Bachelor Thesis

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Detection and Localization of Texture-less Objects with Deep Neural Networks

Pavel Haluza

Supervisor: Ing. Tomáš Hodaň
May 2017

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student:	Pavel H a l u z a
Study programme:	Cybernetics and Robotics
Specialisation:	Robotics
Title of Bachelor Project:	Detection and Localization of Texture-less Objects with Deep Neural Networks

Guidelines:

1. Study the existing state-of-the-art methods for detection and localization of object classes in RGB images. Focus on methods based on deep neural networks, e.g. [1,2,3].
2. Consider application of the existing methods to detection and localization of specific texture-less objects.
3. Analyse properties of a selected method on RGB images from relevant RGB-D datasets, e.g. [4,5,6,7].
4. Propose how to use the D image channel in the selected method. Implement the proposed extension.
5. Experimentally evaluate the extended method on RGB-D images and compare the results with other methods.

Bibliography/Sources:

[1] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." arXiv preprint arXiv:1506.02640 (2015).
[2] Ren, Shaoqing, et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." Advances in neural information processing systems (2015).
[3] Liu, Wei, et al. "SSD: Single Shot MultiBox Detector." arXiv preprint arXiv:1512.02325 (2015).
[4] T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects, http://cmp.felk.cvut.cz/t-less
[5] ICCV2015 Occluded Object Challenge, http://cvlab-dresden.de/iccv2015-occlusion-challenge
[6] UoB Highly Occluded Object Challenge, http://www.cs.bham.ac.uk/research/projects/uob-hooc
[7] Latent-Class Hough Forests for Object Detection and Pose Estimation, http://www.iis.ee.ic.ac.uk/rkouskou/research/LCHF.html

Bachelor Project Supervisor: Ing. Tomáš Hodaň

Valid until: the end of the summer semester of academic year 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic	prof. Ing. Pavel Ripka, CSc.
Head of Department	**Dean**

Prague, January 6, 2017

Acknowledgements

I give a great thanks to my supervisor Ing. Tomáš Hodaň, who was a huge help during the work on this thesis. Furthermore, practical experiments would not be possible without access to Halmos computer acquired by project CEMI of Jiří Matas.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, May 26, 2017

Abstract

This thesis studies Faster R-CNN, the state-of-art method for object detection in RGB images, and proposes its extension to RGB-D images. Solutions to the following problems are proposed and evaluated: filling missing values in depth images, depth encoding (raw depth vs. surface normals), extension of the CNN architecture to accept the extra depth information, and initialization of weights in the extended network. The overall best results were achieved with a network that accepts an extra depth channel, pre-processed by the iterative median filter to fill in the missing values, and has the depth weights in the first convolutional layer initialized with the mean of the color weights that were pretrained on ImageNet. However, the improvement over the original method using only RGB channels is not significant (mAP was increased by $1 - 2\%$), which suggests a need for different incorporation of the depth information.

Keywords: object detection, texture-less, RGB-D, CNN, deep learning

Supervisor: Ing. Tomáš Hodaň

Abstrakt

Tato práce zkoumá Faster R-CNN, moderní metodu pro detekci objektů v RGB snímcích, a navrhuje její rozšíření na RGB-D snímky. Jsou diskutována a vyhodnocena řešení následujících problémů: vyplnění chybějících hodnot v hloubkových snímcích, zakódování hloubkové informace (původní vs. povrchové normály), rozšíření CNN architektury o hloubkové vstupy a inicializace vah v rozšířené síti. Celkově nejlepších výsledků bylo dosaženo se síti pracující s informací o hloubce předzpracované iterativním mediánovým filtrem pro vyplnění chybějících hodnot a hloubkovými váhami v první konvoluční vrstvě inicializovanými průměrem RGB vah předtrénovaných na ImageNetu. Nicméně zlepšení vůči původní metodě využívající pouze RGB kanály je nepatrné (mAP bylo zvýšeno o $1 - 2\%$), což vybízí k jinému přístupu použití hloubkové informace.

Klíčová slova: detekce objektu, bez textury, RGB-D, CNN, hluboké učení

Překlad názvu: Detekce a lokalizace netexturovaných objektů pomocí hlubokých neuronových sítí

Contents

Chapter 1

Introduction

Object detection is one of the core problems of computer vision. The ability to learn the appearance of a specific object or an object category and then find it in a new image is a key for many robotic, augmented-reality and other scene-understanding applications.

The main focus in the field of object detection is currently on 2D detection of object categories (PASCAL VOC Challenge [1], ILSVRC [2], COCO [3]). Given a set of training images, each annotated with 2D bounding boxes and category labels of the visible objects, the goal is to detect the objects in previously unseen images and estimate their 2D bounding boxes (Figure 1.1). As in many other areas of computer vision, the field is currently dominated by methods based on convolutional neural networks (CNN) [4, 5].

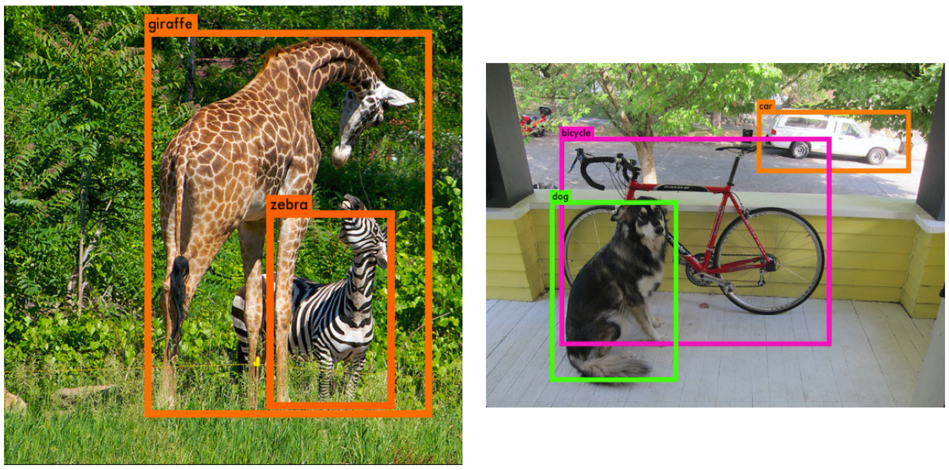


Figure 1.1: 2D detection of object categories – the goal is to find objects from the trained categories and estimate their 2D bounding boxes. Image courtesy of Redmon and Farhadi [6].

In this thesis we focus on detection of specific texture-less objects (Figure 1.2). Appearance of a texture-less object is dominated by its global shape, reflectance properties and the illumination. Unless these are known in advance and precisely controlled, the recognition method needs to be robust to changes in these factors. Traditional recognition methods based on local features does not work well on texture-less objects because interest point detectors typically fail to identify corresponding image regions and common local appearance descriptors are no longer discriminative enough to provide reliable correspondences [7, 8]. Performance of the CNN methods on this type of objects is still unclear.

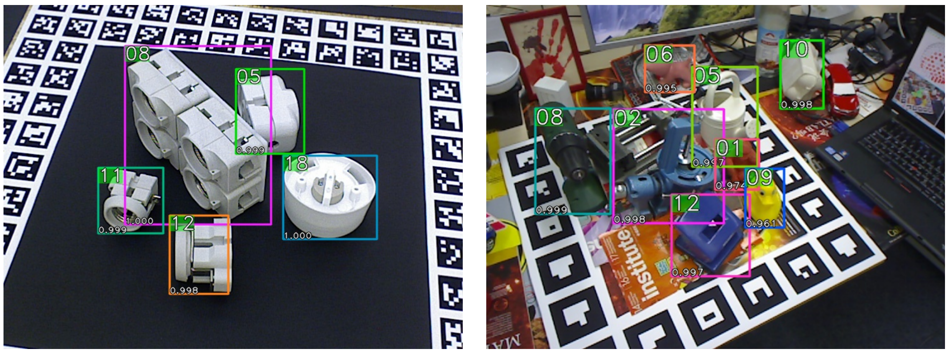


Figure 1.2: 2D detection of specific texture-less objects – the goal is to find instances of the trained objects and estimate their 2D bounding boxes.

The object detection task can be simplified when depth images, which are expected to allow for a more discriminative description, are used as additional input data. RGB-D sensors that capture aligned color and depth images have been available for years, but consumer-level sensors became widely available only recently, with the launch of Microsoft Kinect.

The aim of this thesis is to study the state-of-the-art CNN methods for detection of object categories in RGB images, consider their application to specific texture-less objects, analyze their properties and propose an extension to RGB-D images.

1.1 Structure of the Thesis

Chapter 2 describes theory and fundamentals of CNNs.
Chapter 3 reviews relevant work, Faster R-CNN [4] and YOLO [6].
Chapter 4 proposes an extension of Faster R-CNN to RGB-D images.
Chapter 5 defines datasets and evaluation methodology.
Chapter 6 analyses properties of original Faster R-CNN.
Chapter 7 evaluates the proposed extension on RGB-D datasets.
Chapter 8 concludes the thesis.

Chapter 2
Convolutional Neural Networks

Thanks to the rise of computational power and the introduction of big datasets (e.g. ImageNet [2]), deep artificial neural networks has become a standard tool in machine learning, achieving the state of the art results in various tasks. This chapter focuses on a specific branch represented by convolution neural networks (CNN), as they are used in this work.

2.1 Architecture

Neural networks in general are modeled as collections of neurons that are connected in an acyclic graph, often organized into distinct layers of neurons [9]. There is an input layer (accepting the input image) and an output layer (in our case providing estimates of the position and the class of detected objects). There are multiple hidden layers in between the input and the output layer. The layers are placed in consecutive manner, each processing output from the previous layer and relaying it to the next one (Figure 2.1 left).

A neuron is defined as:

$$y = f(\mathbf{w}^T\mathbf{x} + b), \tag{2.1}$$

where $\mathbf{w}$ is a weight vector, b is a bias, $\mathbf{x}$ is an input vector, f is an activation function, and y is an output (Figure 2.1 right).

The activation function brings non-linearity to the network which increases its discriminative power. The function must be easily differentiable to be able to perform back-propagation. Examples are the sigmoid function $f(x) = 1/1 + e^{-x}$ and ReLU $f(x) = \max(0, x)$, the latter one being a common choice.

There are three main types of layers used to build CNN architectures:

- **Convolutional layer** is defined by a set of learnable filters that are convolved across the width and height of the input volume and the dot product between the weights of the filter and the input is computed at each position. The weights are thus shared over all locations. Its output is called a feature map. Visualization of filters can be found at fig. 7.3.

- **Pooling layer** essentially performs down-sampling. By reducing the spacial size it increases robustness to small deformations, reduces the chance of over-fitting and reduces the complexity of higher layers.

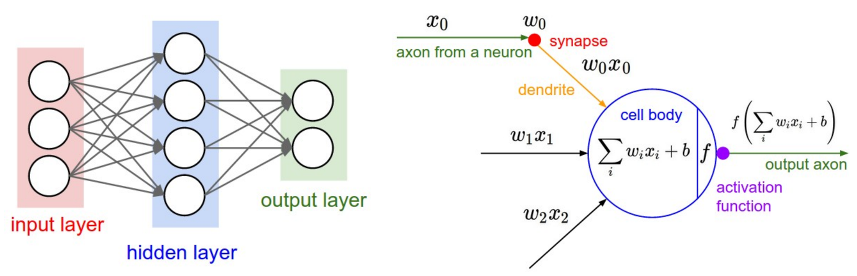


Figure 2.1: A simple neural network with two fully-connected layers (left) and a mathematical model of a neuron (right). Image courtesy of [9].

- **Fully-connected layer** has each of its neurons connected to all neurons of the previous layer. Its job is to do high-level reasoning and is often situated towards the top of the network. CNNs usually use only a few of these as each such layer significantly increases the number of parameters and is therefore relatively hard to train and prone to over-fitting.

2.2 Learning

The most popular technique to learn the CNN weights is stochastic gradient descent. It is an iterative process, which in each iteration 1) randomly selects a subset of training examples, 2) evaluates how the network performs on this subset using a loss function, 3) calculates gradient of the loss function using back-propagation, and 4) use the gradient to update the weights in order to decrease the loss [9].

2.3 Frameworks

When working with neural networks, it is convenient to use some of the freely available frameworks which encompass most of the necessary functionality. Some of the popular frameworks are Caffe, Darknet, Tensorflow, MatConvNet and Torch. The first two are briefly reviewed below.

Caffe. This framework is well known and widely used. It is implemented in C++ and CUDA and has hundreds of contributors on GitHub. The authors praise it for its speed and modularity. Documentation consists mostly of formatted header files, however there are many tutorials and commented examples. It also has Python bindings.

Darknet. It is written in C and CUDA specifically for purposes of YOLO and other projects by Joseph Redmon, who has My Little Pony in his CV. It is light-weight and well optimized but not very flexible for experimenting, as is usual for C.

Chapter 3
Object Detection in RGB Images

As popularity of CNNs grew in the last few years, many attempted to use them for object detection in RGB images. Compared to the traditional methods [10, 11], the CNN methods [12, 13] often achieve better performance with less handcrafting. This chapter describes two state-of-the-art CNN methods.

3.1 Faster R-CNN

Faster R-CNN [4] is built upon Fast R-CNN [14]. It unifies proposal and classification parts into a single neural network which provides nearly cost-free region proposals and allows end-to-end training and testing. Its architecture consists of multiple convolutional layers, Region Proposal Network (RPN), a few fully-connected layers and two output layers – one for classification and one for bounding box regression. Either ZF model [15], which has 5 shareable convolutional layers, or VGG16 model [16], which has 13 shareable convolutional layers, is used. The convolutional layers are interspersed with ReLU and max-pooling layers (Figure 3.1).

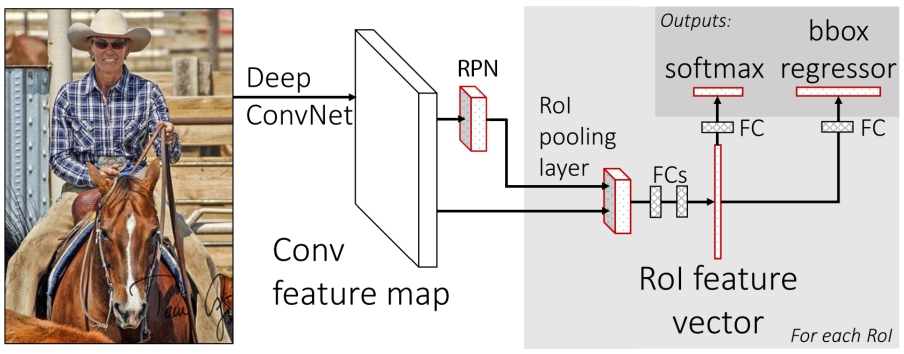


Figure 3.1: Faster R-CNN architecture.

RPN generates object proposals by sliding a small window over the feature maps. Each proposal is given by a bounding-box and objectness score. The bounding boxes are regressed from 9 anchor boxes that improve ability to detect objects of various scales and aspect ratios (fig. 3.2).

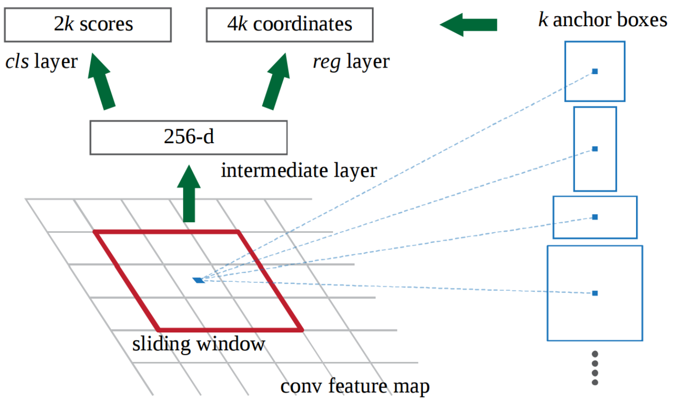


Figure 3.2: Region Proposal Network (RPN).

The feature maps and 300 proposals with the highest objectness score are passed to a region of interest (RoI) pooling layer. The RoI pooling layer uses max-pooling to crop and reshape the regions of feature maps defined by the proposals to achieve the same spacial size, typically 6 by 6. The resized regions are then processed by two fully-connected layers and the resulting feature vector continues to two sibling fully-connected layers. One is followed by softmax and generates probabilities of each class (including background). The other fully-connected layer outputs regression of proposed bounding-boxes.

There are two proposed strategies to train the Faster R-CNN network, both of them use stochastic gradient descent. One alternates between training RPN alone and the rest of network, the other is end-to-end but the gradients for RPN are only approximate.

There are often multiple detections with overlapping or conflicting bounding boxes. To identify unique detections, the non-maximum suppression algorithm is applied. It repeatedly retains the detection with the highest score and removes all those that have large overlap with it, i.e. the IoU is greater than a selected threshold (Figure 3.3). This effectively reduces the number of duplicate detections which improves precision while keeping the recall ideally untouched.

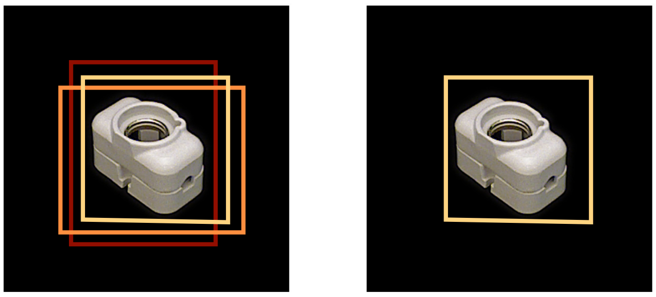


Figure 3.3: Example of NMS outcome. **Left:** Image with all detections. **Right:** NMS algorithm applied to identify the unique detections. Brightness represents score of a detected bounding-box.

3.2 YOLO

YOLO (You Only Look Once) [6] also provides integrated CNN model for object detection, similarly to Faster R-CNN, and generates both bounding-boxes and their classes from an image. It sacrifices precision but aims to be much faster, easier to train and more robust.

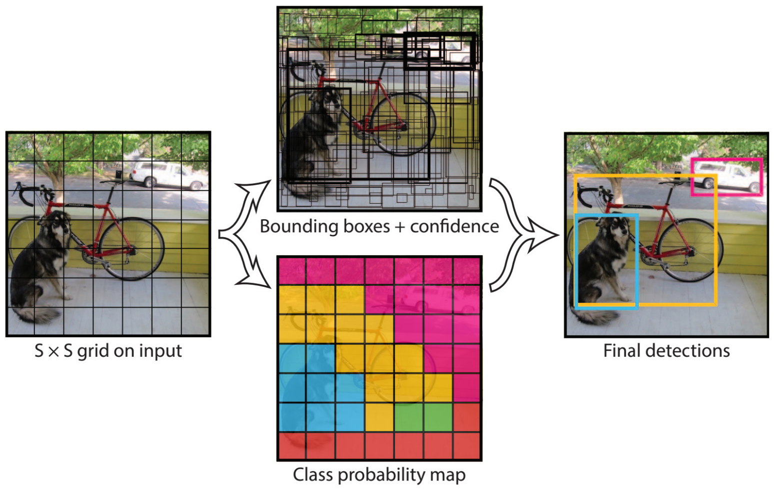


Figure 3.4: YOLO [6] models detection as a regression problem. It divides the image into an $S \times S$ grid cells and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities.

An input image is resized to predefined resolution, fed into 24 convolutional layers (9 for a lighter version) that are followed by two fully-connected layers. The image is divided $S \times S$ grid cells. For each grid cell the network predicts B bounding boxes, confidence for those boxes, and C class probabilities. A bounding-box is defined by five values: x and y coordinate of the center in relation to the grid cell, w and h size of the box as factor of the image size, and confidence that the box contains an object. This means the network outputs $S \times S \times (5 * B + C)$ values. Typical values for detection of objects in the Pascal VOC dataset are $S = 7$, $B = 2$, $C = 20$.

During our examination of YOLO, we bumped into some issues regarding application to our problem. The main one is connected to the fact that each grid cell is trained independently and reasons globally – it looks at the whole image. As a consequence, the method is not invariant to object translation and background variations and therefore requires training data that cover these aspects. Suitable training data was not available and we therefore decided to focus mainly on Faster R-CNN in this thesis, as well as for its mentioned benefits.

Chapter 4

Proposed Extension to RGB-D Images

The CNN models described in the previous chapter were designed to work with RGB images. This chapter discusses their extension to RGB-D images.

4.1 RGB-D Sensors

RGB-D sensors have been available for a long time, but consumer-level sensors became widely available only recently, after the launch of Microsoft Kinect v1 in November 2010. Kinect v1 estimates the depth of the scene using the structured-light principle – a light pattern is projected onto the scene using a near-infrared laser emitter and the light reflected back to a standard off-the-shelf infrared camera is analyzed to estimate the depth of the scene surfaces [17, 18]. There are other consumer-grade RGB-D sensors that are based on the same principle, such as Primesense Carmine or Asus Xtion. The second generation of Microsoft Kinect was released in 2014. This completely new sensor is based on the time-of-flight principle in which the depth of a scene is measured by the absolute time needed by a light wave to travel into the scene and, after reflection, back to the sensor.

The color and the depth sensor have slightly different viewpoints – they are placed a few centimeters apart on the body of a camera. The depth image is typically mapped pixel-by-pixel to the RGB image, using the known relative transformation of the sensors and their intrinsic parameters. The mapping results in "shadows" around depth discontinuities where no depth information is available. Furthermore, the depth measurements are often affected or missing on certain surfaces, especially glossy, transparent or sharply-angled surfaces. Another limitation of the current RGB-D sensors is their restricted sensing range going from tens of centimeters to several meters.

4.2 Filling Missing Depth Measurements

As described in Section 4.1, a typical depth image has missing information on many pixels. In CNN, the representation of invalid values is not trivial and each pixel is rather assumed to have a valid value. Without any special treatment, the missing depth values can yield false filter responses, possibly overshadowing responses from regions with valid values.

In the fields of image transmission and photo recovery, there are various in-painting techniques to reconstruct lost image regions, such as [19, 20]. Nevertheless, these methods usually take from several to tens of seconds per image (depending on the size or regions to be in-painted). Instead, we considered the following simple methods that take only a fraction of second per image and still yield satisfactory results.

Nearest Neighbor

This approach does not consider the depth information at all – an invalid pixel is assigned a value of the closest valid pixel, i.e. an pixel with a valid depth value. The problem is when there are several valid pixels at the same distance. Consider, for example, a 3 by 3 image with missing value at the center pixel. What would be an adequate infill according to the nearest neighbor method? Depending on the implementation, it could be the value of the top, the left, the bottom or the right pixel.

Iterative Median Filter

The median filter is known to efficiently remove salt-and-pepper noise from images. A window with a fixed size is slided across the image and the central pixel of the window is filled with the median of values in the window.

For the purposes of filling holes in depth images, several modifications are needed. First of all, the filter has to be applied only on holes (i.e. regions with missing values), otherwise it would needlessly change the already present information. Secondly, the missing values are represented by zeros and must be ignored for the median calculation. The filter is applied in an iterative manner, gradually filling holes from the borders to the center – the filter is applied if there is at least one valid pixel in the neighborhood.

In our experiments, a sliding window of size 3 by 3 was shown to produce acceptable results. On several manually examined images, the method leads to very similar results as the nearest neighbor method, but was preferred because it does not have the problem of multiple candidate values. Figure 4.1 shows an example result.

4.3 Depth Encoding

One possible way how to input the depth information to the network is to use *absolute depth* – raw or with filled holes – that represents the distance from the sensor.

Another option is to calculate *surface normals*, for which we used the method from [21] – it approximates the surface normal at a pixel by a normal vector of a plane fitted to the pixel neighborhood. The resulting normals are expressed in the camera coordinate system and represented as a

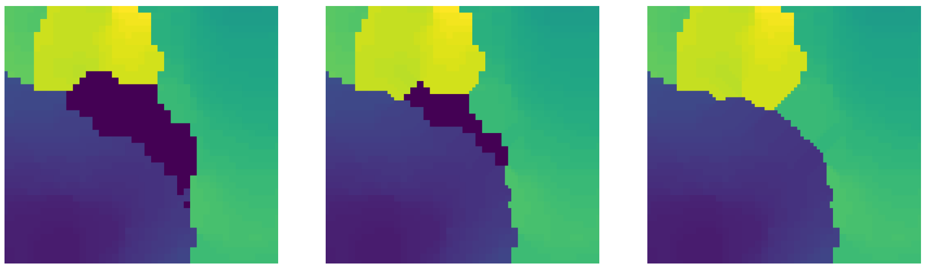
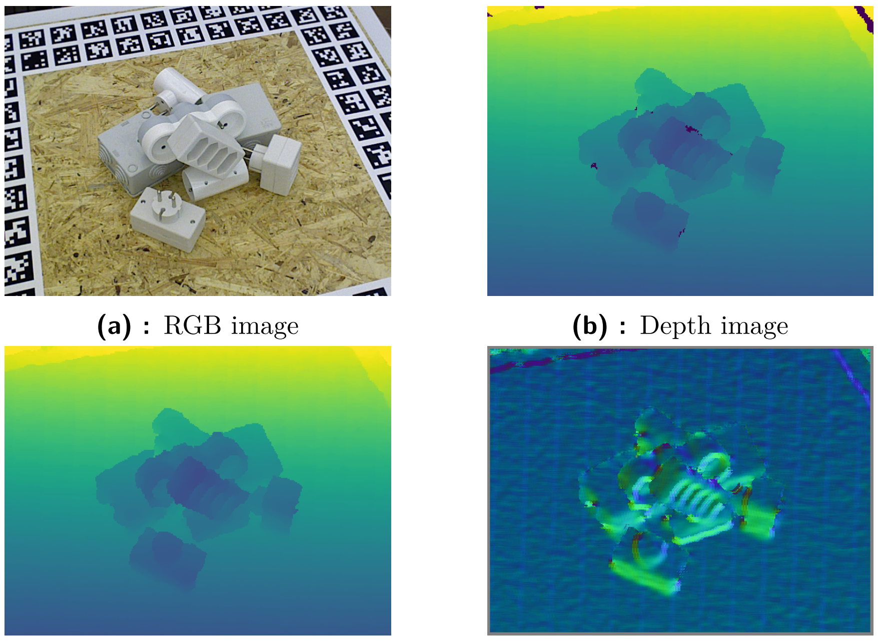

(a) : Raw depth **(b) :** 3 iterations **(c) :** 8 iterations

Figure 4.1: Iterative median filter applied on an image region from the T-LESS dataset. Dark blue represents the missing values. Scaled for better visibility.


(a) : RGB image **(b) :** Depth image

(c) : Depth image with filled holes **(d) :** Surface normals

Figure 4.2: Representation of an RGB-D image.

3-channel image, with one channel per coordinate. To avoid smoothing at depth discontinuities, there is a threshold on the maximum allowed depth difference of points considered for the plane fitting. There is also a threshold on the maximum depth at which the surface normals are calculated – this was disabled in our experiments to avoid creating holes in the surface normal images. A simpler way to extract approximate surface orientation could be to apply a gradient filter, e.g. Sobel, to the depth image. However, this is deemed futile since such filter can be learned in the first convolutional layer.

There are other depth encodings. For example, *HHA* [22] encodes the depth as horizontal disparity, height above the ground and angle of the local surface normal with the inferred gravity direction.

Results obtained with the absolute depth values and the surface normals are compared in chapter 6.

4.4 Extension of the CNN Architecture

To add the depth information, the CNN architecture was altered to accept extra of input channels, not just three for red, green and blue. The images for RGB and depth are pre-processed in advance and saved on disk. During runtime (training and testing), the matching images are loaded from their separate files and inserted to the network as if they were one image. The rest of the architecture, as described in section 3.1, was kept intact.

Another possible extension is a two-branch architecture, as was used in [23], with one branch processing color and one depth information, which allows to learn filters for color and depth independently. It seems promising as it achieves the state-of-the-art semantic segmentation results. Experiments with this architecture are a subject of future work.

4.5 Normalization

The color and depth information have different ranges of values – color being from 0 to 255 and depth from 0 to theoretically infinity (practically 0 to 65 535 as it is saved as a 16-bit unsigned integer). It is important to normalize the input data to make them of approximately equal importance to the learning algorithm. The input values are normalized to be from -1 to 1, using ranges summarized in table 4.1. The depth values are limited to a range covering the captured scene. Values beyond the usable range are capped at the boundary values (-1 or 1).

dataset	channel	used range	target range
all	color (R, G, B)	0 to 255	-1 to 1
	normals (x, y, z)	-1 to 1	
T-LESS train	depth	450 to 850 mm	
T-LESS test	depth	500 to 1050 mm	
LINEMOD test	depth	206 to 1640 mm	

Table 4.1: Ranges used for input normalization.

4.6 Initialization of Network Weights

A common approach is to start with weights that were trained on a related task and then fine-tune them for the given task. In our case, we can use weights that were trained for the original Faster R-CNN on the ImageNet dataset [2], which are provided by authors of Faster R-CNN. The issue is that there are no pretrained values available for non-color weights in the first convolutional layer, i.e. weights that are related to non-color channels.

We considered several strategies for initialization of the network weights. They differ mainly in the way how the non-color weights are initialized. The strategies are evaluated in chapter 6.

Chapter 5

Experimental Setup

5.1 Datasets

The experiments were conducted on two datasets: T-LESS by Hodaň et al. [24] and LINEMOD by Hinterstoisser et al. [25]. Both provide RGB-D images of scenes with texture-less objects. The images are annotated with the ground truth 6D poses of instances of the modeled objects. For the evaluation of the 2D object detection task, bounding boxes of projections of the object models at the ground truth poses were used as the ground truth.

5.1.1 T-LESS Dataset

The T-LESS dataset, which we published in [24], features 30 industry-relevant objects with no significant texture and no discriminative color or reflectance properties. The objects exhibit symmetries and mutual similarities in shape and/or size. The dataset includes training and test images that were captured with three synchronized sensors, specifically a structured-light (Primesense Carmine 1.09) and a time-of-flight RGB-D sensor (Microsoft Kinect v2) and a high-resolution RGB camera (Canon IXUS 950 IS). There are approximately 39K training and 10K test images from each sensor. For experiments in this thesis, only images from the Primesense sensor were used. Training images depict individual objects against a black background. Test images originate from 20 test scenes having varying complexity, which increases from simple scenes with several isolated objects to very challenging ones with multiple instances of several objects and with a high amount of clutter and occlusion. There are 9 test scenes with black background and 11 scenes with cluttered background (Figure 5.1).

Each of the experiments was conducted either on the set of training images (*T-LESS train*) or on the set of test images (*T-LESS test*). The sets were randomly split into equally-sized training and test subsets, i.e. testing of the network was done on different images than it was trained on. The practical scenario for which the dataset was created, i.e. a method is trained on *T-LESS train* and evaluated on *T-LESS test*, was not evaluated in this thesis. This is because the images from the two sets show the objects in different scale and with a background of different distribution. Faster R-CNN cannot handle

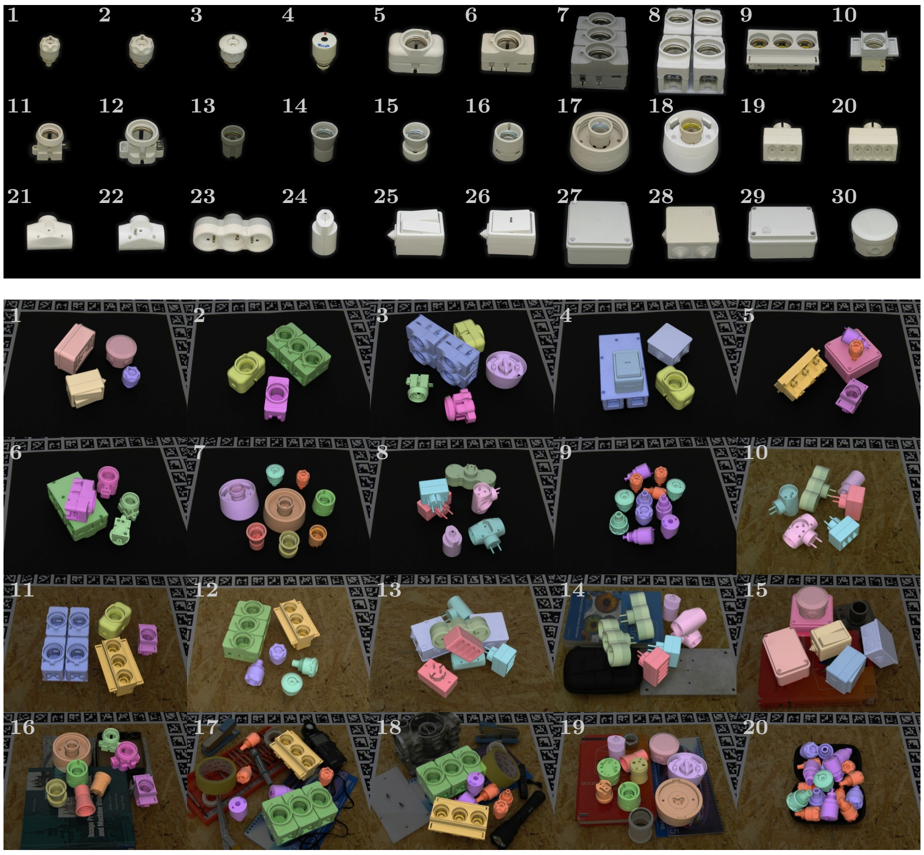


Figure 5.1: T-LESS includes training images and 3D models of 30 objects (top) and test images of 20 scenes (bottom). The shown test images are overlaid with colored renderings of 3D object models at the ground truth poses.

these changes out of the box, without some extra treatment.

5.1.2 LINEMOD Dataset

LINEMOD [25] is a well-established dataset used in most of the recent work on 6D object pose estimation, e.g. [8, 26, 27]. It contains 15 texture-less objects, for which it provides a sequence consisting of around 1200 RGB-D images, each of which includes exactly one instance of the object. The images present significant 2D and 3D clutter and mild occlusion. The objects have discriminative color, shape and/or size, and, compared to the T-LESS objects, their recognition is relatively easy (Figure 5.2). For our experiments, the image set was randomly split into equally-sized training and test subsets.

5.2 Evaluation Methodology

The neural network outputs for each image a set of detections. Each detection is defined by a rectangular boundary, i.e. a bounding-box, an object class

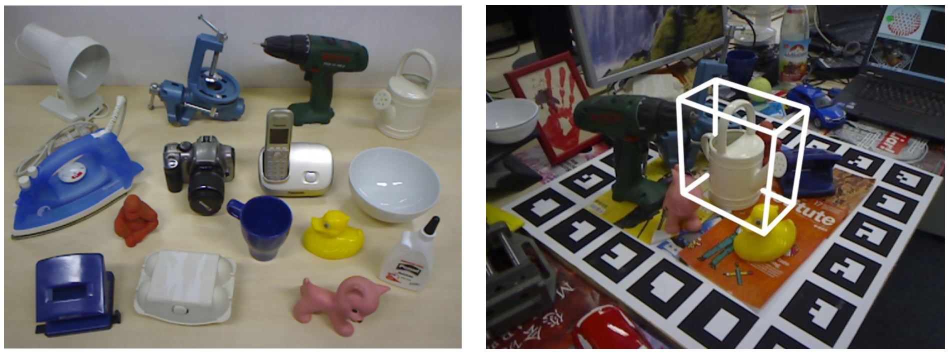


Figure 5.2: LINEMOD includes 15 texture-less objects (left) captured in cluttered scenes (right).

and score which represents probability that the bounding-box contains an object of the assigned class.

Each image is associated with a set of ground truth annotations, each given by a bounding-box and a class of the annotated object. The annotations were not used in any way in the neural network during testing. They were used only for training and true/false positive classification of the final detections.

The process of evaluation consists of the following steps. Steps 1, 2 and 3 are done independently for each image, steps 4 and higher are done across all images at once.

1. Do the forward pass through the network, continue with 100 detections with the highest score. This number was shown sufficient as there is a maximum of 18 objects of interest in each image.
2. Apply non-maximum suppression (section 3.1) on the detections with an IoU threshold of 0.7 (section 5.2.1). This reduces the number of conflicting detections, thus increasing precision (section 5.2.3) while recall remains almost the same.
3. Classify each detection as true or false positive (section 5.2.2).
4. Calculate the precision-recall curve for each object class (section 5.2.3).
5. Compute the average precision (AP) per object class according to PASCAL VOC 2010 [28] (section 5.2.3).
6. Compute the mean average precision (mAP) as the mean of AP rates of the object classes.

5.2.1 Intersection over Union

Intersection over union (IoU) is used to evaluate the alignment of two bounding boxes – IoU $= 1$ for two perfectly aligned rectangles while IoU $= 0$ for no overlap at all. As the name suggests, it is defined as:

$$\text{IoU} = \frac{\text{area of intersection}}{\text{area of union}}. \tag{5.1}$$

To analyze properties of IoU, let us take two identical squares with side $s = 1$, position them over each other and shift one of them by x in one (eq. (5.2)) or both axes (eq. (5.3)). Fig. 5.3 shows that IoU is non-linear w.r.t. the amount of misalignment. Similar relations can be found for scale, rotation and their combination. This finding should be taken into account when deciding IoU thresholds or analyzing results.

$$\text{IoU}_{\text{one}} = \frac{1-x}{1+x} \tag{5.2}$$

$$\text{IoU}_{\text{both}} = \frac{(1-x)^2}{1+2x-x^2} \tag{5.3}$$

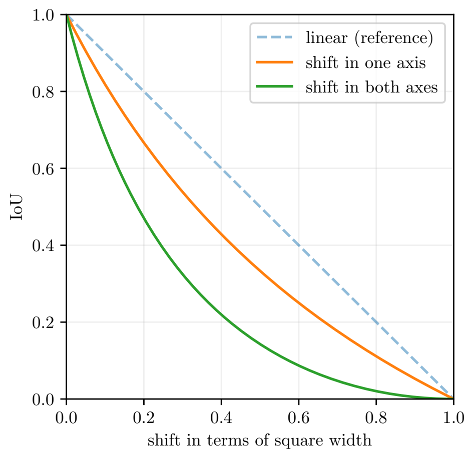


Figure 5.3: IoU of two identical squares when one of them is shifted by x.

5.2.2 Determination of True Positive Detections

The detections are evaluated, i.e. classified as true or false positives, as in [29]. For each image, the detections are sorted by their score in descending order and greedily matched to the ground truth bounding boxes – each detection is matched with a ground truth bounding box with the highest IoU, if IoU> 0.7. Each detection and ground truth can be paired to either zero or one counterpart. Detections which have a partner are true positives, others are false positives. False positives can be further divided into various categories as in [30], however this deep analysis did not found its usage during work on this thesis.

5.2.3 Precision and Recall

The performance of the detection method is measured by recall and precision:

$$\text{recall} = \frac{|\{\text{true positive}\}|}{|\{\text{true positive}\}| + |\{\text{false negative}\}|} \tag{5.4}$$

$$\text{precision} = \frac{|\{\text{true positive}\}|}{|\{\text{true positive}\}| + |\{\text{false positive}\}|} \tag{5.5}$$

Now that we have detections matched up with ground-truths from section 5.2.2, we sort all of them by their score and for each compute recall and precision in a cumulative manner. This produces the precision-recall (PR) curve. Example can be seen in fig. 6.1. Precision of a "well-behaved" detector should monotonically decrease, or ideally stay at 1, while recall rises.

5.2.4 Average Precision

Average precision (AP) of an object class is defined as the area under the precision-recall curve, when considering only detections of that class. To follow the approach used in PASCAL VOC 2010 [28], the curve is smoothed out by setting the precision for recall r to the maximum precision obtained for any recall $r' \geq r$ – as if the new curve was a boundary of shadow cast by the old "bumpy" curve shined at from right to left. The mean average precision (mAP) is calculated as the mean of APs across all object classes.

5.2.5 Confusion Matrix

Another useful insight about recognition performance can be obtained from a confusion matrix. Its columns represent the detected class and rows represent the ground truth class. Each element of the matrix is a number of detections of the column class that are in fact of the row class. An example can be seen in fig. 7.5.

Chapter 6
Evaluation of Faster R-CNN

Before diving into implementing and testing of the proposed extension to RGB-D images, we analyzed what Faster R-CNN is actually capable of, out of the box.

6.1 Implementation

Faster R-CNN is implemented using slightly extended Caffe framework, while bulk of the method (a few extra Caffe layers and overhead) is written in Python. In order to start experimenting, it was necessary to expand the provided source code, namely make it work with datasets in the format of [31] and re-implement the evaluation scripts. The evaluation process follows PASCAL VOC 2010 (section 5.2), but some minor changes were made along the way, such as taking NMS of detections over all classes instead of per class.

Its training and testing pipelines have a number of variants and parameters whose suitable values need to be found manually, such as NMS thresholds of proposals and detections, or the choice of the actual training method and its learning rate. During our work, we used the end-to-end training variant with 100k iterations which was found sufficient to reach a low loss. Training was done mostly on graphics card *NVIDIA GeForce GTX TITAN Black, 6GB* and took about 7.5 hours. Testing reached impressive speed of about 20 images per second and this could be further improved as some layers are implemented in Python.

6.2 Sensitivity to Scale Change

The first experiment analyzes a situation when the network is trained on a set of images and then tested on the same images but scaled differently. The dataset *T-LESS train* was used for this experiment. The images have a black background that prevents removing or adding features in the scaling process.

It can be seen in fig. 6.1 that the network behaves in a predictable way. The more the images are scaled the worse the detections are. Downscaling yields a bit worse results – compare pairs of inverse scales, e.g. 1.2 and $0.83 = 1/1.2$. This might be explained by the fact that scaling an image to a smaller size

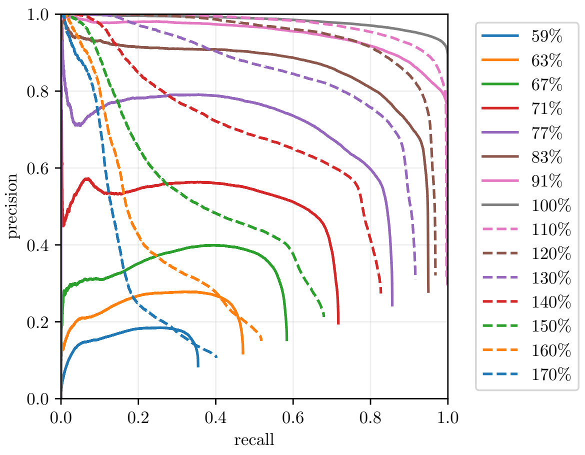


Figure 6.1: Sensitivity to scale change. The network was trained on 100% sized images and tested on multiple others. Curves with the same color represent inversely equivalent scales, e.g. $59\% = \frac{1}{170\%}$.

removes features, while scaling up interpolates additional values.

6.3 Sensitivity to Rotational Change

Similarly to the previous experiment, we analyzed sensitivity of the method to changes in rotation. Results in fig. 6.2 show that change in rotation does not have such severe effect on quality of detections, compared to the scale change. The network can handle about 20° rotation with only a minor drop in performance, while results between 90° to 180° are very similar. The difference between the clockwise and the anti-clockwise rotation is barely noticeable.

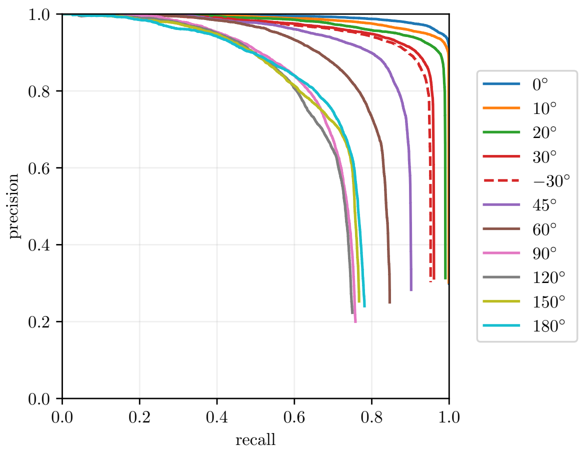


Figure 6.2: Sensitivity to rotational change. The network was trained on the original images and tested on the images rotated in the anti-clockwise direction.

6.4 Sensitivity to Background

We have also examined sensitivity to different background distributions. For this experiment, the *T-LESS test* was divided into these subsets:

- *black* (scenes 1 to 9) - objects on a black background
- *color* (scenes 10 to 20) - objects, including some distractor objects, on a background with various texture and color
- *all* (scenes 1 to 20) - combining the previous two to examine ability of the network to handle both at once

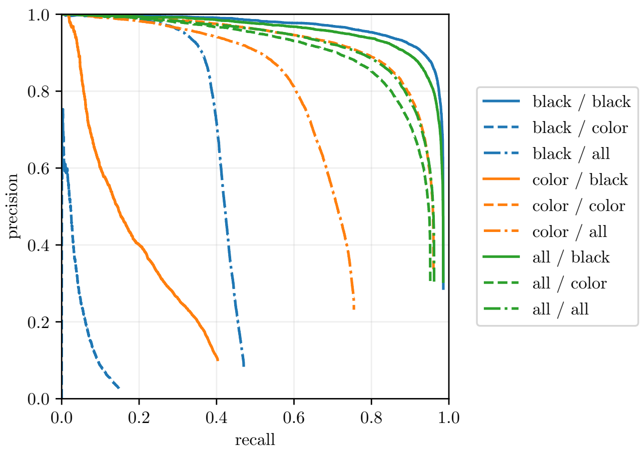


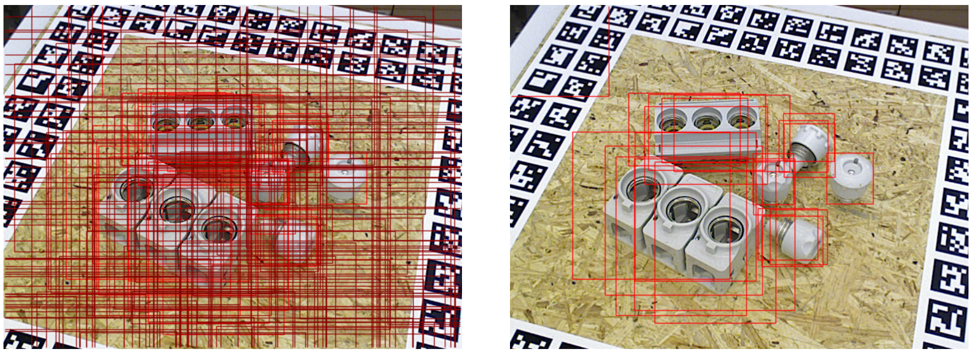
Figure 6.3: Sensitivity to background. Legend: subset used for training / subset used for test.

Fig 6.3 shows that the network has serious difficulties with backgrounds that were not present during training. This indicates an importance of proper background modeling for Faster R-CNN.

6.5 Quality of Object Proposals

The network generates its own object proposals (fig. 6.4) and their objectness score before passing them to the classification part (as described in section 3.1). In this section we examine the proposals in more detail.

One way to evaluate quality of the proposals is to compare their objectness score to IoU with the ground truth bounding boxes. Two ways of pairing proposals with the ground truth bounding boxes were considered: 1) for each proposal find the ground truth with the best IoU (fig. 6.5a), or 2) for each ground truth find the proposal with the best IoU (fig. 6.5b). The rest of detections and ground truth bounding boxes, that were not included in the resulting pairs, were not considered. There are 300 proposals and 1 to 18 ground truth annotations per image.

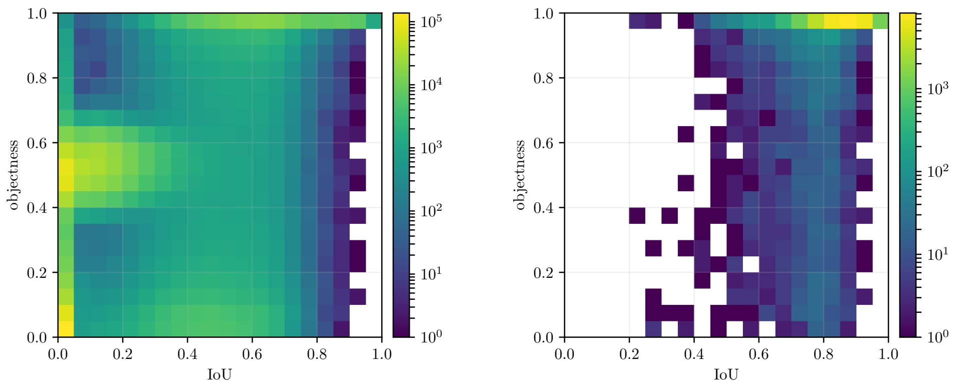


(a) : Objectness > 0.5. **(b) :** Objectness > 0.95.

Figure 6.4: Example object proposals produced by RPN.

Figure 6.5a shows a high number of proposals with low IoU, i.e. proposals on background or distractor objects. A lot of these have relatively large objectness score, meaning the classification part of the network cannot fully rely on them. This is the reason why background needs to be still considered as one of the classes in the classification part.

Fig. 6.5b shows that the best matching proposals tend to have higher objectness score. This demonstrates that the object proposal mechanism works well even on texture-less objects. There is visible decrease of proposals for IoU ≥ 0.9, such values are difficult to reach as they indicate near-perfect alignment (section 5.2.1).



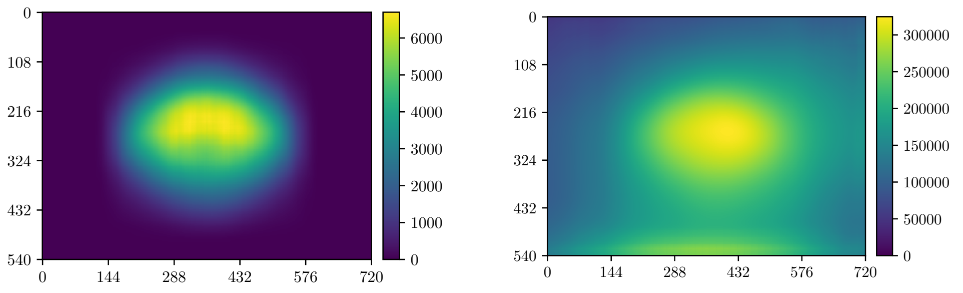
(a) : Every proposal is paired with a ground truth.

(b) : Every ground truth is paired with a proposal.

Figure 6.5: Histograms comparing objectness score of the proposals to IoU with the ground-truth bounding boxes, using dataset *T-LESS test*. The color scale is logarithmic.

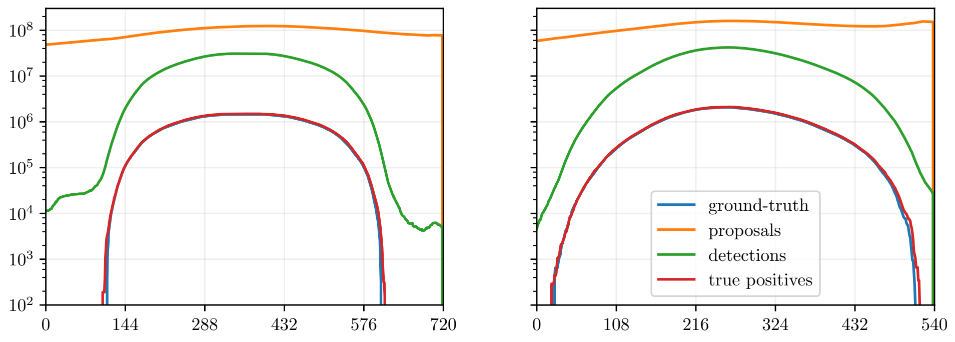
6.6 Bounding-box Distribution

We also examined the distribution of bounding-box positions, both the proposal and the ground truth. Fig. 6.6 shows a distribution where a pixel value is given by the number of bounding-boxes that overlap that pixel. Fig. 6.7 shows the corresponding marginal distributions. One can see that the proposals cover unnecessarily large area, compared to the ground-truth bounding boxes that cover mainly the image center. The classification part of the network can prune a significant part of false proposals, which can be seen by comparing the orange and the green curve in fig. 6.7. Note also that the true positive bounding boxes tend to be slightly larger than the ground truth bounding boxes.



(a) : Ground-truth **(b) :** Proposals

Figure 6.6: Distributions of bounding boxes.



(a) : x axis **(b) :** y axis

Figure 6.7: Marginal distributions of bounding boxes.

Chapter 7

Evaluation of the Proposed Method

This chapter evaluates extensions of the Faster R-CNN method to RGB-D images that were proposed in chapter 4.

7.1 Initialization

First, let us examine the different strategies for initialization of network weights. The following strategies were considered:

- *#1* - all weights pretrained on ImageNet were used, additional weights in the first layer for the non-color channels were initialized with the mean of color weights.
- *#2* - all weights pretrained on ImageNet were used, additional weights in the first layer for the non-color channels were initialized randomly.[1]
- *#3* - all weights in the first layer were initialized randomly, weights from the pretrained model were used for the higher layers.[2]
- *#4* - all weights in all layers were initialized randomly

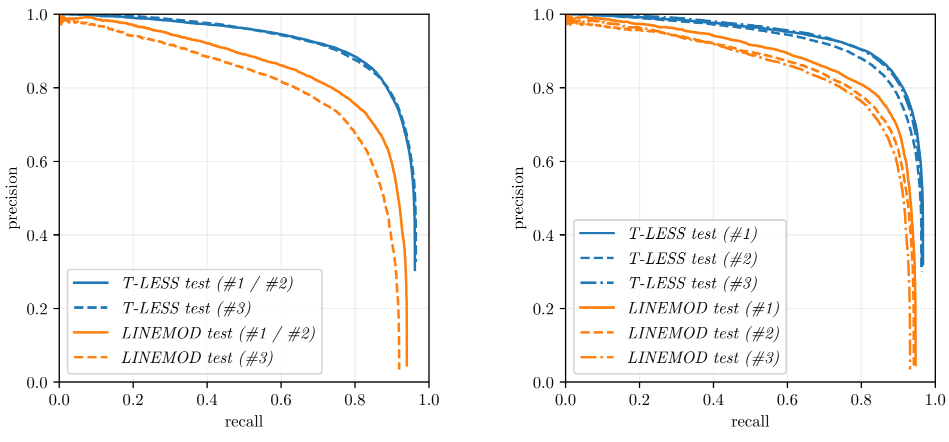
The random weights are drawn from normal distribution $\mathcal{N}\left(0, \frac{1}{n_{in}}\right)$. The corresponding bias values are set to zero.

Figure 7.1 shows that it is beneficial to initialize the weights with values pretrained on the ImageNet dataset. For the RGB-D case, the best results were achieved when the weights in the first layer for the non-color channels were initialized with the mean of the color weights (strategy *#1*).

Randomly initializing all weights in the network (strategy *#4*) was not evaluated as it was found difficult to train on the used datasets. The progress of the learning loss can be seen in fig. 7.2. Too high learning rate is not the cause of the unsuccessful training for strategy *#4*, as we tried multiple values. Most likely it is caused by too small training set, which is significantly smaller than in the case of ImageNet that was used to train the original Faster R-CNN network. The training loss for all initialization strategies reached

[1]Note that the initialization strategies *#1* and *#2* are the same if only RGB is used.
[2]Similarly, strategies *#2* and *#3* are the same if only depth is used.

similar final value, except strategy *#4*. Loss for initialization strategy *#3* is predictably higher for lower iterations.



(a) : RGB images only **(b) :** RGB + filled depth

Figure 7.1: Precision-recall curves for several initialization strategies, two input types and both datasets.

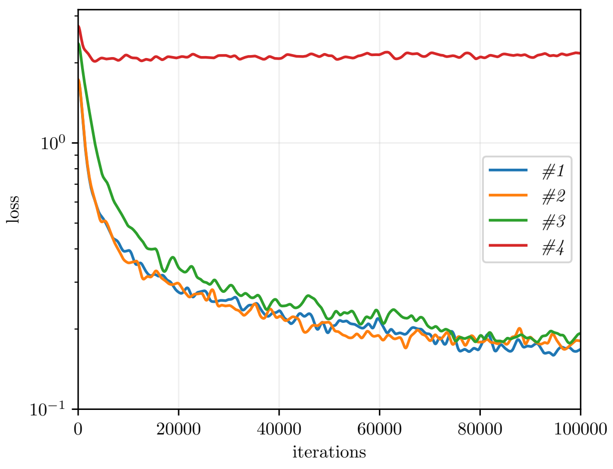
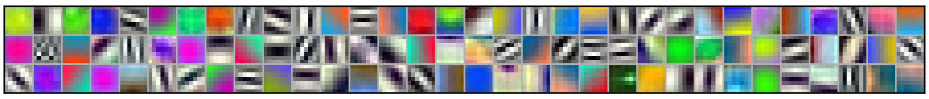


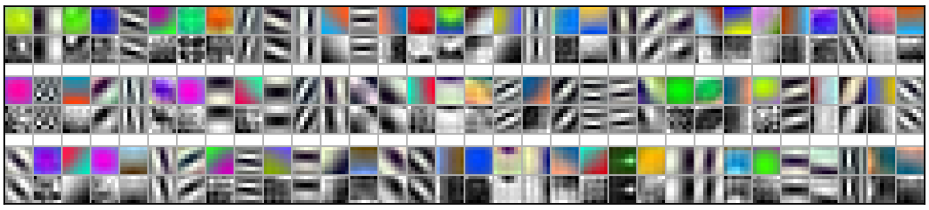
Figure 7.2: Training loss on the *T-LESS test* dataset using RGB and filled depth images with different initialization strategies. Initialization *#4* illustrates how difficult it is to train the network with all weights initialized randomly.

Figure 7.3d shows that filters trained from a random initialization on T-LESS are of a lower frequency when compared to filters trained on ImageNet. This is likely because of the lack of texture, i.e. a higher-frequency pattern, on the T-LESS objects.

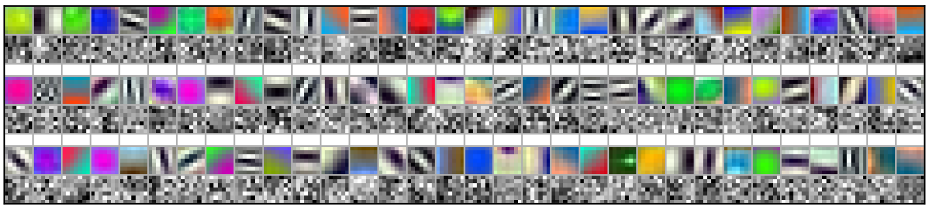
When initialized with random values, the trained depth filters tend to be more noisy than the color filters (fig. 7.3d). Although noisy filters usually indicate an unsuccessful training, the network achieves just slightly worse mAP score when only the depth is used as input (fig. 7.3e).



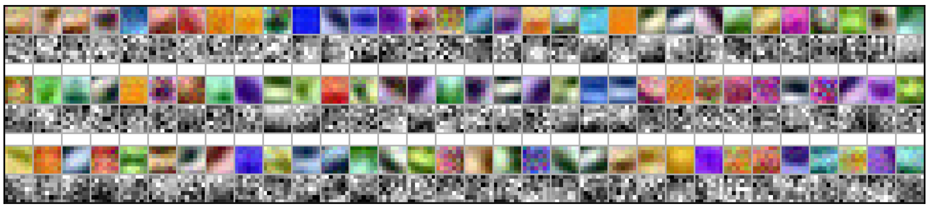
(a) : RGB, initialization #1 / #2, mAP: **0.929** [3]



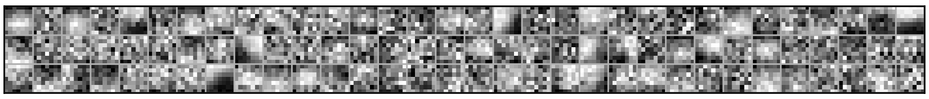
(b) : RGB + filled depth, initialization #1. mAP: **0.938**



(c) : RGB + filled depth, initialization #2, mAP: **0.926**



(d) : RGB + filled depth, initialization #3, mAP: **0.938**



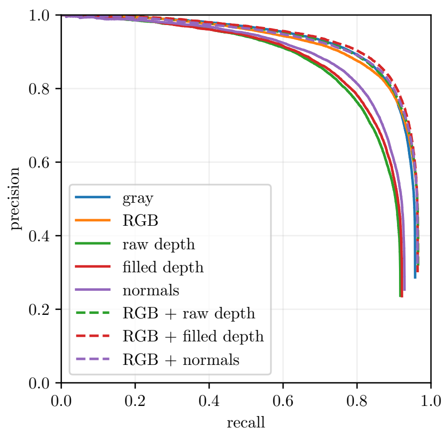
(e) : filled depth, initialization #3, mAP: **0.892**

Figure 7.3: Filters in the first convolutional layer trained on the T-LESS dataset for different network architectures and initialization strategies.

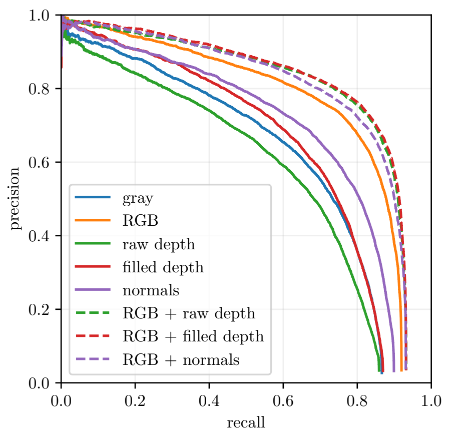
[3]The filters in fig. 7.3a are very similar to those pretrained on ImageNet.

7.2 Benefits of Using Depth

Finally, we compare results of the network for different input modalities. For the depth information, we evaluate three variants (chapter 4): 1) absolute raw depth, 2) absolute depth with holes filled by the iterative median filter, and 3) surface normals. Normalization is applied to all input values so they are in range −1 to 1 (section 4.5).



(a) : *T-LESS test* dataset



(b) : *LINEMOD test* dataset

Figure 7.4: PR curves for different input modalities and datasets. The whole first layer of the network was initialized randomly (initialization strategy *#3*, section 7.1).

training initialization	channels	dataset	
		T-LESS test	*LINEMOD test*
#1	filled depth	0.905	0.738
	RGB + filled depth	**0.938**	**0.856**
#2	RGB	0.929	0.831
#3	grayscale	0.931	0.654
	RGB	0.930	0.785
	raw depth	0.886	0.625
	filled depth	0.892	0.696
	normals	0.892	0.718
	RGB + raw depth	0.933	0.821
	RGB + filled depth	**0.938**	0.824
	RGB + normals	0.933	0.811

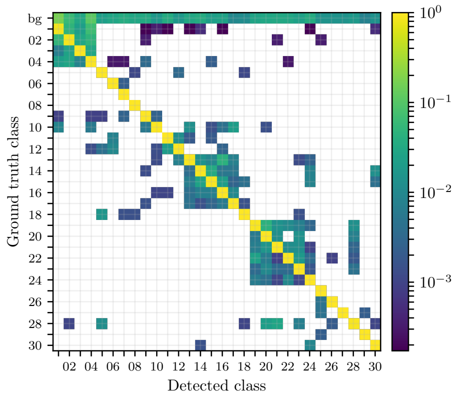
Table 7.1: mAP for the PR curves from fig. 7.4 with additional results for other initialization strategies.

Results in fig. 7.4 show that the combination of RGB and depth with filled holes yields the best results on both datasets. This is followed by RGB with the other depth encodings, which still outperform the RGB-only variant. The depth alone, in all considered encodings, yields the worst results.

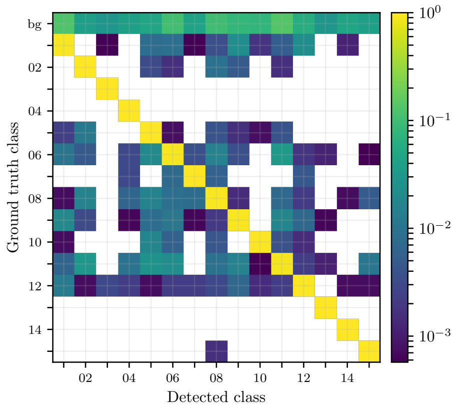
When only grayscale images are used as the input, the network achieves high performance on *T-LESS test*, even slightly higher than for RGB. This is not the case for *LINEMOD test*, which shows the importance of color for this dataset.

Table 7.1 shows results also for other initialization strategies. The overall best results were achieved with initialization stategy *#1* and RGB + filled depth as input.

Confusion matrix for *T-LESS test* in fig. 7.5a show many mistakes between objects 1 to 4, 13 to 17, and 19 to 24. This is expected since these objects are very similar. Figure 7.5b shows mistakes across most of the object classes.



(a) : *T-LESS test* dataset



(b) : *LINEMOD test* dataset

Figure 7.5: Confusion matrices for RGB + filled depth and initialization strategy *#1*. Normalized in rows. Note the logarithmic scale and false positives in the first row.

Chapter 8
Conclusions

In this thesis we have studied the Faster R-CNN method for detection of object categories in RGB images and proposed its extension to RGB-D images.

Analysis of the original Faster R-CNN showed that the method is sensitive to changes in background. When the method was trained on images with only black or only cluttered background and tested on images of the other type, the performance dropped significantly. The object proposals produced by Faster R-CNN were shown to cover the ground truth annotations very well. The well aligned proposals have often high objectness score.

We have discussed possible solutions to several problems of extending Faster R-CNN to RGB-D images: filling missing values in depth images, depth encoding (raw depth vs. surface normals), extension of the CNN architecture to accept the extra depth information, and initialization of weights in the extended network.

The overall best results, on both the T-LESS and the LINEMOD dataset, were achieved with a network that accepts RGB-D channels as input and has the depth weights in the first convolutional layer initialized with the mean of the color weights. The color weights and also all the other network weights were initialized with weights trained on ImageNet, and the depth channel was pre-processed by the iterative median filter to fill in the missing values. This method achieved 93.8% mAP on the T-LESS dataset and 85.6% mAP on the LINEMOD dataset. Note that the improvement over the original method using only RGB channels is not significant (mAP was increased by $1 - 2\%$), which suggests a need for different incorporation of the depth information, e.g. through a dedicated network branch (section 8.1).

Note that the mentioned scores were achieved on a simple task when the test image set was randomly split into two parts, one of which was used for training and one for testing. The method failed when it was trained on the original training images, as they have a black background and the method generalizes poorly to the unseen backgrounds in test images.

8.1 Future Work

We focused only on Faster R-CNN with ZF model of the CNN architecture, which has 5 convolutional layers. The authors of the method proposed also more complex VGG-16 model with 13 convolutional layers. The simpler model was used because of time requirements – one training took almost 8 hours even for the simpler model.

In [23], the state-of-the-art results in semantic segmentation in RGB-D images were achieved with a two-branch CNN architecture – one branch for RGB and one for depth. This late-fusion approach would be interesting to explore even for our task. Other depth encodings could be also explored, such as HHA [22].

The training process was suboptimal in several ways. We used the faster "approximate joint training" instead of the more precise "alternating training", both were proposed by the authors of Faster R-CNN. Learning rates and the number of iterations could be tweaked further. In all experiments, we trained the individual models only once, repetition would assure reliable results.

Appendix A

Bibliography

[1] Mark Everingham et al. “The pascal visual object classes challenge: A retrospective”. In: *International Journal of Computer Vision* 111.1 (2015), pp. 98–136.

[2] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 248–255.

[3] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 740–755.

[4] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *CoRR* abs/1506.01497 (2015). URL: http://arxiv.org/abs/1506.01497.

[5] Joseph Redmon and Ali Farhadi. “YOLO9000: Better, Faster, Stronger”. In: *arXiv preprint arXiv:1612.08242* (2016).

[6] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 779–788.

[7] Federico Tombari, Alessandro Franchi, and Luigi Di Stefano. “BOLD features to detect texture-less objects”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 1265–1272.

[8] Tomas Hodan et al. “Detection and fine 3D pose estimation of texture-less objects in RGB-D images”. In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE. 2015, pp. 4421–4428.

[9] Li Fei-Fei, Justin Johnson, and Serena Yeung. *CS231n Convolutional Neural Networks for Visual Recognition*. 2017. URL: http://cs231n.stanford.edu/.

[10] K. E. A. v. d. Sande, C. G. M. Snoek, and A. W. M. Smeulders. “Fisher and VLAD with FLAIR”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. June 2014, pp. 2377–2384. DOI: 10.1109/CVPR.2014.304.

[11] Xiaoyu Wang et al. “Regionlets for generic object detection”. In: *IEEE transactions on pattern analysis and machine intelligence* 37.10 (2015), pp. 2071–2084.

[12] Wanli Ouyang et al. “Deepid-net: Deformable deep convolutional neural networks for object detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2015, pp. 2403–2412.

[13] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2015, pp. 1–9.

[14] Ross Girshick. “Fast R-CNN”. In: *Proceedings of the IEEE International Conference on Computer Vision.* 2015, pp. 1440–1448.

[15] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *European conference on computer vision.* Springer. 2014, pp. 818–833.

[16] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[17] Jeff Kramer et al. *Hacking the Kinect.* Vol. 268. Springer, 2012.

[18] Kourosh Khoshelham and Sander Oude Elberink. “Accuracy and resolution of kinect depth data for indoor mapping applications”. In: *Sensors* 12.2 (2012), pp. 1437–1454.

[19] Marcelo Bertalmio et al. “Image inpainting”. In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques.* ACM Press/Addison-Wesley Publishing Co. 2000, pp. 417–424.

[20] Connelly Barnes et al. “PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing”. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 28.3 (Aug. 2009).

[21] Stefan Hinterstoisser et al. “Gradient response maps for real-time detection of textureless objects”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.5 (2012), pp. 876–888.

[22] Saurabh Gupta et al. “Learning rich features from RGB-D images for object detection and segmentation”. In: *European Conference on Computer Vision.* Springer. 2014, pp. 345–360.

[23] 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.* 2015, pp. 3431–3440.

[24] Tomas Hodan et al. “T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects”. In: *IEEE Winter Conference on Applications of Computer Vision (WACV)* (2017).

[25] Stefan Hinterstoisser et al. “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.

[26] Eric Brachmann et al. “Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3364–3372.

[27] Stefan Hinterstoisser et al. “Going further with point pair features”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 834–848.

[28] Mark Everingham and John Winn. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Development Kit*. 2012. URL: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/htmldoc/devkit_doc.html.

[29] Tomas Hodan, Jiri Matas, and Stepan Obdrzalek. “On Evaluation of 6D Object Pose Estimation”. In: *Computer Vision–ECCV 2016 Workshops*. 2016.

[30] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. “Diagnosing Error in Object Detectors”. In: (2012). URL: http://dhoiem.web.engr.illinois.edu/projects/detectionAnalysis/.

[31] Tomas Hodan et al. *SIXD Challenge 2017*. 2017. URL: http://cmp.felk.cvut.cz/sixd/challenge_2017/.

Appendix B
Content of the included DVD

Together with the printed copy of this thesis, a DVD with the following content is provided:

- **Faster R-CNN** - Source code of the extended method.
- **Tools** - Miscellaneous Python scripts used as a supplement of Faster R-CNN (e.g. initialization of the network models, evaluation) and plotting the figures.
- **Thesis.pdf** - Electronic version of this work.