and David G Lowe. Flann, fast library for approximate nearest neighbors. In *International Conference on Computer Vision Theory and Applications (VISAPP'09)*, 2009.

[60] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[61] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[62] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(3):4, 2018.

[63] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings*

of the 2017 ACM on Asia conference on computer and communications security, pages 506–519, 2017.

[64] Aayush Prakash, Shaad Boochoon, Mark Brophy, David Acuna, Eric Cameracci, Gavriel State, Omer Shapira, and Stan Birchfield. Structured domain randomization: Bridging the reality gap by context-aware synthetic data. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7249–7255. IEEE, 2019.

[65] Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5822–5830, 2018.

[66] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[67] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[68] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[69] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[70] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[71] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[72] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006*, pages 430–443. Springer, 2006.

[73] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[74] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[75] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Sharathchandra Pankanti, Rogerio Feris, Abhishek Kumar, Raja Giries, and Alex M Bronstein. Repmet: Representative-based metric learning for classification and one-shot object detection. *arXiv preprint arXiv:1806.04728*, 4323, 2018.

[76] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks

via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[77] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[78] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[79] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[80] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[81] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. 1968.

[82] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[83] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[84] Henning Tjaden, Ulrich Schwanecke, and Elmar Schomer. Real-time monocular pose estimation of 3D objects using temporally consistent local color histograms. In *IEEE International Conference on Computer Vision*, 2017.

[85] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

[86] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *IEEE On Computer Vision And Pattern Recognition Workshops*, 2018.

[87] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[88] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[89] Gu Wang, Fabian Manhardt, Jianzhun Shao, Xiangyang Ji, Nassir Navab, and Federico Tombari. Self6d: Self-supervised monocular 6D object pose estimation. *arXiv preprint arXiv:2004.06468*, 2020.

[90] Jianyu Wang, Zhishuai Zhang, Cihang Xie, Yuyin Zhou, Vittal Premachandran, Jun Zhu, Lingxi Xie, and Alan Yuille. Visual concepts and compositional voting. *arXiv preprint arXiv:1711.04451*, 2017.

[91] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1328–1338, 2019.

[92] Xin Wang, Thomas E Huang, Trevor Darrell, Joseph E Gonzalez, and Fisher Yu. Frustratingly simple few-shot object detection. *arXiv preprint arXiv:2003.06957*, 2020.

[93] Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and Liang Lin. Meta r-cnn: Towards general solver for instance-level low-shot learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9577–9586, 2019.

[94] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[95] Yanzhao Zhou, Yi Zhu, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Weakly supervised instance segmentation using class peak response. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3791–3800, 2018.

[96] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[97] Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. *arXiv preprint arXiv:1906.11172*, 2019.